ONE STAGE AUTOENCODERS FOR MULTI-DOMAIN LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoencoders (AEs) are widely being used for representation learning. Empirically AEs are capable of capturing hidden representations of a given domain precisely. However, in principle AEs latent representation might be misleading, especially in the presence of weak encoding constraints. In this paper, we introduce one stage autoencoders (OSAs) to induce searching for patterns while training artificial neural networks. We propose two different frameworks for OSAs; Autoclave Restricted Boltzmann Machines (ACRBMs) and Local Observer Convolution (LOC). Both frameworks are compatible with artificial neural networks and trained via direct backpropagation (end-to-end training). Furthermore, they are scalable and require significantly less number of parameters than traditional AEs. ACRBMs are extensions of RBMs that are able to describe a given domain symmetrically. LOC is a density based clustering algorithm that implicitly draws a spatial graph from input domains. Unlike standard clustering algorithms that require specifying the expected number of clusters, we believe that LOC is the first neural network compatible algorithm capable of dynamically choosing the appropriate number of clusters that best fit a given domain. Both ACRBMs and LOC were evaluated in terms of unsupervised learning. Experiments showed that both structures of shallow ACRBMs and AE-based ACRBMs outperformed K-means for image clustering using the same number of clusters. Similarly, LOC outperformed K-means in terms of unsupervised image segmentation.

1 INTRODUCTION

Computationally, learning process can be coordinated as a conjunction of pattern searching within prior information and pattern generation by exploiting latent representation or an understanding.

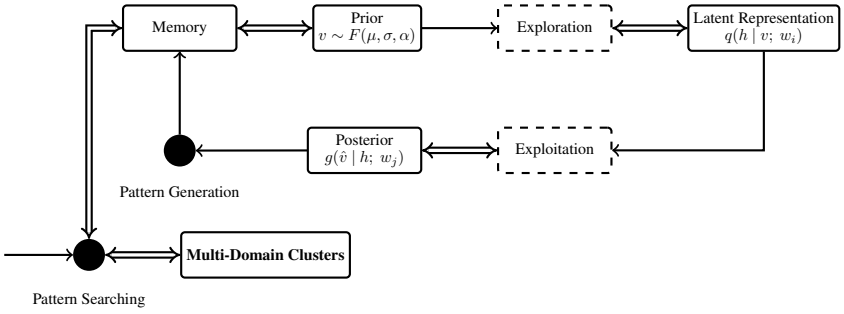


Figure 1: Computational graph for learning process.

As indicated in figure 1, the prior $v \sim F(\mu, \sigma, \alpha)$, follows an arbitrary distribution F with mean μ and a covariance σ, including a domain specific parameter α that works as a domain indicator. The latent representation is inferred by a q function for computing hidden state h given a visible state v. Pattern generation function g exploits the hidden state h, to get the predicted state $\hat{v}$. The functions q and g are optionally parameterized by w_i and w_j respectively.

In machine learning, generative models are commonly being used for patterns generation, however generators incorporated with pattern searching process cannot be trivially obtained. Typically searching process cannot be defined as a continuous function, making it incompatible with gradient based optimization methods. In this paper, we show that a restricted latent representation guarantees a definite exploration that constructs an approximated patterns to imitate inputs domain precisely.

Like stochastic approaches, generative models leverage inputs domain to induce more exploration without introducing too much randomness, however, a large number of parameters could easily lead to undefined exploration behavior or random walks Radhakrishnan et al. (2018). Historically generative models including Hidden Markov model (HMM) (Tran et al., 2016), Bayesian network (Rohekar et al., 2018), Restricted Boltzmann Machines (RBMs) (Salakhutdinov et al., 2007), Deep Boltzmann Machines (DBMs) (Salakhutdinov & Hinton, 2009), Autoencoders (AEs) (Jarrett & van der Schaar, 2020) are indispensable for features-extraction. Likewise, they are commonly used as a base structure for many deep learning problems, such as dimensionality reduction , word embeddings and image segmentation.

In recent years, generative models trained via backpropagation, such as AEs, have been favored as being compatible with other Artificial Neural Networks (ANNs). To approach the point of junction for generative models, we differentiate between RBMs and shallow AEs from a theoretical perspective, reformulating and analyzing their mathematical model.

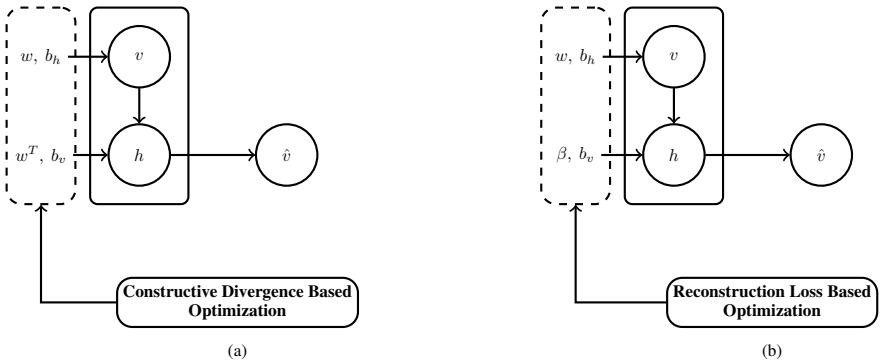


Figure 2: Computational illustration for RBMs (a) and AEs (b).

Figure 2 illustrates a computational representation for RBMs (a) and AEs (b) such that w, β, b_h and b_v are model parameters, while v, h and $\hat{v}$ represent states. Each of RBMs and AEs explore the domain of the visible state v, by utilizing the parameters w and b_h. The exploration process output h is indicated as the latent representation, following equation 1. The predicted state $\hat{v}$ follows equations 2 and 3 for RBMs and AEs respectively. It is obtained by exploiting the latent representation h based on w^T parameter for RBMs, β parameter for AEs, and bias parameter b_v for both. Functions σ_h and σ_v are the nonlinearity activations, e.g., sigmoid function.

$$h = \sigma_h(vw + b_h) \tag{1}$$

$$\hat{v} = \sigma_v(hw^T + b_v) \tag{2}$$

$$\hat{v} = \sigma_v(h\beta + b_v) \tag{3}$$

A frequent problem of AEs is the mutually exclusive encoder-decoder, also known as the exploration-exploitation trade-off. Exploration states that the latent representation h implies new information about input domains, i.e., the more exploration it does, the more interesting features we get. However, unconstrained exploration preserves high entropy which means searching in an infinite space for the latent representation. Therefore, the exploitation is being used to transform the infinite space into a finite one restricted by the neural-network settings. Variational Autoencoder (VAE) (Kingma & Welling, 2014) was slightly able to overcome this problem, nevertheless, VAE

still suffer from the lack of self-consistency problem Cemgil et al. (2020). This paper focuses on the encoder representation in terms of simplicity and explanatory power.

To make searching for a latent vector computationally possible, the searching space must be finite. A remarkable feature of RBMs is that the latent variables are implicitly constrained by how the reconstruction is being computed, and how algorithms such as constructive divergence (CD) (Yuille, 2005) operate relying on this nature. RBMs are able to encode and explain a given domain symmetrically as a one-to-one mapping between the latent variable and the generated sample. This guarantees that the exploration-exploitation trade-off will be approximately zero, however, there is no guarantee that doing much exploration is possible anymore. Unlike AEs, RBMs do not have a smooth reconstruction stage, which makes the optimization process a bit messy, and requires careful settings with many considerations while training. In addition, untimely RBMs fail especially when the ANN compatibility gets into the picture.

In conclusion, it can be considered that AEs outperformed RBMs in terms of unlimited exploration, nevertheless, the trainable decoder, makes the trade-off between the exploration and exploitation out of control. Despite that, if both AEs and RBMs are feasible, it is not strictly fair to say that AEs outperformed RBMs as they are not comparable. Theoretically RBMs still have advantages over AEs, in terms of encoder-decoder symmetry.

As indicated by equations 1, 2 and 3, the key difference between RBMs and AEs is the inferential step of the predicated state. To elaborate the inferential step main differences, we will just consider the affine functions, where most parameterization is being used, neglecting bias parameters and nonlinearity activations (equations 4, 5 and 7).

$$h = vw \tag{4}$$

$$\boxed{\hat{v} = hw^T} \tag{5}$$

$$\boxed{\hat{v} = h\beta} \tag{6}$$

RBMs follow what we call parameter-load, while AEs follow what we call parameter-stress. A parameter is called loaded, if it has many dependencies, in which none of them could be trivially met. This makes the parameter self-regularized, e.g., the same parameter w is used into multiple inferential steps (equations 4 and 5). The parameter-stress could be interpreted in analogous manner, as different parameters w and β are being used for each inferential step, which implies a wider searching space (equations 4 and 6). RBMs implicit restrictions can be shown by reformulating equation 5 into 7, hence, an optimal reconstruction is obtained iff $\boxed{W = I}$, which implies that w is implicitly constrained to be semi-orthogonal.

$$\begin{aligned} &\hat{v} = vww^T, \\ &\hat{v} = v(ww^T) = vW \end{aligned} \tag{7}$$

In order to resolve the exploration-exploitation problem of RBMs and AEs, this paper proposes One Stage Autoencoders (OSAs), with two different ANN compatible frameworks including Autoclave Restricted Boltzmann Machines (ACRBMs) as an RBMs extension, and the Local Observer Convolution (LOC). The major structure of ACRBMs does not require reconstruction criteria as the reconstruction mutually depends on the optimization process of the latent variables. Regardless the existence of the reconstruction criteria, ACRBMs can describe a given domain symmetrically, which means they operate as real generators. Regarding LOC, it implicitly draws a spatial graph from inputs domain, which alerts searching for patterns while training, by synchronizing the pattern searching and pattern generation in a self-regularized way. Since the LOC optimization process requires reconstruction criteria, the generative models based on LOC, are not considered as real generators.

Section 2 exhibits graphical and mathematical representations of ACRBMs and LOC, including their main properties. Section 3 presents baseline models' performance for AE-based ACRBMs and LOC compared to K-means in terms of unsupervised learning.

2 ONE STAGE AUTOENCODERS

The proposed OSAs frameworks are straightforward to train, maintaining the compatibility with other ANNs. They are mainly designed to control AEs' exploration behavior. The number of parameters being used to train OSA-based models are significantly less than AE/VAE.

2.1 AUTOCLAVE RESTRICTED BOLTZMANN MACHINES

ACRBMs nearly have the same forward pass as RBMs, rather what makes ACRBMs an ANN compatible is that the model does not relay on a reconstruction-based optimization. Still the reconstruction can be computed, simulating the encoder-decoder in context.

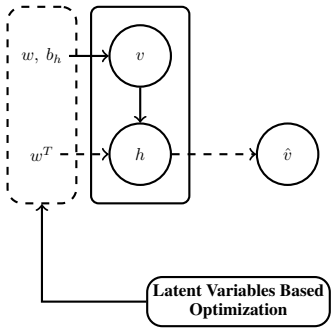


Figure 3: Computational illustration for ACRBMs.

According to figure 3, ACRBMs forward pass follows equations 8 and 9, where h and $\hat{v}$, are ACRBMs hidden and predicted state, respectively

$$h = \sigma_h(vw + b_h) \tag{8}$$

$$\hat{v} = \sigma_v(hw^T) \tag{9}$$

For a smoothly feasible optimization process, the activation function σ_h has been chosen to be a softmax function. Let the optimal reconstruction v^*, given that the nonlinear function σ_v is invertible, then

$$hw^T = \sigma_v^{-1}(v^*) \tag{10}$$

By squaring the left and right hand sides

$$\begin{aligned} (hw^T)^T(hw^T) &= \sigma_v^{-1}(v^*)^T\sigma_v^{-1}(v^*) \\ w(h^Th)w^T &= \sigma_v^{-1}(v^*)^T\sigma_v^{-1}(v^*) \end{aligned} \tag{11}$$

and for a semi-orthogonal hidden states, $\boxed{u = h^Th = I}$, it reduces to

$$ww^T = \sigma_v^{-1}(v^*)^T\sigma_v^{-1}(v^*) \tag{12}$$

The solution to ACRBMs can be expressed as following

$$\eta_i = \frac{u_i}{\sum_{j=1}^{n} u_j}, \quad \text{for } i = 1, \ldots, n \quad \text{and } u = (u_1, \ldots, u_n) \in \mathbb{R}^{n\times n} \tag{13}$$

$$\textbf{min.} \quad -\frac{1}{n}\sum_{i=1}^{n} log(diag(\eta)) \tag{14}$$

The major assumption about ACRBMs states that scaling a latent vector h_i by a constant factor c explicitly adds non-significant variances. Contrarily, orthogonal vectors require a non-trivial transformation, maintaining a considerable variance with a different representation. ACRBMs maximize independence within input examples by imposing orthogonality, i.e., the most dissimilar examples will be encoded as orthogonal ones. Consequently, the orthogonalization will be satisfied for n-samples of h, at which the most dissimilarity occurs in the Euclidean space.

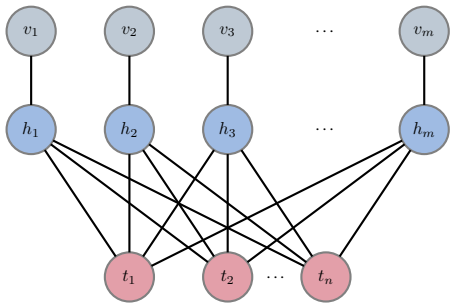
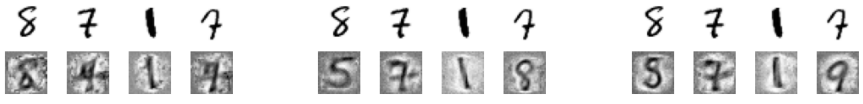


Figure 4: ACRBM probabilistic model (latent dimension size = n).

ACRBMs implicitly inherits some properties of Markov random fields (MRF) (Hamilton et al., 2017), including its graphical representation in abstract form. In figure 4, v_i and h_i are the visible and hidden states for each example i. Initially every hidden state h_i claims that transition probabilities are equal for all transition states t. In order to reach the equilibrium state where all hidden states are uniformly partitioned into n states $(t_1, t_2, \ldots, t_n)$, ACRBM assumes that every hidden state should be assigned to a single transition state. Such lemma can be interpreted as a solution of the following optimization problem

$$\frac{\mathbf{max.} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p(h_i = t_j) \times p(h_i = t_j)}{\mathbf{min.} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} p(h_i = t_j) \times p(h_i = t_k)} \tag{15}$$

In fact the optimization problem (equation 15) is considered a probabilistic reformulation of equation 14, which approximates the empirical distribution that best supports hidden states (observations). For small latent dimension size, ACRBMs transition probabilities tend to be strictly 0 or 1. The maximization term (numerator) asserts that t_j with high confidence is always preferred, therefore the term $p(h_i = t_j) \times p(h_i = t_j)$ represents the probability of h_i starting at t_j and ending at t_j (i.e. no transition). The dominator is a minimization over joint probability summation.



(a) Latent dimension size = 30 (b) Latent dimension size = 100 (c) Latent dimension size = 300

Figure 5: Reconstruction examples by shallow ACRBM for different latent representation sizes.

Figure 5 shows reconstruction examples (2nd row) of randomly sampled Mnist (LeCun et al., 2010) test set examples (1st row). It illustrates that the reconstruction quality is better for a larger latent dimension size, including the capability to generate a neighbor representation of the given input. This is arising, due to implicit equality constraints (equation 12). Yet, in practice these constraints cannot be exactly satisfied. In such case the exact form of equation 12 should be

$$wDw^T = \sigma_v^{-1}(v^*)^T \sigma_v^{-1}(v^*) \tag{16}$$

Where $D = h^T h$ tends to be a sparse square matrix with a non-zero diagonal elements. Theoretically, at reaching the optimal of ACRBMs' objective, D should be a diagonal matrix which implies that ACRBMs' parameters describe relationships between input units v_{ij} globally (the interior of the domain) and locally (domain cluster). This is implicitly similar to Markov chain (Lhritier, 2020),

e.g., ACRBM might consider a reconstructed unit (pixel) as same as input unit which is described by a zero-transition probability.

In figure 6, cluster neighbors are obtained by nearest neighbor search (NNS) on the latent representation of a query image (1st column). It shows that ACRBMs are capable of interpreting neighbors in a rational way even when using less dimensions, e.g., four is interpreted as a nine without loop. This recommends that a reasonably less dimensionality is preferred in case of using ACRBMs within generative adversarial networks (GANs) (Goodfellow et al., 2014).

(a) ACRBM latent dimension = 100 (b) ACRBM latent dimension = 300

Figure 6: Shallow ACRBM cluster neighbors in order.

2.2 Local Observer Convolution

Analogously, LOC implicit restrictions are attained by having spatial conflicts, which are developed by a set of spectral gram matrices G, referred as the observer kernel (see figure 7).

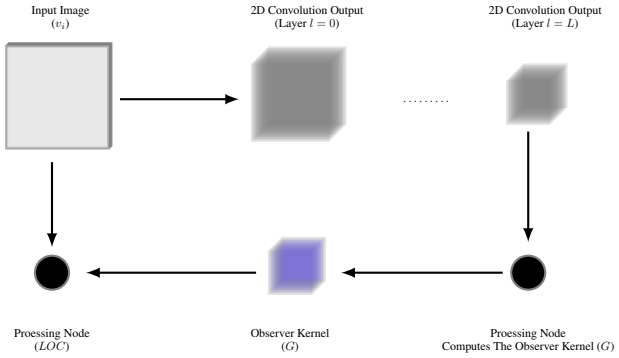


Figure 7: Computational illustration for LOC.

The G_i matrix is a fake observer, who started these conflicts, and then has been employed to resolve them. This game is inspired by the two-way partitioning problem (Schreiber et al., 2018), which can be described by the following mathematical model, where S is the feasible set corresponds to the partition

$$\begin{aligned} &\textbf{min.} \quad w^T v w \\ &\textbf{subject to} : w_i^2 = 1, \quad \text{for } i = 1, \ldots, n \\ &S = \{i \mid w_i = 1\} \cup \{i \mid w_i = -1\} \end{aligned} \tag{17}$$

To reformulate the constrained partitioning problem, into unconstrained one, LOC disposed the parameterization, using the local features a_i, at layer L, approximating the inner product, by the following kernel function

$$K(v_i, a_i) = v_i * a_i \tag{18}$$

A generalization of the previous convolution to an arbitrary number of dimensions, is obtained by repeatedly stacking different versions of a_i. Experiments show that it is possibly sufficient to repeat the same a_i, for tasks such as unsupervised segmentation. We found that many formulations with different properties can be obtained to optimize the fake observer. This simply may include ones such as mean squared error (MSE), which has been chosen in our experiments as the major objective for optimizing the observer. LOC with MSE objective is described by the following minimization problem, which includes an explicitly defined lower bound ρ as a hyper-parameter (equation 21).

$$K(v_i, G_i) = v_i * G_i \tag{19}$$

$$\epsilon = \quad \frac{1}{k}\sum_{i=1}^{k}(v_i - K(v_i, G_i))^2 \tag{20}$$

$$G_{loss} = \begin{cases} \epsilon & if\ \epsilon > \rho \\ 0 & otherwise. \end{cases} \tag{21}$$

$$\mathbf{min}. \quad G_{loss} \tag{22}$$

The ρ hyper-parameter works as a perturbation parameter to control the LOC reductions, i.e., a large perturbation, preserves large partitioning factor. As a result of using MSE objective, we end up with floating points outputs, that can be clipped in case of LOC has been used for clustering. This ends up with statistically correlated cluster labels that follow a standard labeling, e.g., there is no cluster with a label 10 if the total number of clusters are exactly 9.

In case of the problem of image pixels' clustering, a hypothetical solution to the LOC problem would be to form groups of pixels, assigning a representative r_{jk} for each group (e.g., group median seems to be reasonable given the objective function). Still, we believe that more interesting representatives could be found using adversarial attacks on G while training.

3 Experimental Results

The performance of our proposed OSAs frameworks has been evaluated in terms of multi-domain clustering. The settings which have been followed are interchangeable. We restricted the major experiments to a simple ANN modeling in order to elaborate the feasibility of ACRBMs and the LOC in practice (see also appendix B and C for the baseline models' architectures and used hyper-parameters).

3.1 ACRBMs Evaluations

In the following sets of experiments, the performance of the proposed shallow ACRBM and AE-based ACRBM architectures have been evaluated against K-means and AE-based K-means for the problem of unsupervised image clustering. Mnist dataset (LeCun et al., 2010) has been used for evaluating the clustering performance where the original splits were maintained, as $60K$ and $10K$ for train and test sets respectively.

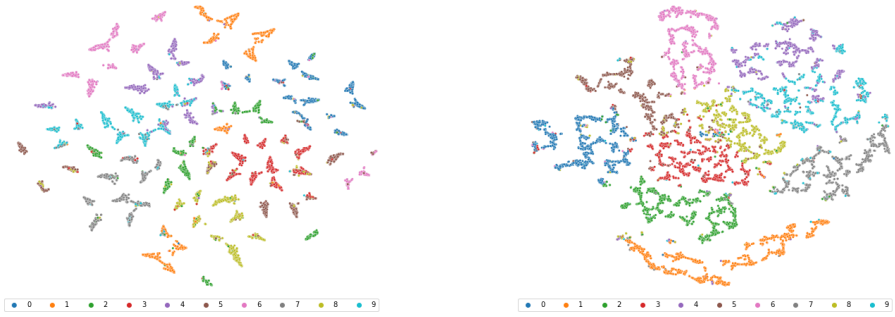
ACRBMs have been trained without considering a reconstruction criterion, either as AE-based or shallow ANN model. The cluster label of an example i is obtained based on ACRBM hidden state h_i activation (energy), such that the index of the most activated dimension j is being assigned as a cluster label (for $j = 1, \ldots, n$). Mapping clusters to ground truth labels is obtained by mapping the

Table 1: Unsupervised clustering results on Mnist test set

	#Clusters	#Parameters	Accuracy
K-means	10	-	57.32
AE + K-means	10	$+200K$	79.56
Shallow ACRBM	10	$+23K$	61.44
Shallow ACRBM	30	$+23K$	76.95
AE + ACRBM	10	$+197K$	86.80 ± 1.06
AE + ACRBM	30	$+850K$	92.86 ± 0.89

whole cluster label c_i for each example v_i to the corresponding ground truth label y_i that minimizes the total cluster entropy, (see appendix A equation 23 for computing average accuracy).

As reported in table 1, for 10 clusters the classification accuracy of shallow ACRBM is significantly better than K-means, and similarly AE-based ACRBM outperformed AE-based K-means. Results indicated that a substantial performance gain is obtained by increasing the number of clusters.



(a) latent dimentions size = 100

(b) latent dimentions size = 1000

Figure 8: A 2D projection of shallow ACRBMs latent representation.

Figure 8, shows 2D projection of shallow ACRBMs latent representation with different sizes. It illustrates that by increasing latent dimension size (number of clusters) the distortion decreases significantly. We observed that large number of clusters serve as a smoothing parameter for ACRBMs, in which more restrictions are being defined to the semi-orthogonalization objective, and frustrating ACRBMs confidence about clusters assignments (figure 9), see also Müller et al. (2019).

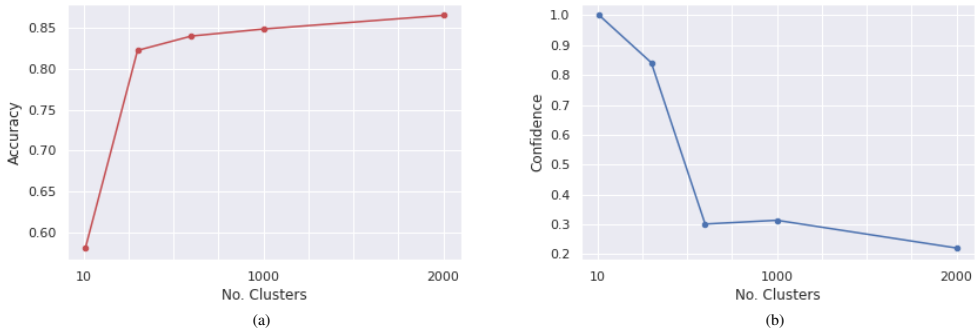


Figure 9: The trade-off between accuracy, confidence, and number of clusters.

3.2 LOC EVALUATIONS

The proposed LOC has been experimented in terms of unsupervised segmentation compared with K-means. The flowers dataset (Nilsback & Zisserman, 2010) has been used for evaluations. We randomly resampled a test set of size 2189 examples, and the remaining were used for training.

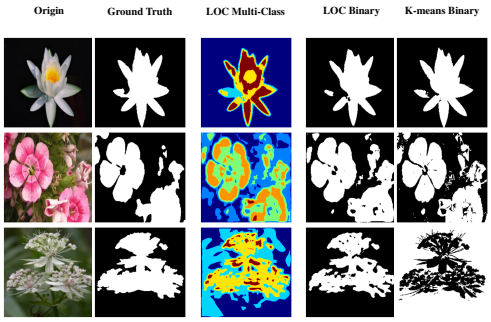


Figure 10: Flowers segmentation examples for LOC and K-means.

Figure 10 shows randomly sampled results of LOC in terms of multi-class segmentation and binary segmentation. Relying on the statistical properties of LOC with MSE objective, we were able to approximate the multi-class segmentation to binary segmentation according to the median cluster label for each example separately. The first 50% of sorted clusters labels have been mapped to zero cluster (background), while others assigned to one cluster (foreground).

Table 2: Unsupervised segmentation results on flowers

	Accuracy	IoU	DICE
K-means	61.48	52.23	55.02
LOC	81.22	61.93	71.80

As shown in table 2 the unsupervised segmentation performance of LOC is significantly better than K-means, see also appendix A for evaluation metrics.

Unlike standard unsupervised learning algorithms, LOC has many extra advantages, these includes:

1. The number of clusters are begin determined automatically by the ANN proportional to the perturbation parameter, i.e., each image may have been segmented with a different number of clusters.
2. The underlying structure is simple and compatible with ANN.
3. The latent representation (observer kernel) of LOC mainly reflects clustering results.

4 CONCLUSIONS

To address the non-descriptive learning problem of AEs, we have investigated the learning representation under implicit constraints, and its impact on pattern searching process. The proposed formulations of ACRBMs and LOC have shown their capability of working as discrete searching processes in an ANN compatible form. Our proposed frameworks provide baseline settings to imitate input domain which has been demonstrated for unsupervised learning tasks. In addition to the discrete searching capability of ACRBM, we believe that it has a scalable potential for operating as real generators, especially by being incorporated with GANs to automatically sample from feature representations via reconstruction. LOC has been designed to be incorporated with ANN to make use of earlier stage layers gradients as feedback. It instructs searching for patterns within the local domain (features) while being conditionally dependent on global domain (examples).

REFERENCES

B. Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5:622–633, 2012.

A. Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy Dvijotham, Sven Gowal, and Pushmeet Kohli. Autoencoding variational autoencoder. *CoRR*, abs/2012.03715, 2020. URL https://arxiv.org/abs/2012.03715.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Linus Hamilton, Frederic Koehler, and Ankur Moitra. Information theoretic properties of markov random fields, and their algorithmic applications. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/8fb5f8be2aa9d6c64a04e3ab9f63feee-Paper.pdf.

Daniel Jarrett and Mihaela van der Schaar. Target-embedding autoencoders for supervised representation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BygXFkSYDH.

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Alix Lhritier. Pcmc-net: Feature-based pairwise choice markov chains. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgWE1SFwS.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019. URL http://arxiv.org/abs/1906.02629.

Maria-Elena Nilsback and Andrew Zisserman. Delving deeper into the whorl of flower segmentation. *Image Vision Comput.*, 28(6):10491062, June 2010. ISSN 0262-8856. doi: 10.1016/j.imavis.2009.10.001. URL https://doi.org/10.1016/j.imavis.2009.10.001.

Adityanarayanan Radhakrishnan, Mikhail Belkin, and Caroline Uhler. Downsampling leads to image memorization in convolutional autoencoders. *CoRR*, abs/1810.10333, 2018. URL http://arxiv.org/abs/1810.10333.

R. Y. Rohekar, Shami Nisimov, G. Koren, Yaniv Gurwicz, and Gal Novik. Constructing deep neural networks by bayesian network structure learning. In *NeurIPS*, 2018.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL https://proceedings.mlr.press/v5/salakhutdinov09a.html.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *In Machine Learning, Proceedings of the Twenty-fourth International Conference (ICML 2004). ACM*, pp. 791–798. AAAI Press, 2007.

Ethan L. Schreiber, Richard E. Korf, and Michael D. Moffitt. Optimal multi-way number partitioning. *J. ACM*, 65(4), July 2018. ISSN 0004-5411. doi: 10.1145/3184400. URL https://doi.org/10.1145/3184400.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Unsupervised neural hidden markov models. *CoRR*, abs/1609.09007, 2016. URL http://arxiv.org/abs/1609.09007.

Alan L Yuille. The convergence of contrastive divergences. In L. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/75e33da9b103b7b91dcd8da0abe1354b-Paper.pdf.

A Evaluation Metrics

$$A(y,\hat{y}) = \begin{cases} \frac{\sum_{i=1}^{m} \mathbf{1}_{\{\mathrm{y_i} = \hat{\mathrm{y}}_\mathrm{i}\}}}{m} & y,\hat{y} \in \mathbb{R}^m \\ \frac{\sum_{i=1}^{m}\sum_{j=1}^{h}\sum_{k=1}^{w} \mathbf{1}_{\{\mathrm{y_{ijk}} = \hat{\mathrm{y}}_\mathrm{ijk}\}}}{m \times h \times w} & y,\hat{y} \in \mathbb{R}^{m\times h\times w} \end{cases} \tag{23}$$

$$J(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m} \frac{\sum_{j=1}^{h}\sum_{k=1}^{w} \mathbf{1}_{\{\mathrm{y_{ijk}} \wedge \hat{\mathrm{y}}_\mathrm{ijk}\}}}{\sum_{j=1}^{h}\sum_{k=1}^{w} \mathbf{1}_{\{\mathrm{y_{ijk}} \vee \hat{\mathrm{y}}_\mathrm{ijk}\}}}, \quad y,\hat{y} \in \mathbb{R}^{m\times h\times w} \tag{24}$$

$$S(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m} \frac{2 \times \sum_{j=1}^{h}\sum_{k=1}^{w} \mathbf{1}_{\{\mathrm{y_{ijk}} \wedge \hat{\mathrm{y}}_\mathrm{ijk}\}}}{\sum_{j=1}^{h}\sum_{k=1}^{w} y_{ijk} + \hat{y}_{ijk}}, \quad y,\hat{y} \in \mathbb{R}^{m\times h\times w} \tag{25}$$

The average accuracy (Accuracy), intersection-over-union (IoU) and Sϕrensen-Dice coefficient (DICE) metrics follows equations 23, 24 and 25 respectively. The ground truth is indicated by y_i, while $\hat{y}_i$ represents the predicted state. If $y, \hat{y} \in \mathbb{R}^m$, then y_i and $\hat{y}_i$ are considered labels, otherwise they are considered binary masks (i.e. $y, \hat{y} \in \mathbb{R}^{m\times h\times w}$).

B ACRBMs Implementation Details

The architecture illustrated in figure 11 is the one used for evaluating AE-based ACRBMs performance. Both encoder and decoder are composed of two convolution layers, with ReLU activation function. It is composed of 2 layers for both of encoder and decoder, with ReLU activation function. Encoder is followed by AE latent representation h_{AE} layer with a linear activation. The h_{AE} layer is attached to ACRBM latent representation h_{ACRBM}. Both of AE and ACRBM latent dimensions have been merged and passed to the decoder.

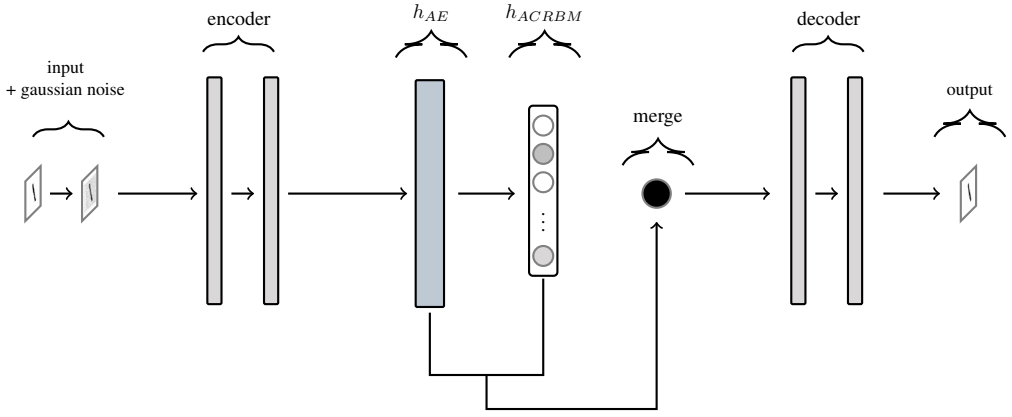


Figure 11: ACRBM within AE architecture.

The same architecture has been used for AE-based K-means experiment, except for the h_{ACRBM} layer which has been replaced by a dense layer with 10 hidden units and ReLU activation function.

Table 3 states the hyper-parameters that have been used for ACRBM major experiments. We ran 500 K-means initializations Bahmani et al. (2012) on AE latent representation, where the one that achieved the maximum accuracy has been selected.

Table 3: ACRBM hyperparameters

	#Clusters	#Hidden Units	Optimizer	#Epochs	Batch Size
AE + K-means	10	Encoder = 100 Latent dimensions = 100 Decoder = 100	Adam $\beta_1 = 0.05$ $\beta_2 = 0.999$ $\alpha = 10^{-3}$	500	256
Shallow ACRBM	10	-	Adam $\beta_1 = 0.01$ $\beta_2 = 0.999$ $\alpha = 10^{-2}$	20	256
Shallow ACRBM	30	-	Adam $\beta_1 = 0.01$ $\beta_2 = 0.999$ $\alpha = 10^{-2}$	20	256
AE + ACRBM	10	Encoder = 100 Latent dimensions = 90 Decoder = 100	Adam $\beta_1 = 0.05$ $\beta_2 = 0.999$ $\alpha = 10^{-3}$	500	256
AE + ACRBM	30	Encoder = 300 Latent dimensions = 300 Decoder = 300	Adam $\beta_1 = 0.1$ $\beta_2 = 0.999$ $\alpha = 10^{-3}$	500	256

C LOC Implementation Details

Architecture illustrated in figure 12 has been used for evaluating LOC performance. The convolution blocks composed of $2D$ convolution layer followed by batch normalization and ReLU activation function. The parameters k, s and nf represents kernel size, strides and number of filters respectively. The hyper-parameters that have been used for LOC experiment are shown in table 4.

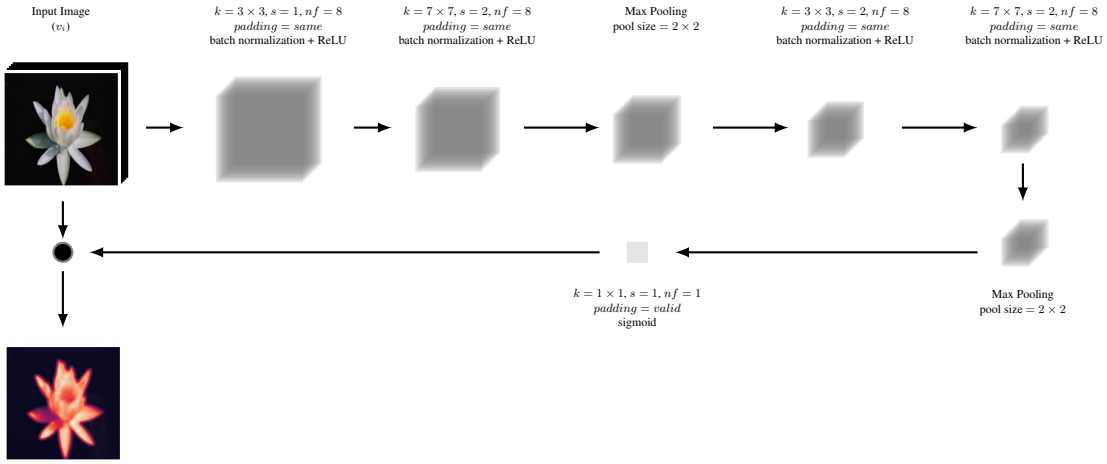


Figure 12: LOC auto-segmentation architecture.

Table 4: LOC hyperparameters

	ρ	Optimizer	#Epochs	Batch Size
LOC	0.5	Adam $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\alpha = 10^{-3}$	50	64