A APPENDIX

A.1 MONGE PROBLEM

Define two discrete measures on two arbitrary sets $\mathcal{X}$ and $\mathcal{Y}$, respectively, i.e.,

$$\alpha = \sum_{i=1}^{n} a_i \delta_{x_i}, \quad \beta = \sum_{j=1}^{m} b_j \delta_{y_j},$$

where $\boldsymbol{a} = (a_i)_{i=1}^n \in \Sigma^n$ and $\boldsymbol{b} = (b_j)_{j=1}^m \in \Sigma^m$.

We continue to define a *surjective* map $T : \mathcal{X} \to \mathcal{Y}$ that associates each x_i to y_j, such that for all $j \in [\![m]\!]$, $b_j = \sum_{Tx_i = y_j} a_i$. Induced by mass transport, a transport of discrete probability measures can be denoted in a compact *push-forward* form $T_\sharp \alpha = \beta$. Furthermore, the *Monge problem* is to seek an optimal T to minimize the transportation cost with respect to a non-negative cost function $c(x, y)$ (satisfying the three properties of a legitimate distance metric) defined on $\mathcal{X} \times \mathcal{Y}$, i.e.,

$$\min_T \Big\{ \sum_i c(x_i, Tx_i) \big| T_\sharp \alpha = \beta \Big\}. \tag{10}$$

In particular, when $n = m$, then T is a bijection from *source* $\mathcal{X}$ to *target* $\mathcal{Y}$, and can induce a permutation $\sigma \in \text{Perm}(n)$, such that $Tx_i = y_{\sigma(i)}$.

Remark 4. *Note that the Monge problem may not even have a solution when the measures α and β are not compatible, which is always the case when the target measure has more points than the source measure, i.e., $n < m$. Besides, it is also the case when $n = m$ and there is at least one points in the target measure with the mass that does not match any point the source measure, i.e., $\exists\ b_j, s.t.\ b_j \neq a_i, \forall i$, and vice versa. We leave more complex situations for the readers due to the deviation from our main topic.*

A.2 CONVERGENCE OF ENTROPIC REGULARIZED OT

The following results given by Peyré et al. (2019), Proposition 4.1, indicating the relationship between the original OT problem (1) and the entropic regularized OT (2).

Proposition 2 (Convergence of ε)**.** *The unique solution $\mathbf{P}_\varepsilon$ converges to the optimal solution with maximal entropy within the set of all optimal solutions of the OT problem (1), namely*

$$\mathbf{P}_\varepsilon \overset{\varepsilon \to 0}{\longrightarrow} \underset{\mathbf{P} \in U(\boldsymbol{a}, \boldsymbol{b})}{\arg\min} \{ -\mathbf{H}(\mathbf{P}) \mid \langle \mathbf{P}, \mathbf{C} \rangle = \mathbf{L}_\mathbf{C}(\boldsymbol{a}, \boldsymbol{b}) \}, \tag{11}$$

so that in particular

$$\mathbf{P}_\varepsilon \overset{\varepsilon \to 0}{\longrightarrow} \mathbf{L}_\mathbf{C}(\boldsymbol{a}, \boldsymbol{b}).$$

On the other hand, we have

$$\mathbf{P}_\varepsilon \overset{\varepsilon \to \infty}{\longrightarrow} \boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^\intercal.$$

Remark 5. *A key insight is, as ε increases, the optimal coupling of the problem (2) becomes less and less sparse, which in turn has the effect of both accelerating computational algorithms, and leading to faster statistical convergence. Besides, we can decrease the magnitude of ε to restore the original OT problem. In fact, the entropic regularization term provides a trade-off between the computational efficiency and the sparsity of solution.*

A.3 FROM SUPERVISED LEARNING TO UNSUPERVISED LEARNING

Most of the content in this section is the reorganization of the seminal work of Asano et al. (2019). We present the most relevant results here as an essential preliminary, in order to help the readers to clearly understand our OT-LLP framework.

In the standard supervised learning, after mapped by a deep neural network (DNN) $\Phi_\theta(\cdot)$, the high-level representation $\Phi_\theta(\mathbf{x})$ is followed by the fully connected multi-layer perceptron (MLP) h_φ to yield the class-specific logits. Based on these logits, DNN obtains the corresponding posterior class probabilities with a *softmax* operation. Denote the multi-class classification head by $g_\varphi : \mathbb{R}^D \to \mathbb{R}^K$, then $g_\varphi = \mathrm{softmax} \circ h_\varphi$. In other words, the probabilistic output for an instance $\mathbf{x}$ can be summarized as

$$p_\phi(\boldsymbol{y}|\mathbf{x}) = g_\varphi(\Phi_\theta(\mathbf{x})) = \mathrm{softmax} \circ h_\varphi(\Phi_\theta(\mathbf{x})) \in 2^K, \tag{12}$$

where $\phi = (\varphi, \theta)$ is the parameters of the model. Furthermore, the representation part and head parameters (θ, φ) of DNN are learned with a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N) | (\mathbf{x}_i, y_i) \in \mathcal{X} \times [\![K]\!]\}$, by minimizing the average cross-entropy loss with respect to (θ, φ)

$$CE(y, p|\mathcal{D}, \theta, \varphi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(y_i = k)\log p(k|\mathbf{x}_i, \theta, \varphi) = -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i|\mathbf{x}_i, \theta, \varphi). \tag{13}$$

On the other hand, the standard semi-supervised learning normally leverages a transductive scheme to incorporate the information of unlabeled data through self-labeling framework. The key idea is, when the labels are unavailable, we require a self-labeling mechanism to assign the labels automatically.

To be concrete, suppose $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$ is the labeled data, and $\{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \cdots, \mathbf{x}_{l+u}\}$ is the unlabeled data, i.e., $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \bigcup \{\mathbf{x}_j\}_{j=l+1}^{l+u}$. The representation part and head parameters (φ, θ) are learned by minimizing the average cross-entropy loss $\mathcal{L}$ on $\mathcal{D}$

$$\mathbb{E}_{P(X,Y)}(\mathcal{L}(g(X), Y) \backsimeq \hat{\mathbb{E}}(\mathcal{L}(g(\mathbf{x}), y)) = -\frac{1}{l+u}\sum_{i=1}^{l+u}\sum_{k=1}^{K}\mathbb{I}(y_i = k)\log p_\phi^k(\mathbf{x}_i). \tag{14}$$

Although the formulation remains the same in (13) and (14), different to (13) in supervised learning, the transductive scheme in semi-supervised learning is to proceed joint optimization on (14) with respect to $\phi = (\theta, \varphi)$ and the labeling $\{y_i\}_{i=l+1}^{l+u}$ of the unlabeled data simultaneously, according to the fully supervised information in labeled data, i.e., alternately conducting optimization on the following two problems by fixing the other parameters

$$\min_{(\varphi,\theta)} \; -\frac{1}{l+u}\sum_{i=1}^{l+u}\log p_\phi^{y_i}(\mathbf{x}_i) = -\frac{1}{l+u}\sum_{i=1}^{l+u}\log(\mathrm{softmax} \circ h_\varphi(\Phi_\theta(\mathbf{x}_i))_{y_i}, \tag{15}$$

and

$$\min_{\boldsymbol{y}\in[\![K]\!]^u} \; -\frac{1}{u}\sum_{i=l+1}^{l+u}\sum_{k=1}^{K}\mathbb{I}(y_i = k)\log p_\phi^k(\mathbf{x}_i) = -\frac{1}{u}\sum_{i=l+1}^{l+u}\sum_{k=1}^{K}\mathbb{I}(y_i = k)\log(\mathrm{softmax} \circ h_\varphi(\Phi_\theta(\mathbf{x}_i))_k, \tag{16}$$

where $\boldsymbol{y} = (y_{l+1}, y_{l+2}, \cdots, y_{l+u})^\intercal$ is the unknown labels for the unlabeled data.

The above formulation can be naturally extended to fully unsupervised learning scenario, with training data $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, by performing the clustering and representation learning simultaneously. However, the intact transfer of this mechanism without constraints on labeling will lead to degeneration to the final

result: Assigning all the data points to a single (arbitrary) label. To avoid this issue, Asano et al. (2019) introduce the following constrained self-labeling framework to solve the clustering.

First, the *soft* encoding of the labels is to utilize the probabilistic (non-mutually-exclusive) classifier to inference the posterior distribution (pseudo-labeling), based on the following cross-entropy loss:

$$CE(p,q) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{K} q(y|\mathbf{x}_i)\log p(y|\mathbf{x}_i). \tag{17}$$

To be concrete, for fixed p, the constrained minimization problem of q according to $CE(p,q)$ is

$$\begin{aligned} &\min_{q} \;\; CE(p,q) \\ &s.t. \;\; \sum_{i=1}^{N} q(y|\mathbf{x}_i) = \frac{N}{K}, q(y|\cdot) \in [0,1], \forall y \in [\![K]\!]. \end{aligned} \tag{18}$$

Accordingly, we obtain a combinatorial programming and thus may appear very difficult to optimize. However, we can transfer (18) to a standard optimal transport problem, and solve it quickly, thanks to the Sinkhorn's algorithm (Cuturi, 2013). This can be concluded as the **self-labeling** step.

Formally, let $\mathbf{Q} = (Q_{ij}) \in \mathbb{R}_{+}^{K\times N}$ and $Q_{ij} = q_i(\mathbf{x}_j)/N$. Similarly, let $\mathbf{P} = (P_{ij}) \in \mathbb{R}_{+}^{K\times N}$ and $P_{ij} = p_i(\mathbf{x}_j)/N$. In addition, denote

$$\boldsymbol{a} = \frac{1}{K}\mathbb{1}_K, \;\; and \;\; \boldsymbol{b} = \frac{1}{N}\mathbb{1}_N.$$

Hence,

$$U(\boldsymbol{a},\boldsymbol{b}) = \Big\{\mathbf{Q} \in \mathbb{R}_{+}^{K\times N} \mid \mathbf{Q}\mathbb{1}_N = \boldsymbol{a}, \; \mathbf{Q}^{\intercal}\mathbb{1}_K = \boldsymbol{b}\Big\}.$$

Then, we can give an equivalent problem for (18) with the following OT problem:

$$\min_{\mathbf{Q}\in U(\boldsymbol{a},\boldsymbol{b})} \langle \mathbf{Q}, -\log \mathbf{P}\rangle = CE(p,q) + \log N. \tag{19}$$

After updating q in (18), we can perform unconstrained optimization on $CE(p,q)$ with repect to the parameter in p, i.e., $\min_{\phi} CE(p_\phi, q)$. The can be denoted as the *representation learning* step.

We can alternately conduct the above *self-labeling* and *representation learning* steps, to realize the unsupervised classification.

A.4 The Network architectures using in the experiments

In Table 2, we deliver the convolution neural network architecture used in our experiments for all the methods. More specifically, it is a 13-layer convolutional neural network, where Leaky ReLU is chosen as the activation function. Note that the network architecture is for the CIFAR-10 and CIFAR-100 dataset, while a dense-based architecture is applied for the MNIST, K-MINST, and F-MNIST datasets as shown in Table 3. In detail, it is a fully connected network with 5 hidden layers. In particular, the *Dense* in the table means the fully connected layer followed by ReLU as the activation function.

Table 2: The baseline architecture for CIFAR-10.

Input 32×32 RGB image
3×3 conv. 128 followed by LeakyReLU
3×3 conv. 128 followed by LeakyReLU
3×3 conv. 128 followed by LeakyReLU
2×2 max-pooling with stride 2, dropout with $p = 0.5$
3×3 conv. 256 followed by LeakyReLU
3×3 conv. 256 followed by LeakyReLU
3×3 conv. 256 followed by LeakyReLU
2×2 max-pooling with stride 2, dropout with $p = 0.5$
3×3 conv. 512 followed by LeakyReLU
1×1 conv. 256 followed by LeakyReLU
1×1 conv. 128 followed by LeakyReLU
Global Mean pooling
Dense 10
10-way Softmax

Table 3: The baseline's architectures for MNIST, K-MNIST and F-MNIST.

Input 28×28 gray image
Dense 28×28 $\rightarrow$ 1000 followed by ReLU
Dense 1000 $\rightarrow$ 500 followed by ReLU
Dense 500 $\rightarrow$ 250 followed by ReLU
Dense 250 $\rightarrow$ 250 followed by ReLU
Dense 250 $\rightarrow$ 250 followed by ReLU
Dense 250 $\rightarrow$ 10
10-way Softmax