Reparameterizable Subset Sampling via Continuous Relaxations

Sang Michael Xie and **Stefano Ermon**
Stanford University
{xie, ermon}@cs.stanford.edu

Abstract

Many machine learning tasks require sampling a subset of items from a collection based on a parameterized distribution. The Gumbel-softmax trick can be used to sample a single item, and allows for low-variance reparameterized gradients with respect to the parameters of the underlying distribution. However, stochastic optimization involving subset sampling is typically not reparameterizable. To overcome this limitation, we define a continuous relaxation of subset sampling that provides reparameterization gradients by generalizing the Gumbel-max trick. We use this approach to sample subsets of features in an instance-wise feature selection task for model interpretability, subsets of neighbors to implement a deep stochastic k-nearest neighbors model, and sub-sequences of neighbors to implement parametric t-SNE by directly comparing the identities of local neighbors. We improve performance in all these tasks by incorporating subset sampling in end-to-end training.

1 Introduction

Sampling a single item from a collection is common in machine learning problems such as generative modeling with latent categorical variables, attention mechanisms [Kingma *et al.*, 2014; Xu *et al.*, 2015]. These tasks involve optimizing an expectation objective over a latent categorical distribution parameterized by a neural network. Score-based methods such as REINFORCE [Williams, 1992] for estimating the gradient of such objectives typically have high variance. The reparameterization trick [Kingma and Welling, 2013] allows for low variance gradients for certain distributions, not typically including categorical distributions. The Gumbel-softmax trick [Jang *et al.*, 2016] or Concrete distribution [Maddison *et al.*, 2017] are continuous relaxations that allow for reparameterized gradients with respect to the parameters of the distribution. Among many others, this enabled generative modeling with latent categorical variables without costly marginalization and modeling sequences of discrete elements with GANs [Jang *et al.*, 2016; Kusner and Hernández-Lobato, 2016].

In this paper, we consider the more general problem of sampling a subset of *multiple items* from a collection without replacement. As an example, choosing a subset is important in instance-wise feature selection [Chen *et al.*, 2018], where the goal is to select a subset of features that best explain the model's output for each example. Sampling subsets of neighbors also enables implementing stochastic k-nearest neighbors end-to-end with deep features. Stochastic optimization involving subset sampling, however, does not typically have relaxations with low-variance reparameterization gradients as in Gumbel-softmax. To overcome this limitation, we develop a continuous relaxation for approximate reparameterized gradients with respect to the parameters of a subset distribution to enable learning with backpropagation. In our setting, the Gumbel-max trick (and thus Gumbel-softmax) is not directly applicable since it requires treating every possible subset as a category, requiring a combinatorial number of categories. We use an extension to the Gumbel-max trick which perturbs the log-probabilities of a categorical distribution with Gumbel noise and takes the top-k elements to produce samples without replacement. Ignoring ordering in these samples allows for sampling from a family of subset distributions using the same algorithm. We give a general algorithm that produces continuous relaxations with reparameterization gradients using top-k relaxations. We then show that a recent top-k relaxation [Plötz and Roth, 2018] can be used in our algorithm and study the consistency of this top-k relaxation.

Our main contributions are the following:

- We give an algorithm for a reparameterizable continuous relaxation to sampling subsets using top-k relaxations and a extension to the Gumbel-max trick.
- We show that the top-k relaxation of [Plötz and Roth, 2018] is *consistent* in the sense that the ordering of inputs is preserved in the output in many practical settings.
- We test our algorithm as a drop-in replacement for subset selection routines in explaining deep models through feature subset selection, training stochastic neural k-nearest neighbors, and implementing parametric t-SNE without Student-t distributions by directly comparing neighbor samples. We improve performance on all tasks using the same architectures and metrics as the original [1].

[1] Code available at https://github.com/ermongroup/subsets.

Algorithm 1 Weighted Reservoir Sampling (non-streaming)

```
Input: Items $x_1, \dots, x_n$, weights $\mathbf{w} = [w_1, \dots, w_n]$, reservoir size $k$
Output: $S_{wrs} = [\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}]$ a sample from $p(S_{wrs}|\mathbf{w})$
$\mathbf{r} \leftarrow [\,]$
for $i \leftarrow 1$ to $n$ do
    $u_i \leftarrow$ Uniform(0, 1)
    $r_i \leftarrow u_i^{1/w_i}$                # Sample random keys
    $\mathbf{r}$.append($r_i$)
end for
$[\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}] \leftarrow$ TopK($\mathbf{r}$, $k$)
return $[\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}]$
```

2 Preliminaries

2.1 Weighted Reservoir Sampling

Reservoir sampling is a family of streaming algorithms that is used to sample k items from a collection of n items, $x_1, \dots, x_n$, where n may be infinite [Vitter, 1985]. We consider finite n and only produce samples after processing the entire stream. In weighted reservoir sampling, every x_i is associated with a weight $w_i \geq 0$. Let $\mathbf{w} = [w_1, \dots, w_n]$ and $Z = \sum_{i=1}^n w_i$ be the normalizing constant. Let $\mathbf{e}^j = [e_1^j, \dots, e_n^j] = [0, \cdots, 0, 1, 0, \cdots, 0] \in \{0,1\}^n$ be a 1-hot vector, i.e., a vector with only one nonzero element at index j, where $e_j^j = 1$. We define a weighted reservoir sample (WRS) as $S_{wrs} = [\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}]$, a sequence of k 1-hot (standard basis) vectors where $\mathbf{e}^{i_j}$ represents selecting element x_{i_j} in the j-th sample. We wish to sample S_{wrs} from

$$p(S_{wrs} \mid \mathbf{w}) = \frac{w_{i_1}}{Z} \frac{w_{i_2}}{Z - w_{i_1}} \cdots \frac{w_{i_k}}{Z - \sum_{j=1}^{k-1} w_{i_j}}, \quad (1)$$

which corresponds to sampling without replacement with probabilities proportional to item weights. Modeling samples without replacement allows for sampling a sequence of distinct items. For $k = 1$, $p(S_{wrs}|\mathbf{w})$ is the standard softmax distribution with logits given by $\log(w_i)$.

[Efraimidis and Spirakis, 2006] give an algorithm for weighted reservoir sampling (Algorithm 1). Each item x_i is given a random key $r_i = u_i^{1/w_i}$ where u_i is drawn from a uniform distribution between $[0, 1]$ and w_i is the weight of item x_i. Let the top k keys over the n items be $r_{i_1}, \dots, r_{i_k}$. We define the function TopK$(\mathbf{r}, k)$ which takes keys $\mathbf{r} = [r_1, \dots, r_n]$ and returns $[\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}]$ associated with the top-k keys. The algorithm uses TopK to return the items $S_{wrs} = [\mathbf{e}^{i_1}, \dots, \mathbf{e}^{i_k}]$ as the WRS. Efraimidis and Spirakis proved (Proposition 5 in [Efraimidis and Spirakis, 2006]) that the output of Algorithm 1 is distributed according to $p(S_{wrs}|\mathbf{w})$.

2.2 Gumbel-max Trick

Given $\mathbf{w}$ as in (1), $\log(w_i)$ are logits for a softmax distribution $p(x_i|\mathbf{w}) = w_i/Z$. The Gumbel-max trick [Yellott, 1977] generates random keys $\hat{r}_i = \log(w_i) + g_i$ by perturbing logits with Gumbel noise $g_i \sim$ Gumbel$(0, 1)$, then taking x_{i^*} such that $i^* = \arg\max_i \hat{r}_i$ as a sample. These samples are distributed according to $p(x_i|\mathbf{w}) = w_i/Z$. The idea is to reparameterize the sample as a deterministic transformation of the parameters $\mathbf{w}$ and some independent noise g_i. Then by relaxing the deterministic transformation (from *max* to *softmax*), the Gumbel-softmax trick allows for training with backpropagation [Maddison *et al.*, 2017; Jang *et al.*, 2016]. Similarly, we use an extension of the Gumbel-max trick to decouple the deterministic transformation of the parameters (in our case, a top-k selection function) and the randomness (Gumbel noise g_i), and we relax the top-k function to allow for backpropagation.

3 Reparameterizable Continuous Relaxation for Subset Sampling

3.1 Setup

We represent a subset $S \in \{0,1\}^n$ as a k-hot vector, which is a vector with exactly k nonzero elements that are all equal to 1. We define the probability of a subset S as the sum of the probabilities of all WRS with the same elements

$$p(S \mid \mathbf{w}) = \sum_{S_{wrs} \in \Pi(S)} p(S_{wrs}|\mathbf{w}) \quad (2)$$

where $\Pi(S) = \{S_{wrs} : S = \sum_{j=1}^k S_{wrs}[j]\}$ is the set of all permutations of elements in S represented by sequences of 1-hot vectors. Here, $S_{wrs}[j]$ is the j-th 1-hot vector in the sequence. By simply ignoring the order of elements in a WRS, we can also sample from $p(S|\mathbf{w})$ using the same algorithm. Note that this is a restricted family of subset distributions. Since each distribution is over $\binom{n}{k}$ subsets of size k, the full space of distributions requires $\binom{n}{k} - 1$ free parameters. Here, we reduce the number of free parameters to $n - 1$. While this is a restriction, we gain tractability in our algorithm.

3.2 Gumbel-max Extension

We extend the Gumbel-max trick to sample from $p(S|\mathbf{w})$. The main intuition is that the outputs of the reservoir sampling Algorithm 1 only depend on the *ordering* of the random keys r_i and not their values. We show that random keys $\hat{r}_i$ generated from the Gumbel-max trick are monotonic transformations of the random keys r_i from Algorithm 1. Because a monotonic transformation preserves the ordering, the elements that achieve the top-k Gumbel-max keys $\hat{r}_i$ have the same distribution as the elements that achieve the top-k weighted reservoir sampling keys r_i. Therefore we can sample from $p(S|\mathbf{w})$ by taking the top-k elements of $\hat{r}_i$ instead.

To make the procedure differentiable with respect to $\mathbf{w}$, we replace top-k selection with a differentiable approximation. We define a relaxed k-hot vector $\mathbf{a} = [a_1, \dots, a_n]$ to have $\sum_{i=1}^n a_i = k$ and $0 \leq a_i \leq 1$. We relax S_{wrs} by replacing all $\mathbf{e}^{i_j}$ with relaxed 1-hot vectors and relax S by a relaxed k-hot vector. Our continuous relaxation will approximate sampling from $p(S|\mathbf{w})$ by returning relaxed k-hot vectors. We use a top-k relaxation RelaxedTopK, which is a *differentiable* function that takes $\hat{r}_i$, k, and a temperature parameter $t > 0$ and returns a relaxed k-hot vector $\mathbf{a}$ such that as $t \to 0$, RelaxedTopK$(\hat{\mathbf{r}}, k, t) \to \sum_{j=1}^k$ TopK$(\hat{\mathbf{r}}, k)[j]$, where TopK$(\hat{\mathbf{r}}, k)[j] = \mathbf{e}^{i_j}$ is the 1-hot vector associated with

Algorithm 2 Relaxed Subset Sampling

```
Input: Items $x_1, \dots, x_n$, weights $\mathbf{w} = [w_1, \dots, w_n]$, subset size $k$, temperature $t > 0$
Output: Relaxed $k$-hot vector $\mathbf{a} = [a_1, \dots, a_n]$, where $\sum_{i=1}^n a_i = k, 0 \leq a_i \leq 1$
$\hat{\mathbf{r}} \leftarrow [\,]$
for $i \leftarrow 1$ to $n$ do
    $u_i \leftarrow$ Uniform(0, 1)            # Random Gumbel keys
    $\hat{r}_i \leftarrow -\log(-\log(u_i)) + \log(w_i)$
    $\hat{\mathbf{r}}$.append($\hat{r}_i$)
end for
$\mathbf{a} \leftarrow$ RelaxedTopK($\hat{\mathbf{r}}$, $k$, $t$)
return $\mathbf{a}$
```

the j-th top key in $\hat{\mathbf{r}}$. Thus Algorithm 2 produces relaxed k-hot samples that, as $t \to 0$ converge to exact samples from $p(S|\mathbf{w})$. Note that we can also produce approximate samples from $p(S_{wrs}|\mathbf{w})$ if an intermediate output of RelaxedTopK is a sequence of k relaxed 1-hot vectors $[\mathbf{a}^{i_1}, \dots, \mathbf{a}^{i_k}]$ such that as $t \to 0$, $[\mathbf{a}^{i_1}, \dots, \mathbf{a}^{i_k}] \to \text{TopK}(\hat{\mathbf{r}}, k)$. This means that the intermediate output converges to a WRS.

Proposition 1. *Let RelaxedTopK be defined as above. Given n items $x_1, \dots, x_n$, a subset size k, and a distribution over subsets described by weights $w_1, \dots, w_n$, Algorithm 2 gives exact samples from $p(S|\mathbf{w})$ as in (2) as $t \to 0$.*

Proof. Let the random keys in Algorithm 2 be $\hat{\mathbf{r}}$ and the random keys in weighted reservoir sampling be $\mathbf{r}$. For any i,

$$\begin{aligned}\hat{r}_i &= -\log(-\log(u_i)) + \log(w_i)\\ &= -\log(-\frac{1}{w_i}\log(u_i))\\ &= -\log(-\log(u_i^{1/w_i})) = -\log(-\log(r_i)).\end{aligned}$$

Fixing u_i, since $-\log(-\log(a))$ is monotonic in a, $\hat{r}_i$ is a monotonic transformation of r_i and $\text{TopK}(\hat{\mathbf{r}}, k) = \text{TopK}(\mathbf{r}, k)$. Let $\text{TopK}(\mathbf{r}, k)$ be samples from Algorithm 1. By construction of (2), $\sum_{j=1}^k \text{TopK}(\mathbf{r}, k)[j]$ is distributed as $p(S|\mathbf{w})$. As $t \to 0$, Algorithm 2 produces samples from $p(S|\mathbf{w})$ since $\text{RelaxedTopK}(\hat{\mathbf{r}}, k, t) \to \sum_{j=1}^k \text{TopK}(\hat{\mathbf{r}}, k)[j] = \sum_{j=1}^k \text{TopK}(\mathbf{r}, k)[j]$. □

This fact has been shown previously in [Vieira, 2014] for $k = 1$ and in [Kim *et al.*, 2016] for sampling from $p(S_{wrs}|\mathbf{w})$ without the connection to reservoir sampling. Kool *et al.* concurrently developed a similar method for ordered sampling without replacement for stochastic beam search. Note that Algorithm 2 is general to any choice of top-k relaxation.

3.3 Differentiable Top-k Procedures

A vital component of Algorithm 2 is a top-k relaxation that is *differentiable* with respect to the input keys $\hat{r}_i$ (a random function of $\mathbf{w}$). This allows for parameterizing $\mathbf{w}$, which governs $p(S|\mathbf{w})$, using neural networks and training using backpropagation. We propose to use a recent top-k relaxation based on successive applications of the softmax function [Plötz and Roth, 2018]. For some temperature $t > 0$, define for all $i = 1, \dots, n$

$$\alpha_i^1 := \log(\hat{r}_i), \;\; \alpha_i^{j+1} := \alpha_i^j + \log(1 - a_i^j) \tag{3}$$

where a_i^j is a sample at step j from the distribution

$$p(a_i^j = 1) = \frac{\exp(\alpha_i^j/t)}{\sum_{m=1}^n \exp(\alpha_i^j/t)} \tag{4}$$

for $j = 1, \dots, k$ steps. In the relaxation, the a_i^j is replaced with its expectation, $p(a_i^j = 1)$, such that the new update is

$$\alpha_i^{j+1} := \alpha_i^j + \log(1 - p(a_i^j = 1)) \tag{5}$$

and the output is $a_i = \sum_{j=1}^k p(a_i^j = 1)$ for each i. Let $\mathbf{a}^j = [p(a_1^j = 1), \dots, p(a_n^j = 1)]$ be relaxed 1-hot outputs at step j and $\mathbf{a} = \sum_{j=1}^k \mathbf{a}^j$ be the relaxed k-hot output. Plötz and Roth show that as $t \to 0$, $[\mathbf{a}^1, \dots, \mathbf{a}^k] \to \text{TopK}(\hat{\mathbf{r}}, k)$ and thus $\mathbf{a} \to \sum_{j=1}^k \text{TopK}(\hat{\mathbf{r}}, k)[j]$ so that this is a valid RelaxedTopK. Thus the relaxation can be used for approximately sampling from both $p(S|\mathbf{w})$ and $p(S_{wrs}|\mathbf{w})$. Next we show that the magnitude of values in the output relaxed k-hot vector $\mathbf{a}$ preserves order of input keys $\hat{r}_i$ for $t \geq 1$.

Theorem 1. *Given keys $\hat{\mathbf{r}}$, the top-k relaxation of [Plötz and Roth, 2018] produces a relaxed k-hot vector $\mathbf{a}$ where if $\hat{r}_i \leq \hat{r}_j$, then $a_i \leq a_j$ for any temperature $t \geq 1$ and $k \leq n$.*

Proof. By induction on k. Fix any $\hat{r}_i \leq \hat{r}_j$. For step $k = 1$, we have $a_i^1 \leq a_j^1$ since the softmax function preserves ordering. Assuming the statement holds for $0, \dots, k$, we want to show that $\alpha_i^{k+1} \leq \alpha_j^{k+1}$, which suffices to imply $a_i^{k+1} \leq a_j^{k+1}$ by the order-preserving property of softmax. Define $\hat{\alpha}_i^k = \frac{\exp(\alpha_i^k/t)}{\sum_{m=1}^n \exp(\alpha_i^k/t)} = p(a_i^j = 1)$. Then

$$\begin{aligned}\alpha_i^{k+1} &= \alpha_i^k + \log\left(1 - \hat{\alpha}_i^k\right)\\ \exp(\alpha_i^{k+1}) &= \exp(\alpha_i^k)\left(1 - \hat{\alpha}_i^k\right).\end{aligned}$$

Comparing α_j^{k+1} and α_i^{k+1} through the ratio,

$$\frac{\exp(\alpha_j^{k+1})}{\exp(\alpha_i^{k+1})} = \frac{\exp(\alpha_j^k)}{\exp(\alpha_i^k)}\left(\frac{\exp(\alpha_i^k/t) + c}{\exp(\alpha_j^k/t) + c}\right)$$

where we can view $c = \sum_{m=1}^n \exp(\alpha_m^k/t) - \exp(\alpha_i^k/t) - \exp(\alpha_j^k/t) \geq 0$ as a non-negative constant in this analysis. Note that $\frac{\exp(\alpha_j^k)}{\exp(\alpha_i^k)} = \exp(\alpha_j^k - \alpha_i^k) \geq \exp((\alpha_j^k - \alpha_i^k)/t) = \frac{\exp(\alpha_j^k/t)}{\exp(\alpha_i^k/t)}$ for $t \geq 1$. Therefore

$$\frac{\exp(\alpha_j^{k+1})}{\exp(\alpha_i^{k+1})} \geq \frac{\exp(\alpha_j^k/t)}{\exp(\alpha_i^k/t)}\left(\frac{\exp(\alpha_i^k/t) + c}{\exp(\alpha_j^k/t) + c}\right) \geq 1$$

for any $c \geq 0$. Thus $\alpha_i^{k+1} \leq \alpha_j^{k+1}$, implying $a_i^{k+1} \leq a_j^{k+1}$. □

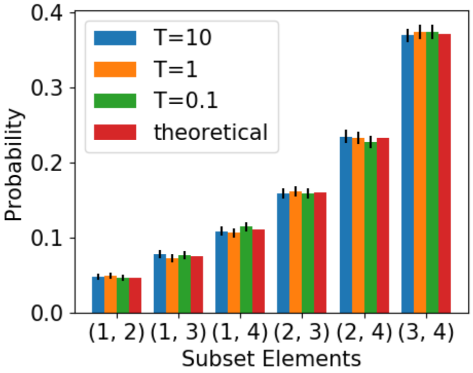


Figure 1: Empirical subset distributions on a toy subset distribution.

Therefore, this top-k relaxation is *consistent* in the sense that the indices with the top k weights in w are also the top k values in the relaxed k-hot vector for many reasonable t. In the limit as $t \to 0$ the consistency property holds for the exact k-hot vector, but the relaxation is not consistent in general for $0 < t < 1$. To see a concrete example of the loss of consistency at lower t, take $\hat{\mathbf{r}} = [e, e^2]$, $k = 2$, and $t = 0.4$. The output on this input is $[1.05, 0.95]$, which has inconsistent ordering. Consistency at higher temperatures is important when considering the bias-variance tradeoff in the gradients produced by the approximation. While higher temperatures allow for lower variance gradients, the bias introduced may cause the model to optimize something far from the true objective. Consistency in ordering suggests that the top-k information is not lost at higher temperatures.

Note that any top-k relaxation can be used in Algorithm 2. For example, [Grover *et al.*, 2019] give a relaxation of the sorting procedure through a relaxed permutation matrix P. To produce a relaxed k-hot vector in this framework, we can take $\mathbf{a} = \sum_{j=1}^{k} P_j$ where P_j is the j-th row of P. Similarly, when the associated temperature parameter approaches 0, P approaches the exact permutation matrix so that the relaxed k-hot vector $\mathbf{a}$ also approaches the exact top-k k-hot vector.

4 Experiments

4.1 Synthetic Experiments

We check that generating samples from Algorithm 2 results in samples approximately from $p(S|\mathbf{w})$. We define a subset distribution using weights $\mathbf{w} = [0.1, 0.2, 0.3, 0.4]$ and take subset size $k = 2$. Using Algorithm 2 and the top-k relaxation from [Plötz and Roth, 2018], we sample 10000 relaxed k-hot samples for each temperature in $\{0.1, 1, 10\}$ and take the the top-k values in the relaxed k-hot vector as the "chosen" subset. We plot the empirical histogram of subset occurrences and compare with the true probabilities from the subset distribution, with 95% confidence intervals. The relaxation produces subset samples with empirical distribution within 0.016 in total variation distance of $p(S|\mathbf{w})$ for all t. This agrees with Theorem 1, which states that even for higher temperatures, taking the top-k values in the relaxed k-hot vector should produce true samples from (2).

i just saw this at the - film festival it was the most wonderful movie the best i've seen in quite a while the - character of is an open - and as - in the film love filled person portrayed by the beautiful - the - music is - and - a - to the soul each character seems to be - in some way and their - w conflicts make for a great story the tale told by a k - as the old man was most - done i wanted it to go on and on that - seemed to remember his place throughout added get power to the story a refreshing change to the - headed plots of many modern writers all and all an excellent film go see

Figure 2: An example of an IMDB review where our explainer model predicts the $k = 10$ words and the sentiment model outputs a similar prediction (positive) when only taking the 10 words as input. Words not in the vocabulary are replaced with dashes.

read the book forget the movie

Figure 3: A review where the L2X explainer model [Chen *et al.*, 2018] can select 10 words (whole review) but subsamples, reducing the sentiment model's confidence from 0.93 to 0.52 when taking the subset of words as input. Our explainer returns the whole review.

4.2 Model Explanations

We follow the L2X model [Chen *et al.*, 2018] and set up the problem of explaining instance-wise model outputs by training an auxiliary *explainer* model to output the k features with the highest mutual information with the model output. For example, given a movie review and a sentiment model, we want to select up to k words from the review that best explains the sentiment model's output on the particular review (see Figure 2). Since optimizing the mutual information is intractable, L2X optimizes a variational lower bound instead:

$$\max_{\mathcal{E},q} \mathbb{E}[\log q(X_S)] \quad \text{s.t. } S \sim \mathcal{E}(X) \tag{6}$$

where $\mathcal{E} : \mathbb{R}^d \to \mathcal{P}_k$ is an explainer model, parameterized as a neural network, mapping from an input to the space of all k-hot vectors S. The approximating distribution is $q(X_S)$ where q is a neural network and $X_S = S \odot X \in \mathbb{R}^d$ is X with the elements not corresponding to S zeroed out. L2X approximates the subset sampling procedure by sampling k independent times from a Concrete distribution [Maddison *et al.*, 2017] and taking the elementwise maximum. Since each independent Concrete sample is a relaxed 1-hot vector, the L2X explainer model may suffer by independently sampling the same feature many times, resulting in selecting less than k features. Our model differs by replacing this sampling procedure with Algorithm 2. Figure 3 shows an instance where the L2X explainer model chooses an ambiguous subset of the review even when the review is shorter than the maximum number of words to be selected (k), which reduces the sentiment model's confidence significantly (0.93 to 0.52). Our model selects the entire review in this case, so the sentiment model's output is not affected.

We test our results on the Large Movie Review Dataset (IMDB) for sentiment classification [Maas *et al.*, 2011], where we select the most important words or sentences that contribute to the sentiment prediction for the review.

Model	IMDB-word	IMDB-sent
L2X	90.7 ± 0.004	82.9 ± 0.005
RelaxSubSample	$\mathbf{91.7 \pm 0.003}$	$\mathbf{83.2 \pm 0.004}$

Table 1: Post-hoc accuracy (%, 95% interval) on explaining sentiment predictions on the IMDB Large Movie Review Dataset. L2X refers to the model from [Chen *et al.*, 2018] while RelaxSubSample is our method.

Model	MNIST	Fashion-MNIST	CIFAR-10
Stochastic NeuralSort	**99.4**	93.4	89.5
RelaxSubSample	99.3	**93.6**	**90.1**
CNN (no kNN)	99.4	93.4	95.1

Table 2: Classification test accuracy (%) of deep stochastic k-nearest neighbors using the NeuralSort relaxation [Grover *et al.*, 2019] and our method (RelaxSubSample).

The original model for word-based sentiment classification (IMDB-word) is a convolutional neural network [Kim, 2014], while the original model for sentence-based sentiment prediction is a hierarchical LSTM (IMDB-sent) [Li *et al.*, 2015]. The explainer and variational distribution models are CNNs with the same architectures as in L2X [Chen *et al.*, 2018]. Following L2X, we use $k = 10$ for IMDB-word and $k = 1$ sentences for IMDB-sent. At test time, all explainer models deterministically choose subsets based on the highest weights instead of sampling. We evaluate using *post-hoc accuracy*, which is the proportion of examples where the original model evaluated on masked features X_S matches the model on unmasked X. We use cross validation to choose temperatures $t \in \{0.1, 0.5, 1, 2, 5\}$ according to the validation loss. Our model (RelaxSubSample) improves upon L2X by up to 1% by only changing the sampling procedure (Table 1).

4.3 Stochastic K-Nearest Neighbors

We give a stochastic k-NN algorithm where we use deep features tuned end-to-end for computing neighbors. We follow the setup of Grover *et al.*, using $m = 100$ randomly sampled neighbor candidates and the same loss. We defer details, including architecture, to Grover *et al.*.

We compare to NeuralSort, which implements kNN using a relaxation of the sorting operator [Grover *et al.*, 2019]. We fix $k = 9$ nearest neighbors to choose from m candidates and search over temperatures $t = \{0.1, 1, 5, 16, 64\}$ using the validation set, whereas NeuralSort searches over both k and t. RelaxSubSample approaches the accuracy of a CNN trained using the standard cross entropy loss on MNIST (99.3% vs. 99.4%) and increases accuracy by 0.6% over the NeuralSort implementation on CIFAR-10 (Table 2).

Model	$m = 100$	$m = 1000$	$m = 5000$
NeuralSort	**0.010s**	0.073s	3.694s
RelaxSubSample	**0.010s**	**0.028s**	**0.110s**

Table 3: Forward pass average runtimes (100 trials) for a small CNN selecting $k = 5$ neighbors from m candidates using the NeuralSort relaxation [Grover *et al.*, 2019] and our method (RelaxSubSample). Results were obtained from a Titan Xp GPU.

Model	d	MNIST	20 Newsgroups
Par. t-SNE ($\alpha = 1$)	2	0.926	0.720
RSS-SNE (no pretrain)	2	**0.929**	**0.763**
RSS-SNE (pretrain)	2	**0.947**	**0.764**
Par. t-SNE ($\alpha = 1$)	10	0.983	0.854
RSS-SNE (no pretrain)	10	**0.998**	**0.912**
RSS-SNE (pretrain)	10	**0.999**	**0.905**
Par. t-SNE ($\alpha = 1$)	30	0.983	0.866
RSS-SNE (no pretrain)	30	**0.999**	**0.929**
RSS-SNE (pretrain)	30	**0.999**	**0.965**

Table 4: Trustworthiness(12) of low dimensional embeddings of size $d \in \{2, 10, 30\}$ for parametric t-SNE (Par. t-SNE) and RelaxSubSample SNE (RSS-SNE) on MNIST and 20 Newsgroups. Pretrain refers to layer-wise pretraining using autoencoders.

Model	d	MNIST	20 Newsgroups
Par. t-SNE ($\alpha = 1$)	2	9.90	34.30
RSS-SNE (no pretrain)	2	11.80	36.80
RSS-SNE (pretrain)	2	**8.31**	35.11
Par. t-SNE ($\alpha = 1$)	10	5.38	24.40
RSS-SNE (no pretrain)	10	**4.97**	29.39
RSS-SNE (pretrain)	10	**4.56**	28.50
Par. t-SNE ($\alpha = 1$)	30	5.41	24.88
RSS-SNE (no pretrain)	30	**3.51**	29.39
RSS-SNE (pretrain)	30	**3.05**	28.90

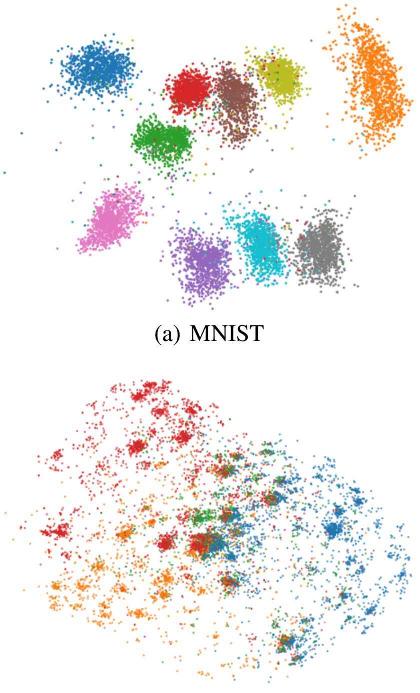
Table 5: Test errors (%) of 1-NN classifiers trained on low dimensional embeddings of size $d \in \{2, 10, 30\}$ generated by parametric t-SNE (Par. t-SNE) and our model (RSS-SNE).

Note that NeuralSort implements a relaxation to the sorting procedure, while in kNN we only require the top-k elements. We use the top-k relaxation from [Plötz and Roth, 2018], which computes k softmaxes for a runtime and storage of $O(km)$. NeuralSort requires $O(m^2)$ time and storage as it produces a $m \times m$ permutation matrix for each input. Table 3 shows forward pass runtimes of a CNN using our method and NeuralSort for different values of m and $k = 5$ neighbors. While runtimes for small m are comparable, our method scales much better for larger m (Table 3).

4.4 Stochastic Neighbor Embeddings

We consider the problem of learning a parametric embedding which maps a high-dimensional space into a lower-dimensional space while preserving local neighbor structure. This problem is addressed by parametric t-SNE [van der Maaten, 2009], which represents pairwise densities of datapoints in the high-dimensional space as symmetrized Gaussians with variances tuned so that the perplexity of each point is equal. The pairwise densities in the low-dimensional space are modeled by Student-t distributions to address the *crowding problem*, a result of the volume shrinkage from the high to low-dimensional spaces. Student-t distributions have heavier tails, allowing for distant points in the high-dimensional space to be modeled as far apart in the low-dimensional space. The objective is to minimize the KL divergence between the pairwise distributions in the two spaces.

We propose to learn such a mapping without the use of Student-t distributions. Our insight is that the goal of preserving local structure can be achieved by preserving the neighbor *rankings*, which is insensitive to scaling of distances. In our RelaxSubSample-based stochastic neighbor embed-



(a) MNIST

(b) 20 Newsgroups

Figure 4: 2D embeddings generated by our model (with pretrain) on 10000 MNIST datapoints and 15000 20 Newsgroups datapoints. Colors represent different classes. (Best in color.)

ding (RSS-SNE), we aim to preserve the distribution of k neighbors around each point. We define the neighbor distributions as follows. Let $X = \{x_1, \ldots, x_n\}$ be the training data. Let $w(i,j) = \exp(-\|x_i - x_j\|_2^2)$ be exponentiated negative pairwise squared distances. For datapoint x_i, let $\mathbf{w}(i) \in \mathbb{R}^{n-1}$ be the pairwise distances from x_i to other points. Then we model the neighbor distribution for x_i as $p(S_{wrs}|\mathbf{w}(i))$ as in (1). Note that we sample sub-sequences because we want to preserve neighbor rankings. Let h be our parametric embedding function. Similarly, letting $\hat{w}(i,j) = \exp(-\|h(x_i) - h(x_j)\|_2^2)$ and the pairwise distances involving $h(x_i)$ be $\hat{\mathbf{w}}(i)$, the neighbor distribution in the low dimensional space is $p(S_{wrs}|\hat{\mathbf{w}}(i))$. We aim to match these distributions by comparing the neighbor *samples* to avoid the crowding problem. For each x_i, let a sample from $p(S_{wrs}|\mathbf{w}(i))$ be $[\mathbf{e}^{i_1}, \ldots, \mathbf{e}^{i_k}]$ where $\mathbf{e}^{i_j}$ are 1-hot vectors corresponding to a selected neighbor x_{i_j}. Let a relaxed sample from $p(S_{wrs}|\hat{\mathbf{w}}(i))$ be $[\mathbf{a}^{i_1}, \ldots, \mathbf{a}^{i_k}]$, k intermediate relaxed 1-hot outputs of the top-k relaxation from [Plötz and Roth, 2018]. We minimize the objective

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{1}{e^{j-1}} < \mathbf{e}^{i_j}, -\log(\mathbf{a}^{i_j}) > \qquad (7)$$

where we aim to match samples from the neighbor distributions in the two spaces, putting more weight on matching closer neighbors. While we can directly match the neighbor distributions $p(S_{wrs}|\mathbf{w}(\mathbf{i}))$ and $p(S_{wrs}|\hat{\mathbf{w}}(i))$ without sampling, they are defined in terms of pairwise distances that cannot be matched due to the crowding problem. We find that sampling from both distributions is necessary to keep the loss agnostic to scaling of distances in the different spaces.

We compare with parametric t-SNE [van der Maaten, 2009] on the MNIST [LeCun and Cortes, 2010] and a small version of the 20 Newsgroups dataset [Roweis, 2009][2]. Following van der Maaten, we compare the trustworthiness and performance of 1-NN classifiers of the low dimensional embeddings on the *test* set. Trustworthiness [Venna and Kaski, 2006] measures preservation of local structure in the low dimensional embedding and is defined as

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)}\sum_{i=1}^{n}\sum_{j\in N_i^{(k)}}\max(r(i,j)-k, 0)$$

where $r(i,j)$ is the rank of datapoint j according to distances between datapoint i and other datapoints in the high-dimensional space, and $N_i^{(k)}$ is the set of k nearest neighbors in the low dimensional space. Trustworthiness decreases when a datapoint is in the k nearest neighbors in the low dimensional space but not the original space. As in van der Maaten, we use $T(12)$, comparing 12 nearest neighbors.

We use the same feedforward networks as in van der Maaten as embedding functions. For all experiments, we set $t = 0.1$ and train for 200 epochs with a batch size of 1000, choosing the model with the best training loss. We sample neighbors only within each training batch. We find that $k = 1$ is sufficient to learn the local structure. For $k > 1$, we observe a trade-off where 1-NN accuracy decreases but trustworthiness either stays the same or increases. It was important to add a small bias ($1\mathrm{e}-8$) to the relaxed k-hot vectors for better optimization. We compare two versions of RSS-SNE, one trained from scratch and another using layerwise pretraining. We pretrain layer l by treating layers $1, \ldots, l$ as an encoder and adding a 1 layer decoder to the original space; then we optimize the MSE autoencoder objective for 10 epochs. Note that the original parametric t-SNE used a similar layerwise pretraining scheme using RBMs. RSS-SNE models consistently have higher trustworthiness and competititive 1-NN test errors when compared to parametric t-SNE (Tables 4, 5). Since trustworthiness compares 12 nearest neighbors, this suggests that our embedding has better overall structure as opposed to focusing on the immediate neighbor.

5 Conclusion

We present an algorithm for relaxing samples from a distribution over subsets such that the procedure can be included in deep models trained with backpropagation. We use the algorithm as a drop-in replacement in tasks requiring subset sampling to boost performance. Our algorithm has the potential to improve any task requiring subset sampling by tuning the model end-to-end with the subset procedure in mind.

Acknowledgements

This research was supported by NSF (#1651565, #1733686), ONR, AFOSR (FA9550- 19-1-0024), and Bloomberg. We are thankful to Aditya Grover for helpful comments.

[2] Data at https://cs.nyu.edu/~roweis/data/20news_w100.mat

References

[Chen *et al.*, 2018] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 883–892, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[Efraimidis and Spirakis, 2006] Pavlos S. Efraimidis and Paul G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181 – 185, 2006.

[Grover *et al.*, 2019] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019.

[Jang *et al.*, 2016] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144, 2016.

[Kim *et al.*, 2016] Carolyn Kim, Ashish Sabharwal, and Stefano Ermon. Exact sampling with integer linear programs and random perturbations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3248–3254. AAAI Press, 2016.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. cite arxiv:1312.6114.

[Kingma *et al.*, 2014] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

[Kool *et al.*, 2019] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3499–3508, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[Kusner and Hernández-Lobato, 2016] Matt J. Kusner and José Miguel Hernández-Lobato. GANS for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051, 2016.

[LeCun and Cortes, 2010] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[Li *et al.*, 2015] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL (1)*, pages 1106–1115. The Association for Computer Linguistics, 2015.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Maddison *et al.*, 2017] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

[Plötz and Roth, 2018] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1095–1106. Curran Associates, Inc., 2018.

[Roweis, 2009] Sam Roweis. Data for matlab hackers, 2009.

[van der Maaten, 2009] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 384–391, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[Venna and Kaski, 2006] Jarkko Venna and Samuel Kaski. Visualizing gene interaction graphs with local multidimensional scaling. 2006.

[Vieira, 2014] Tim Vieira. Gumbel-max trick and weighted reservoir sampling, 2014.

[Vitter, 1985] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org, 2015.

[Yellott, 1977] John I. Yellott. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2):109 – 144, 1977.