Learning Latent Permutations with Gumbel-Sinkhorn Networks

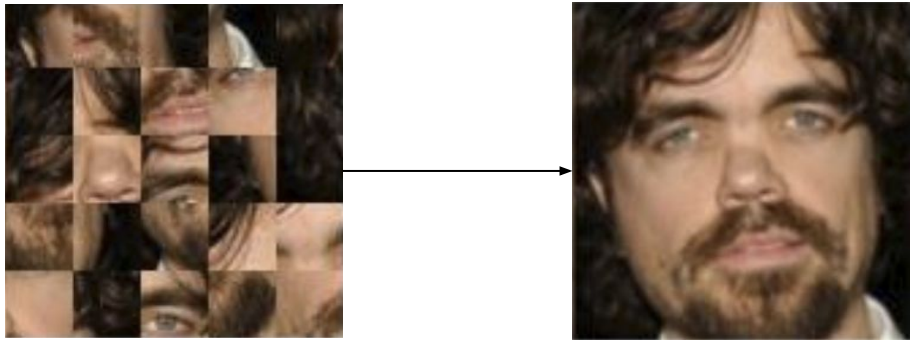
Gonzalo E. Mena
David Belanger
Scott Linderman
Jasper Snoek

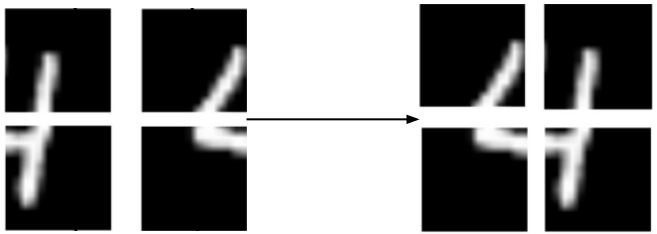
Presented by: Lawson Fulton, Trefor Evans

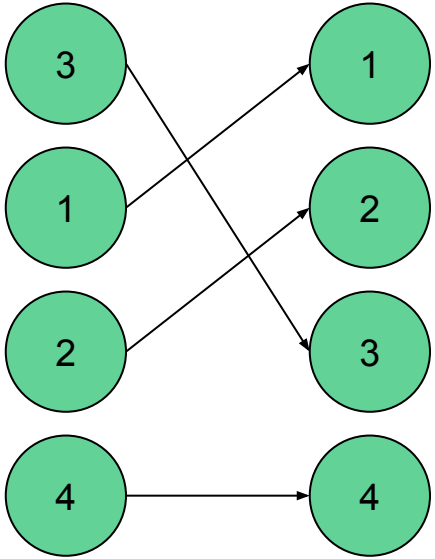
Why Latent Permutations & Matchings?

Align, **canonicalize**, and **sort** data

Want to learn the matching ***without*** explicit labels







Representing Permutations

Given a permutation mapping

$$\pi : \{1, \ldots, m\} \to \{1, \ldots, m\}$$

$$\begin{pmatrix} 1 & 2 & \cdots & m \\ \pi(1) & \pi(2) & \cdots & \pi(m) \end{pmatrix}$$

We can represent it as multiplication of the identity matrix with permuted rows

$$P_\pi = \begin{bmatrix} \mathbf{e}_{\pi(1)} \\ \mathbf{e}_{\pi(2)} \\ \mathbf{e}_{\pi(3)} \\ \mathbf{e}_{\pi(4)} \\ \mathbf{e}_{\pi(5)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$P_\pi \mathbf{g} = \begin{bmatrix} \mathbf{e}_{\pi(1)} \\ \mathbf{e}_{\pi(2)} \\ \vdots \\ \mathbf{e}_{\pi(n)} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix} = \begin{bmatrix} \pi(1) \\ \pi(2) \\ \vdots \\ \pi(n) \end{bmatrix}$$

Parameterizing Permutations

Matching operator gives mapping from unconstrained matrices to permutations.

$$M(X) = \underset{P \in \mathcal{P}_N}{\arg\max} \langle P, X \rangle_F$$

Non-differentiable and requires considering $\boldsymbol{n!}$ permutations.

Relaxing Permutations

The set of *doubly stochastic* matrices gives the *Birkhoff Polytope*

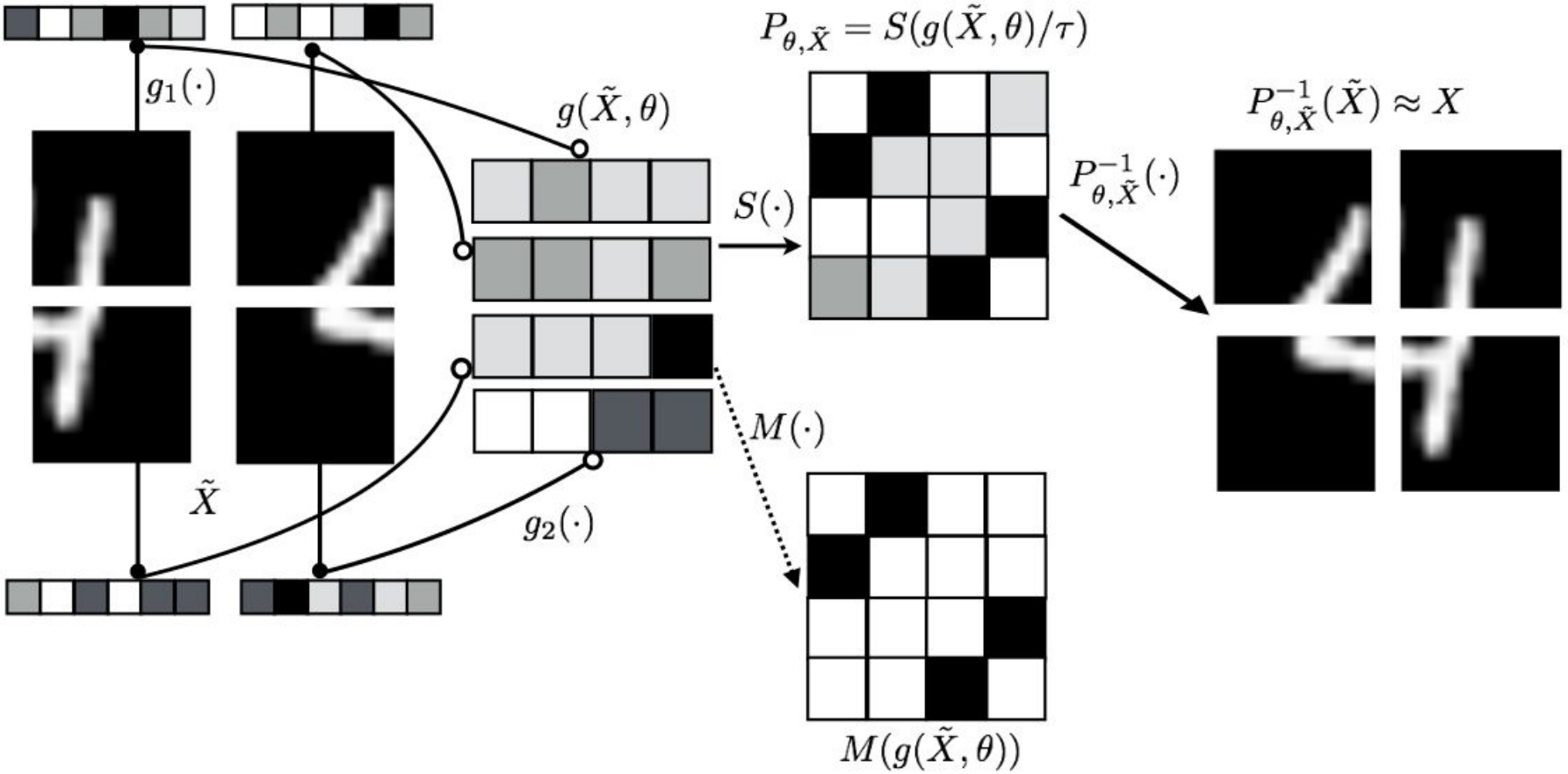
$$\mathcal{B}_N = \left\{ A \in \mathbb{R}^{N \times N} \mid \sum_i a_{ij} = \sum_j a_{ij} = 1 \right\}$$

The Birkhoff-von Neumann theorem states that this defines the convex hull of *N*x*N* permutation matrices.

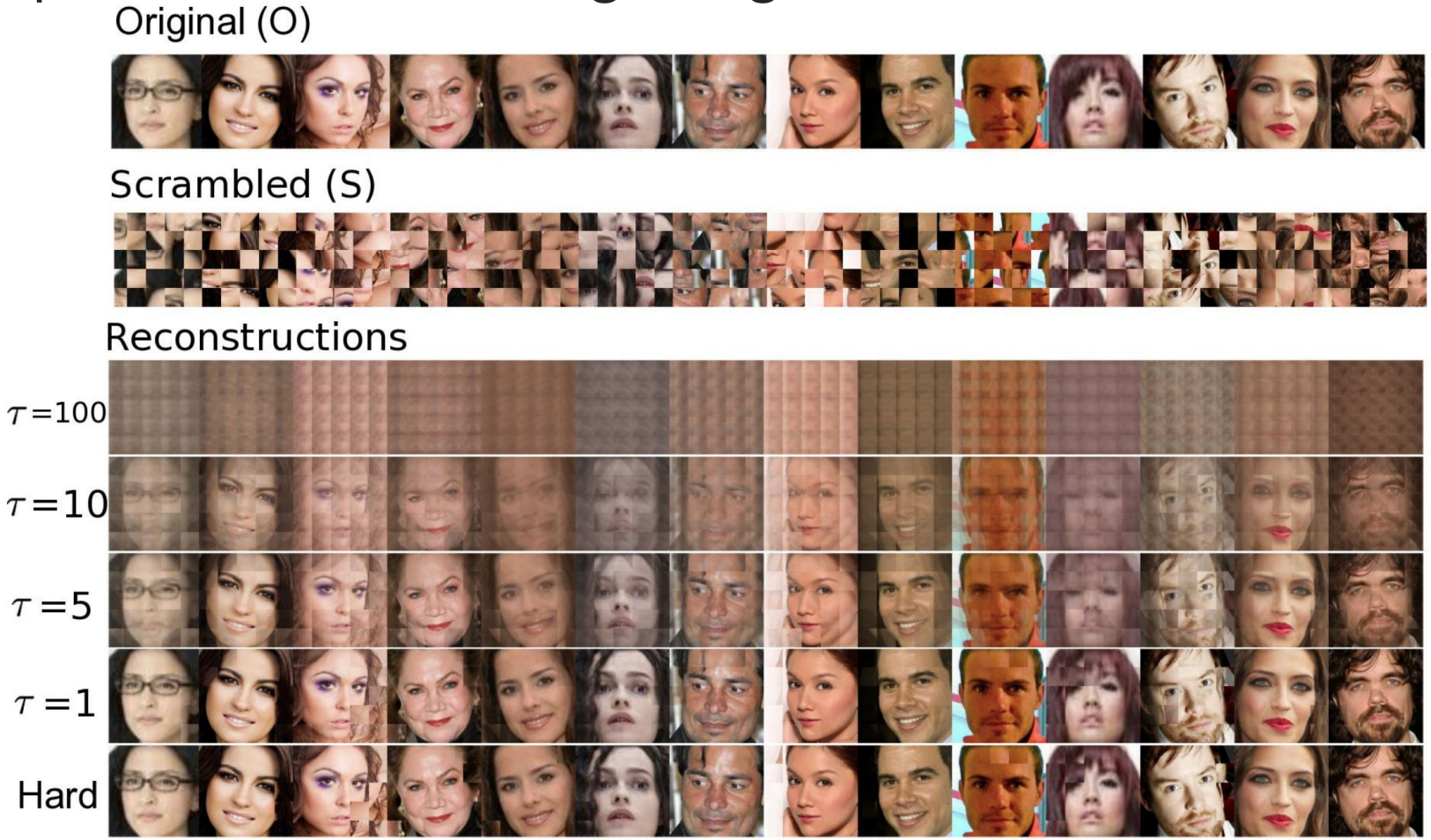
Sinkhorn Relaxation & Gumbel-Sinkhorn Reparam

See Demo in Jupyter notebook

Sinkhorn Networks for Matching



Experiments - Shuffling Images



Experiments - Sorting

Test distribution	$N = 5$	$N = 10$	$N = 15$	$N = 80$	$N = 100$	$N = 120$
$U(0, 1)$	**.0**	**.0**	**.0**	**.0**	**.0**	**.01**
$U(0, 1)$ (Vinyals et al., 2015)	.06	0.43	0.9	-	-	-
$U(0, 10)$	.0	.0	.0	.0	.02	.03
$U(0, 1000)$	.0	.0	.0	.01	.02	.04
$U(1, 2)$	.0	.0	.0	.01	.04	.08
$U(10, 11)$	.0	.0	.0	.08	.08	.6
$U(100, 101)$	.0	.0	.01	.02	.99	1.
$U(1000, 1001)$	.0	.0	.07	1.	1.	1.

Table 1: Results on the number sorting task measured using Prop. any wrong. In the top two rows we compare to Vinyals et al. (2015), showing that our approach can sort far more inputs at significantly higher accuracy. In the bottom rows we evaluate generalization to different intervals on the real line.

Thank You