

Improve SGD Training via Aligning Min-batches

Xiangrui Li¹ Deng Pan¹ Xin Li¹ Dongxiao Zhu¹

Abstract

Deep neural networks (DNNs) for supervised learning can be viewed as a pipeline of a feature extractor (i.e. last hidden layer) and a linear classifier (i.e. output layer) that is trained jointly with stochastic gradient descent (SGD). In each iteration of SGD, a mini-batch from the training data is sampled and the true gradient of the loss function is estimated as the noisy gradient calculated on this mini-batch. From the feature learning perspective, the feature extractor should be updated to learn meaningful features with respect to the entire data, and reduce the accommodation to noise in the mini-batch. With this motivation, we propose In-Training Distribution Matching (ITDM) to improve DNN training and reduce overfitting. Specifically, along with the loss function, ITDM regularizes the feature extractor by matching the moments of distributions of different mini-batches in each iteration of SGD, which is fulfilled by minimizing the maximum mean discrepancy. As such, ITDM does not assume any explicit parametric form of data distribution in the latent feature space. Extensive experiments are conducted to demonstrate the effectiveness of our proposed strategy.

1. Introduction

Recently, deep neural networks (DNNs) have achieved remarkable performance improvements in a wide range of challenging tasks in computer vision (Krizhevsky et al., 2012; He et al., 2016; Huang et al., 2019), natural language processing (Sutskever et al., 2014; Chorowski et al., 2015) and healthcare informatics (Miotto et al., 2018). Modern architectures of DNNs usually have an extremely large number of model parameters, which often outnumbers the available training data. Recent studies in theoretical deep learning have shown that DNNs can achieve good generalization even with the over-parameterization (Neyshabur et al., 2017; Olson et al., 2018). Although over-parameterization may not

be very damaging to DNN’s overall generalizability, DNNs can still overfit the noise within the training data (e.g. sampling noise in data collection) due to its highly expressive power. This makes DNNs sensitive to small perturbations in testing data, for example, adversarial samples (Goodfellow et al., 2014). To alleviate overfitting of DNNs, many methods have been proposed. These include classic ones such as early stopping, L_1 and L_2 regularization (Goodfellow et al., 2016), and more recent ones such as dropout (Srivastava et al., 2014), batch normalization (Ioffe & Szegedy, 2015) and data-augmentation types of regularization (e.g. cutout (DeVries & Taylor, 2017), shake-shake (Gastaldi, 2017)). There are also other machine learning regimes that can achieve regularization effect such as transfer learning (Pan & Yang, 2009) and multi-task learning (Caruana, 1997; Ruder, 2017).

For supervised learning, DNNs can be viewed as a feature extractor followed by a linear classifier on the latent feature space, which is jointly trained using stochastic gradient descent (SGD). When DNNs overfit the training data and a large gap between training and testing loss (e.g. cross-entropy loss for classification) is observed, from the feature learning perspective, it implies mismatching of latent feature distributions between the training and testing data extracted by the feature extractor. Regularization methods mentioned above can reduce such mismatching and hence improve DNNs performance, as the linear classifier can accommodate itself to the latent features to achieve good performance.

In this paper, we propose a different regularization method, called In-Training Distribution Matching (ITDM), that specifically aims at reducing the fitting of noise for feature extraction during SGD training. The idea behind ITDM is motivated by a simple interpretation of the mini-batch update (in addition to the approximation perspective).

Specifically, in each iteration of SGD, a mini-batch of m samples $\{(x_i, y_i)\}_{i=1}^m$ is sampled from the training data $\{(x_i, y_i)\}_{i=1}^n$ ($n > m$). The gradient of loss function $L(x, \theta)$ is calculated on the mini-batch, and network parameter θ is updated via one step of gradient descent (learning

¹Computer Science, Wayne State University. Correspondence to: Dongxiao Zhu <dzhu@wayne.edu>.

rate α):

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} L(x_i, \theta) &\approx \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} L(x_i, \theta), \\ \theta &\leftarrow \theta - \alpha \cdot \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} L(x_i, \theta). \end{aligned} \quad (1)$$

This update (Eq.(1)) can be interpreted from two perspectives. (1) From the conventional approximation perspective, the true gradient of the loss function (i.e. gradient on the entire training data) is approximated by the mini-batch gradient. As each mini-batch contains useful information for the learning tasks and its gradient computation is cheap, large DNNs can be efficiently and effectively trained with modern computing infrastructures. Moreover, theoretical studies in deep learning have shown that the noisiness in the estimated gradient using the randomly sampled mini-batch plays a crucial role in DNNs generalizability (Ge et al., 2015; Daneshmand et al., 2018). (2) **Eq. (1) can also be interpreted as an exact gradient descent update on the mini-batch.** In other words, SGD updates network parameter θ to achieve maximum improvement in fitting the mini-batch. As each mini-batch is noisy, such exact update inevitably introduces the undesirable mini-batch-dependent noise. In terms of feature learning, the DNN feature extractor can encode the mini-batch noise into the feature representations.

These two perspectives enable us to decompose the SGD update intuitively as:

Training improvement + (possible) Mini-batch overfitting.

A natural question then to ask is “*Can we reduce the mini-batch overfitting?*” to reduce the mini-batch dependence in SGD update Eq. (1). One solution to this problem is batch normalization (BN) (Ioffe & Szegedy, 2015). In their seminal paper, the internal covariate shift is observed due to the distribution shift of the activation of each hidden layer from mini-batch to mini-batch. Under our decomposition, this phenomenon is closely related to the mini-batch overfitting as networks have to adjust parameter θ to fit the mini-batches. To reduce the distribution shift, (Ioffe & Szegedy, 2015) introduces the BN layer that fixes the means and variances of activations of hidden layers.

Different from BN, the proposed ITDM directly reduces the mini-batch overfitting by matching its latent feature distribution with another mini-batch. In this paper, we only consider the feature representation from the last hidden layer. Ideally, if the distribution $P(h)$ of latent feature h is known as a prior, we could explicitly match the mini-batch feature h_{mb} with $P(h)$ via maximum likelihood. However, in practice, $P(h)$ is not known or does not even have an analytic form. To tackle this problem, we utilize the maximum mean

discrepancy (MMD) (Gretton et al., 2012) from statistical hypothesis testing for the two-sample problem. Our motivation of using MMD as the matching criterion is: *if the SGD update using one mini-batch A is helpful for DNNs learning good feature representations with respect to the entire data, then for another mini-batch B, the mismatch of latent feature distributions between A and B should not be significant.* In this way, we can reduce mini-batch overfitting by forcing accommodation of SGD update to B and reducing dependence of the network on A. In terms of model training, MMD has two advantages: (1) it enables us to avoid the presumption for $P(h)$ and (2) the learning objective of MMD is differentiable which can be jointly trained with $L(x, \theta)$ by backpropagation. Note that ITDM is not a replacement of BN. In fact, ITDM can benefit from BN when BN helps improving the feature learning for DNNs.

We summarize our contributions as follows. (1) We propose a training strategy ITDM for training DNNs. ITDM augments conventional SGD with regularization effect by additionally forcing feature matching of different mini-batches to reduce mini-batch overfitting. ITDM can be combined with existing regularization approaches and applied on a broad range of network architectures and loss functions. (2) We conduct extensive experiments to evaluate ITDM. Results on different benchmark datasets demonstrate that training with ITDM can significantly improve DNN performances, compared with conventional training strategy (i.e. perform SGD only on the loss function).

2. Related Work

In this section, we first review regularization methods in deep learning. Our work utilizes MMD and hence is also related to the topic of distribution matching, so we also review its related works that is widely studied under the context of domain adaption and generative modeling.

With limited amount of training data, training DNNs with a large number of parameters usually requires regularization to reduce overfitting. Those regularization methods include class ones such as L_1/L_2 -norm penalties and early stopping (Hastie et al., 2009; Goodfellow et al., 2016). For deep learning, many new approaches are proposed motivated by the SGD training dynamics. For example, dropout (Srivastava et al., 2014) and its variants (Gao et al., 2019; Ghiasi et al., 2018) achieves regularization effect by reducing the co-adaption of hidden neurons of DNNs. In the training process, dropout randomly sets some hidden neurons’ activation to zero, resulting in an averaging effect of a number of sub-networks. (Ioffe & Szegedy, 2015) proposes batch normalization (BN) to reduce the internal covariate shift caused by SGD. By maintaining the mean and variance of mini-batches, BN regularizes DNNs by discouraging the adaption to the mini-batches. Label smoothing (Szegedy et al., 2016)

is another regularization technique that discourage DNNs' over-confident predictions for training data. Our proposed ITDM is partially motivated by the covariate shift observed by (Ioffe & Szegedy, 2015). ITDM achieves regularization by reducing DNN's accommodation to each mini-batch in SGD as it is an exact update for that mini-batch.

To match the distribution of different mini-batches, ITDM uses MMD as its learning objective. MMD (Gretton et al., 2007; 2012) is a probability metric for testing whether two finite sets of samples are generated from the same distribution. With the kernel trick, minimizing MMD encourages to match all moments of the data empirical distributions. MMD has been widely applied in many machine learning tasks. For example, (Li et al., 2015) and (Li et al., 2017) use MMD to train unsupervised generative models by matching the generated distribution with the data distribution. Another application of MMD is for the domain adaption. To learn domain-invariant feature representations, (Long et al., 2015) uses MMD to explicitly match feature representations from different domains. Our goal is different from those applications. In ITDM, we do not seek exact distribution matching. Instead, we use MMD as a regularization to improve SGD training.

3. In-Training Distribution Matching

In this section, we first provide an introduction of maximum mean discrepancy for the two-sample problem from the statistical hypothesis testing. Then we present our proposed ITDM for training DNNs using SGD, along with some details in implementation.

3.1. Maximum Mean Discrepancy

Given two finite sets of samples $S_1 = \{x_i\}_{i=1}^n$ and $S_2 = \{y_i\}_{i=1}^m$, MMD (Gretton et al., 2007; 2012) is constructed to test whether S_1 and S_2 are generated from the same distribution. MMD compares the sample statistics between S_1 and S_2 , and if the discrepancy is small, S_1 and S_2 are then likely to follow the same distribution.

Using the kernel trick, the empirical estimate of MMD (Gretton et al., 2007) w.r.t S_1 and S_2 can be rewritten as:

$$\text{MMD}(S_1, S_2) = \left[\frac{1}{n^2} \sum_{i,j=1}^n k(x_i, x_j) + \frac{1}{m^2} \sum_{i,j=1}^m k(y_i, y_j) - \frac{2}{mn} \sum_{i=1}^n \sum_{j=1}^m k(x_i, y_j) \right]^{1/2},$$

where $k(\cdot, \cdot)$ is a kernel function. (Gretton et al., 2007) shows that if k is a characteristic kernel, then asymptotically $\text{MMD} = 0$ if and only S_1 and S_2 are generated from the same distribution. A typical choice of k is the Gaussian

kernel with bandwidth parameter σ :

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma}\right).$$

The computational cost of MMD is $O((m+n)^2)$. In ITDM, this is not problematic as typically only a small number of samples in each mini-batch (e.g. 100) is used in SGD.

3.2. Proposed ITDM

The idea of ITDM, as explained in Section 1, is to reduce the DNN adaption to each mini-batch if we view the SGD iteration as an exact update for that mini-batch. In terms of feature learning, we attempt to train the feature extractor to encode less mini-batch dependent noise into the feature representation. From the distribution point of view, the latent feature distribution of the mini-batch should approximately match with, or more loosely, should not deviate much from that of the entire data. However, matching with the entire data has some disadvantages. If MMD is used as matching criterion and training data size (say n) is large, the time complexity for MMD is not desirable (i.e. $O(n^2)$). For computational efficiency, an analytic form of the latent feature distribution can be assumed but we will be at the risk of misspecification. As such, we propose to use a different mini-batch only for latent feature matching (and not for classification loss function). As seen in the experiments, this strategy can significantly improve the performance in terms of loss values on the independent testing data.

More formally, let $f_\theta(x)$ be a convolutional neural network model for classification that is parameterized by θ . It consists of a feature extractor $h = E_{\theta_e}(x)$ and a linear classifier $C_{\theta_c}(h)$ parameterized by θ_e and θ_c respectively. Namely, $f_\theta(x) = C_{\theta_c}(E_{\theta_e}(x))$ and $\theta = \{\theta_e, \theta_c\}$. Without ambiguity, we drop θ in f , E and C for notational simplicity.

In each iteration of SGD, let $S_1 = \{(x_i^1, y_i^1)\}_{i=1}^{m_1}$ be the mini-batch of m_1 samples. Then the loss function using cross-entropy (CE) on S_1 can be written as

$$L_{mb}(\theta) = -\frac{1}{m_1} \sum_{i=1}^{m_1} \log f_{y_i^1}(x_i^1), \quad (2)$$

where $f_{y_i^1}(x_i^1)$ is the predicted probability for x_i^1 's true label y_i^1 . SGD performs one gradient descent step on L_{mb} w.r.t θ using Eq. (1).

To reduce θ 's dependence on S_1 in this exact gradient descent update, we sample from the training data another mini-batch $S_2 = \{(x_i^2, y_i^2)\}_{i=1}^{m_2}$ to match the latent feature distribution between S_1 and S_2 using MMD:

$$\begin{aligned} H_1 &= \{h_i^1 = E(x_i^1) : i = 1, \dots, m_1\}, \\ H_2 &= \{h_i^2 = E(x_i^2) : i = 1, \dots, m_2\}, \\ \text{Match}_j(\theta_e; H_1, H_2) &= \text{MMD}(H_1, H_2). \end{aligned} \quad (3)$$

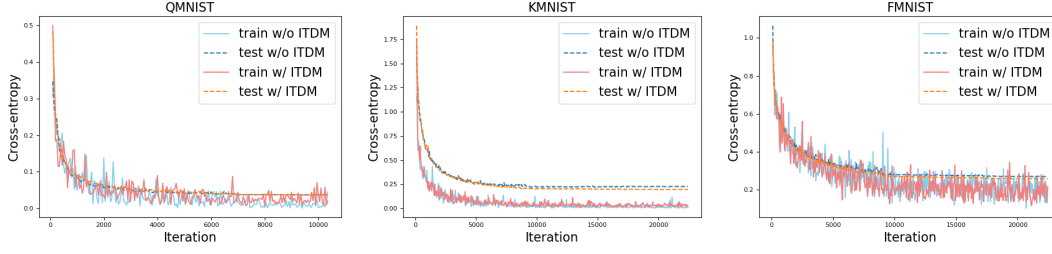


Figure 1. Training and testing cross-entropy for QMNIST, KMNIST and FMNIST.

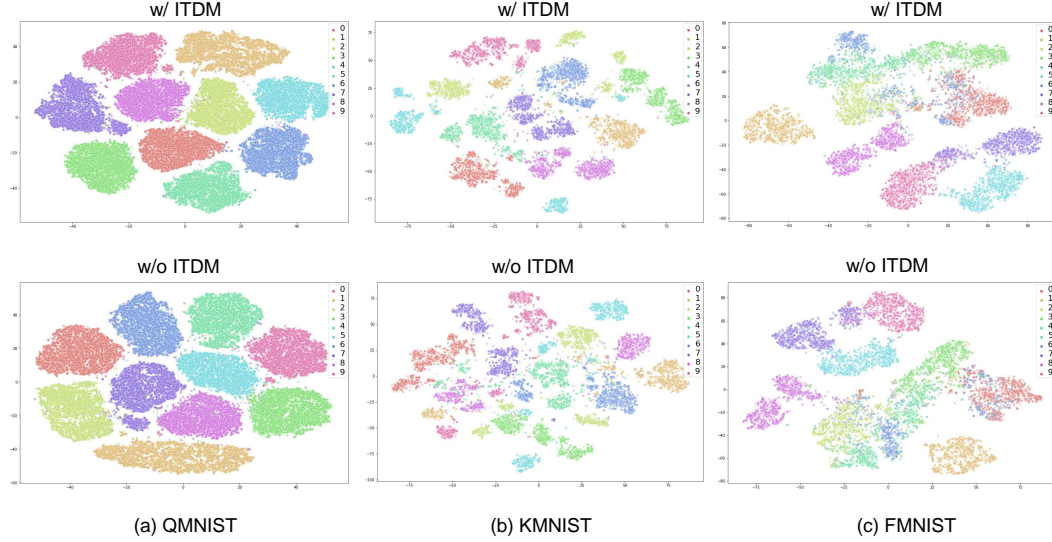


Figure 2. T-SNE (Maaten & Hinton, 2008) plot for training CNN with and without ITDM-j using cross-entropy as loss function on the standard testing data of QMNIST, KMNIST and FMNIST. Compared with the conventional training that only performs SGD on the loss function, ITDM has slightly clearer margin and boundaries in the visualization.

Our proposed ITDM modifies the conventional gradient descent step in SGD by augmenting the cross-entropy loss (Eq. (2)) with the matching loss, which justifies the name of ITDM:

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} [L_{mb}(\theta) + \lambda \text{Match}_j(\theta_e; H_1, H_2)], \quad (4)$$

where λ is the tuning parameter controlling the contribution of Match_j . We call the update in Eq. (4) as **ITDM-j** since we match the joint distribution of (x, y) without differentiating each class in the mini-batches S_1 and S_2 . Note that in ITDM-j, mini-batch S_2 is not used in the calculation of cross-entropy loss $L_{mb}(\theta)$.

Initial results using ITDM-j To test the effectiveness of ITDM, we performed initial experiments using three MNIST-type datasets: QMNIST (Yadav & Bottou, 2019), KMNIST (Clanuwat et al., 2018) and FMNIST (Xiao et al., 2017). We trained a simple CNN of two convolutional layers with and without ITDM under the exactly same setting. Experiment details are provided in the supplemental material.

Table 1. Classification accuracy (in %) and CE loss trained with and without ITDM-j, on the testing data of QMNIST, KMNIST and FMNIST. Results are reported as the average of last 10 iterations.

		QMNIST	KMNIST	FMNIST
Acc	w/ ITDM	98.94	94.42	90.79
	w/o ITDM	98.86	94.19	90.52
CE	w/ ITDM	0.037	0.196	0.257
	w/o ITDM	0.037	0.227	0.269

Table 1 shows the classification accuracy and cross-entropy loss (i.e. negative log-likelihood) on the standard testing data. Figure 1 plots the curve of cross-entropy loss for both training and testing data. It can be seen that ITDM-j achieves better results compared with the conventional training. Notably, ITDM-j has overall smaller loss on the testing data which implies the model makes correct predictions with larger probability. To verify this, we plot the T-SNE

Algorithm 1 IN-Training Distribution Matching

```

1: Input: training data  $\{(x_i, y_i)\}$ 
2: Initialization: model parameter  $\theta$ 
3: for each epoch do
4:   for each mini-batch  $S_1$  do
5:     Sample another mini-batch  $S_2$ 
6:     Calculate  $L_{mb}$  using  $S_1$  (Eq. (2))
7:     Calculate  $H_1$  and  $H_2$  for  $S_1$  and  $S_2$  (Eq. (3))
8:     Calculate  $\text{Match}_j$  or  $\text{Match}_c$  (Eq. (3) or (5))
9:     Perform ITDM update using Eq. (4) or (6)
10:   end for
11: end for
    
```

embedding of latent features in Figure 2. We see that with ITDM, the latent feature for each class has slightly clearer margin and boundaries. This implies that ITDM can help SGD converge to a better local minimum.

Class-conditional ITDM For classification tasks, we could utilize the label information and further refine the match loss as a sum of class-conditional match loss, termed as **ITDM-c** (using the notation in Eq. (3)):

$$\begin{aligned}
 H_1^k &= \{h_i^1 : y_i = k, i = 1, \dots, m_1\} (k = 1, \dots, K), \\
 H_2^k &= \{h_i^2 : y_i = k, i = 1, \dots, m_2\} (k = 1, \dots, K), \\
 \text{Match}_c(\theta_e; H_1, H_2) &= \frac{1}{K} \sum_{k=1}^K \text{Match}_j(\theta_e; H_1^k, H_2^k),
 \end{aligned} \tag{5}$$

where K is the total number of classes and $y_i = k$ the true label of sample x_i . The ITDM-c update in SGD is

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} [L_{mb}(\theta) + \lambda \text{Match}_c(\theta_e; H_1, H_2)]. \tag{6}$$

The overall training procedure of ITDM is summarized in Algorithm 1.

Compared with ITDM-j, ITDM-c has two advantages. (1) With the implicitly utilization of label information of mini-batch S_2 , ITDM-c can help DNN learn better feature representation by focusing on the in-class distribution matching. (2) With in-class matching, the computational cost for calculating MMD is reduced from $O((m_1 + m_2)^2)$ to $O(Km_H^2)$ where $m_H = \max_k \{\#H_i^k : i = 1, 2, k = 1, \dots, K\}$.

3.3. Implementation Considerations

Bandwidth parameter σ in Gaussian kernel The performance MMD as a metric of matching two samples is sensitive to the choice of the bandwidth parameter σ when Gaussian kernel is used. Since we generally do not have the prior knowledge about the latent feature distribution, we follow the practice in (Gretton et al., 2007; 2012; Long et al., 2015) that takes the heuristic of setting σ as the median squared distance between two samples. In ITDM, σ is

not prefixed but rather estimated in each iteration of SGD w.r.t two mini-batches.

We check the gradient of Gaussian kernel $k(x, y)$ to justify this choice of σ :

$$\nabla_x k(x, y) = -2k(x, y) \frac{x - y}{\sigma}. \tag{7}$$

For a fixed σ , if x and y are either close to or far from each other, $\nabla_x k(x, y)$ is small and hence provides little information in the backpropagation. By setting σ as the running median squared distance between random mini-batches, the MMD loss can automatically adapt itself and maintain useful gradient information.

It is worth mentioning that MMD with Gaussian kernel may not effectively carry gradient information for hard samples, as the latter are usually close to the decision boundary and far away from the majority of samples. The reason is due to small $\nabla_x k(x, y)$ if x and y is far from each other. To remedy this, we use a mixture of g Gaussian kernels with different ranges of σ s:

$$k_{\text{mix}}(x, y) = \frac{1}{g} \sum_{i=1}^g k_{\sigma_i}(x, y).$$

Mini-batch size When used as the training objective for distribution matching, MMD usually requires large batch-size for effective learning. For example, (Li et al., 2015) sets the batch size to 1000. However, our goal of ITDM is not for exact distribution matching, but rather as a regularization to reduce the mini-batch overfitting in SGD update. In our experiments, we set the batch size following the common practice (e.g 150) and it works well in practice without introducing many computational burdens.

4. Experiments

In this section, we evaluate the ITDM strategy and compare its performance with the vanilla SGD training on several benchmark datasets of image classification, i.e. **w/ ITDM v.s w/o ITDM**. Specifically, ITDM-c is tested as it provides implicit label information with better supervision in the training process. We implement our codes in Pytorch (Paszke et al., 2019) and utilize Nvidia RTX 2080TI GPU for computation acceleration.

4.1. Datasets

We test ITDM on four benchmark datasets Kuzushiji-MNIST (KMNIST) (Clanuwat et al., 2018), Fashion-MNIST (FMNIST) (Xiao et al., 2017), CIFAR10 (Krizhevsky et al., 2009) and STL10 (Coates et al., 2011). KMNIST and FMNIST are two gray-scale image datasets that are intended as alternatives to MNIST. Both datasets

consist of 70000 (28×28) images from 10 different classes of Japanese character and clothing respectively, among which 60000 are used for training data and the remaining 10000 for testing data. CIFAR10 is a colored image dataset of 32×32 resolution. It consists of 50000 training and 10000 testing images from 10 classes. STL10 is another colored image dataset where each image is of size 96×96 . Original STL10 has 100000 unlabeled images, 5000 labeled for training and 8000 labeled for testing. In our experiment, we only use the labeled subset for evaluation.

4.2. Implementation Details

Through all experiments, the optimization algorithm is the standard stochastic gradient descent with momentum and the loss function is cross-entropy (CE) loss. In ITDM-c, CE loss is further combined with the matching loss (Eq. (6) in each iteration.

On KMNIST and FMNIST, we build a 5-layer convolutional neural network (CNN) with batch normalization applied. Detailed architecture is provided in the supplemental material. Momentum is set to 0.5, batch size 150, number of epochs 50, initial learning rate 0.01 and multiplied by 0.2 at 20th and 40th epoch. No data augmentation is applied. For CIFAR10 and STL10, we use publicly available implementation of VGG13 (Simonyan & Zisserman, 2014), Resnet18 (He et al., 2016) and Mobilenet (Howard et al., 2017). All models are trained with 150 epochs, SGD momentum is set to 0.5, initial learning rate is 0.5 and multiplied by 0.1 every 50 epochs, batch size 150. We resize STL10 to 32×32 . For colored image datasets, we use random crop and horizontal flip for data augmentation.

In all experiments, networks are trained with vanilla SGD and ITDM SGD under the exactly same setting (learning rate, batch size et al.). For the tuning parameter λ in ITDM, we test $\{0.2, 0.4, 0.6, 0.8, 1\}$ for checking ITDM’s sensitivity to it. Note that when $\lambda = 0$, ITDM is equivalent to vanilla SGD training. For the bandwidth parameter in the mixture of Gaussian kernels, we use 5 kernel k_{σ_i} with $\{\sigma_i = 2^i \sigma_{\text{Med}} : i = 0, \dots, 4\}$. We utilize the standard train/test split given in the benchmark datasets and train all models once on the training data. Performances are evaluated on the testing data and reported as the average of the last 10 iterations.

4.3. Results¹

For predictive performance in the table, we report the best (B) and worst (W) Top-1 accuracy trained with ITDM, and their corresponding λ and cross-entropy (CE) loss values, which is equivalent to negative log-likelihood. For better comparison, the performance difference of Δ between with

Table 2. Accuracy (in %, larger is better) and CE (smaller is better) on KMNIST testing data.

	λ	ACC	Δ	CE	Δ
w/o ITDM	-	95.57	-	0.183	-
w/ ITDM (B)	0.8	95.79	0.22 \uparrow	0.170	0.013 \uparrow
w/ ITDM (W)	0.4	95.59	0.02	0.162	0.031 \uparrow

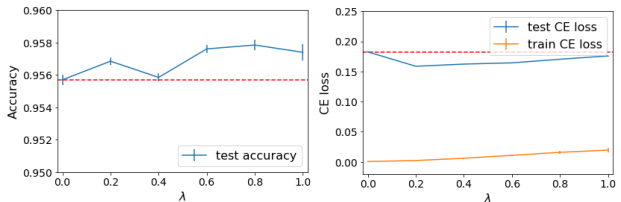


Figure 3. Accuracy and CE loss (with one standard deviation error bar) w.r.t different λ values on KMNIST. $\lambda = 0$ is equivalent to the vanilla SGD training.

and without ITDM is also reported with \uparrow indicating significant improvement from ITDM and \downarrow otherwise.

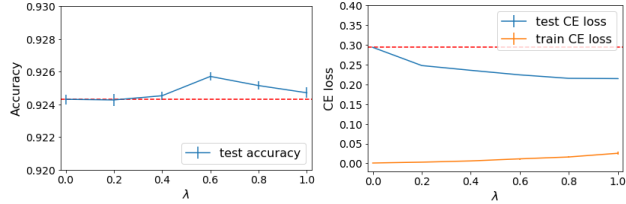
KMNIST Table 2 shows the predictive performance for KMNIST. From the table, we see that training with ITDM achieves better results in terms of accuracy and CE. Even in the worst case, ITDM has comparable accuracy with that of the vanilla SGD training. In Figure 3, we also plot the accuracy, training and testing loss (after optimization converges) against λ . From the figure, we have the following observations. (1) On KMNIST, training with ITDM is not very sensitive to λ , which at least has comparable performance in terms of accuracy, and always has smaller CE loss. As CE is equivalent to negative log-likelihood, smaller CE value implies the network makes predictions on testing data with higher confidence on average. (2) As λ increases, the training CE loss also increases. This is expected as in each iteration of ITDM, there is a tradeoff between the CE and match loss. Since a larger CE implies smaller likelihood, ITDM has a regularization effect by alleviating the over-confident predictions on training data.

FMNIST Table 3 shows the predictive performance for FMNIST. We plot in Figure 4 the accuracy, training and testing loss. As can be seen from the table and figure, ITDM generally does not damage the predictive accuracy. Similar to KMNIST, ITDM always has smaller CE values. However it does not necessarily lead to accuracy gain. The possible reason is that, FMNIST has a significant number of hard samples (e.g. those from pullover, coat and shirt classes). Though ITDM can always lead to prediction with stronger confidence, it still misses those hard samples as MMD may not be able to effectively capture their information in the

¹Complete results are provided in supplemental materials.

Table 3. Accuracy (in %, larger is better) and CE loss (smaller is better) on FMNIST testing data.

	λ	ACC	Δ	CE	Δ
w/o ITDM	-	92.43	-	0.294	-
w/ ITDM (B)	0.6	92.57	0.14 \uparrow	0.224	0.070 \uparrow
w/ ITDM (W)	0.2	92.42	0.01	0.248	0.046 \uparrow


 Figure 4. Accuracy and CE (with error bar) w.r.t different λ values on FMNIST. $\lambda = 0$ is equivalent to the vanilla SGD training.

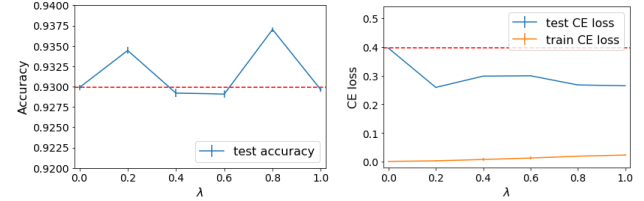
training process (Eq. (7)).

CIFAR10 In Table 4, we present the performance of Resnet18, VGG13 and Mobilenet on CIFAR10. For Resnet and Mobilenet, the overall performances of training with and without ITDM in terms of accuracy are comparable across all λ values. In particular, when λ is set with a relatively large value of 0.8 or 1, ITDM can further improve the accuracy by a margin 0.71% for Resnet and 0.82% for Mobilenet. For VGG13, training with ITDM gives higher accuracy and worse when $\lambda = 0.8$ and $\lambda = 0.2$ respectively. We also plot the CE loss for different λ s in Figure 5 w.r.t Resnet (for illustration purpose). Comparing with vanilla SGD training, we see that training with ITDM results in significant gain in CE, regardless of network architecture: Resnet 32.6%(0.129/0.396), VGG 29.4% and Mobilenet 30.6%. This pattern also holds even if ITDM does not outperform vanilla SGD training in terms of accuracy: Resnet 24.5%(0.097/0.396), VGG 25.8% and Mobilenet 17.6%. On the other hand, as λ increases, the training loss also increases. A closer gap between training and testing loss usually implies better generalization as it means a closer distribution match between train and testing data. From this perspective, ITDM can regularize DNNs to learn better feature representations with better generalizability.

STL10 The results on STL10 is shown in Table 5. Similar to CIFAR10, a larger value of λ results in higher accuracy with significant margin for ITDM, i.e., Resnet 1.9%, VGG13 1.4% and Mobilenet 2.93%, whereas a smaller value leads to performance drop, i.e., Resnet 2.47% and Mobilenet 1.23%. In terms of CE loss, ITDM always outperforms vanilla SGD training, i.e., Resnet 35.6%(0.581/1.630), VGG 39.5% and Mobilenet 20.6% (Figure 7).

Table 4. Accuracy (in %, larger is better) and CE loss (smaller is better) of Resnet18, VGG13 and Mobilenet on CIFAR10.

RESNET18					
	λ	ACC	Δ	CE	Δ
w/o ITDM	-	92.99	-	0.396	-
w/ ITDM (B)	0.8	93.70	0.71 \uparrow	0.267	0.129 \uparrow
w/ ITDM (W)	0.6	92.91	0.08 \downarrow	0.299	0.097 \uparrow
VGG13					
	λ	ACC	Δ	CE	Δ
w/o ITDM	-	92.49	-	0.473	-
w/ ITDM (B)	0.8	92.72	0.23 \uparrow	0.334	0.139 \uparrow
w/ ITDM (W)	0.2	92.34	0.15 \downarrow	0.351	0.122 \uparrow
MOBILENET					
	λ	ACC	Δ	CE	Δ
w/o ITDM	-	88.55	-	0.615	-
w/ ITDM (B)	1.0	89.37	0.82 \uparrow	0.427	0.188 \uparrow
w/ ITDM (W)	0.2	88.76	0.21 \uparrow	0.507	0.108 \uparrow


 Figure 5. Resnet18 accuracy and CE (with error bar) w.r.t different λ values on CIFAR10.

4.4. Analysis

Through the extensive experimental results across a broad range of datasets, we observe that ITDM with larger λ values tends to have better performances when compared with smaller λ values, and outperforms the vanilla SGD training. Since we use ITDM-c in the experiments, a plausible reason for this phenomenon is that ITDM-c provides implicit supervision in the learning process by matching two random, noisy mini-batches from the *same* class. With larger λ s, ITDM can benefit from the stronger implicit supervision and hence improve network performance.

Another phenomenon is that ITDM can reduce the testing CE loss significantly, in particular for CIFAR10 and STL10 datasets. Given a sample $(x, y = k)$, its CE loss is calculated as $-\log f_k(x)$, where f_k is the predicted probability for x 's true class label k . A smaller CE value implies a larger probability f_k . From the geometric perspective, samples from the same class should stay close and those from different classes are expected to stay far apart in the feature space

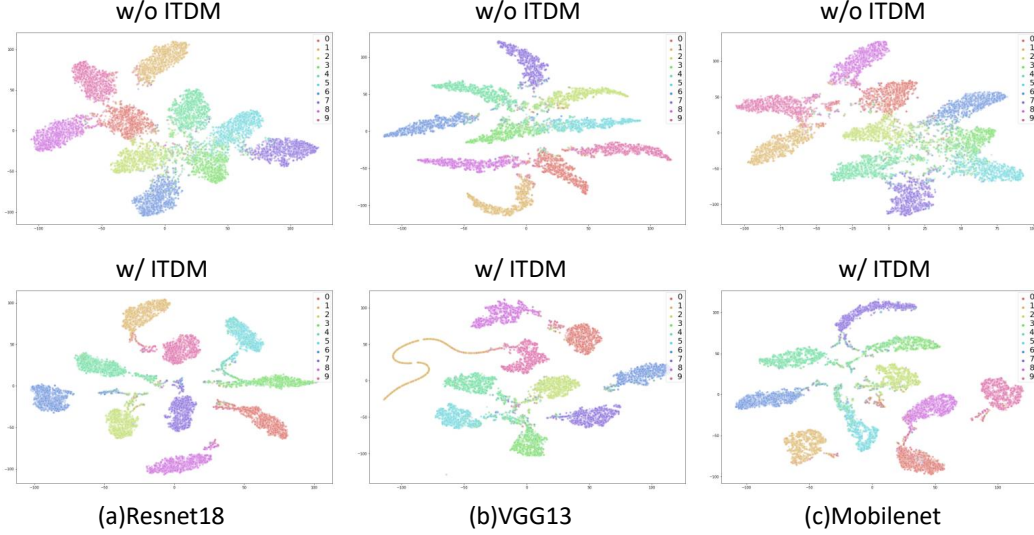


Figure 6. T-SNE plot for CIFAR10 testing data. Networks are trained with λ that achieves best accuracy in Table 4.

Table 5. Accuracy (in %, larger is better) and CE loss (smaller is better) of Resnet18, VGG13 and Mobilenet on STL10 testing data.

RESNET18					
	λ	Acc	Δ	CE	Δ
w/o ITDM	-	70.88	-	1.630	-
w/ ITDM (B)	0.6	72.78	1.90 \uparrow	1.049	0.581 \uparrow
w/ ITDM (W)	0.8	71.29	0.41 \uparrow	1.048	0.582 \uparrow
VGG13					
	λ	Acc	Δ	CE	Δ
w/o ITDM	-	74.40	-	1.545	-
w/ ITDM (B)	0.8	75.80	1.40 \uparrow	0.934	0.611 \uparrow
w/ ITDM (W)	0.4	74.46	0.06	1.110	0.435 \uparrow
MOBILENET					
	λ	Acc	Δ	CE	Δ
w/o ITDM	-	59.09	-	2.144	-
w/ ITDM (B)	0.6	62.02	2.93 \uparrow	1.603	0.541 \uparrow
w/ ITDM (W)	0.8	58.93	0.16 \downarrow	1.638	0.506 \uparrow

(so that f_k output by softmax is large). To confirm this, we visualize the distribution of CIFAR10 testing samples with T-SNE (Maaten & Hinton, 2008) in Figure 6. From the figure, ITDM learns feature representation that is much tighter with clearer inter-class margin than that learned by vanilla SGD training. We also can gain some insight on why ITDM achieves impressive improvement in CE loss but not as much in accuracy: For each class, ITDM effectively captures the “typical pattern” of each class and the majority of samples are hence clustered closely, but ITDM also

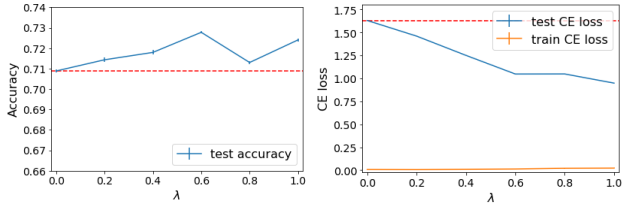


Figure 7. Resnet18 accuracy and CE loss (with error bar) w.r.t different λ values on STL10.

misses some hard samples that overlap with other classes. Overall, ITDM still outperforms vanilla SGD training and can be used as a promising training prototype that is capable of learning more discriminative features.

5. Conclusion

In this paper, we propose a new training strategy with regularization effect, ITDM, as an alternative to vanilla SGD training. ITDM augments vanilla SGD with a matching loss that uses MMD as the objective function. By forcing the matching of two different mini-batches, ITDM reduces the possible mini-batch overfitting in vanilla SGD. Experimental results demonstrate its excellent performance on classification tasks, as well as its impressive feature learning capacity. There are two possible directions for our future studies. The first one is to improve ITDM that can learn hard sample more effectively. The second one is potential ITDM application in learning from poisoned datasets as ITDM tends to capture the major pattern in the dataset.

References

- Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.
- Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pp. 577–585, 2015.
- Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature, 2018.
- Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011.
- Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. Escaping saddles with stochastic gradients. *arXiv preprint arXiv:1803.05999*, 2018.
- DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
- Gao, H., Pei, J., and Huang, H. Demystifying dropout. In *The 36th International Conference on Machine Learning (ICML 2019)*, 2019.
- Gastaldi, X. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.
- Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle point online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.
- Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 10727–10737, 2018.
- Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.
- Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. J. A kernel method for the two-sample problem. In *Advances in neural information processing systems*, pp. 513–520, 2007.
- Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.
- Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L., and Weinberger, K. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pp. 2203–2213, 2017.
- Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In *International Conference on Machine Learning*, pp. 1718–1727, 2015.
- Long, M., Cao, Y., Wang, J., and Jordan, M. I. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.
- Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.
- Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. T. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.
- Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

- Olson, M., Wyner, A., and Berk, R. Modern neural networks generalize on small data sets. In *Advances in Neural Information Processing Systems*, pp. 3619–3628, 2018.
- Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2009.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.
- Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
- Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- Yadav, C. and Bottou, L. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems*, pp. 13443–13452, 2019.