A. Relation to Denoising and Contractive Auto-encoders

Our method is related to denoising auto-encoders (Vincent et al., 2008). Auto-encoders maximize a lower bound of mutual information (Cover & Thomas, 2012) between inputs and their hidden representations (Vincent et al., 2008), while the denoising mechanism regularizes the auto-encoders to be locally invariant. However, such a regularization does not necessarily impose the invariance on the *hidden representations* because the decoder network also has the flexibility to model the invariance to data perturbations. SAT is more direct in imposing the intended invariance on hidden representations predicted by the encoder network.

Contractive auto-encoders (Rifai et al., 2011) directly impose the local invariance on the encoder network by minimizing the Frobenius norm of the Jacobian with respect to the weight matrices. However, it is empirically shown that such regularization attained lower generalization performance in supervised and semi-supervised settings than VAT, which regularizes neural networks in an end-to-end fashion (Miyato et al., 2016). Hence, we adopted the end-to-end regularization in our unsupervised learning. In addition, our regularization, SAT, has the flexibility of modeling other types invariance such as invariance to affine distortion, which cannot be modeled with the contractive regularization. Finally, compared with the auto-encoders approaches, our method does not require learning the decoder network; thus, is computationally more efficient.

B. Penalty Method and its Implementation

Our goal is to optimize the constrained objective of Eq. (10):

$$\min_{\theta} \mathcal{R}_{\mathrm{SAT}}(\theta; T) + \lambda H(Y|X),$$
$$\text{subject to } \mathrm{KL}[p_\theta(y) || \, q(y)] \leq \delta.$$

We use the penalty method (Bertsekas, 1999) to solve the optimization. We introduce a scalar parameter μ and consider minimizing the following unconstrained objective:

$$\mathcal{R}_{\mathrm{SAT}}(\theta; T) + \lambda H(Y|X) + \mu \max\{\mathrm{KL}[p_\theta(y) || \, q(y)] - \delta, 0\}. \tag{19}$$

We gradually increase μ and solve the optimization of Eq. (19) for a fixed μ. Let μ^* be the smallest value for which the solution of Eq. (19) satisfies the constraint of Eq. (10). The penalty method ensures that the solution obtained by solving Eq. (19) with $\mu = \mu^*$ is the same as that of the constrained optimization of Eq. (10).

In experiments in Section 4.2, we increased μ in the order of $\lambda, 2\lambda, 4\lambda, 6\lambda, \ldots$ until the solution of Eq. (19) satisfied the constraint of Eq. (10).

C. On the Mini-batch Approximation of theMmarginal Distribution

The mini-batch approximation can be validated for the clustering scenario in Eq. (10) as follows. By the convexity of the KL divergence (Cover & Thomas, 2012) and Jensen's inequality, we have

$$\mathbb{E}_{\mathcal{B}}[\mathrm{KL}[\widehat{p_\theta}^{(\mathcal{B})}(y) || q(y)]] \geq \mathrm{KL}[p_\theta(y) || q(y)] \geq 0, \tag{20}$$

where the first expectation is taken with respect to the randomness of the mini-batch selection. Therefore, in the penalty method, the constraint on the exact KL divergence, i.e., $\mathrm{KL}[p_\theta(y) || \, q(y)] \leq \delta$ can be satisfied by minimizing its upper bound, which is the approximated KL divergence $\mathbb{E}_{\mathcal{B}}[\mathrm{KL}[\widehat{p_\theta}^{(\mathcal{B})}(y) || q(y)]]$. Obviously, the approximated KL divergence is amenable to the mini-batch setting; thus, can be minimized with SGD.

D. Implementation Detail

We set the size of mini-batch to 250, and ran 50 epochs for each dataset. We initialized weights following He et al. (2015): each element of the weight is initialized by the value drawn independently from Gaussian distribution whose mean is 0, and standard deviation is $scale \times \sqrt{2/fan_{in}}$, where fan_{in} is the number of input units. We set the $scale$ to be 0.1-0.1-0.0001 for weight matrices from the input to the output. The bias terms were all initialized with 0.

E. Datasets Description

- **MNIST**: A dataset of hand-written digit classification (LeCun et al., 1998). The value of each pixel was transformed linearly into an interval [-1, 1].

- **Omniglot**: A dataset of hand-written character recognition (Lake et al., 2011), containing examples from 50 alphabets ranging from well-established international languages. We sampled 100 types of characters from four alphabets, Magi, Anglo-Saxon Futhorc, Arcadian, and Armenian. Each character contains 20 data points. Since the original data have high resolution (105-by-105 pixels), each data point was down-sampled to 21-by-21 pixels. We also augmented each data point 20 times by thestochastic affine distortion explained in Appendix F.

- **STL**: A dataset of 96-by-96 color images acquired from labeled examples on ImageNet (Coates et al., 2010). Features were extracted using 50-layer pre-trained deep residual networks (He et al., 2016) available online as a caffe model. Note that since the residual network is also trained on ImageNet, we expect that each class is separated well in the feature space.

- **CIFAR10**: A dataset of 32-by-32 color images with ten object classes, which are from the Tiny image dataset (Torralba et al., 2008). Features were extracted using the 50-layer pre-trained deep residual networks (He et al., 2016).

- **CIFAR100**: A dataset 32-by-32 color images with 100 refined object classes, which are from the Tiny image dataset (Torralba et al., 2008). Features were extracted using the 50-layer pre-trained deep residual networks (He et al., 2016).

- **SVHN**: A dataset with street view house numbers (Netzer et al., 2011). Training and test images were both used. Each image was represented as a 960-dimensional GIST feature (Oliva & Torralba, 2001).

- **Reuters**: A dataset with English news stories labeled with a category tree (Lewis et al., 2004). Following DEC (Xie et al., 2016), we used four categories: corporate/industrial, government/social, markets, and economics as labels. The preprocessing was the same as that used by Xie et al. (2016), except that we removed stop words. As Xie et al. (2016) did, 10000 documents were randomly sampled, and *tf-idf* features were used.

- **20news**: A dataset of newsgroup documents, partitioned nearly evenly across 20 different newsgroups[2]. As Reuters dataset, stop words were removed, and the 2000 most frequent words were retained. Documents with less than ten words were then removed, and *tf-idf* features were used.

For the STL, CIFAR10 and CIFAR100 datasets, each image was first resized into a 224-by-224 image before its feature was extracted using the deep residual network.

F. Affine Distortion for the Omniglot Dataset

We applied stochastic affine distortion to data points in Omniglot. The affine distortion is similar to the one used by Koch (2015), except that we applied the affine distortion on down-sampled images in our experiments. The followings are the stochastic components of the affine distortion used in our experiments. Our implementation of the affine distortion is based on scikit-image[3].

- Random scaling along x and y-axis by a factor of (s_x, s_y), where s_x and s_y are drawn uniformly from interval $[0.8, 1.2]$.
- Random translation along x and y-axis by (t_x, t_y), where t_x and t_y are drawn uniformly from interval $[-0.4, 0.4]$.
- Random rotation by θ, where θ is drawn uniformly from interval $[-10^\circ, 10^\circ]$.
- Random shearing along x and y-axis by (ρ_x, ρ_y), where ρ_x and ρ_y are drawn uniformly from interval $[-0.3, 0.3]$.

Figure. 3 shows examples of the random affine distortion.

[2]http://qwone.com/~jason/20Newsgroups/
[3]http://scikit-image.org/

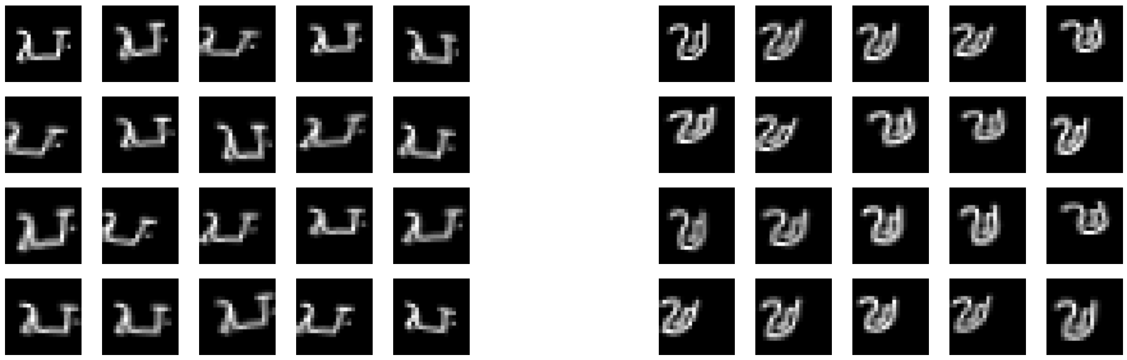

Figure 3. Examples of the random affine distortion used in our experiments. Images in the top left side are stochastically transformed using the affine distortion.

Table 6. Comparison of hash performance for 32-bit hash codes (%). Averages and standard deviations over ten trials were reported. Experimental results of Deep Hash and the previous methods are excerpted from Erin Liong et al. (2015).

Method	Hamming ranking (mAP)		precision @ sample = 500		precision @ r = 2	
(Network dimensionality)	MNIST	CIFAR10	MNIST	CIFAR10	MNIST	CIFAR10
Spectral hash (Weiss et al., 2009)	25.7	12.4	61.3	19.7	65.3	20.6
PCA-ITQ (Gong et al., 2013)	43.8	16.2	74.0	25.3	73.1	15.0
Deep Hash (80-50)	45.0	16.6	74.7	26.0	73.3	15.8
Linear RIM	29.7 (0.4)	**21.2 (3.0)**	68.9 (0.9)	16.7 (0.8)	60.9 (2.2)	15.2 (0.9)
Deep RIM (80-50)	34.8 (0.7)	14.2 (0.3)	72.7 (2.2)	24.0 (0.9)	72.6 (2.1)	23.5 (1.0)
Deep RIM (200-200)	36.5 (0.8	14.1 (0.2)	76.2 (1.7)	23.7 (0.7)	75.9 (1.6)	23.3 (0.7)
Deep RIM (400-400)	37.0 (1.2)	14.2 (0.4)	76.1 (2.2)	23.9 (1.3)	75.7 (2.3)	23.7 (1.2)
IMSAT (VAT) (80-50)	55.4 (1.4)	20.0 (5.5)	87.6 (1.3)	23.5 (3.4)	88.8 (1.3)	22.4 (3.2)
IMSAT (VAT) (200-200)	62.9 (1.1)	18.9 (0.7)	96.1 (0.6)	29.8 (1.6)	95.8 (0.4)	**29.1 (1.4)**
IMSAT (VAT) (400-400)	**64.8 (0.8)**	18.9 (0.5)	**97.3 (0.4)**	**30.8 (1.2)**	**96.7 (0.6)**	**29.2 (1.2)**

G. Hyper-parameter Selection

In Figure 4 we report the experimental results for different hyper-parameter settings. We used Eq. (21) as a criterion to select hyper-parameter, β^*, which performed well across the datasets.

$$\beta^* = \arg\max_{\beta} \sum_{\text{dataset}} \frac{\text{ACC}(\beta, \text{dataset})}{\text{ACC}(\beta^*_{\text{dataset}}, \text{dataset})}, \tag{21}$$

where β^*_{dataset} is the best hyper-parameter for the dataset, and $\text{ACC}(\beta, \text{dataset})$ is the clustering accuracy when hyper-parameter β is used for the dataset. According to the criterion, we set 0.005 for decay rates in both Linear RIM and Deep RIM. Also, we set $\lambda = 1.6, 0.05$ and 0.1 for Linear IMSAT (VAT), IMSAT (RPT) and IMSAT (VAT), respectively.

H. Experimental Results on Hash Learning with 32-bit Hash Codes

Table 6 lists the results on hash learning when 32-bit hash codes were used. We observe that IMSAT with the largest network sizes (400-400) exhibited competitive performance in both datasets. The performance of IMSAT improved significantly when we used slightly larger networks (200-200), while the performance of Deep RIM did not improve much with the larger networks.

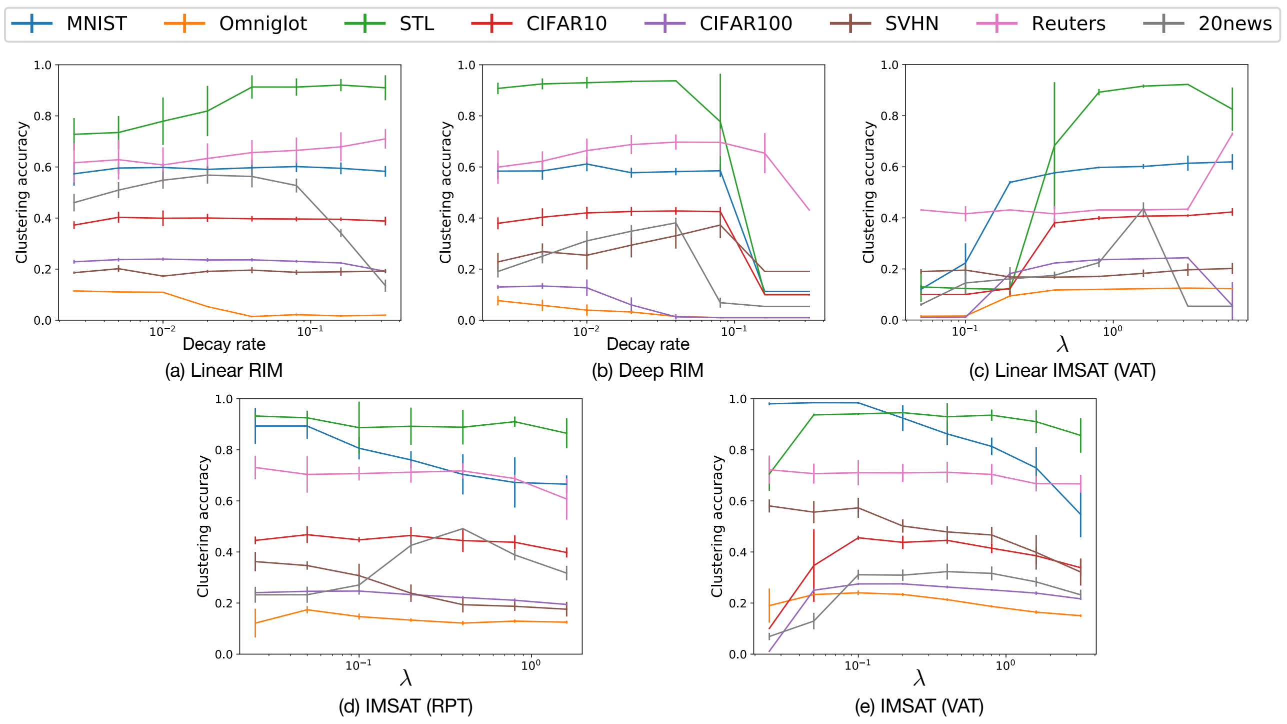

Figure 4. Relationship between hyper-parameters and clustering accuracy for 8 benchmark datasets with different methods: (a) Linear RIM, (b) Deep RIM, (c) Linear IMSAT (VAT), (d) IMSAT (RPT), and (e) IMSAT (VAT).

I. Comparisons of Hash Learning with Different Regularizations and Network Sizes Using Toy Dataset

We used a toy dataset to illustrate that IMSAT can benefit from larger networks sizes by better modeling the local invariance. We also illustrate that weight-decay does not benefit much from the increased flexibility of neural networks.

For the experiments, we generated a spiral-shaped dataset, each arc containing 300 data points. For IMSAT, we used VAT regularization and set $\epsilon = 0.3$ for all the data points. We compared IMSAT with Deep RIM, which also uses neural networks but with weight-decay regularization. We set the decay rate to 0.0005. We varied three settings for the network dimensionality of the hidden layers: 5-5, 10-10, and 20-20.

Figure 5 shows the experimental results. We see that IMSAT (VAT) was able to model the complicated decision boundaries by using the increased network dimensionality. On the contrary, the decision boundaries of Deep RIM did not adapt to the non-linearity of data even when the network dimensionality was increased. This observation may suggest why IMSAT (VAT) benefited from the large networks in the benchmark datasets, while Deep RIM did not.

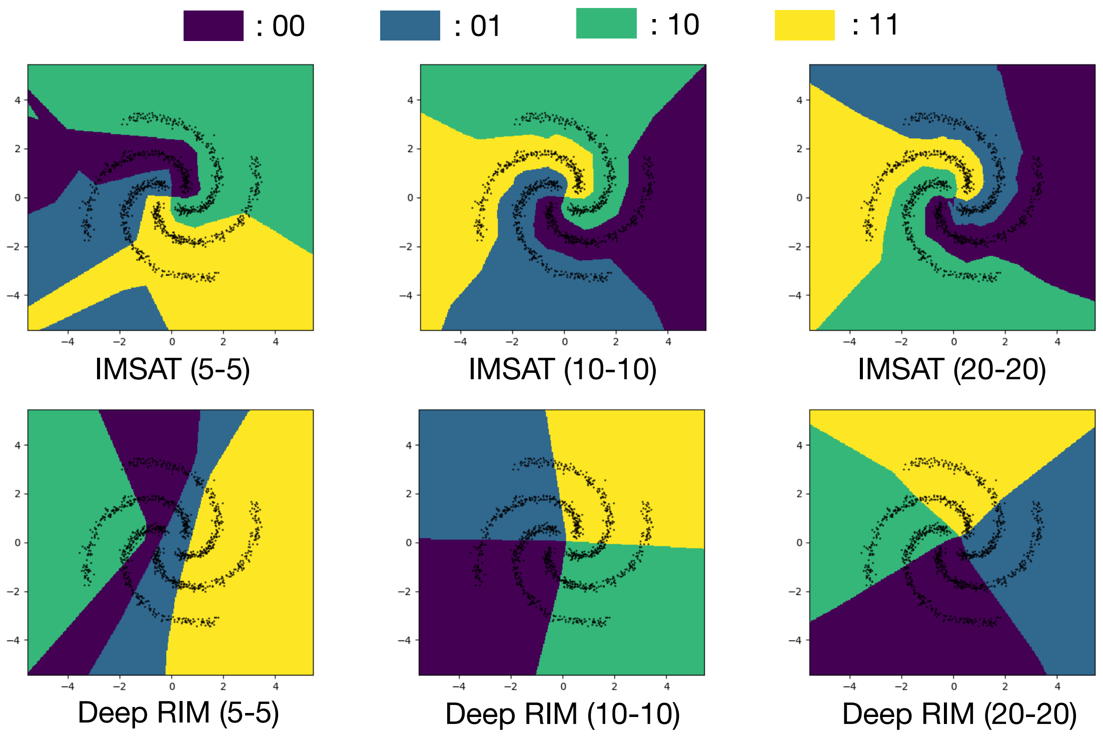

Figure 5. Comparisons of hash learning with the different regularizations and network sizes using toy datasets.