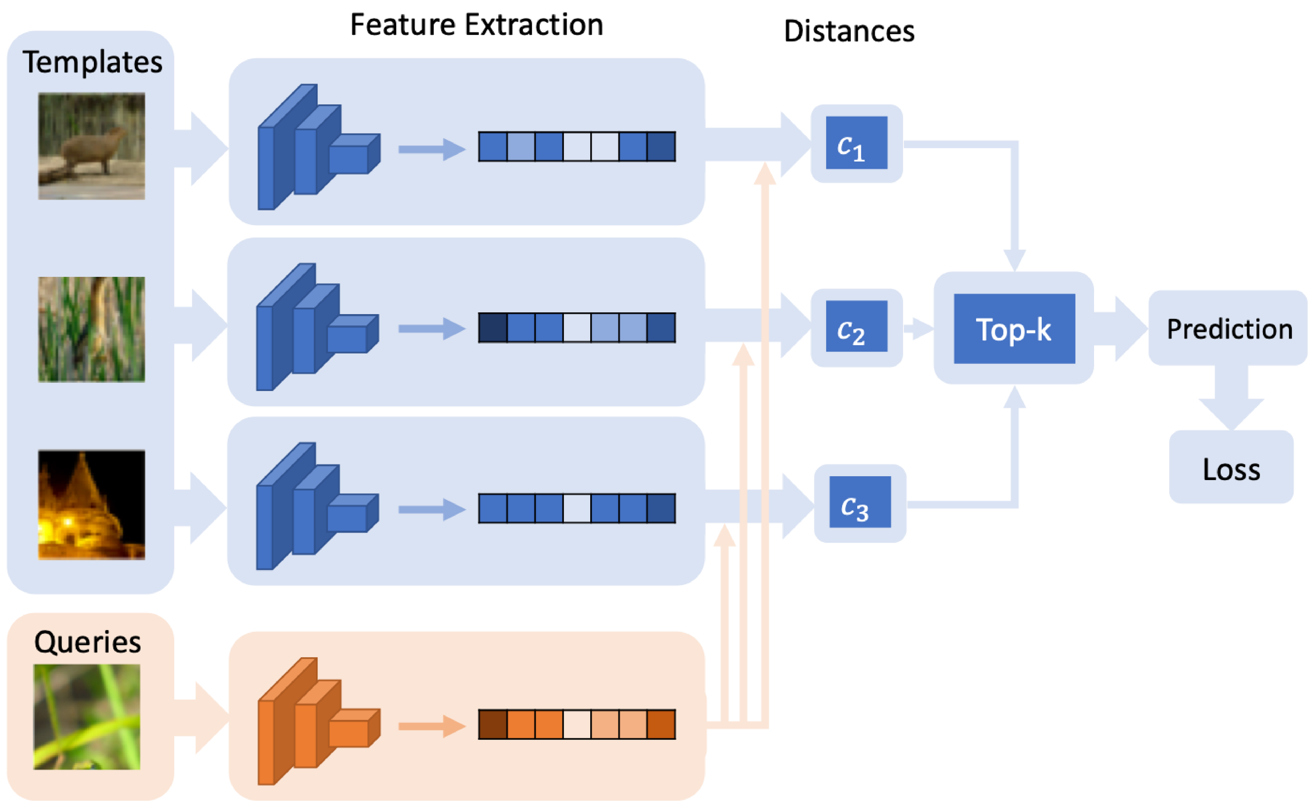
Differentiable Top-k with Optimal Transport

Yujia Xie[1], Hanjun Dai[2], Minshuo Chen[1], Bo Dai[2], Tuo Zhao[1], Hongyuan Zha[1], Wei Wei[3], Tomas Pfister[3]

[1]Georgia Institute of Technology [2]Google Brian [3]Google Cloud AI Research

Motivating Example 1 – Deep kNN

- Deep kNN classification.



End-to-end training is prohibitive using first-order methods, e.g., SGD, since top-k operation is **not differentiable**!

- Bubble? Heap? Quicksort partition? – gradient cannot be computed.
- Consider top-k as a function returns an indicator vector?

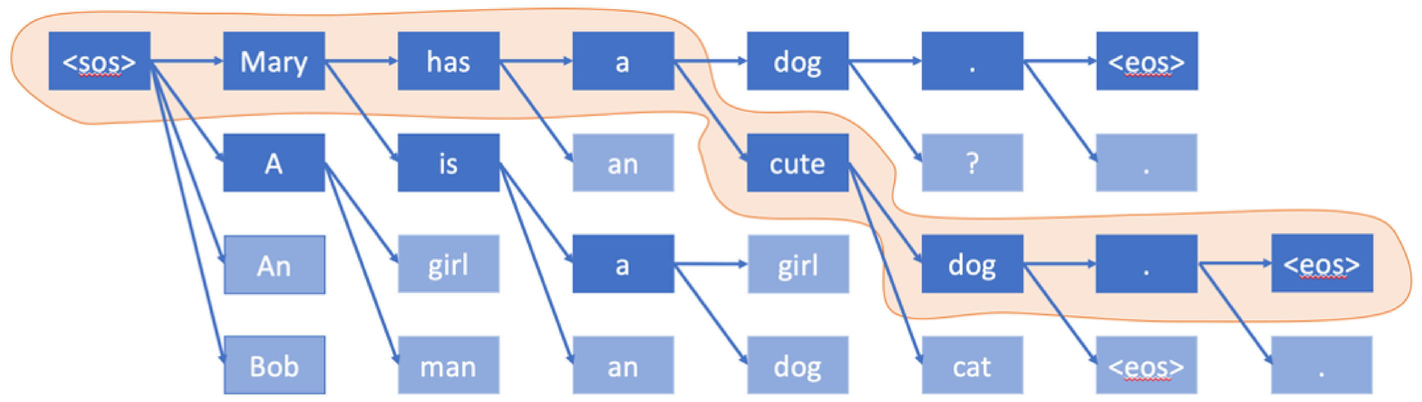
$x_1 = [4, 1, 2, 3, 0, 1, 2]^\top$ → **Top-1** → The first entry is the largest $[\mathbf{1}, 0, 0, 0, 0, 0, 0]^\top$

$x_2 = [5, 3, 1, 0, 2, 2, 1]^\top$ → **Top-1**

Distinct inputs — Zero gradient — Identical output

Motivating Example 2 – Beam Search

- A popular **inference** method in machine translation tasks.
- Recursively keeps k sequences with the largest likelihoods, and feeds them into the decoder to predict the next token.



▷ Misalignment between training and inference.
- In the training stage, the ground truth sequence is fed into the decoder;
- In the inference stage, the tokens generated by the decoder are used.

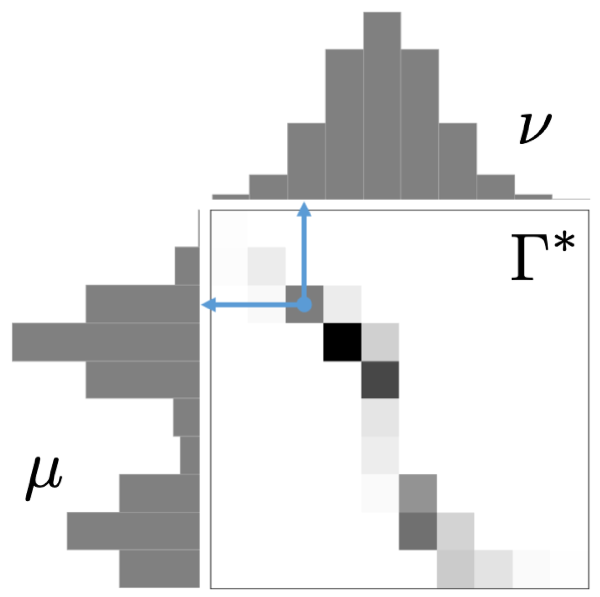
End-to-end training requires beam search to be **"differentiable"**!

Preliminary: Optimal transport (OT)

OT aims to find the optimal plan to transport mass between two distributions.

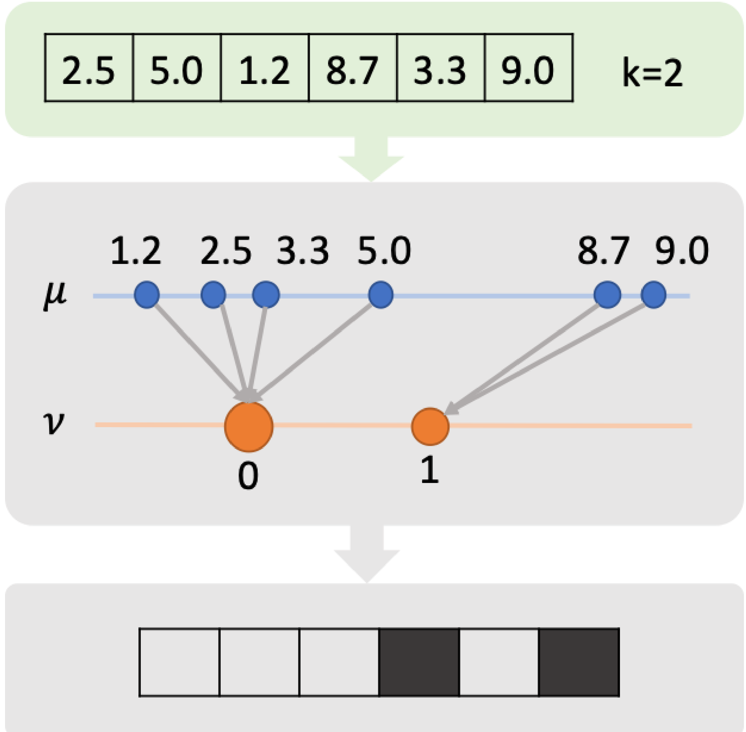
$$\Gamma^* = \operatorname*{argmin}_{\Gamma \geq 0} \langle C, \Gamma \rangle,$$
$$\text{s.t.}, \quad \Gamma \mathbf{1}_m = \mu, \ \Gamma^\top \mathbf{1}_n = \nu,$$

- μ, ν source and target distributions;
- C cost matrix;
- Γ transport plan.



Differentiability — SOFT Top-k

- Parameterizing Top-k Operator as an OT Problem:



Input vector

μ: empirical distribution of the input
ν: Bernoulli(k/n)

Output indicator vector A: which entries are aligned to 1

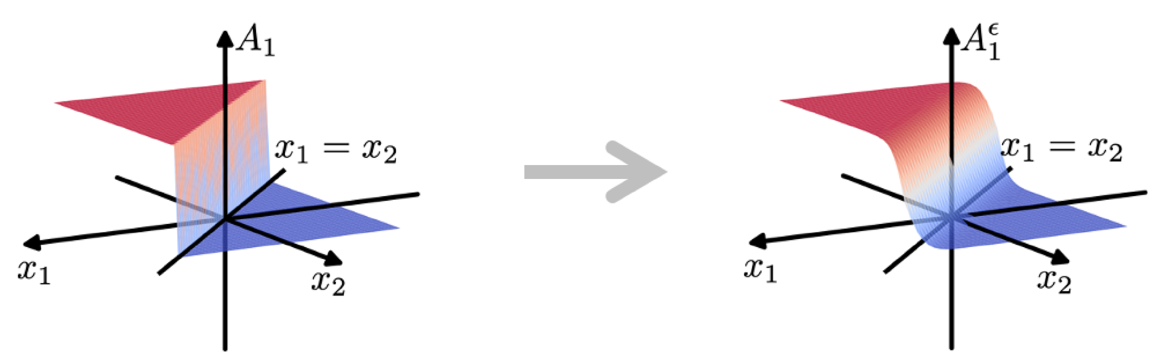
- **Smoothing** using Entropy Regularization:

$$\Gamma^{*,\epsilon} = \operatorname*{argmin}_{\Gamma \geq 0} \langle C, \Gamma \rangle + \epsilon H(\Gamma), \quad \text{s.t.}, \quad \Gamma \mathbf{1}_m = \mu, \ \Gamma^\top \mathbf{1}_n = \nu,$$

where $H(\Gamma) = \sum_{i,j} \Gamma_{i,j} \log \Gamma_{i,j}$, m, n are the input and output dimensions.

SOFT top-k operator: input vector $\mapsto A^\epsilon := n\Gamma^{*,\epsilon} \cdot [0, 1]^\top$.

Theorem 1.
(1) *(Nonzero gradient) Under mild conditions, SOFT top-k operator is differentiable; its Jacobian matrix* **always has nonzero entries**.
(2) *(Small approximation error)* $\|\Gamma^{*,\epsilon} - \Gamma^*\|_F = O\left(\frac{\epsilon \log n}{n \cdot gap_k}\right)$, *where* gap_k *denotes the gap between the* $(k+1)$*-th and the* k*-th largest input entries.*



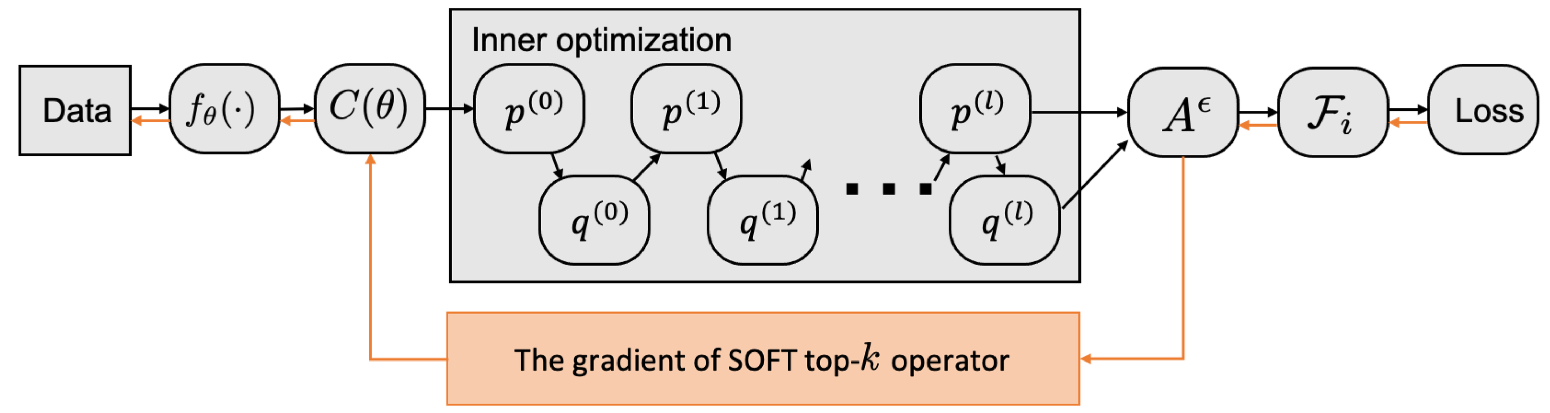
Efficient Implementation

- Consider deep kNN in classification. Minimizing the loss for each query sample is a **bilevel optimization** problem:

$$\mathcal{F}_i(\theta) = \sum_{j=1}^{n_t} A_j^\epsilon(\theta) \ell(y_i^q, y_j^t), \quad \text{s.t.}, \quad A^\epsilon(\theta) = n\Gamma^{*,\epsilon}(\theta) \cdot [1, 0]^\top,$$
$$\Gamma^{*,\epsilon}(\theta) = \operatorname*{argmin}_{\Gamma \geq 0} \langle C(\theta), \Gamma \rangle + \epsilon H(\Gamma), \quad \Gamma \mathbf{1}_m = \mu, \ \Gamma^\top \mathbf{1}_n = \nu.$$

- $\{x_i^q, y_i^q\}_{i=1}^{n_q}$ are query samples, $\{x_j^t, y_j^t\}_{j=1}^{n_t}$ are template samples;
- $\ell(\cdot, \cdot)$ is the zero-one loss;
- $C_{ij}(\theta) = c(f_\theta(x_i^q), f_\theta(x_j^t))$, $c(\cdot, \cdot)$ is the squared Euclidean distance;
- $f_\theta(\cdot)$ is the feature extractor parametrized by θ.

- **Efficient gradient computation**. When optimizing $\min_\theta \sum_{i=1}^{n_q} \mathcal{F}_i(\theta)$ using SGD, KKT conditions yield *closed-form expression* of $\nabla_{C(\theta)} A^\epsilon(\theta)$:



- **Computationally efficient**: simple matrix operations;
- **Memory efficient**: no need to store intermediate steps.

Experiment – Deep kNN

Backbone: ResNet-18

Algorithm	MNIST	CIFAR10
kNN	97.2%	35.4%
kNN+PCA	97.6%	40.9%
kNN+pretrained CNN	98.4%	91.1%
RelaxSubSample	99.3%	90.1%
kNN+NeuralSort	**99.5%**	90.7%
kNN+Cuturi (2019)	99.0%	84.8%
kNN+Softmax k times	99.3%	92.2%
CE+CNN (He, 2016)	99.0%	91.3%
kNN+**SOFT Top-k**	99.4%	**92.6%**

Experiment – Beam Search

WMT14 Single LSTM

Algorithm	BLEU
Luong (2014)	33.10
Durrani (2014)	30.82
Cho (2014)	34.54
Sutskever (2014)	30.59
Bahdanau (2014)	28.45
Jean (2014)	34.60
Bahdanau (2014) (Our implementation)	35.38
Beam Search + SOFT Top-k	**36.27**