Semi-Supervised Pairing via Basis-Sharing Wasserstein Matching Auto-Encoder

Ziyin Liu[†], **Yao-Hung Hubert Tsai**[‡], **Makoto Yamada***, **Ruslan Salakhutdinov**[‡]
[†]University of Tokyo, [‡]Carnegie Mellon University, *Kyoto University/RIKEN AIP
zliu@cat.phys.s.u-tokyo.ac.jp ,yaohungt@cs.cmu.edu,
myamada@i.kyoto-u.ac.jp,rsalakhu@cs.cmu.edu

1 Introduction

Semi-supervised pairing, learning explicit pairing relation between unlabeled input and target distribution in an otherwise semisupervised setting, has recently attracted lots of attention [16, 12], since data labeling is often time-consuming and requires lots of human labor. In this paper, we propose Basis-Sharing Wasserstein Matching Auto-Encoder to tackle the problem of semi-supervised pairing. Our model inspired by the success of robust representation learning for matching cross-modal latent space distribution [15, 10] and good statistical properties of optimal transport [1, 4]. In particular, Wasserstein distance in the optimal transport family has been shown to work on GANs and autoencoders [2, 14]. We propose to match the latent code distribution of the unlabeled dataset, with the labeled data points as "anchor points". This should help the network making association between the distributions from two different domains. A similar idea that uses a different method has been shown to work on some language tasks [3]; our work explores a new approach based on distribution matching. Through preliminary experiments, we show that the proposed algorithm can successfully incorporate the unlabeled data for improving the classification accuracy on MNIST and CIFAR10 datasets.

2 Proposed Method

2.1 Wasserstein Training of Cross-Domain Auto-encoders

We denote unlabeled training set as $\mathcal{D}_{UL} = \{\{\boldsymbol{x}_i\}_{i=1}^n, \{\boldsymbol{y}_j\}_{j=1}^m\}$, on which we want to find a pairing relation, and labeled training set as $\mathcal{D} = \{(\boldsymbol{x}_i', \boldsymbol{y}_i')\}_{i=1}^r$. Data points $\boldsymbol{x}_\cdot, \boldsymbol{x}_\cdot' \sim \mathcal{X} \in \mathbb{R}^{d_x}$ and $\boldsymbol{y}_\cdot, \boldsymbol{y}_\cdot' \sim \mathcal{Y} \in \mathbb{R}^{d_y}$, where $\mathbb{R}^{d_x}$ can be different from $\mathbb{R}^{d_y}$. Our model contains two auto-encoders on the input domain $\{\boldsymbol{x}_i\}_{i=1}^n$ and the target domain $\{\boldsymbol{y}_j\}_{j=1}^m$ respectively:

$$\widehat{\boldsymbol{x}}_i = \text{Dec}_{\boldsymbol{x}}\Big(\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}_i)\Big), \quad \widehat{\boldsymbol{y}}_i = \text{Dec}_{\boldsymbol{y}}\Big(\text{Enc}_{\boldsymbol{y}}(\boldsymbol{y}_i)\Big),$$

where $\widehat{\boldsymbol{x}}_\cdot$ and $\widehat{\boldsymbol{y}}_\cdot$ represent reconstructed data. For inference across domains $x \to y$, we switch the decoder of the two, and notice that backward inference can also be done easily ($y \to x$):

$$\widehat{\boldsymbol{y}}_i = \text{Dec}_{\boldsymbol{y}}\Big(\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}_i)\Big), \quad \widehat{\boldsymbol{x}}_i = \text{Dec}_{\boldsymbol{x}}\Big(\text{Enc}_{\boldsymbol{y}}(\boldsymbol{y}_i)\Big).$$

We also propose that the encoders share the latent structure across domains:

$$\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}_i) = \boldsymbol{B}\boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x}_i), \quad \text{Enc}_{\boldsymbol{y}}(\boldsymbol{y}_i) = \boldsymbol{B}\boldsymbol{a}_{\boldsymbol{y}}(\boldsymbol{y}_i), \tag{1}$$

where $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k] \in \mathbb{R}^{d_h \times k}$ are shared low-rank decomposed basis vectors with rank k, and $\boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x})$ and $\boldsymbol{a}_{\boldsymbol{y}}(\boldsymbol{y})$ are standard deep encoders. This auto-encoder model with shared low rank representation allows us to explicitly share the model parameters between domains $\mathcal{H}_{\boldsymbol{x}} = \{\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}_i)\}_i^n$, and $\mathcal{H}_{\boldsymbol{y}} = \{\text{Enc}_{\boldsymbol{y}}(\boldsymbol{y}_i)\}_i^m$. On this shared space we minimize the Wasserstein distance(WD) between

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

the input domain and the target domain, i.e. $\ell_{wass} = WD(\mathcal{H}_{\boldsymbol{x}}, \mathcal{H}_{\boldsymbol{y}})$. So our objective function for the training is:

$$\min_{\boldsymbol{\Theta}} \quad E(\boldsymbol{\Theta}, \mathcal{D}) + \rho R(\boldsymbol{\Theta}, \mathcal{D}_{UL}) + \gamma WD(\mathcal{H}_{\boldsymbol{x}}, \mathcal{H}_{\boldsymbol{y}}) \tag{2}$$

where $\boldsymbol{\Theta}$ is the model parameter of auto-encoder models, $E(\boldsymbol{\Theta}, \mathcal{D})$ is the supervised pairing (i.e., $\|\boldsymbol{y}'_i - \text{Dec}_{\boldsymbol{y}}\big(\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}'_i)\big)\|_2^2$) loss on the labeled data, $R(\boldsymbol{\Theta}, \mathcal{D}_{UL})$ is the auto-encoder reconstruction loss (on unpaired data), and $\gamma, \rho \geq 0$ are regularizaiton parameters. For the algorithm for Wasserstein training, see Algorithm 1 in Appendix.

2.2 Calculating Wasserstein Distance with the Sinkhorn algorithm

An optimal transport method allows us to find the pairing relation explicitly because of its reliance on finding the pairing kernel p_{ij} between x_i and y_j. Hence, Wasserstein distance (WD) has been proposed as a reliable distance measure between distributions. Because it also considers the geometrical properties of the distributions [17], it serves as a much better measure for learning to match two different distributions than information distances such as the KL divergence [7]. Our key motivation is that minimizing Wasserstein Distance allows the model to learn a pairing relation in a explicit and robust way. The Sinkhorn algorithm is based on the minimization of the following Lagrangian with the constraint that the marginal distribution is specified [4]; following [5], we constrain the marginal to be uniform. The Wasserstein distance with entropic smoothing ($WD()$ in Equation 2) is then defined as $WD(\mathcal{H}_x, \mathcal{H}_y) = \sum_{i=1}^n \sum_{j=1}^m p_{ij} c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}})$ where p_{ij} is found by:

$$\underset{\boldsymbol{P} \geq 0}{\arg\min} \quad \sum_{i=1}^n \sum_{j=1}^m p_{ij} c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}}) + \lambda \sum_{i=1}^n \sum_{j=1}^m p_{ij} \log p_{ij}, \text{ s.t.} \quad \sum_{i=1}^n p_{ij} = r_j, \sum_{j=1}^m p_{ij} = c_i, \forall\ i, j. \tag{3}$$

where $\lambda \geq 0$ is a hyperparameter. In this paper, we let $c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}}) = \|\text{Enc}_{\boldsymbol{x}}(\boldsymbol{x}_i) - \text{Enc}_{\boldsymbol{y}}(\boldsymbol{y}_j)\|_2^2$ and set $r_j = \frac{1}{m}$ and $c_j = \frac{1}{n}$, respectively. While the original WD requires linear programming (i.e., takes $O(n^2)$ complexity) to solve, the Sinkhorn algorithm [5] finds a smooth solution in only $O(n \log n)$.

2.3 Wasserstein distance with weight decay (WDWD)

It is not trivial to compute WD. Inspired by the well-known Sinkhorn algorithm, we relax the entropic smoothing constraint further, and reparametrize the transportation plan p_{ij} as a function of its potentials ϕ_{ij}. We propose Weight Decay Wasserstein Distance (WDWD) on the latent distribution $\mathcal{H}_x$ and $\mathcal{H}_y$:

$$\underset{\boldsymbol{P} \geq 0}{\arg\min} \quad \sum_{i=1}^n \sum_{j=1}^m p_{ij} c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}}) + \lambda \sum_{i=1}^n \sum_{j=1}^m \log^2 p_{ij}, \text{ s.t.} \quad \sum_{i=1}^n p_{ij} = r_j, \sum_{j=1}^m p_{ij} = c_i, \forall\ i, j, \tag{4}$$

Assume $p_{ij} = \text{softmax}(\phi_{ij})$ and $\sum_{i=1}^n \sum_{j=1}^m p_{ij} = 1$, and let $\boldsymbol{x}, \boldsymbol{y}$ be any high dimensional vector, and $c(\cdot)$ a distance function between them, then we have

$$\sum_{i=1}^n \sum_{j=1}^m p_{ij} c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}}) + \lambda \sum_{i=1}^n \sum_{j=1}^m \log^2 p_{ij} \geq \sum_{i=1}^n \sum_{j=1}^m p_{ij} c(\boldsymbol{h}_i^{\boldsymbol{x}}, \boldsymbol{h}_j^{\boldsymbol{y}}) + \lambda \sum_{i=1}^n \sum_{j=1}^m (\phi_{ij} - \log nm)^2, \tag{5}$$

and LHS equals RHS when we have a maximum entropy distribution, i. e., a uniform distribution over i, j. Namely, we have $\phi_{ij} \to \log nm$ for large λ, and we achieve a maximum entropy solution $p_{ij} = \frac{1}{nm}$, and in this case the inequality becomes an equality. Recalling that the Sinkhorn algorithm [4] is also based on entropic smoothing; hence, our method is qualitatively similar to it. In future, we will investigate the property of this regularizer in detail, some initial experiments are included in Appendix. Our algorithm for computing WDWD is given in Algorithm 2 in Appendix.

3 Experiment

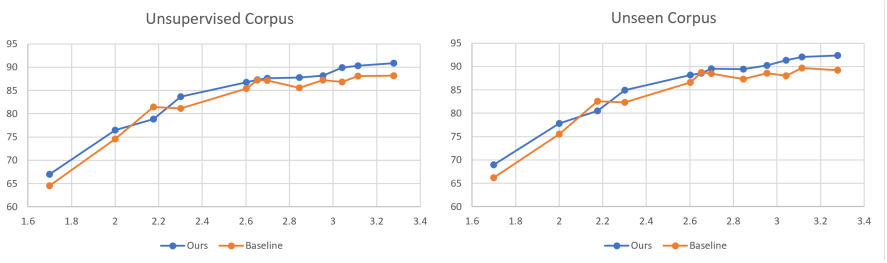
We demonstrate the proof of concept on MNIST. We use the 60000 images in the training set as $\mathcal{D}_{UL}$ The rest of the standard testing set (consisting of 9900 images) are used as $\mathcal{D}_{Te}$. We use the first 100 images of the standard testing set as $\mathcal{D}$. It is on $\mathcal{D}_{UL}$ that we want to find an association relation between the input (images) and target (labels) distributional, i.e. the paring relation. See Tab. 1 for our result. We notice that baseline model that trains an autoencoder on $\mathcal{D}_{UL}$ improves over the pure supervised model by a large margin, and both Wasserstein matching methods improve further, suggesting that it learned better than the baseline to make connections between the labeled set and unlabeled set. In Figure 1, we show how the method scales towards larger proportion of

Table 1: Performance on MNIST. We set the dimensionality of the latent space to 20; a "−" in the table means that no distribution matching is used; Unsup. refers to whether we add reconstruction loss to the autoencoders; Unsup. Acc. refers to the classification accuracy on $\mathcal{D}_{UL}$. Acc. refers to the classification accuracy on test set.

Basis-Sharing	**Distance**	**Unsup.**	**Unsup. Acc.**	**Testing. Acc.**
No	-	No	62.89%	63.10%
No	-	Yes	75.14%	**76.11%**
No	Sinkhorn [4]	Yes	**75.25%**	75.77%
No	WDWD	Yes	73.25%	74.25%
Yes	-	No	61.66%	61.91%
Yes	-	Yes	75.99%	77.20%
Yes	Sinkhorn [4]	Yes	76.48%	77.92%
Yes	WDWD	Yes	**78.83%**	**79.80%**

Table 2: Experiment on CIFAR10. Performance of the cited/proposed methods are shown in difference from the baseline.

Method	**Unsup. Acc.(%)**	**Testing. Acc.(%)**
Supervised	70.02%	70.81%
Sinkhorn	**+2.10**	+1.91
WDWD	+2.01	**+3.30**

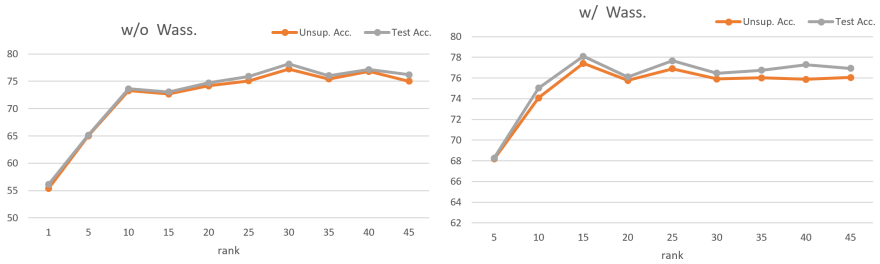


(a) Comparison on Unsupervised Corpus (b) Comparison on Supervised Corpus

Figure 1: Scaling effect of our method. The y-axis is the classification accuracy, and the x-axis is the log-size of the supervised corpus. The first point refers to sample per label = 5 and the last refers to sample per label = 200.

labeled set. We notice that our method consistently perform better than a baseline with access to $\mathcal{D}_{UL}$ without space sharing or Wasserstein Matching. In particular, when the sample size is 1000 in total, Wasserstein training achieves 93% accuracy, 5% higher than the baseline. The use of a low-rank decomposed latent space for distribution matching is also new to this work (see Equation 1), see Figure 2 for the effect of changing rank. Notice that both models see plateauing effect from around rank 15; The fact that the benefit of increasing rank plateaus at the order $\sim O(10)$ seems to suggest the inherent data complexity of the task. We also perform a study of our method on CIFAR10. We implement this part exactly as stated in [11]. In particular, we use 4000 images from CIFAR10 as $\mathcal{D}$. The model we use is Wide-Resnet 28-2 [18]. We notice that both Wasserstein training methods achieve better accuracy than a model that uses pure supervised data.

Besides these, a few more detailed studies of our method are presented in the Appendix, including a visualizing of the WDWD transport plan kernel p_{ij} (Section 6.2), a more detailed ananlysis of the plateauing effect of the low-rank decomposition (Section 6.3), a visualization of the learned low-rank basis (Section 6.4), and a more detailed description of the CIFAR10 experiment with respect to other methods in the field (Section 6.1).



(a) Without Wasserstein Matching (b) With Wasserstein Matching

Figure 2: Performance of the model as we vary the rank of the low rank factorization matrix $\mathbf{B}$ (see Equation 1).

References

[1] S.-i. Amari, R. Karakida, and M. Oizumi. Information Geometry Connecting Wasserstein Distance and Kullback-Leibler Divergence via the Entropy-Relaxed Transportation Problem. *ArXiv e-prints*, September 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *ArXiv e-prints*, page arXiv:1701.07875, January 2017.

[3] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

[4] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.

[5] A. Genevay, G. Peyré, and M. Cuturi. Learning Generative Models with Sinkhorn Divergences. *ArXiv e-prints*, June 2017.

[6] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.

[7] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[8] S. Laine and T. Aila. Temporal Ensembling for Semi-Supervised Learning. *ArXiv e-prints*, October 2016.

[9] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *ArXiv e-prints*, April 2017.

[10] T. Mukherjee, M. Yamada, and T. M. Hospedales. Learning Unsupervised Word Translations Without Adversaries. In *EMNLP*, 2018.

[11] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *ArXiv e-prints*, April 2018.

[12] M. Rochan and Y. Wang. Learning Video Summarization Using Unpaired Data. *ArXiv e-prints*, May 2018.

[13] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *ArXiv e-prints*, March 2017.

[14] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. *ArXiv e-prints*, page arXiv:1711.01558, November 2017.

[15] Y.-H. H. Tsai, L.-K. Huang, and R. Salakhutdinov. Learning Robust Visual-Semantic Embeddings. *ArXiv e-prints*, March 2017.

[16] Vanya Valindria, Nick Pawlowski, Martin Rajchl, Ioannis Lavdas, Eric Aboagye, Andrea G. Rockall, Daniel Rueckert, and Ben Glocker. Multi-modal learning from unpaired images: Application to multi-organ segmentation in ct and mri. pages 547–556, 03 2018.

[17] Cédric Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer, 2009 edition, September 2008.

[18] S. Zagoruyko and N. Komodakis. Wide Residual Networks. *ArXiv e-prints*, May 2016.

Algorithm 1 Wasserstein Training (WaT)

Input: $\mathcal{D}_{UL} = \{\{\boldsymbol{x}_i\}_{i=1}^{n}\{\boldsymbol{y}_j\}_{j=1}^{m}\}$, and $\mathcal{D} = \{(\boldsymbol{x}'_i, \boldsymbol{y}'_i)\}_{i=1}^{r}$.
Intialize: $\boldsymbol{\Theta}$, $\ell \leftarrow 0$, α, γ, and ρ.
while ℓ not converged **do**:
    $\boldsymbol{h}^{\boldsymbol{x}} \leftarrow \text{Enc}_{\boldsymbol{x}}(\boldsymbol{x})$ ▷ for all $\boldsymbol{x}$ in a minibatch $\mathcal{D}_{\boldsymbol{x}}$
    $\boldsymbol{h}^{\boldsymbol{y}} \leftarrow \text{Enc}_{\boldsymbol{y}}(\boldsymbol{y})$ ▷ for all $\boldsymbol{y}$ in a minibatch $\mathcal{D}_{\boldsymbol{y}}$
    $p_{ij} \leftarrow \text{Wasserstein-Distance}(\boldsymbol{h}^{\boldsymbol{x}}, \boldsymbol{h}^{\boldsymbol{y}})$ ▷ see alg. 2
    $\ell \leftarrow \gamma \sum_{i,j} p_{ij} \|\boldsymbol{h}^{\boldsymbol{x}}_i - \boldsymbol{h}^{\boldsymbol{y}}_j\|_2^2 + \rho R(\boldsymbol{\Theta}, \mathcal{D}_{UL}) + \alpha E(\boldsymbol{\Theta}, \mathcal{D})$ ▷ cf. Eq. 2
    Update $\boldsymbol{\Theta}$ by Adam.
Output: ℓ, $\boldsymbol{\Theta}$.

Algorithm 2 Wasserstein-Distance (WDWD algorithm)

Input: $(\{\boldsymbol{h}^{\boldsymbol{x}}_i\}_{i=1}^{n'}, \{\boldsymbol{h}^{\boldsymbol{y}}_j\}_{j=1}^{m'})$
Intialize: $\phi_{ij} \leftarrow \mathbf{0}$, $\ell \leftarrow 0$, λ
while ℓ not converged **do**:
    $p_{ij} \leftarrow \text{softmax}(\phi_{ij})$
    $\ell \leftarrow \sum_{i,j} p_{ij} \|\boldsymbol{h}^{\boldsymbol{x}}_i - \boldsymbol{h}^{\boldsymbol{y}}_j\|_2^2$
    $\Delta\phi_{ij} \leftarrow -\nabla_{\phi_{ij}} \ell - \lambda\phi_{ij}$
    $\phi_{ij} \leftarrow \phi_{ij} + \Delta\phi_{ij}$
Output: ℓ, p_{ij} ▷ ℓ is the computed WD, p_{ij} is the transportation plan.

Appendix

4 Hyperparameters

We first train with Adam with $lr = 0.01$ and $decay = 0.001$ for 100 epochs and then switch to SGD with $lr = 0.0005$, $momentum = 0.9$ and $decay = 0.0001$ for 20 more epochs because we find that starting with ADAM speeds up the convergence a lot and switching to SGD with small learning rate at the end improves the performance of most models by about 2%; we also notice that the amount of improvement is independent of the model and so this technique is not used just to give our model unfair advantage. Also, we do not perform any sort of data augmentation on the supervised set to simulate a setting in which the labeled data is truly sparse.

5 Pseudo-codes

See alg. 1 for our code to perform Wasserstein training. See alg. 2 for the algorithm to perform WDWD calculation of the Wasserstein distance.

5.1 Derivation of Eq. (5)

Our modified lagriangian replaces the entropic smoothing with squared log probability. While it might look bizarre at the moment, but the intention will become clear soon:

$$L_{WDWD} = \lambda \sum_{ij} \log^2 p_{ij} + \sum_{ij} p_{ij} c_{ij} + constraint \tag{6}$$

To understand what this Lagrangian does, we first show that it encourages a maximum entropy solution, and so qualitatively by minimizing this we should get similar result to that of the Sinkhorn divergence. We write without loss of generality p_{ij} in terms of its softmax potentials $p_{ij} = Softmax(\phi_{ij})$:

$$\log p_{ij} = \phi_{ij} - \log \sum_{kl} \exp(\phi_{kl}) = \phi_{ij} - 2\log n - \log \overline{\exp(\phi_{kl})}$$

where $\overline{\exp(\phi_{kl})}$ denotes its average over k, l, and now we can apply Jensen's inequality:

$$\log p_{ij} \geq \phi_{ij} - 2\log n - \overline{\phi_{ij}}$$

but recall that the softmax probability is invariant to adding a constant in its potential and so we can subtract away the mean from the potentials and so $\overline{\phi_{ij}} = 0$, and plug into the original Lagrangian we get a variational lower bound of it:

$$L_{WDWD} \geq \lambda \sum_{ij} (\phi_{ij} - 2\log n)^2 + \sum_{ij} p_{ij} c_{ij} + constraint \tag{7}$$

and we see that the term $\lambda \sum_{ij} (\phi_{ij} - 2\log n)^2$ encourages the potentials of our transport plan to be close to the potentials of a uniform distribution, i. e. the maximum entropy distribution. Also, notice that the bound is tight exactly when p_{ij} conforms to a uniform distribution. In this sense, we expect similar qualitative behaviour when minimizing over our Lagrangian as compared to the original Lagrangian.

6 Additional Experiments

6.1 CIFAR10

In this section, we give more information about our CIFAR10 experiments. To allow for fair comparison with other models, we implement this part exactly as stated in [11]. In particular, we use 4000 images from CIFAR10 as $\mathcal{D}$. The model we use is Wide-Resnet 28-2. Please refer to [11] for exact details about the implementation. We show the performance difference between each method and a supervised baseline. However, due to time and resource constraint, we have not yet been able to reproduced the baseline performance as stated in [11], and so our result is not directly comparable to the cited results. Currently, our method is compared to our current best achieving baseline. We notice that our method is capable of achieving similar performance increase over the baseline as some of the published methods, but at this stage it is hard to conclude whether the improvement is significant. While the original paper perform 1000 grid search on each of the method, our results in this section is very preliminary and only result from 5 to 10 rounds of hyperparameter tuning. In the full paper, we will present a complete and much more detailed version of this experiment.

Method	**-**	**Testing. Acc. (%)**
Supervised	-	79.74
Π-M [8]	-	+3.89
Mean Teacher [13]	-	+4.39
VAT [9]	-	+6.40
VAT-EM [9]	-	**+7.13**
Pseudo-label [6]	-	+2.48

Ours	**Unsup. Acc.(%)**	**Testing. Acc.(%)**
Supervised	70.02%	70.81%
Sinkhorn	**+2.10**	+1.91
WDWD	+2.01	**+3.30**

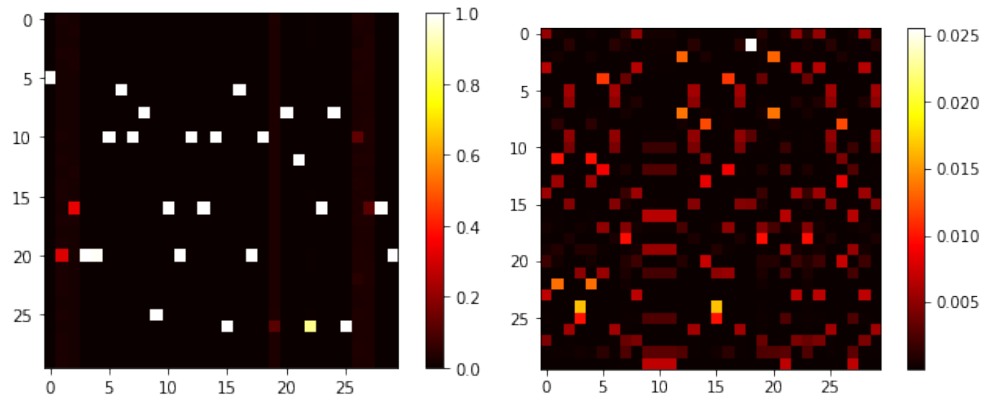
Table 3: Experiment on CIFAR10. Performance of the cited/proposed methods are shown in difference from the baseline.

6.2 Effect of different strengths of Weight Decay

In this section, we study the effect of the strength of weight decay on the Wasserstein potential matrix. We show the effect of larger strength has the effect of making the joint distribution “softer". This shows that using weight decay on the potential matrix has qualitatively the same effect as tuning the λ parameter in the Sinkhorn divergence formulation. Just like increaing λ makes the learned Sinkhorn distribution “softer", using a stronger weight decay does similar thing. See fig. 3. This shows that the weight decay formulation is qualitatively similar to the Sinkhorn algorithm formulation, in which a special parameter can be tuned to control the strength of entropic smoothing.

6.3 The Effect of Rank

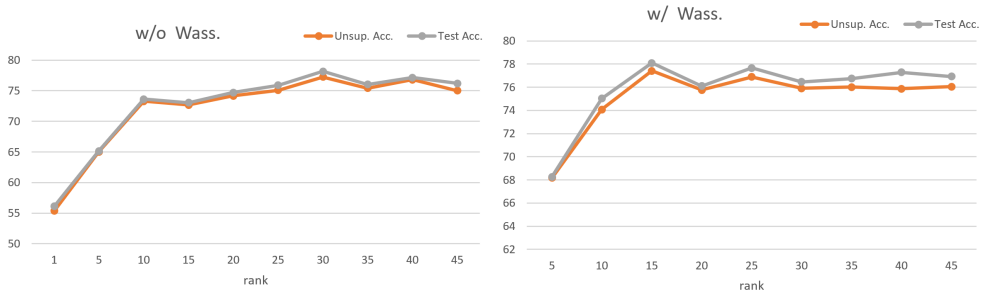
Since the rank of the latent space decomposition is a hyperparameter for our model, we explore the effect of varying the rank on the performance of the model. In fact, the result is quite intuitive to



(a) Joint distribution when the weight decay strength is 0.1

(b) when the strength is 0.9

Figure 3: Comparison of the effect of using different strength of weight decay to calculate the joint distribution matrix of a set of 25 input pictures in the optimal transport formulation. The x-axis refers to the input images and the y-axis refers to the target labels. Note that in the MNIST expriments we do not force every label to be assigned a probability mass since there is degeneracy in the label (a set of 25 target might contain multiply many same label).



(a) Without Wasserstein Matching

(b) With Wasserstein Matching

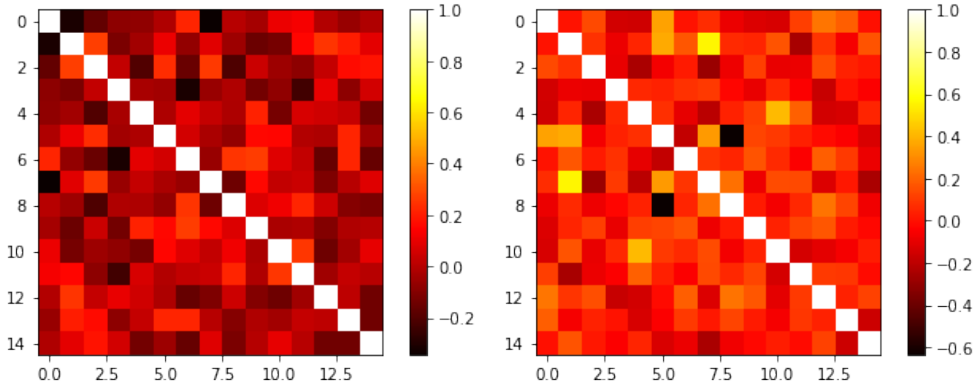
Figure 4: Performance of the model as we vary the rank of the low rank factorization matrix $\mathbf{B}$ (see Equation 1). Notice that both models see plateauing effect from around rank 15; suggesting the intrinsic rank of the task.

expect. We expect that with very low rank, the model would not have enough capacity to capture the dynamic of the task and the richness of the input and output distribution; with high rank, we would have much larger parameter to explore than necessary and so finding the optimal solution would be hard and so the performance should plateau or even decrease. For this part of the experiment, we fix the dimension of the latent space to be 50 and we vary the rank of the latent representations in this space. For the result see part (a) of 2 (w/o wasserstein distance) and part (b) of 2 (w/ wasserstein distancec), and the model works just as we expected in both cases.

Also, we find that in both settings, the performance increase stagnates around rank 10 to rank 15, suggesting at the intrinsic complexity of the task. This also agrees with what one would expect. For simple tasks such as MNIST, the number of subspaces needed to represent the data distribution should be close to the number of actual classes since there is not much intra class variation.

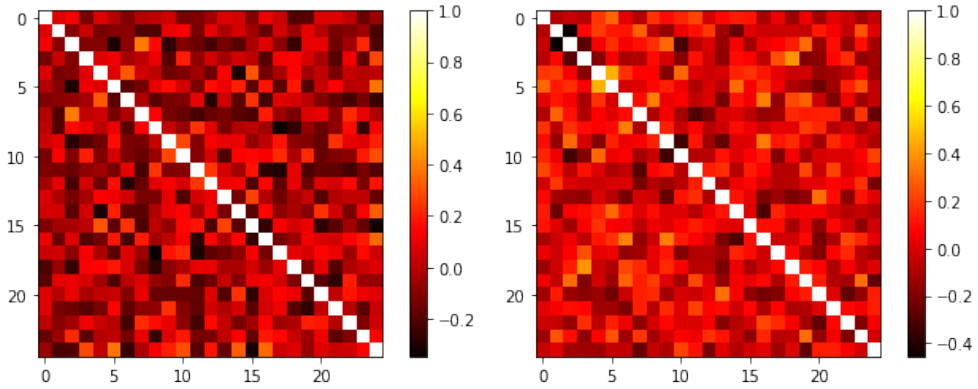
6.4 Visualization of the Learned Low Rank Basis

In this section, we visualize the learned low-rank basis of rank 15 in a 50 dimensional space. The low rank basis is initialzized with 0 mean and 0.1 variance gaussian noise. We calculate the cosine angle between the 15 basis vector and plot the heatmap in fig. 5. We hypothesize that this is because with low rank approximation, the model has learned correlation and similarity between different classes (for example, images with label 5 have greater similarity with 3 and less so with 1) and so it captures



(a) Cosine similarity of the basis before training (b) at Epoch 120

Figure 5: Heatmap for rank 15. Notice that before training the off diagonal elements are much darker, averaging around 0.13; after the training the off diagonal elemnents are brighter, with similarity averaging around 0.17.



(a) Cosine similarity of the basis before training (b) at Epoch 120

Figure 6: Heat map for rank 25. Notice that for rank 25 the difference between the two plots begin to diminish , with both cosine angle averaging around 0.15.

the interaction between the different labels better. Also, this effect diminishes as the rank is increased as we enter into the non-low-rank regime; possibly because such correlation is harder to learn with higher dimensions, and this is why rank 15 performs better than rank 25 (see fig. 2 in the preceding subsection). See fig. 6 for comparison.