Implicit MLE: Backpropagating Through Discrete Exponential Family Distributions

Mathias Niepert
NEC Laboratories Europe
mathias.niepert@neclab.eu

Pasquale Minervini
University College London
p.minervini@ucl.ac.uk

Luca Franceschi
Istituto Italiano di Tecnologia
University College London
ucablfr@ucl.ac.uk

Abstract

Combining discrete probability distributions and combinatorial optimization problems with neural network components has numerous applications but poses several challenges. We propose Implicit Maximum Likelihood Estimation (I-MLE), a framework for end-to-end learning of models combining discrete exponential family distributions and differentiable neural components. I-MLE is widely applicable as it only requires the ability to compute the most probable states and does not rely on smooth relaxations. The framework encompasses several approaches such as perturbation-based implicit differentiation and recent methods to differentiate through black-box combinatorial solvers. We introduce a novel class of noise distributions for approximating marginals via perturb-and-MAP. Moreover, we show that I-MLE simplifies to maximum likelihood estimation when used in some recently studied learning settings that involve combinatorial solvers. Experiments on several datasets suggest that I-MLE is competitive with and often outperforms existing approaches which rely on problem-specific relaxations.

1 Introduction

While deep neural networks excel at perceptual tasks, they tend to generalize poorly whenever the problem at hand requires some level of symbolic manipulation or reasoning, or exhibit some (known) algorithmic structure. Logic, relations, and explanations, as well as decision processes, frequently find natural abstractions in discrete structures, ill-captured by the continuous mappings of standard neural nets. Several application domains, ranging from relational and explainable ML to discrete decision-making [Mišić and Perakis, 2020], could benefit from general-purpose learning algorithms whose inductive biases are more amenable to integrating symbolic and neural computation. Motivated by these considerations, there is a growing interest in end-to-end learnable models incorporating discrete components that allow, e.g., to sample from discrete latent distributions [Jang et al., 2017, Paulus et al., 2020] or solve combinatorial optimization problems [Pogančić et al., 2019, Mandi et al., 2020]. Discrete energy-based models (EBMs) [LeCun et al., 2006] and discrete world models [Hafner et al., 2020] are additional examples of neural network-based models that require the ability to backpropagate through discrete probability distributions.

For complex discrete distributions, it is intractable to compute the exact gradients of the expected loss. For combinatorial optimization problems, the loss is discontinuous, and the gradients are zero almost everywhere. The standard approach revolves around problem-specific smooth relaxations, which allow one to fall back to (stochastic) backpropagation. These strategies, however, require tailor-made relaxations, presuppose access to the constraints and are, therefore, not always feasible nor tractable for large state spaces. Moreover, reverting to discrete outputs at test time may cause unexpected behavior. In other situations, discrete outputs are required at training time because one has to make one of a number of discrete choices, such as accessing discrete memory or deciding on an action in a game.

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

With this paper, we take a step towards the vision of general-purpose algorithms for hybrid learning systems. Specifically, we consider settings where the discrete component(s), embedded in a larger computational graph, are discrete random variables from the constrained exponential family[1]. Grounded in concepts from Maximum Likelihood Estimation (MLE) and perturbation-based implicit differentiation, we propose Implicit Maximum Likelihood Estimation (I-MLE). To approximate the gradients of the discrete distributions' parameters, I-MLE computes, at each update step, a *target distribution* q that depends on the loss incurred from the discrete output in the forward pass. In the backward pass, we approximate maximum likelihood gradients by treating q as the empirical distribution. We propose ways to derive target distributions and introduce a novel family of noise perturbations well-suited for approximating marginals via perturb-and-MAP. I-MLE is general-purpose as it only requires the ability to compute the most probable states and not faithful samples or probabilistic inference. In summary, we make the following contributions:

1. We propose implicit maximum likelihood estimation (I-MLE) as a framework for computing gradients with respect to the parameters of discrete exponential family distributions;
2. We show that this framework is useful for backpropagating gradients through *both* discrete probability distributions and discrete combinatorial optimization problems;
3. I-MLE requires two ingredients: a family of target distribution q and a method to sample from complex discrete distributions. We propose two families of target distributions and a family of noise-distributions for Gumbel-max (perturb-and-MAP) based sampling.
4. We show that I-MLE simplifies to *explicit* maximum-likelihood learning when used in some recently studied learning settings involving combinatorial optimization solvers.
5. Extensive experimental results suggest that I-MLE is flexible and competitive compared to the straight-through and relaxation-based estimators.

Instances of the I-MLE framework can be easily integrated into modern deep learning pipelines, allowing one to readily utilize several types of discrete layers with minimal effort. We provide implementations and Python notebooks at `https://github.com/nec-research/tf-imle`

2 Problem Statement and Motivation

We consider models described by the equations

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \quad \boldsymbol{z} \sim p(\boldsymbol{z}; \boldsymbol{\theta}), \quad \boldsymbol{y} = f_{\boldsymbol{u}}(\boldsymbol{z}), \quad (1)$$

where $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ denote feature inputs and target outputs, $h_{\boldsymbol{v}} : \mathcal{X} \to \Theta$ and $f_{\boldsymbol{u}} : \mathcal{Z} \to \mathcal{Y}$ are smooth parameterized maps, and $p(\boldsymbol{z}; \boldsymbol{\theta})$ is a discrete probability distribution.

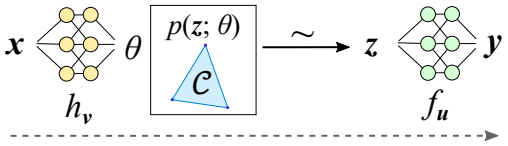

Figure 1: Illustration of the addressed learning problem. $\boldsymbol{z}$ is the discrete (latent) structure.

Given a set of examples $\mathcal{D} = \{(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_j)\}_{j=1}^N$, we are concerned with learning the parameters $\boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u})$ of (1) by finding approximate solutions of $\min_{\boldsymbol{\omega}} \sum_j L(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_j; \boldsymbol{\omega})/N$. The training error L is typically defined as:

$$L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) = \mathbb{E}_{\hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; \hat{\boldsymbol{\theta}})} \left[\ell(f_{\boldsymbol{u}}(\hat{\boldsymbol{z}}), \hat{\boldsymbol{y}}) \right] \quad \text{with} \quad \hat{\boldsymbol{\theta}} = h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}), \quad (2)$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a point-wise loss function. Fig. 1 illustrates the setting. For example, an interesting instance of (1) and (2) arises in *learning to explain* user reviews [Chen et al., 2018] where the task is to infer a target sentiment score (e.g. w.r.t. the quality of a product) from a review while also providing a concise explanation of the predicted score by selecting a subset of exactly k words (cf. Example 2). In Section 6, we present experiments precisely in this setting. As anticipated in the introduction, we restrict the discussion to instances in which $p(\boldsymbol{z}; \boldsymbol{\theta})$ belongs to the (constrained) *discrete exponential family*, which we now formally introduce.

Let $\boldsymbol{Z}$ be a vector of discrete random variables over a state space $\mathcal{Z}$ and let $\mathcal{C} \subseteq \mathcal{Z}$ be the set of states that satisfy a given set of *linear constraints*.[2] Let $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m$ be a real-valued parameter vector.

[1] This includes integer linear programs via a natural link that we outline in Example 3.

[2] For the sake of simplicity we assume $\mathcal{Z} \subseteq \{0, 1\}^m$. The set $\mathcal{C}$ is the integral polytope spanned by the given, problem-specific, linear constraints.

The probability mass function (PMF) of a discrete constrained exponential family r.v. is:

$$p(\boldsymbol{z};\boldsymbol{\theta}) = \begin{cases} \exp\left(\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle/\tau - A(\boldsymbol{\theta})\right) & \text{if } \boldsymbol{z} \in \mathcal{C}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here, $\langle \cdot, \cdot \rangle$ is the inner product and τ the temperature, which, if not mentioned otherwise, is assumed to be 1. $A(\boldsymbol{\theta})$ is the log-partition function defined as $A(\boldsymbol{\theta}) = \log\left(\sum_{\boldsymbol{z}\in\mathcal{C}} \exp\left(\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle/\tau\right)\right)$. We call $\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle$ the *weight* of the state $\boldsymbol{z}$. The *marginals* (expected value, mean) of the r.v.s $\mathbf{Z}$ are defined as $\boldsymbol{\mu}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\hat{\boldsymbol{z}}\sim p(\boldsymbol{z};\boldsymbol{\theta})}[\hat{\boldsymbol{z}}]$. Finally, the most probable or Maximum A-Posteriori (MAP) states are defined as $\texttt{MAP}(\boldsymbol{\theta}) \coloneqq \arg\max_{\boldsymbol{z}\in\mathcal{C}} \langle \boldsymbol{z}, \boldsymbol{\theta}\rangle$. The family of probability distributions we define here captures a broad range of settings and subsumes probability distributions such as positive Markov random fields and statistical relational formalisms [Wainwright and Jordan, 2008, Raedt et al., 2016]. We now discuss some examples which we will use in the experiments. Crucially, in Example 3 we establish the link between the constrained exponential family and integer linear programming (ILP) identifying the ILP cost coefficients with the distribution's parameters $\boldsymbol{\theta}$.

Example 1 (Categorical Variables)**.** *An m-way (one-hot) categorical variable corresponds to $p(\boldsymbol{z};\boldsymbol{\theta}) = \exp\left(\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle - A(\boldsymbol{\theta})\right)$, subject to the constraint $\langle \boldsymbol{z}, \mathbf{1}\rangle = 1$, where $\mathbf{1}$ is a vector of ones.*

As $\mathcal{C} = \{\mathbf{e}_i\}_{i=1}^m$, where $\mathbf{e}_i$ is the i-th vector of the canonical base, the parameters of the above distribution coincide with the weights, which are often called *logits* in this context. The marginals $\boldsymbol{\mu}$ coincide with the PMF and can be expressed through a closed-form smooth function of $\boldsymbol{\theta}$: the softmax. This facilitates a natural relaxation that involves using $\boldsymbol{\mu}(\boldsymbol{\theta})$ in place of $\boldsymbol{z}$ [Jang et al., 2017]. The convenient properties of the categorical distribution, however, quickly disappear even for slightly more complex distributions, as the following example shows.

Example 2 (k-subset Selection)**.** *Assume we want to sample binary m-dimensional vectors with k ones. This amounts to replacing the constraint in Example 1 by the constraint $\langle \boldsymbol{z}, \mathbf{1}\rangle = k$.*

Here, a closed-form expression for the marginals does not exist: sampling from this distribution requires computing the $\binom{m}{k} = O(m^k)$ weights (if $k \leq m/2$). Computing MAP states instead takes time linear in m.

Example 3 (Integer Linear Programs)**.** *Consider the combinatorial optimization problem given by the integer linear program $\arg\min_{\boldsymbol{z}\in\mathcal{C}}\langle \boldsymbol{z}, \mathbf{c}\rangle$, where $\mathcal{C}$ is an integral polytope and $\mathbf{c} \in \mathbb{R}^m$ is a vector of cost coefficients, and let $\boldsymbol{z}^*(\mathbf{c})$ be the set of its solutions. We can associate to the ILP the family (indexed by $\tau > 0$) of probability distributions $p(\boldsymbol{z};\boldsymbol{\theta})$ from* (3)*, with $\mathcal{C}$ the ILP polytope and $\boldsymbol{\theta} = -\mathbf{c}$. Then, for every $\tau > 0$, the solutions of the ILP correspond to the MAP states: $\texttt{MAP}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{z}\in\mathcal{C}}\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle = \boldsymbol{z}^*(\mathbf{c})$ and for $\tau \to 0$ one has that $\Pr(\mathbf{Z} \in \boldsymbol{z}^*(\mathbf{c})) \to 1$.*

Many problems of practical interest can be expressed as ILPs, such as finding shortest paths, planning and scheduling problems, and inference in propositional logic.

3 The Implicit Maximum Likelihood Estimator

In this section, we develop and motivate a family of general-purpose gradient estimators for Eq. (2) that respect the structure of $\mathcal{C}$.[3] Let $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \in \mathcal{D}$ be a training example and $\hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}))$. The gradient of L w.r.t. $\boldsymbol{u}$ is given by $\nabla_{\boldsymbol{u}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) = \mathbb{E}_{\hat{\boldsymbol{z}}}[\partial_{\boldsymbol{u}} f_{\boldsymbol{u}}(\hat{\boldsymbol{z}})^\intercal \nabla_{\boldsymbol{y}} \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})]$ with $\boldsymbol{y} = f_{\boldsymbol{u}}(\hat{\boldsymbol{z}})$, which may be estimated by drawing one or more samples from p. Regarding $\nabla_{\boldsymbol{v}} L$, one has

$$\nabla_{\boldsymbol{v}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) = \partial_{\boldsymbol{v}} h_{\boldsymbol{v}}(\hat{\boldsymbol{x}})^\intercal \nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}), \tag{4}$$

where the major challenge is to compute $\nabla_{\boldsymbol{\theta}} L$. A standard approach is to employ the score function estimator (SFE) which typically suffers from high variance. Whenever a pathwise derivative estimator (PDE) is available it is usually the preferred choice [Schulman et al., 2015]. In our setting, however, the PDE is not readily applicable since $\boldsymbol{z}$ is discrete and, therefore, every (exact) reparameterization path would be discontinuous. Various authors developed (biased) adaptations of the PDE for discrete r.v.s (see Section 5). These involve either smooth approximations of $p(\boldsymbol{z};\boldsymbol{\theta})$ or approximations of the derivative of the reparameterization map. *Our proposal departs from these two routes and instead involves the formulation of an implicit maximum likelihood estimation problem.* In a nutshell, I-MLE

[3] The derivations are adaptable to other types of losses defined over the outputs of Eq. (1).

Algorithm 1 Instance of I-MLE with perturbation-based implicit differentiation.

function FORWARDPASS($\boldsymbol{\theta}$)
  // Sample from the noise distribution $\rho(\boldsymbol{\epsilon})$
  $\boldsymbol{\epsilon} \sim \rho(\boldsymbol{\epsilon})$
  // Compute a MAP state of perturbed $\boldsymbol{\theta}$
  $\hat{\boldsymbol{z}} = \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$
  save $\boldsymbol{\theta}$, $\boldsymbol{\epsilon}$, and $\hat{\boldsymbol{z}}$ for the backward pass
  return $\hat{\boldsymbol{z}}$

function BACKWARDPASS($\nabla_{\boldsymbol{z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{z}), \hat{\boldsymbol{y}}), \lambda$)
  load $\boldsymbol{\theta}$, $\boldsymbol{\epsilon}$, and $\hat{\boldsymbol{z}}$ from the forward pass
  // Compute target distribution parameters
  $\boldsymbol{\theta}' = \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{z}), \hat{\boldsymbol{y}})$
  // Single sample I-MLE gradient estimate
  $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}') = \hat{\boldsymbol{z}} - \texttt{MAP}(\boldsymbol{\theta}' + \boldsymbol{\epsilon})$
  return $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}')$

is a (biased) estimator that replaces $\nabla_{\boldsymbol{\theta}} L$ in Eq. (4) with $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}$, where $\mathcal{L}$ is an implicitly defined MLE objective and $\widehat{\nabla}$ is an estimator of the gradient.

We now focus on deriving the (implicit) MLE objective $\mathcal{L}$. Let us assume we can, for any given $\hat{\boldsymbol{y}}$, construct an exponential family distribution $q(\boldsymbol{z}; \boldsymbol{\theta}')$ that, ideally, is such that

$$\mathbb{E}_{\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}; \boldsymbol{\theta}')} \left[\ell(f_{\boldsymbol{u}}(\hat{\boldsymbol{z}}), \hat{\boldsymbol{y}}) \right] \leq \mathbb{E}_{\hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; \boldsymbol{\theta})} \left[\ell(f_{\boldsymbol{u}}(\hat{\boldsymbol{z}}), \hat{\boldsymbol{y}}) \right]. \tag{5}$$

We will call q the *target distribution*. The idea is that, by making p more similar to q we can (iteratively) reduce the model loss $L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega})$. To this purpose, we define $\mathcal{L}$ as the MLE objective[4] between the model distribution p with parameters $\boldsymbol{\theta}$ and the target distribution q with parameters $\boldsymbol{\theta}'$:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') \coloneqq -\mathbb{E}_{\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}; \boldsymbol{\theta}')} [\log p(\hat{\boldsymbol{z}}; \boldsymbol{\theta})] = \mathbb{E}_{\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}; \boldsymbol{\theta}')} [A(\boldsymbol{\theta}) - \langle \hat{\boldsymbol{z}}, \boldsymbol{\theta} \rangle] \tag{6}$$

Now, exploiting the fact that $\nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) = \boldsymbol{\mu}(\boldsymbol{\theta})$, we can compute the gradient of $\mathcal{L}$ as

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \boldsymbol{\mu}(\boldsymbol{\theta}) - \mathbb{E}_{\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}; \boldsymbol{\theta}')} [\hat{\boldsymbol{z}}] = \boldsymbol{\mu}(\boldsymbol{\theta}) - \boldsymbol{\mu}(\boldsymbol{\theta}'), \tag{7}$$

that is , the difference between the marginals of the current distribution p and the marginals of the target distribution q, also equivalent to the gradient of the KL divergence between p and q.

We will not use Eq. (7) directly, as computing the marginals is, in general, a #P-hard problem and scales poorly with the dimensionality m. MAP states are typically less expensive to compute (e.g. see Example 2) and are often used directly to approximate $\boldsymbol{\mu}(\boldsymbol{\theta})$[5] or to compute perturb-and-MAP approximations, where $\boldsymbol{\mu}(\boldsymbol{\theta}) \approx \mathbb{E}_{\boldsymbol{\epsilon} \sim \rho(\boldsymbol{\epsilon})} \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ where $\boldsymbol{\epsilon} \sim \rho(\boldsymbol{\epsilon})$ is an appropriate *noise distribution* with domain $\mathbb{R}^m$. In this work we follow – and explore in more detail in Section 3.2 – the latter approach (also referred to as the Gumbel-max trick [cf. Papandreou and Yuille, 2011]), a strategy that retains most of the computational advantages of the pure MAP approximation but may be less crude. Henceforth, we only assume access to an algorithm to compute MAP states (such as a standard ILP solver in the case of Example 3) and rephrase Eq. (1) as

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \quad \boldsymbol{z} = \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \text{ with } \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}), \quad \boldsymbol{y} = f_{\boldsymbol{u}}(\boldsymbol{z}). \tag{8}$$

With Eq. (8) in place, the general expression for the I-MLE estimator is $\widehat{\nabla}_{\boldsymbol{v}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_{\boldsymbol{v}} h_{\boldsymbol{v}}(\hat{\boldsymbol{x}})^{\intercal} \widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ with $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\hat{\boldsymbol{x}})$ where, for $S \in \mathbb{N}^+$:

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{S} \sum_{i=1}^{S} [\texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_i) - \texttt{MAP}(\boldsymbol{\theta}' + \boldsymbol{\epsilon}_i)], \text{ with } \boldsymbol{\epsilon}_i \sim \rho(\boldsymbol{\epsilon}) \text{ for } i \in \{1, \ldots, S\}. \tag{9}$$

If the states of both the distributions p and q are binary vectors, $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') \in [-1, 1]^m$ and when $S = 1$ $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') \in \{-1, 0, 1\}^m$. In the following, we discuss the problem of constructing families of target distributions q. We will also analyze under what assumptions the inequality of Eq. (5) holds.

3.1 Target Distributions via Perturbation-based Implicit Differentiation

The efficacy of the I-MLE estimator hinges on a proper choice of q, a hyperparameter of our framework. In this section we derive and motivate a class of general-purpose target distributions, rooted in perturbation-based implicit differentiation (PID):

$$q(\boldsymbol{z}; \boldsymbol{\theta}') = p(\boldsymbol{z}; \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{z}} \ell(f_{\boldsymbol{u}}(\overline{\boldsymbol{z}}), \hat{\boldsymbol{y}})) \text{ with } \overline{\boldsymbol{z}} = \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \text{ and } \boldsymbol{\epsilon} \sim \rho(\boldsymbol{\epsilon}), \tag{10}$$

[4] We expand on this in Appendix A where we also review the classic MLE setup [Murphy, 2012, Ch. 9].

[5] This is known as the *perceptron learning rule* in standard MLE.

where $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\hat{\boldsymbol{x}})$, $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \in \mathcal{D}$ is a data point, and $\lambda > 0$ is a hyperparameter that controls the perturbation intensity.

To motivate Eq. (10), consider the setting where the inputs to f are the marginals of $p(\boldsymbol{z}; \boldsymbol{\theta})$ (rather than discrete perturb-and-MAP samples as in Eq. (8)), that is, $\boldsymbol{y} = f_{\boldsymbol{u}}(\boldsymbol{\mu}(\boldsymbol{\theta}))$ with $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\hat{\boldsymbol{x}})$, and redefine the training error L of Eq. (2) accordingly. A seminal result by Domke [2010] shows that, in this case, we can obtain $\nabla_{\boldsymbol{\theta}} L$ by perturbation-based differentiation as:

$$\nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) = \lim_{\lambda \to 0} \left\{ \frac{1}{\lambda} \left[\boldsymbol{\mu}(\boldsymbol{\theta}) - \boldsymbol{\mu}\left(\boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\mu}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega})\right) \right] \right\}, \tag{11}$$

where $\nabla_{\boldsymbol{\mu}} L = \partial_{\boldsymbol{\mu}} f_{\boldsymbol{u}}(\boldsymbol{\mu})^{\intercal} \nabla_{\boldsymbol{y}} \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$. The expression inside the limit may be interpreted as the gradient of an implicit MLE objective (see Eq. (7)) between the distribution p with (current) parameters $\boldsymbol{\theta}$ and p with parameters perturbed in the negative direction of the downstream gradient $\nabla_{\boldsymbol{\mu}} L$. Now, we can adapt (11) to our setting of Eq. (8) by resorting to the straight-through estimator (STE) assumption [Bengio et al., 2013]. Here, the STE assumption translates into reparameterizing $\boldsymbol{z}$ as a function of $\boldsymbol{\mu}$ and approximating $\partial_{\boldsymbol{\mu}} \boldsymbol{z} \approx \boldsymbol{I}$. Then, $\nabla_{\boldsymbol{\mu}} L = \partial_{\boldsymbol{\mu}} \boldsymbol{z}^{\intercal} \nabla_{\boldsymbol{z}} L \approx \nabla_{\boldsymbol{z}} L$ and we approximate Eq. (11) as:

$$\nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) \approx \frac{1}{\lambda} \left[\boldsymbol{\mu}(\boldsymbol{\theta}) - \boldsymbol{\mu}\left(\boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{z}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega})\right) \right] = \frac{1}{\lambda} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{z}} L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}; \boldsymbol{\omega})), \tag{12}$$

for some $\lambda > 0$. From Eq. (12) we derive (10) by taking a single sample estimator of $\nabla_{\boldsymbol{z}} L$ (with perturb-and-MAP sampling) and by incorporating the constant $1/\lambda$ into a global learning rate. I-MLE with PID target distributions may be seen as a way to generalize the STE to more complex distributions. Instead of using the gradients $\nabla_{\boldsymbol{z}} L$ to backpropagate directly, I-MLE uses them to construct a target distribution q. With that, it defines an implicit maximum likelihood objective, whose gradient (estimator) propagates the supervisory signal upstream, critically, taking the constraints into account. When using Eq. (10) with $\rho(\boldsymbol{\epsilon}) = \delta_0(\boldsymbol{\epsilon})$[6], the I-MLE estimator also recovers a recently proposed gradient estimation rule to differentiate through black-box combinatorial optimization problems [Pogančić et al., 2019]. I-MLE unifies existing gradient estimation rules in one framework. Algorithm 1 shows the pseudo-code of the algorithm implementing Eq. (9) for $S = 1$, using the PID target distribution of Eq. (10). The simplicity of the code also demonstrates that instances of I-MLE can easily be implemented as a layer.

We will resume the discussion about target distributions in Section 4, where we analyze more closely the setup of Example 3. Next, we focus on the perturb-and-MAP strategies and derive a class of noise distributions that is particularly apt to the settings we consider in this work.

3.2 A Novel Family of Perturb-and-MAP Noise Distributions

When p is a complex high-dimensional distribution, obtaining Monte Carlo estimates of the gradient in Eq. (7) requires approximate sampling. In this paper, we rely on perturbation-based sampling, also known as perturb and MAP [Papandreou and Yuille, 2011]. In this Section we propose a novel way to design tailored noise perturbations. While the proposed family of noise distributions works with I-MLE, the results of this section are of independent interest and can also be used in other (relaxed) perturb-and-MAP based gradient estimators [e.g. Paulus et al., 2020]. First, we start by revisiting a classic result by Papandreou and Yuille [2011] which theoretically motivates the perturb-and-MAP approach (also known as the Gumbel-max trick), which we generalize here to consider also the temperature parameter τ.

Proposition 1. *Let $p(\boldsymbol{z}; \boldsymbol{\theta})$ be a discrete exponential family distribution with integer polytope $\mathcal{C}$ and temperature τ, and let $\langle \boldsymbol{z}, \boldsymbol{\theta} \rangle$ be the unnormalized weight of each $\boldsymbol{z} \in \mathcal{C}$. Moreover, let $\tilde{\boldsymbol{\theta}}$ be such that, for all $\boldsymbol{z} \in \mathcal{C}$, $\langle \boldsymbol{z}, \tilde{\boldsymbol{\theta}} \rangle = \langle \boldsymbol{z}, \boldsymbol{\theta} \rangle + \epsilon(\boldsymbol{z})$ with each $\epsilon(\boldsymbol{z})$ sampled i.i.d. from $\mathrm{Gumbel}(0, \tau)$. Then we have that $\Pr(\texttt{MAP}(\tilde{\boldsymbol{\theta}}) = \boldsymbol{z}) = p(\boldsymbol{z}; \boldsymbol{\theta})$.*

All proofs can be found in Appendix B. The proposition states that if we can perturb the weights $\langle \boldsymbol{z}, \boldsymbol{\theta} \rangle$ of each $\boldsymbol{z} \in \mathcal{C}$ with independent $\mathrm{Gumbel}(0, \tau)$ noise, then obtaining MAP states from the perturbed model is equivalent to sampling from $p(\boldsymbol{z}; \boldsymbol{\theta})$ at temperature[7] τ. For complex exponential

[6] δ_0 is the Dirac delta centered around 0 – this is equivalent to approximating the marginals with `MAP`.

[7] Note that the temperature here is different to the temperature of the Gumbel softmax trick [Jang et al., 2017] which scales *both* the sum of the logits *and* the samples from $\mathrm{Gumbel}(0, 1)$.

distributions, perturbing the weights $\langle \boldsymbol{z}, \boldsymbol{\theta} \rangle$ for each state $\boldsymbol{z} \in \mathcal{C}$ is at least as expensive as computing the marginals exactly. Hence, one usually resorts to *local perturbations* of each $[\boldsymbol{\theta}]_i$ (the i-th entry of the vector $\boldsymbol{\theta}$) with Gumbel noise. Fortunately, we can prove that, for a large class of distributions, it is possible to design more suitable *local* perturbations. First, we show that, for any $\kappa \in \mathbb{N}^+$, a Gumbel distribution can be written as a finite sum of κ i.i.d. (implicitly defined) random variables.

Lemma 1. *Let $X \sim \mathrm{Gumbel}(0, \tau)$ and let $\kappa \in \mathbb{N}^+$. Define the Sum-of-Gamma distribution as*

$$\mathrm{SoG}(\kappa, \tau, s) \coloneqq \frac{\tau}{\kappa} \left\{ \sum_{i=1}^{s} \{\mathrm{Gamma}(1/\kappa, \kappa/i)\} - \log(s) \right\}, \tag{13}$$

where $s \in \mathbb{N}^+$ and $\mathrm{Gamma}(\alpha, \beta)$ is the Gamma distribution with shape α and scale β, and let $\mathrm{SoG}(\kappa, \tau) \coloneqq \lim_{s \to \infty} \mathrm{SoG}(\kappa, \tau, s)$. Then we have that $X \sim \sum_{j=1}^{\kappa} \boldsymbol{\epsilon}_j$, with $\boldsymbol{\epsilon}_j \sim \mathrm{SoG}(\kappa, \tau)$.

Based on Lemma 1, we can show that for exponential family distributions where every $\boldsymbol{z} \in \mathcal{C}$ has exactly k non-zero entries we can design perturbations of $\langle \boldsymbol{z}, \boldsymbol{\theta} \rangle$ following a Gumbel distribution.

Theorem 1. *Let $p(\boldsymbol{z}; \boldsymbol{\theta})$ be a discrete exponential family distribution with integer polytope $\mathcal{C}$ and temperature τ. Assume that if $\boldsymbol{z} \in \mathcal{C}$ then $\langle \boldsymbol{z}, \mathbf{1} \rangle = k$ for some constant $k \in \mathbb{N}^+$. Let $\tilde{\boldsymbol{\theta}}$ be the perturbation obtained by $[\tilde{\boldsymbol{\theta}}]_j = [\boldsymbol{\theta}]_j + \boldsymbol{\epsilon}_j$ with $\boldsymbol{\epsilon}_j \sim \mathrm{SoG}(k, \tau)$ from Eq. (13). Then, $\forall \boldsymbol{z} \in \mathcal{C}$ we have that $\langle \boldsymbol{z}, \tilde{\boldsymbol{\theta}} \rangle = \langle \boldsymbol{z}, \boldsymbol{\theta} \rangle + \epsilon(\boldsymbol{z})$, with $\epsilon(\boldsymbol{z}) \sim \mathrm{Gumbel}(0, \tau)$.*

Many problems such as k-subset selection, traveling salesman, spanning tree, and graph matching strictly satisfy the assumption of Theorem 1. We can, however, also apply the strategy in cases where the variance of $\langle \mathbf{Z}, \mathbf{1} \rangle$ is small (e.g. shortest weighted path). The Sum-of-Gamma perturbations provide a more fine-grained approach to noise perturbations. For $\tau = \kappa = 1$, we obtain the standard Gumbel perturbations. In contrast to the standard $\mathrm{Gumbel}(0, 1)$ noise, the proposed local Sum-of-Gamma perturbations result in weights' perturbations that follow the Gumbel distribution. Fig. 2 shows histograms of 10k samples, where each sample is either the sum of 5 samples from $\mathrm{Gumbel}(0, 1)$ (the standard approach) or the sum of $k = 5$ samples from $\mathrm{SoG}(5, 1, 10) = \frac{1}{5} \sum_{i=1}^{10} \{\mathrm{Gamma}(1/5, 5/i) - \log(10)\}$. While we still cannot sample faithfully from $p(\boldsymbol{z}; \boldsymbol{\theta})$ as the perturbations are *not* independent, we can counteract the problem of partially dependent perturbations by increasing the temperature τ and, therefore, the variance of the noise distribution. We explore and verify the importance of tuning τ empirically. In the appendix, we also show that the infinite series from Lemma 1 can be well approximated by a finite sum using convergence results for the Euler-Mascheroni series [Mortici, 2010].

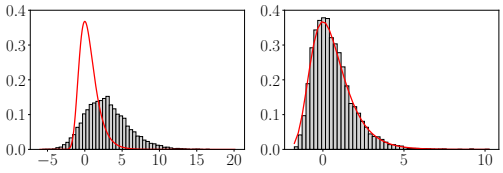

Figure 2: Histograms for 10k samples where each sample is (left) the sum of 5 $\epsilon_j \sim \mathrm{Gumbel}(0, 1)$ or (right) the sum of 5 $\epsilon_j \sim \mathrm{SoG}(5, 1, 10)$.

4 Target Distributions for Combinatorial Optimization Problems

In this section, we explore the setting where the discrete computational component arises from a combinatorial optimization (CO) problem, specifically an integer linear program (ILP). Many authors have recently considered the setup where the CO component occupies the last layer of the model defined by Eq. (1) (where $f_{\boldsymbol{u}}$ is the identity) and the supervision is available in terms of examples of either optimal solutions [e.g. Pogančić et al., 2019] or optimal cost coefficients (conditioned on the inputs) [e.g. Elmachtoub and Grigas, 2020]. We have seen in Example 3 that we can naturally associate to each ILP a probability distribution (see Eq. (3)) with $\boldsymbol{\theta}$ given by the negative cost coefficients $\boldsymbol{c}$ of the ILP and $\mathcal{C}$ the integral polytope. Letting $\tau \to 0$ is equivalent to taking the MAP in the forward pass. Furthermore, in Section 3.1 we showed that the I-MLE framework subsumes a recently propose method by Pogančić et al. [2019]. Here, instead, we show that, for a certain choice of the target distribution, I-MLE estimates the gradient of an explicit maximum likelihood learning loss $\mathcal{L}$ where the data distribution is ascribed to either (examples of) optimal solutions or optimal cost coefficients.

Let $q(\boldsymbol{z}; \boldsymbol{\theta}')$ be the distribution $p(\boldsymbol{z}; \boldsymbol{\theta}')$, with parameters

$$[\boldsymbol{\theta}']_i := \begin{cases} [\boldsymbol{\theta}]_i & \text{if } [\nabla_{\boldsymbol{z}} L]_i = 0 \\ -[\nabla_{\boldsymbol{z}} L]_i & \text{otherwise.} \end{cases} \tag{14}$$

In the first CO setting, we observe training data $\mathcal{D} = \{(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_j)\}_{j=1}^N$ where $\hat{\boldsymbol{y}}_j \in \mathcal{C}$ and the loss ℓ measures a distance between a discrete $\hat{\boldsymbol{z}}_j \sim p(\boldsymbol{z}; \boldsymbol{\theta}_j)$ with $\boldsymbol{\theta}_j = h_v(\hat{\boldsymbol{x}})$ and a given optimal solution of the ILP $\hat{\boldsymbol{y}}_j$. An example is the Hamming loss ℓ_H [Pogančić et al., 2019] defined as $\ell_H(\boldsymbol{z}, \boldsymbol{y}) = \boldsymbol{z} \circ (\mathbf{1} - \boldsymbol{y}) + \boldsymbol{y} \circ (\mathbf{1} - \boldsymbol{z})$, where $\circ$ denotes the Hadamard (or entry-wise) product.

Fact 1. *If one uses ℓ_H, then* I-MLE *with the target distribution of Eq.* (14) *and $\rho(\boldsymbol{\epsilon}) = \delta_0$ is equivalent to the perceptron-rule estimator of the MLE objective between $p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}_j))$ and $\hat{\boldsymbol{y}}_j$.*

It follows that the method by Pogančić et al. [2019] returns, for a large enough λ, the maximum-likelihood gradients (scaled by $1/\lambda$) approximated by the perceptron rule. The proofs are given in Appendix B.

In the second CO setting, we observe training data $\mathcal{D} = \{(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{c}}_j)\}_{j=1}^N$, where $\hat{\boldsymbol{c}}_j$ is the optimal cost conditioned on input $\hat{\boldsymbol{x}}_j$. Here, various authors [e.g. Elmachtoub and Grigas, 2020, Mandi et al., 2020, Mandi and Guns, 2020] use as point-wise loss the regret $\ell_R(\boldsymbol{\theta}, \boldsymbol{c}) = \boldsymbol{c}^\top (\boldsymbol{z}(\boldsymbol{\theta}) - \hat{\boldsymbol{z}}^*(\boldsymbol{c}))$ where $\boldsymbol{z}(\boldsymbol{\theta})$ is a state sampled from $p(\boldsymbol{z}; \boldsymbol{\theta})$ (possibly with temperature $\tau \to 0$, that is, a MAP state) and $\hat{\boldsymbol{z}}^*(\boldsymbol{c}) \in \boldsymbol{z}^*(\boldsymbol{c})$ is an optimal state for $\boldsymbol{c}$.

Fact 2. *If one uses ℓ_R then* I-MLE *with the target distribution of Eq.* (14) *is equivalent to the perturb-and-MAP estimator of the MLE objective between $p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}_j))$ and $p(\boldsymbol{z}; -\hat{\boldsymbol{c}}_j)$.*

This last result also implies that when using the target distribution q from (14) in conjunction with the regret, I-MLE performs maximum-likelihood learning minimizing the KL divergence between the current distribution and the distribution whose parameters are the optimal cost.

Moreover, both facts imply that, when sampling from the MAP states of the distribution q defined by Eq. (14), we have that $\ell(\hat{\boldsymbol{z}}, \hat{\boldsymbol{y}}) = 0$ for $\hat{\boldsymbol{z}} \in \texttt{MAP}(\boldsymbol{\theta}')$. Therefore, $\ell(\hat{\boldsymbol{z}}, \hat{\boldsymbol{y}}) = 0 \leq \mathbb{E}_{\hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; \boldsymbol{\theta})} [\ell((\hat{\boldsymbol{z}}, \hat{\boldsymbol{y}})]$, meaning that the inequality of Eq. (5) is satisfied for $\tau \to 0$.

5 Related Work

Several papers address the gradient estimation problem for discrete r.v.s, many resorting to relaxations. Maddison et al. [2017], Jang et al. [2017] propose the Gumbel-softmax distribution to relax categorical r.v.s; Paulus et al. [2020] study extensions to more complex probability distributions. The concrete distribution (the Gumbel-softmax distribution) is only directly applicable to categorical variables. For more complex distributions, one has to come up with tailor-made relaxations or use the straight-through or score function estimators (see for instance Kim et al. [2016], Grover et al. [2019]). In our experiments, we compare with the Gumbel-softmax estimator in Figure 4 (left and right). We show that the k-subset VAE trained with I-MLE achieves loss values that are similar to those of the categorical (1-subset) VAE trained with the Gumbel-softmax gradient estimator. Tucker et al. [2017], Grathwohl et al. [2018] develop parameterized control variates (the former was named REBAR) based on continuous relaxations for the score-function estimator. In contrast, we focus explicitly on problems where *only* discrete samples are used during training. Moreover, REBAR is tailored to categorical distributions. I-MLE is intended for models with complex distributions (e.g. those with with many constraints).

Approaches that do not rely on relaxations are specific to certain distributions [Bengio et al., 2013, Franceschi et al., 2019, Liu et al., 2019] or assume knowledge of $\mathcal{C}$ [Kool et al., 2020]. We provide a general-purpose framework that does not require access to the linear constraints and the corresponding integer polytope $\mathcal{C}$. Experiments in the next section show that while I-MLE only requires a MAP solver, it is competitive and sometimes outperforms tailor-made relaxations. SparseMAP [Niculae et al., 2018] is an approach to structured prediction and latent variables, replacing the exponential distribution (specifically, the softmax) with a sparser distribution. Similar to our work, it only presupposes the availability of a MAP oracle. LP-SparseMAP [Niculae and Martins, 2020] is an extension that uses a relaxation of the optimization problem rather than a MAP solver. Sparsity can also be exploited for efficient marginal inference in latent variable models [Correia et al., 2020].

A series of works about differentiating through CO problems [Wilder et al., 2019, Elmachtoub and Grigas, 2020, Ferber et al., 2020, Mandi and Guns, 2020] relax ILPs by adding L^1, L^2 or log-barrier regularization terms and differentiate through the KKT conditions deriving from the application of the cutting plane or the interior-point methods. These approaches are conceptually linked to techniques for differentiating through smooth programs [Amos and Kolter, 2017, Donti et al., 2017,

Agrawal et al., 2019, Chen et al., 2020, Domke, 2012, Franceschi et al., 2018] that arise not only in modelling but also in hyperparameter optimization and meta-learning. Pogančić et al. [2019], Rolínek et al. [2020], Berthet et al. [2020] propose methods that are not tied to a specific ILP solver. As we saw above, the former two, originally derived from a continuous interpolation argument, may be interpreted as special instantiations of I-MLE. The latter addresses the theory of perturbed optimizers and discusses perturb and MAP in the context of the Fenchel-Young loss. All the CO-related works assume that either optimal costs or solutions are given as training data, while I-MLE may be also applied in the absence of such supervision by making use of implicitly generated target distributions. Other authors focus on devising differentiable relaxations for specific CO problems such as SAT [Evans and Grefenstette, 2018] or MaxSAT [Wang et al., 2019]. Machine learning intersects with CO also in other contexts, e.g. in learning heuristics to improve the performances of CO solvers or differentiable models such as GNNs to "replace" them; see Bengio et al. [2020] and references therein.

Direct Loss Minimization [DLM, McAllester et al., 2010, Song et al., 2016] is also related to our work, but the assumption there is that examples of optimal states $\hat{\boldsymbol{z}}$ are given. Lorberbom et al. [2019] extend the DLM framework to discrete VAEs using coupled perturbations. Their approach is tailored to VAEs and not general-purpose. Under a methodological viewpoint, I-MLE inherits from classical MLE [Wainwright and Jordan, 2008] and perturb-and-MAP [Papandreou and Yuille, 2011]. The theory of perturb-and-MAP was used to derive general-purpose upper bounds for log-partition functions [Hazan and Jaakkola, 2012, Shpakova and Bach, 2016].

6 Experiments

The set of experiments can be divided into three parts. First, we analyze and compare the behavior of I-MLE with (i) the score function and (ii) the straight-through estimator using a toy problem. Second, we explore the latent variable setting where both $h_{\boldsymbol{v}}$ and $f_{\boldsymbol{u}}$ in Eq. (1) are neural networks and the optimal structure is *not* available during training. Finally, we address the problem of differentiating through black-box combinatorial optimization problems, where we use the target distribution derived in Section 4. More experimental details for available in the appendix.

Synthetic Experiments. We conducted a series of experiments with a tractable 5-subset distribution (see Example 2) where $\mathbf{z} \in \{0,1\}^{10}$. We set the loss to $L(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\boldsymbol{z}} \sim p(\mathbf{z};\boldsymbol{\theta})}[\|\hat{\boldsymbol{z}} - \mathbf{b}\|^2]$, where $\mathbf{b}$ is a fixed vector sampled from $\mathcal{N}(0, \mathbf{I})$. In Fig. 3 (Top), we plot optimization curves with means and standard deviations, comparing the proposed estimator with the straight-through (STE) and the score function (SFE) estimators. [8] For STE and I-MLE, we use Perturb-and-MAP (PaM) with Gumbel and $\mathrm{SoG}(1, 5, 10)$ noise, respectively. The SFE uses faithful samples and exact marginals (which is feasible only when m is very small) and converges much more slowly than the other methods, while the STE converges to worse solutions than those found using I-MLE. Fig. 3 (Bottom) shows the benefits of using SoG rather than Gumbel perturbations with I-MLE. While the best configurations for both are comparable, SoG noise achieves in average (over 100 runs) strictly better final values of L for more than 50% of the tested configurations (varying λ from Eq. (10) and the learning rate) and exhibit smaller variance (see Fig. 6). Additional details and results in Appendix C.1.

Learning to Explain. The BEERADVOCATE dataset [McAuley et al., 2012] consists of free-text reviews and ratings for 4 different aspects of beer: appearance, aroma, palate, and taste. Each sentence in the test set has annotations providing the words that best describe the various aspects. Following the experimental setup of recent work [Paulus et al., 2020], we address the problem introduced by the L2X paper [Chen et al., 2018] of learning a distribution over k-subsets of words that best explain a given aspect rating. The complexity of the MAP problem for the k-subset distribution is linear in k. sThe training set has 80k reviews for the aspect APPEARANCE and 70k reviews for all other aspects. Since the original dataset [McAuley et al., 2012] did not provide separate validation and test sets, we compute 10 different evenly sized validation/test splits of the 10k held out set and compute mean and standard deviation over 10 models, each trained on one split. Subset precision was computed using a subset of 993 annotated reviews. We use pre-trained word embeddings from Lei et al. [2016]. Prior work used non-standard neural networks for which an implementation is not available [Paulus

[8] Hyperparameters are optimized against L for all methods independently. Statistics are over 100 runs. We found STE slightly better with Gumbel rather than SoG noise. SFE failed with all tested PaM strategies.

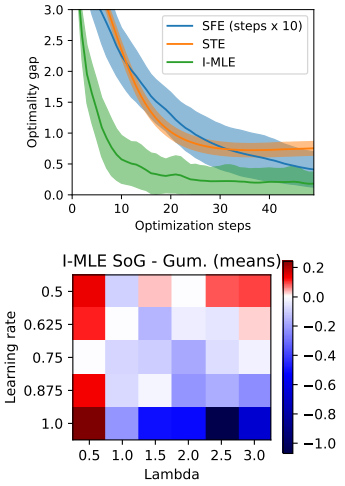

Figure 3: Top: Gradient-based optimization of L with various estimators. Bottom: Mean difference of the final value of L between I-MLE with SoG or Gumbel $\rho(\epsilon)$, varying λ and the learning rate (blue = better SoG).

Method	Test MSE		Subset Precision	
	Mean	Std. Dev.	Mean	Std. Dev.
$k = 10$				
L2X ($t = 0.1$)	6.68	1.08	26.65	9.39
SoftSub ($t = 0.5$)	**2.67**	0.14	44.44	2.27
STE ($\tau = 30$)	4.44	0.09	38.93	0.14
I-MLE MAP	4.08	0.91	14.55	0.04
I-MLE Gumbel	**2.68**	0.10	39.28	2.62
I-MLE ($\tau = 30$)	**2.71**	0.10	**47.98**	2.26
$k = 5$				
L2X ($t = 0.1$)	5.75	0.30	33.63	6.91
SoftSub ($t = 0.5$)	**2.57**	0.12	**54.06**	6.29
I-MLE ($\tau = 5$)	**2.62**	0.05	**54.76**	2.50
$k = 15$				
L2X ($t = 0.1$)	7.71	0.64	23.49	10.93
SoftSub ($t = 0.5$)	**2.52**	0.07	37.78	1.71
I-MLE ($\tau = 30$)	2.91	0.18	**39.56**	2.07

Table 1: Detailed results for the aspect AROMA. Test MSE and subset precision, both $\times 100$, for $k \in \{5, 10, 15\}$.

et al., 2020]. Instead, we used the neural network from the L2X paper with 4 convolutional and one dense layer. This neural network outputs the parameters $\boldsymbol{\theta}$ of the distribution $p(\boldsymbol{z}; \boldsymbol{\theta})$ over k-hot binary latent masks with $k \in \{5, 10, 15\}$. We compare to relaxation-based baselines L2X [Chen et al., 2018] and SoftSub [Xie and Ermon, 2019]. We also compare the straight-through estimator (STE) with Sum-of-Gamma (SoG) perturbations. We used the standard hyperparameter settings of Chen et al. [2018] and choose the temperature parameter $t \in \{0.1, 0.5, 1.0, 2.0\}$. For I-MLE we choose $\lambda \in \{10^1, 10^2, 10^3\}$, while for both I-MLE and STE we choose $\tau \in \{k, 2k, 3k\}$ based on the validation MSE. We used the standard Adam settings. We trained separate models for each aspect using MSE as point-wise loss ℓ.

Table 1 lists detailed results for the aspect AROMA. I-MLE's MSE values are competitive with those of the best baseline, and its subset precision is significantly higher than all other methods (for $\tau = 30$). Using only MAP as the approximation of the marginals leads to poor results. This shows that using the tailored perturbations with tuned temperature is crucial to achieve state of the art results. The Sum-of-Gamma perturbation introduced in this paper outperforms the standard local Gumbel perturbations. More details and results can be found in the appendix.

Discrete Variational Auto-Encoder. We evaluate various perturbation strategies for a discrete k-subset Variational Auto-Encoder (VAE) and compare them to the straight-through estimator (STE) and the Gumbel-softmax trick. The latent variables model a probability distribution over k-subsets of (or top-k assignments too) binary vectors of length 20. The special case of $k = 1$ is equivalent to a categorical variable with 20 categories. For $k > 1$, we use I-MLE using the class of PID target distributions of Eq. (10) and compare various perturb-and-MAP noise sampling strategies. The experimental setup is similar to those used in prior work on the Gumbel softmax tricks [Jang et al., 2017]. The loss is the sum of the reconstruction losses (binary cross-entropy loss on output pixels) and the KL divergence between the marginals of the variables and the uniform distribution. The encoding and decoding functions of the VAE consist of three dense layers (encoding: 512-256-20x20; decoding: 256-512-784). We do not use temperature annealing. Using Eq. (9) with $S = 1$, we use either $\mathrm{Gumbel}(0, 1)$ perturbations (the standard approach)[9] or Sum-of-Gamma (SoG) perturbations at a temperature of $\tau = 10$. We run 100 epochs and record the loss on the test data. The difference in training time is negligible. Fig. 4 shows that using the SoG noise distribution is beneficial. The test loss using the SoG perturbations is lower despite the perturbations having higher variance and, therefore, samples of the model being more diverse. This shows that using perturbations of the weights that follow a proper Gumbel distribution is indeed beneficial. I-MLE significantly outperforms the

[9]Increasing the temperature τ of $\mathrm{Gumbel}(0, \tau)$ samples increased the test loss.

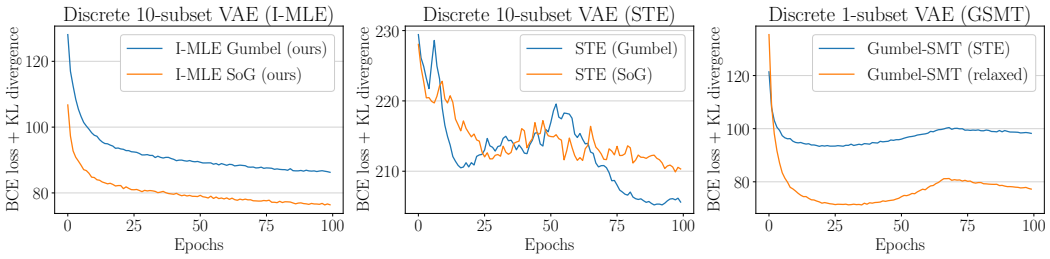

Figure 4: Plots of the sum of the binary reconstruction loss and the KL divergence as a function of the number of epochs (lower is better). (Left) Discrete 10-subset VAE trained with I-MLE with $\lambda = 10$ (I-MLE). (Center) Discrete 10-subset VAE trained with the straight-through estimator (STE). (Right) Discrete 1-subset VAE using the Gumbel softmax trick (GSMT). The down-up-down artifact is due to temperature annealing. Sum-of-Gamma (SoG) perturbations have the lowest test loss for the 10-subset VAEs. For $\lambda = 10$ and SoG perturbations, the test loss is similar to that of the categorical (1-subset) VAE trained with the Gumbel softmax trick.

STE, which does not work in this setting and is competitive with the Gumbel-Softmax trick for the 1-subset (categorical) distribution where marginals can be computed in closed form.

Differentiating through Combinatorial Solvers. In these experiments, proposed by Pogančić et al. [2019], the training datasets consists of 10,000 examples of randomly generated images of terrain maps from the Warcraft II tile set [Guyomarch, 2017]. Each example has an underlying $K \times K$ grid whose cells represent terrains with a fixed cost. The shortest (minimum cost) path between the top-left and bottom-right cell in the grid is encoded as an indicator matrix and serves as the target output. An image of the terrain map is presented to a CNN, which produces a $K \times K$ matrix of vertex costs. These costs are then given to *Dijkstra's algorithm* (the MAP solver) to compute the shortest path. We closely follow the evaluation protocol of Pogančić et al. [2019]. We considered two instantiations of I-MLE: one derived from Fact 1 (`M-M` in Table 2) using ℓ_H and one derived from Fact 2 ($\boldsymbol{\mu}$-$\boldsymbol{\mu}$) using ℓ_R, with $\rho(\epsilon) = \mathrm{SoG}(k, 1, 10)$ where k is the empirical mean of the path lengths (different for each grid size K). We compare with the method proposed by Pogančić et al. [2019] (BB[10]) and Berthet et al. [2020] (DPO). The results are listed in Table 2. I-MLE obtains results comparable to (BB) with `M-M` and is more accurate with $\boldsymbol{\mu}$-$\boldsymbol{\mu}$. We believe that the $\boldsymbol{\mu}$-$\boldsymbol{\mu}$ advantage may be partially due to an implicit form of data augmentation since we know from Fact 2 that, by using I-MLE, we obtain samples from the distribution whose parameters are the optimal cost. Training dynamics, showing faster convergence of I-MLE ($\boldsymbol{\mu}$-$\boldsymbol{\mu}$), and additional details are available in Table 4.

Table 2: Results for the Warcraft shortest path task. Reported is the accuracy, i.e. percentage of paths with the optimal costs. Standard deviations are over five runs.

K	I-MLE ($\boldsymbol{\mu}$-$\boldsymbol{\mu}$)	I-MLE (`M-M`)	BB	DPO
12	$\mathbf{97.2} \pm 0.5$	95.2 ± 0.3	95.2 ± 0.7	94.8 ± 0.3
18	$\mathbf{95.8} \pm 0.7$	94.4 ± 0.5	94.7 ± 0.4	92.3 ± 0.8
24	$\mathbf{94.3} \pm 1.0$	93.2 ± 0.2	93.8 ± 0.3	91.5 ± 0.4
30	93.6 ± 0.4	$\mathbf{93.7} \pm 0.6$	93.6 ± 0.5	91.5 ± 0.8

7 Conclusions

I-MLE is an efficient, simple-to-implement, and general-purpose framework for learning hybrid models. I-MLE is competitive with relaxation-based approaches for discrete latent-variable models and with approaches to backpropagate through CO solvers. Moreover, we showed empirically that I-MLE outperforms the straight-through estimator. A limitation of the work is its dependency on computing MAP states which is, in general, an NP-hard problem (although for many interesting cases there are efficient algorithms). Future work includes devising target distributions when $\nabla_{\boldsymbol{z}} L$ is not available, studying the properties (including the bias) of the proposed estimator, developing adaptive strategies for τ and λ, and integrating and testing I-MLE in several challenging application domains.

[10]Note that this is the same as using I-MLE with PID target distribution form Eq. (10) and $\rho(\epsilon) = \delta_0$.

References

A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019.

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

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J. Vert, and F. R. Bach. Learning with differentiable pertubed optimizers. In *NeurIPS*, 2020.

J. Chen, L. Song, M. Wainwright, and M. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.

X. Chen, Y. Zhang, C. Reisinger, and L. Song. Understanding deep architecture with reasoning layer. *Advances in Neural Information Processing Systems*, 33, 2020.

G. M. Correia, V. Niculae, W. Aziz, and A. F. Martins. Efficient marginalization of discrete and structured latent variables via sparsity. *Advances in Neural Information Processing Systems*, 2020.

J. Domke. Implicit differentiation by perturbation. In *Advances in Neural Information Processing Systems 23*, pages 523–531. 2010.

J. Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

P. L. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, 2017.

A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize", 2020.

R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

A. Ferber, B. Wilder, B. Dilkina, and M. Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1504–1511, 2020.

L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2018.

L. Franceschi, M. Niepert, M. Pontil, and X. He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *ICLR*, 2018.

A. Grover, E. Wang, A. Zweig, and S. Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

J. Guyomarch. Warcraft II Open-Source Map Editor. `http://github.com/war2/war2edit`, 2017.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

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

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

N. L. Johnson and N. Balakrishnan. Advances in the theory and practice of statistics, 1998.

C. Kim, A. Sabharwal, and S. Ermon. Exact sampling with integer linear programs and random perturbations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

W. Kool, H. van Hoof, and M. Welling. Estimating gradients for discrete random variables by sampling without replacement. *ICLR*, 2020.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

T. Lei, R. Barzilay, and T. Jaakkola. Rationalizing neural predictions. In *EMNLP*, 2016.

R. Liu, J. Regier, N. Tripuraneni, M. Jordan, and J. Mcauliffe. Rao-blackwellized stochastic gradients for discrete distributions. In *International Conference on Machine Learning*, pages 4023–4031. PMLR, 2019.

G. Lorberbom, A. Gane, T. Jaakkola, and T. Hazan. Direct optimization through argmax for discrete variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 6203–6214, 2019.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017.

J. Mandi and T. Guns. Interior point solving for lp-based prediction+optimisation. In *Advances in Neural Information Processing Systems*, 2020.

J. Mandi, E. Demirović, P. J. Stuckey, and T. Guns. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1603–1610, 2020.

D. A. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, volume 1, page 3, 2010.

J. McAuley, J. Leskovec, and D. Jurafsky. Learning attitudes and attributes from multi-aspect reviews. *2012 IEEE 12th International Conference on Data Mining*, pages 1020–1025, 2012.

V. V. Mišić and G. Perakis. Data analytics in operations management: A review. *Manufacturing & Service Operations Management*, 22(1):158–169, 2020.

C. Mortici. Fast convergences towards Euler-Mascheroni constant. *Computational & Applied Mathematics*, 29, 00 2010.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

V. Niculae and A. F. T. Martins. Lp-sparsemap: Differentiable relaxed optimization for sparse structured prediction. In *ICML*, 2020.

V. Niculae, A. F. T. Martins, M. Blondel, and C. Cardie. Sparsemap: Differentiable sparse structured inference. In *ICML*, 2018.

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

M. B. Paulus, D. Choi, D. Tarlow, A. Krause, and C. J. Maddison. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*, 2020.

M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.

L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016.

M. Rolínek, P. Swoboda, D. Zietlow, A. Paulus, V. Musil, and G. Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *ECCV*, 2020.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3528–3536, 2015.

T. Shpakova and F. Bach. Parameter learning for log-supermodular distributions. In *Advances in Neural Information Processing Systems*, volume 29, pages 3234–3242, 2016.

Y. Song, A. Schwing, R. Urtasun, et al. Training deep neural networks via direct loss minimization. In *International Conference on Machine Learning*, pages 2169–2177, 2016.

G. Tucker, A. Mnih, C. J. Maddison, D. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 2017.

M. J. Wainwright and M. I. Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.

B. Wilder, B. Dilkina, and M. Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

Xi'an. Which pdf of x leads to a gumbel distribution of the finite-size average of x? Cross Validated, 2016. URL `https://stats.stackexchange.com/q/214875`. URL:https://stats.stackexchange.com/q/214875 (version: 2016-05-27).

S. M. Xie and S. Ermon. Reparameterizable subset sampling via continuous relaxations. In *IJCAI*, 2019.

A Standard Maximum Likelihood Estimation and Links to I-MLE

In the standard MLE setting [see, e.g., Murphy, 2012, Ch. 9] we are interested in learning the parameters of a probability distribution, here assumed to be from the (constrained) exponential family (see Eq. (3)), given a set of example states. More specifically, given training data $\mathcal{D} = \{\hat{\mathbf{z}}_j\}_{j=1}^N$, with $\hat{\mathbf{z}}_j \in \mathcal{C} \subseteq \{0,1\}^m$, maximum-likelihood learning aims to minimize the empirical risk

$$\mathcal{L}(\boldsymbol{\theta}, q_{\mathcal{D}}) = \mathbb{E}_{\hat{\boldsymbol{z}} \sim q_{\mathcal{D}}}[-\log p(\hat{\mathbf{z}}; \boldsymbol{\theta})] = \frac{1}{N}\sum_{j=1}^{N} -\log p(\hat{\mathbf{z}}_j; \boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N} \left(A(\boldsymbol{\theta}) - \langle \hat{\mathbf{z}}_j, \boldsymbol{\theta} \rangle\right) \tag{15}$$

with respect to $\boldsymbol{\theta}$, where $q_{\mathcal{D}}(\boldsymbol{z}) = \sum_j \delta_{\hat{\boldsymbol{z}}_j}(\boldsymbol{z})/N$ is the empirical data distribution and $\delta_{\hat{\boldsymbol{z}}}$ is the Dirac delta centered in $\hat{\boldsymbol{z}}$. In Eq. (15), the point-wise loss ℓ is the negative log likelihood $-\log p(\hat{\boldsymbol{z}}, \boldsymbol{\theta})$, and $q_{\mathcal{D}}$ may be seen as a (data/empirical) *target distribution*. Note that in the main paper, as we assumed q to be from the exponential family with parameters $\boldsymbol{\theta}'$, we used the notation $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ to indicate the MLE objective rather than $\mathcal{L}(\boldsymbol{\theta}, q)$. These two definitions are, however, essentially equivalent.

Eq. (15) is a smooth objective that can be optimized with a (stochastic) gradient descent procedure. For a data point $\hat{\mathbf{z}}$, the gradient of the point-wise loss is given by $\nabla_{\boldsymbol{\theta}} \ell = \boldsymbol{\mu}(\boldsymbol{\theta}) - \hat{\mathbf{z}}$, since $\nabla_{\boldsymbol{\theta}} A = \boldsymbol{\mu}$. For the entire dataset, one has

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, q_{\mathcal{D}}) = \boldsymbol{\mu}(\boldsymbol{\theta}) - \frac{1}{N}\sum_{j=1}^{N} \hat{\boldsymbol{z}}_j = \boldsymbol{\mu}(\boldsymbol{\theta}) - \mathbb{E}_{\hat{\boldsymbol{z}} \sim q_{\mathcal{D}}}[\hat{\boldsymbol{z}}_j] \tag{16}$$

which is (cf. Eq. (7)) the difference between the marginals of $p(\boldsymbol{z}; \boldsymbol{\theta})$ (the mean of $\mathbf{Z}$) and the empirical mean of $\mathcal{D}$, $\boldsymbol{\mu}(q_{\mathcal{D}}) = \mathbb{E}_{\hat{\boldsymbol{z}} \sim \mathcal{D}}[\hat{\boldsymbol{z}}]$. As mentioned in the main paper, the main computational challenge when evaluating Eq. (16) is to compute the marginals (of $p(\boldsymbol{z}; \boldsymbol{\theta})$). There are many approximate schemes, one of which is the so-called *perceptron rule*, which approximate Eq. (16) as

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, q_{\mathcal{D}}) = \texttt{MAP}(\boldsymbol{\theta}) - \frac{1}{N}\sum_{j=1}^{N} \hat{\boldsymbol{z}}_j$$

and it is frequently employed in a stochastic manner by sampling one or more points from $\mathcal{D}$, rather than computing the full dataset mean.

We may interpret the standard MLE setting described in this section from the perspective of the problem setting we presented in Section 2. The first indeed amounts to the special case of the latter where there are no inputs ($\mathcal{X} = \emptyset$), the (target) output space coincides with the state space of the distribution ($\mathcal{Y} = \mathcal{Z}$), f is the identity mapping, ℓ is the negative log-likelihood and the model's parameter coincide with the distribution parameters, that is $\boldsymbol{\omega} = \boldsymbol{\theta}$.

B Proofs of Section 3.2 and Section 4

This section contains the proofs of the results relative to the perturb and map section (Section 3.2) and the section on optimal target distributions for typical loss functions when backpropagating through combinatorial optimization problems (section 4). We repeat the statements here, for the convenience of the reader.

Proposition 1. *Let $p(\boldsymbol{z}; \boldsymbol{\theta})$ be a discrete exponential family distribution with constraints $\mathcal{C}$ and temperature τ, and let $\langle \boldsymbol{\theta}, \boldsymbol{z} \rangle$ be the unnormalized weight of each $\boldsymbol{z}$ with $\boldsymbol{z} \in \mathcal{C}$. Moreover, let $\tilde{\boldsymbol{\theta}}$ be such that, for all $\boldsymbol{z} \in \mathcal{C}$,*

$$\langle \boldsymbol{z}, \tilde{\boldsymbol{\theta}} \rangle = \langle \boldsymbol{z}, \boldsymbol{\theta} \rangle + \epsilon(\boldsymbol{z})$$

with each $\epsilon(\boldsymbol{z})$ i.i.d. samples from $\mathrm{Gumbel}(0, \tau)$. *Then,*

$$\Pr\left(\texttt{MAP}(\tilde{\boldsymbol{\theta}}) = \boldsymbol{z}\right) = p(\boldsymbol{z}; \boldsymbol{\theta}).$$

Proof. Let $\epsilon_i \sim \text{Gumbel}(0, \tau)$ i.i.d. and $\tilde{\theta}_i = \theta_i + \epsilon_i$. Following a derivation similar to one made in Papandreou and Yuille [2011], we have:

$$
\begin{aligned}
&\Pr\{\arg\max(\tilde{\theta}_1, \ldots, \tilde{\theta}_m) = n\} = \\
=&\Pr\{\tilde{\theta}_n \geq \max_{j \neq n}\{\tilde{\theta}_j\}\} \\
=&\int_{-\infty}^{+\infty} g(t; \theta_n) \prod_{j \neq n} G(t; \theta_j)\, dt \\
=&\int_{-\infty}^{+\infty} \frac{1}{\tau} \exp\left(\frac{\theta_n - t}{\tau} - e^{\frac{\theta_n - t}{\tau}}\right) \prod_{j \neq n} \exp\left(-e^{\frac{\theta_j - t}{\tau}}\right)\, dt \\
=&\int_{-\infty}^{+\infty} \frac{1}{\tau} e^{\frac{\theta_n - t}{\tau}} \exp\left(-e^{\frac{\theta_n - t}{\tau}}\right) \prod_{j \neq n} \exp\left(-e^{\frac{\theta_j - t}{\tau}}\right)\, dt \\
=&\int_0^1 \prod_{j \neq n} z^{\exp\left(\frac{\theta_j - \theta_n}{\tau}\right)}\, dz \qquad \text{with } z \triangleq \exp\left(-e^{\frac{\theta_n - t}{\tau}}\right) \\
=&\frac{1}{1 + \sum_{j \neq n} e^{\frac{\theta_j - \theta_n}{\tau}}} \\
=&\frac{e^{\frac{\theta_n}{\tau}}}{\sum_{j=1}^m e^{\frac{\theta_j}{\tau}}},
\end{aligned}
$$

where g and G are respectively the Gumbel probability density function and the Gumbel cumulative density function. The proposition now follows from arguments made in Papandreou and Yuille [2011] using the maximal equivalent re-parameterization of $p(\boldsymbol{z}; \boldsymbol{\theta})$ where we specify a parameter $\langle \boldsymbol{\theta}, \boldsymbol{z} \rangle$ for each $\boldsymbol{z}$ with $\mathcal{C}(\boldsymbol{z})$ and perturb these parameters. □

Lemma 1. *Let $X \sim$ Gumbel$(0, \tau)$ and let $\kappa \in \mathbb{N} \setminus \{0\}$. Then we can write*

$$
X \sim \sum_{j=1}^{\kappa} \frac{\tau}{\kappa} \left[\lim_{s \to \infty} \sum_{i=1}^{s} \{\text{Gamma}(1/\kappa, \kappa/i)\} - \log(s) \right],
$$

where $\text{Gamma}(\alpha, \beta)$ *is the gamma distribution with shape α and scale β.*

Proof. Let $\kappa \in \mathbb{N} \setminus \{0\}$ and let $X \sim \text{Gumbel}(0, \tau)$. Its moment generating function has the form

$$
\mathbb{E}[\exp(tX)] = \Gamma(1 - \tau t). \tag{17}
$$

As mentioned in Johnson and Balakrishnan [p. 443, 1998] we know that we can write the Gamma function as

$$
\Gamma(1 - \tau t) = e^{\gamma \tau t} \prod_{i=1}^{\infty} \left(1 - \frac{\tau t}{i}\right)^{-1} e^{\frac{-\tau t}{i}} \tag{18}
$$

where γ is the Euler-Mascheroni constant. We have that

$$
\left(1 - \frac{\tau t}{i}\right)^{-1} = \frac{i}{i - \tau t} = \frac{\frac{i}{\tau}}{\frac{i - \tau t}{\tau}} = \frac{\frac{i}{\tau}}{\frac{i}{\tau} - \frac{\tau t}{\tau}} = \frac{\frac{i}{\tau}}{\frac{i}{\tau} - t}.
$$

The last term is the moment generating function of an exponential distribution with scale $\frac{\tau}{i}$. We can now take the logarithm on both sides of Eq. (18) and obtain

$$
tX = \gamma \tau t + \lim_{s \to \infty} \sum_{i=1}^{s} \left(t \,\text{Exp}(\tau / i) - \frac{\tau t}{i} \right),
$$

where $\mathrm{Exp}(\alpha)$ is the exponential distribution with scale α. Hence,

$$\begin{aligned}
X &\sim \lim_{s\to\infty} \sum_{i=1}^{s} \left(\mathrm{Exp}(\tau/i) - \frac{\tau}{i}\right) + \gamma\tau \\
&= \lim_{s\to\infty} \sum_{i=1}^{s} \left(\mathrm{Exp}(\tau/i) - \frac{\tau}{i}\right) + \tau \lim_{s\to\infty} \sum_{i=1}^{s} \frac{1}{i} - \log(s) \\
&= \lim_{s\to\infty} \sum_{i=1}^{s} \left(\mathrm{Exp}(\tau/i) - \frac{\tau}{i} + \frac{\tau}{i}\right) - \tau\log(s) \\
&= \lim_{s\to\infty} \sum_{i=1}^{s} \mathrm{Exp}(\tau/i) - \tau\log(s)
\end{aligned}$$

Since $\mathrm{Exp}(\alpha) \sim \mathrm{Gamma}(1, \alpha)$, and due to the scaling and summation properties of the Gamma distribution (with shape-scale parameterization), we can write for all $r > 1$:

$$\mathrm{Exp}(\alpha) \sim \sum_{j=1}^{r} \mathrm{Gamma}(1/r, \alpha r)/r.$$

Hence, picking $r = \kappa$ from the hypothesis, we have

$$\begin{aligned}
X &\sim \lim_{s\to\infty} \left\{ \sum_{i=1}^{s} \sum_{j=1}^{\kappa} \mathrm{Gamma}(1/\kappa, \tau\kappa/i)/\kappa \right\} - \tau\log(s) \\
&= \lim_{s\to\infty} \left\{ \sum_{j=1}^{\kappa} \frac{\tau}{k} \sum_{i=1}^{s} \mathrm{Gamma}(1/\kappa, \kappa/i) \right\} - \sum_{j=1}^{\kappa} \frac{\tau}{\kappa}\log(s) \\
&= \lim_{s\to\infty} \sum_{j=1}^{\kappa} \frac{\tau}{\kappa} \left\{ \left[\sum_{i=1}^{s} \mathrm{Gamma}(1/\kappa, \kappa/i) \right] - \log(s) \right\} \\
&= \sum_{j=1}^{\kappa} \frac{\tau}{\kappa} \left\{ \lim_{s\to\infty} \left[\sum_{i=1}^{s} \mathrm{Gamma}(1/\kappa, \kappa/i) \right] - \log(s) \right\}
\end{aligned}$$

This concludes the proof. Parts of the proof are inspired by a post on stackexchange Xi'an [2016]. □

Theorem 1. *Let $p(\boldsymbol{z}; \boldsymbol{\theta})$ be a discrete exponential family distribution with constraints $\mathcal{C}$ and temperature τ, and let $k \in \mathbb{N} \setminus \{0\}$. Let us assume that if $\mathcal{C}(\boldsymbol{z})$ then $\langle \boldsymbol{z}, \mathbf{1}\rangle = k$. Let $\tilde{\boldsymbol{\theta}}$ be the perturbation obtained by $\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i + \epsilon_i$ with*

$$\epsilon_i \sim \frac{\tau}{k} \left[\lim_{s\to\infty} \sum_{i=1}^{s} \{\mathrm{Gamma}(1/k, k/i)\} - \log(s) \right], \tag{19}$$

where $\mathrm{Gamma}(\alpha, \beta)$ is the gamma distribution with shape α and scale β. Then, for every $\boldsymbol{z}$ we have that $\langle \boldsymbol{z}, \tilde{\boldsymbol{\theta}}\rangle = \langle \boldsymbol{z}, \boldsymbol{\theta}\rangle + \epsilon(\boldsymbol{z})$ with $\epsilon(\boldsymbol{z}) \sim \mathrm{Gumbel}(0, \tau)$.

Proof. Since we perturb each θ_i by ε_i we have, by assumption, that $\langle \boldsymbol{\theta}, \mathbf{1}\rangle = k$, for every $\boldsymbol{z}$ with $\mathcal{C}(\boldsymbol{z})$, that

$$\langle \boldsymbol{z}, \tilde{\boldsymbol{\theta}}\rangle = \langle \boldsymbol{z}, \boldsymbol{\theta}\rangle + \sum_{j=1}^{k} \varepsilon_j. \tag{20}$$

Since by Lemma 1 we know that $\sum_{j=1}^{k} \varepsilon_i \sim \mathrm{Gumbel}(0, \tau)$, the statement of the theorem follows. □

The following theorem shows that the infinite series from Lemma 1 can be well approximated by a finite sum using convergence results for the Euler-Mascheroni series.

Theorem 2. *Let* $X \sim \text{Gumbel}(0, \tau)$ *and* $\tilde{X}(m) \sim \sum_{j=1}^{\kappa} \frac{\tau}{\kappa} \left[\sum_{i=1}^{m} \{\text{Gamma}(1/\kappa, \kappa/i)\} - \log(m)\right]$. *Then*

$$\frac{\tau}{2(m+1)} < \mathbb{E}[\tilde{X}(m)] - \mathbb{E}[X] < \frac{\tau}{2m}.$$

Proof. We have that

$$\begin{aligned}
\mathbb{E}[\tilde{X}(m)] &= \\
&= \mathbb{E}\left[\sum_{j=1}^{\kappa} \frac{\tau}{\kappa} \left[\sum_{i=1}^{m} \{\text{Gamma}(1/\kappa, \kappa/i)\} - \log(m)\right]\right] \\
&= \sum_{i=1}^{m} \mathbb{E}\left[\text{Gamma}(1/\kappa, \tau\kappa/i)\right] - \tau \log(m) \\
&= \left[\sum_{i=1}^{m} \frac{1}{\kappa} \frac{\tau\kappa}{i} - \tau \log(m)\right] \\
&= \tau \left[\sum_{i=1}^{m} \frac{1}{i} - \log(m)\right].
\end{aligned}$$

If $X \sim \text{Gumbel}(0, \tau)$, we know that $\mathbb{E}[X] = \tau\gamma$. Hence,

$$\begin{aligned}
\mathbb{E}[\tilde{X}(m)] - \mathbb{E}[X] &= \tau \left[\sum_{i=1}^{m} \frac{1}{i} - \log(m)\right] - \tau\gamma \\
&= \tau \left[\sum_{i=1}^{m} \frac{1}{i} - \log(m) - \gamma\right].
\end{aligned}$$

The theorem now follows from convergence results of the Euler-Mascheroni series Mortici [2010]. □

Fact 1. *If one uses* ℓ_H*, then* I-MLE *with the target distribution of Eq.* (14) *and* $\rho(\boldsymbol{\epsilon}) = \delta_0(\boldsymbol{\epsilon})$ *is equivalent to the perceptron-rule estimator of the MLE objective between* $p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}_j))$ *and* $\hat{\boldsymbol{y}}_j$.

Proof. Rewriting the definition of the Hamming loss gives us

$$\ell_H(\boldsymbol{z}, \boldsymbol{y}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{z}_i + \boldsymbol{y}_i - 2\boldsymbol{z}_i\boldsymbol{y}_i\right).$$

Hence, we have that

$$\nabla_{\boldsymbol{z}_i} \ell_H = \frac{1}{m} \left(1 - 2\boldsymbol{y}_i\right).$$

Therefore, $\nabla_{\boldsymbol{z}_i} \ell_H = -\frac{1}{m}$ if $\boldsymbol{y}_i = 1$ and $\nabla_{\boldsymbol{z}_i} H = \frac{1}{m}$ if $\boldsymbol{y}_i = 0$. Since, by definition $\boldsymbol{y} \in \mathcal{C}$, we have that

$$\texttt{MAP}\left(-\nabla_{\boldsymbol{z}} \ell_H\right) = \boldsymbol{y}.$$

Now, when using I-MLE with $S = 1$ and $\rho(\boldsymbol{\epsilon}) = \delta_0(\boldsymbol{\epsilon})$ we approximate the gradients as

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \texttt{MAP}(\boldsymbol{\theta}) - \texttt{MAP}(\boldsymbol{\theta}') = \texttt{MAP}(\boldsymbol{\theta}) - \texttt{MAP}\left(-\nabla_{\boldsymbol{z}} \ell_H\right) = \texttt{MAP}(\boldsymbol{\theta}) - \boldsymbol{y}.$$

This concludes the proof. □

Fact 2. *If one uses* ℓ_R *then* I-MLE *with the target distribution of Eq.* (14) *is equivalent to the perturb-and-MAP estimator of the MLE objective between* $p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}_j))$ *and* $p(\boldsymbol{z}; -\hat{\boldsymbol{c}}_j)$.

Proof. We have that $\nabla_{\boldsymbol{z}_i} \ell_R = \boldsymbol{c}_i$ for all i. Now, when using I-MLE with target distribution $\hat{q}(\boldsymbol{z}; \boldsymbol{\theta}')$ of Eq. (14) (and without loss of generality, for $S = 1$) we have that $\hat{q}(\boldsymbol{z}; \boldsymbol{\theta}') = p(\boldsymbol{z}; -\boldsymbol{c})$, and we approximate the gradients as

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_i) - \texttt{MAP}(\boldsymbol{\theta}' + \boldsymbol{\epsilon}_i) = \texttt{MAP}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_i) - \texttt{MAP}(-\boldsymbol{c} + \boldsymbol{\epsilon}_i), \text{ where } \boldsymbol{\epsilon}_i \sim \rho(\boldsymbol{\epsilon}).$$

Hence, I-MLE approximates the gradients of the maximum likelihood estimation problem between $p(\boldsymbol{z}; h_{\boldsymbol{v}}(\hat{\boldsymbol{x}}_j))$ and $p(\boldsymbol{z}; -\hat{\boldsymbol{c}}_j)$ using perturb-and-MAP. This concludes the proof. □

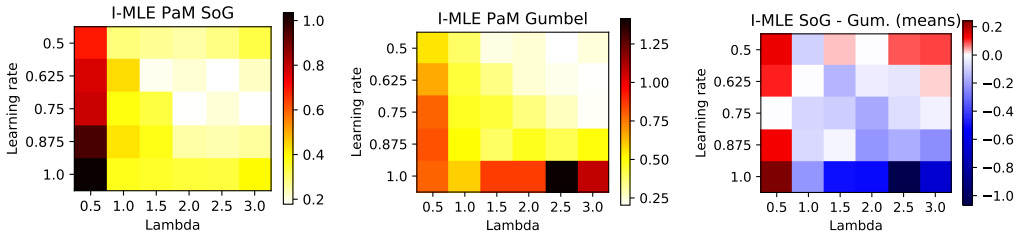

Figure 5: Average (over 100 runs) values of $L(\boldsymbol{\theta})$ after 50 steps of stochastic gradient descent (with momentum) using single-sample I-MLE with $\mathrm{SoG}(1, 5, 10)$ noise (left) and $\mathrm{Gumbel}(0, 1)$ noise (center) varying the perturbation intensity λ (see Eq. (10)) and learning rate. The rightmost heat-map depicts the (point-wise) difference between the two methods (blue = better SoG).

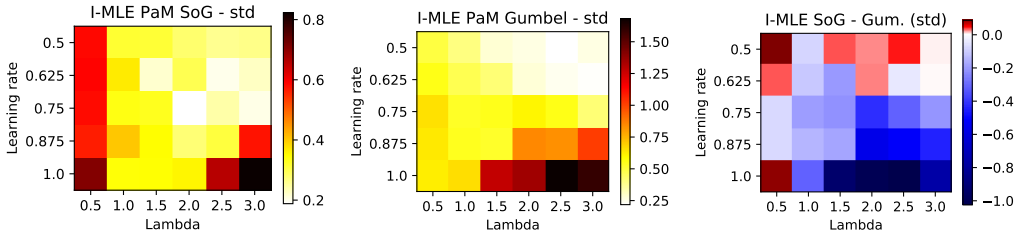

Figure 6: Same as above, but reporting standard deviations.

C Experiments: Details and Additional Results

C.1 Synthetic Experiments

In this series of experiments we analyzed the behaviour of various discrete gradient estimators, comparing our proposed I-MLE with standard straight-trhough (STE) and score-function (SFE) estimators. We also study of the effect of using Sum-of-Gamma perturbations rather than standard Gumbel noise. In order to be able to compute exactly (up to numerical precision) all the quantities involved, we chose a tractable 5-subset distribution (see Example 2) of size $m = 10$. We set the loss to $L(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\boldsymbol{z}} \sim p(\mathbf{z};\boldsymbol{\theta})}[\|\hat{\boldsymbol{z}} - \mathbf{b}\|^2]$, where $\mathbf{b}$ is a fixed vector sampled (only once) from $\mathcal{N}(0, \mathbf{I})$. This amounts to an unconditional setup where there are no input features (as in the standard MLE setting of Appendix A), but where the point-wise loss $\ell(\boldsymbol{z})$ is the Euclidean distance between the distribution output and a fixed vector $\mathbf{b}$. In Fig. 7 we plot the runtime (mean and standard deviation over 10 evaluations) of the full objective L (Expect., in the plot), of a (faithful) sample of ℓ and of a perturb-and-MAP sample with Sum-of-Gamma noise distribution (P&M) for increasing size m, with $k = m/2$. As it is evident from the plot, the runtime for both expectation and faithful samples, which require computing all the states in $\mathcal{C}$, increases exponentially, while perturb and MAP remains almost constant.

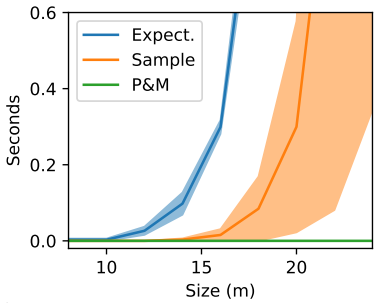

Figure 7: Runtime (mean and standard deviation) for computing L and samples of it, as the dimensionality of the k-subset distribution increases (with $k = m/2$).

Within this setting, the one-sample I-MLE estimator is

$$\widehat{\nabla}_{\text{I-MLE}} L(\boldsymbol{\theta}) = \texttt{MAP}(\boldsymbol{\theta} + \epsilon) - \texttt{MAP}(\boldsymbol{\theta}' + \epsilon), \text{ with } \epsilon \sim \mathrm{SoG}(1, 5, 10)$$

where $\boldsymbol{\theta}' = \boldsymbol{\theta} - \lambda[2(\hat{\boldsymbol{z}} - \mathbf{b})]$, where $\hat{\boldsymbol{z}} = \texttt{MAP}(\boldsymbol{\theta} + \epsilon)$ is a (perturb-and-MAP) sample, while the one-sample straight through estimator is

$$\widehat{\nabla}_{\text{STE}} L(\boldsymbol{\theta}) = 2(\hat{\boldsymbol{z}} - \mathbf{b}), \text{ with } \hat{\boldsymbol{z}} = \texttt{MAP}(\boldsymbol{\theta} + \epsilon), \ \epsilon \sim \mathrm{Gumbel}(0, 1).$$

For the score function estimator, we have used an expansive faithful sample/full marginal implementation given by

$$\widehat{\nabla}_{\text{SFE}} L(\boldsymbol{\theta}) = \|\hat{\boldsymbol{z}} - \mathbf{b}\|^2 \nabla_{\boldsymbol{\theta}} \log p(\hat{\boldsymbol{z}}; \boldsymbol{\theta}) = \|\hat{\boldsymbol{z}} - \mathbf{b}\|^2 [\hat{\boldsymbol{z}} - \boldsymbol{\mu}(\boldsymbol{\theta})], \text{ with } \hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; \boldsymbol{\theta})$$

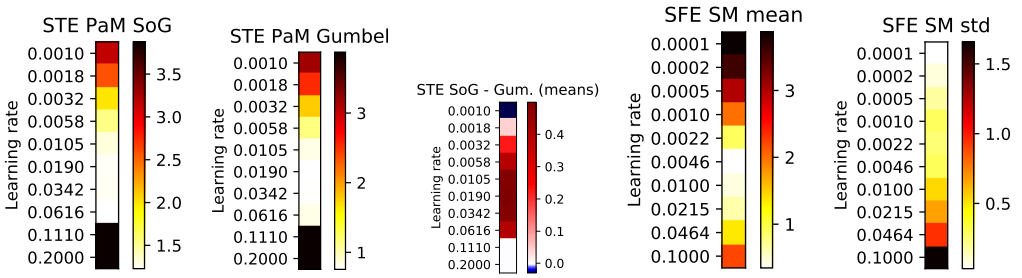

Figure 8: First three plots, from left to right: average (over 100 runs) final values of $L(\boldsymbol{\theta})$ after 50 steps of optimization using the straight-through estimator with SoG noise, varying the learning rate; same but using Gumbel noise; difference of averages between the first and the second heat-maps. Last two plots, from left to right: average (over 20 runs) final values of $L(\boldsymbol{\theta})$ after 500 steps of optimization using the score function estimator (with faithful samples and exact marginals), varying the learning rate; standard deviation for the same setting.

Method	Appearance		Palate		Taste	
	Test MSE	Subset precision	Test MSE	Subset precision	Test MSE	Subset precision
L2X ($t = 0.1$)	10.70 ± 4.82	30.02 ± 15.82	6.70 ± 0.63	50.39 ± 13.58	6.92 ± 1.61	32.23 ± 4.92
SoftSub ($t = 0.5$)	$\mathbf{2.48} \pm 0.10$	52.86 ± 7.08	$\mathbf{2.94} \pm 0.08$	39.17 ± 3.17	$\mathbf{2.18} \pm 0.10$	$\mathbf{41.98} \pm 1.42$
I-MLE ($\tau = 30$)	$\mathbf{2.51} \pm 0.05$	$\mathbf{65.47} \pm 4.95$	$\mathbf{2.96} \pm 0.04$	40.73 ± 3.15	2.38 ± 0.04	$\mathbf{41.38} \pm 1.55$

Table 3: Experimental results (mean $\pm$ std. dev.) for the learning to explain experiments for $k = 10$ and various aspects.

since, in preliminary experiments, we did not manage to obtain meaningful results with SFE using perturb-and-MAP for sampling and/or marginals approximation. These equations give the formulae for the estimators which we used for the results plotted in Fig. 3 (top) in the main paper.

In Fig. 5 we plot the heat-maps for the sensitivity results comparing between I-MLE with SoG and I-MLE with Gumbel perturbations. The two leftmost heat-maps depict the average value (over 100 runs) of $L(\boldsymbol{\theta})$ after 50 steps of stochastic gradient descent, for various choices of λ and learning rates (momentum factor was fixed at 0.9 for all experiments). The rightmost plot of Fig. 5 is the same as the one in the main paper, and represents the difference between the first and the second heat-maps. Fig. 6 refers to the same setting, but this time showing standard deviations. The rightmost plot of Fig. 6 suggests that using SoG perturbations results also in reduced variance (of the final loss) for most of the tried hyperparameter combinations. Finally, in Fig. 8 we show sensitivity plots for STE (both with SoG and Gumbel perturbations) and SFE, where we vary the learning rate.

C.2 Learning to Explain

Experiments were run on a server with Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz, 4 GeForce GTX 1080 Ti, and 128 GB RAM.

The pre-trained word embeddings and data set can be found here: `http://people.csail.mit.edu/taolei/beer/`. Figure 10 depicts the neural network architecture used for the experiments. As in prior work, we use a batch size of 40. The maximum review length is 350 tokens. We use the standard neural network architecture from prior work Chen et al. [2018], Paulus et al. [2020]. The dimensions of the token embeddings (of the embedding layers) are 200. All 1D convolutional layers have 250 filters with a kernel size of 3. All dense layers have a dimension of 100. The dropout layer has a dropout rate of 0.2. The layer Multiply perform the multiplication between the token mask (output of I-MLE) and the embedding matrix. The Lambda layer computes the mean of the selected embedding vectors. The last dense layer has a sigmoid activation. IMLESubsetkLayer is the layer implementing I-MLE. We train for 20 epochs using the standard Adam settings in Tensorflow 2.4.1 (learning rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-07, amsgrad=False), and no learning rate schedule. The training time (for the 20 epochs) for I-MLE, with sum-of-Gamma perturbations, is 380 seconds, for SoftSub 360 seconds, and for L2X 340 seconds. We always evaluate the model with the best validation MSE among the 20 epochs.

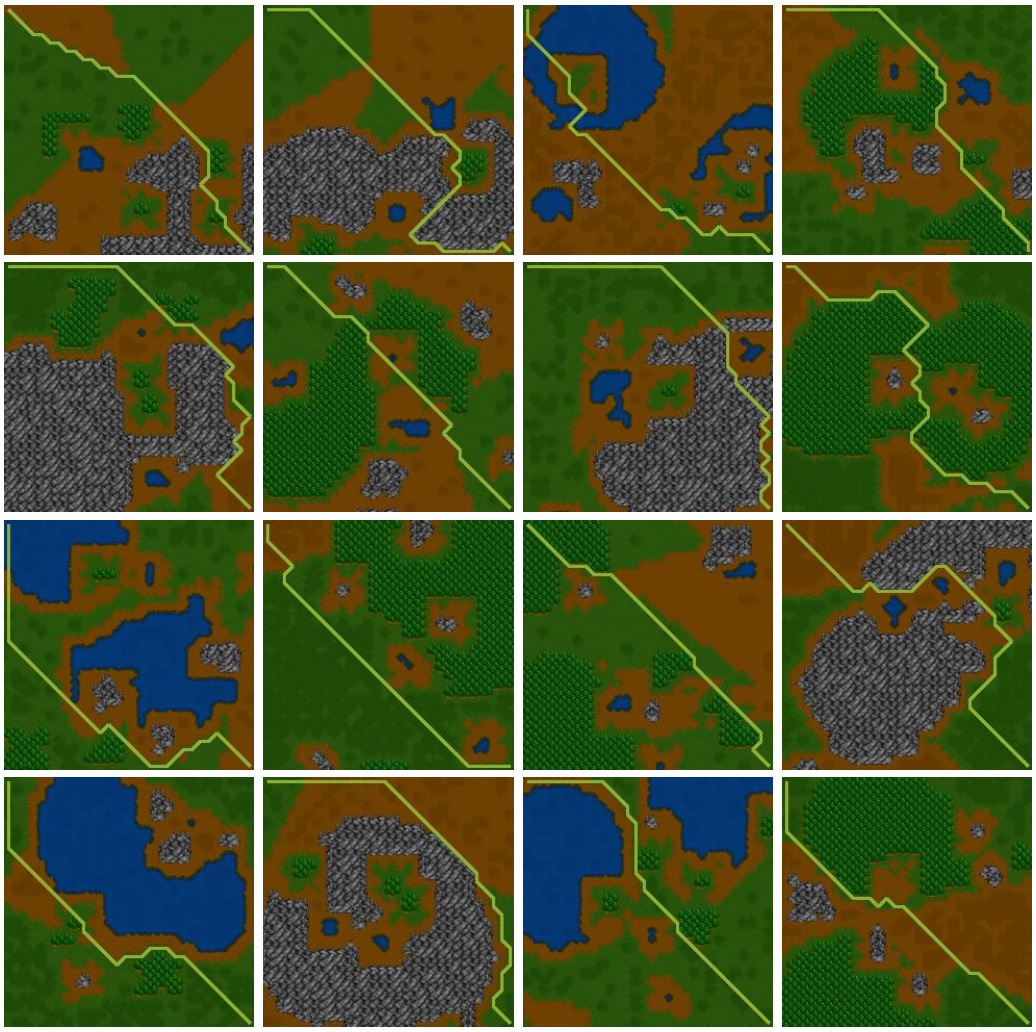

Table 4: Sample of Warcraft maps, and corresponding shortest paths from the upper left to the lower right corner of the map.

Implementations of I-MLE and all experiments will soon be made available. Table 3 lists the results for L2X, SoftSub, and I-MLE for three additional aromas and $k = 10$.

C.3 Discrete Variational Auto-Encoder

Experiments were run on a server with Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz, 4 GeForce GTX 1080 Ti, and 128 GB RAM.

The data set can be loaded in Tensorflow 2.x with *tf.keras.datasets.mnist.load_data()*. As in prior work, we use a batch size of 100 and train for 100 epochs, plotting the test loss after each epoch. We use the standard Adam settings in Tensorflow 2.4.1 (learning rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-07, amsgrad=False), and no learning rate schedule. The MNIST dataset consists in black-and-white 28×28 pixels images of hand-written digits. The encoder network consists of an input layer with dimension 784 (we flatten the images), a dense layer with dimension 512 and ReLu activation, a dense layer with dimension 256 and ReLu activation, and a dense layer with dimension 400 (20×20) which outputs the $\boldsymbol{\theta}$ and no non-linearity. The IMLESubsetkLayer takes $\boldsymbol{\theta}$ as input and outputs a discrete latent code of size 20×20. The decoder network, which takes this discrete latent code as input, consists of a dense layer with dimension 256 and ReLu activation, a dense layer with dimension 512 and ReLu activation, and finally a dense layer with dimension 784 returning the logits for the output pixels. Sigmoids are applied to these logits and the binary cross-entropy

Figure 9: Original MNIST digits from the test set and their reconstructions using the discrete 10-subset VAE trained with Sum-of-Gamma perturbations for $\lambda = 1$ (center) and $\lambda = 10$ (right).

Table 5: Results for the Warcraft shortest path task using I-MLE with two target distributions, namely Eq. (10) and Eq. (14). Reported is the accuracy, i.e. percentage of paths with the optimal costs. Standard deviations are over five runs.

K	$\boldsymbol{\mu}$-$\boldsymbol{\mu}$, Eq. (14)	M-M, Eq. (14)	$\lambda = 20$, Eq. (10)	$\lambda = 20, \tau = 0.01$, Eq. (10)
12	$\mathbf{97.2} \pm 0.5$	95.2 ± 0.3	95.2 ± 0.7	95.1 ± 0.4
18	$\mathbf{95.8} \pm 0.7$	94.4 ± 0.5	94.7 ± 0.4	94.4 ± 0.4
24	$\mathbf{94.3} \pm 1.0$	93.2 ± 0.2	93.8 ± 0.3	93.7 ± 0.4
30	93.6 ± 0.4	93.7 ± 0.6	93.6 ± 0.5	$\mathbf{93.8} \pm 0.3$

loss is computed. The training time (for the 100 epochs) was 21 minutes with the sum-of-Gamma perturbations and 18 minutes for the standard Gumbel perturbations.

C.4 Differentiating through Combinatorial Solvers

The experiments were run on a server with Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz CPUs, 4 NVIDIA Titan RTX GPUs, and 256 GB main memory.

Table 4 shows a set of 30×30 Warcraft maps, and the corresponding shortest paths from the upper left to the lower right corner of the map. In these experiments, we follow the same experimental protocol of Pogančić et al. [2019]: optimisation was carried out via the Adam optimiser, with scheduled learning rate drops dividing the learning rate by 10 at epochs 30 and 40. The initial learning rate was 5×10^{-4}, and the models were trained for 50 epochs using 70 as the batch size. As in [Pogančić et al., 2019], the $K \times K$ weights matrix is produced by a subset of ResNet18 [He et al., 2016], whose weights are trained on the task. For training BB, in all experimental results in Section 6 and Table 4, the hyperparameter λ was set to $\lambda = 20$.

Fig. 11 shows the training dynamics of different models, including the method proposed by Pogančić et al. [2019] (BB) with different choices of the λ hyperparameter, the ResNet18 baseline proposed by Pogančić et al. [2019], and I-MLE.

Furthermore, we experimented with two different target distributions, namely Eq. (10) and Eq. (14), where noise samples were drawn from a sum-of-Gamma distribution. Results are summarised in Table 5. In our experiments, the two target distributions yield very similar results for $\tau = 0.01$, and results tend to degrade for larger values of τ. This is to be expected since the target distribution in Eq. (14) is meaningful in the context described in Section 4, where there are forms of explicit supervision over the discrete states. Code and data for all the experiments described in this paper are available online, at `https://github.com/nec-research/tf-imle`.

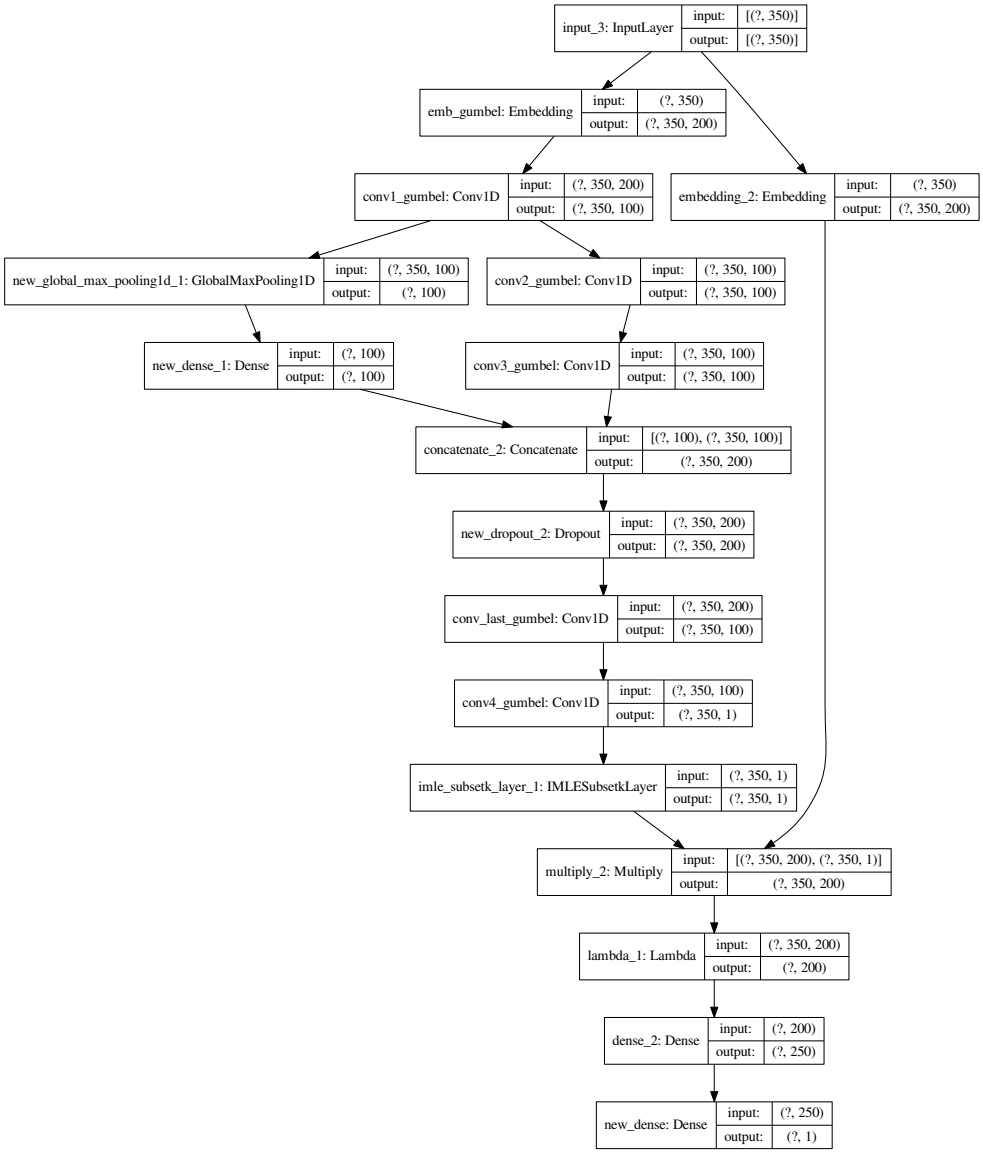

Figure 10: The neural network architecture for the learning to explain experiments. (Please zoom into the vector graphic for more details.) We use the standard architecture and settings from prior work Chen et al. [2018]. The maximum review length is 350 tokens. The dimensions of the token embeddings (of the embedding layers) are 200. All 1D convolutional layers have 250 filters with a kernel size of 3. All dense layers have a dimension of 100. The dropout layer has a dropout rate of 0.2. The layer Multiply perform the multiplication between the token mask (output of I-MLE) and the embedding matrix. The Lambda layer computes the mean of the selected embedding vectors The last dense layer has a sigmoid activation. IMLESubsetkLayer is the layer implementing I-MLE. Code is available in the submission system.

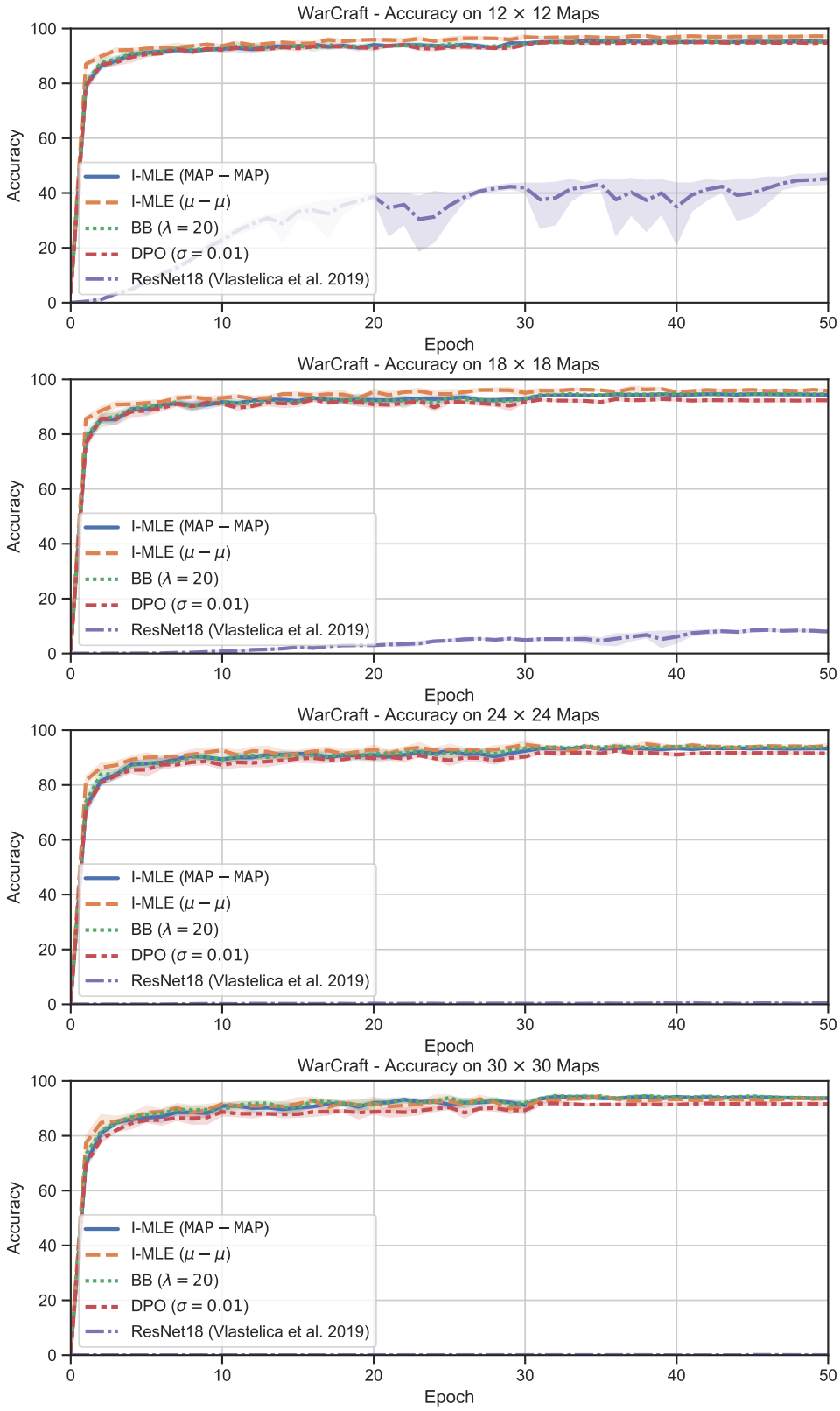

Figure 11: Training dynamics for different models on $K \times K$ shortest path tasks on Warcraft maps, with $K \in \{12, 18, 24, 30\}$.