MIXTURE MODEL AUTO-ENCODERS: DEEP CLUSTERING THROUGH DICTIONARY LEARNING

Alexander Lin[⋆] *Andrew H. Song*[†] *Demba Ba*[⋆]

[⋆]School of Engineering and Applied Sciences, Harvard University, Boston, MA, USA
[†]Massachusetts Institute of Technology, Cambridge, MA, USA

ABSTRACT

State-of-the-art approaches for clustering high-dimensional data utilize deep auto-encoder architectures. Many of these networks require a large number of parameters and suffer from a lack of interpretability, due to the black-box nature of the auto-encoders. We introduce *Mixture Model Auto-Encoders* (MixMate), a novel architecture that clusters data by performing inference on a generative model. Built on ideas from *sparse dictionary learning* and *mixture models*, MixMate comprises several auto-encoders, each tasked with reconstructing data in a distinct cluster, while enforcing sparsity in the latent space. Through experiments on various image datasets, we show that MixMate achieves competitive performance versus state-of-the-art deep clustering algorithms, while using orders of magnitude fewer parameters.

Index Terms— deep clustering, auto-encoder, dictionary learning, mixture model, sparsity

1. INTRODUCTION

Clustering is a fundamental task for dividing data into groups without supervised labels. *Deep clustering* is a recent line of work that leverages deep neural networks to improve clustering for high-dimensional data, such as images. One line of research uses a single auto-encoder to project data into a low-dimensional space that is friendly for simple clustering algorithms [1, 2]. Another line of work employs a mixture-of-experts approach [3], where K different auto-encoders partition the dataset into K parts [4, 5]. While more accurate than simple clustering algorithms, these frameworks typically have black-box architectures that lack interpretability and require a large number of parameters.

Model-based deep learning is an emergent methodology for marrying the interpretability of signal processing models with the learning efficiency and representational power of neural networks [6]. It has shown success in various imaging applications [7, 8, 9, 10]. The model-based approach starts with a generative model of data, which enables the incorporation of domain knowledge. The unfolding of classical optimization algorithms then leads to multi-layer neural networks for conducting inference. This connection enables the use of deep learning tools (e.g. backpropagation, graphics processing units) for accelerated learning [11], while maintaining the interpretability of the original model.

We introduce *Mixture Model Auto-Encoders* (MixMate), a novel architecture for *model-based deep clustering* of images. It is derived from a mixture of sparse dictionary learning models, inspired by the wealth of evidence for the sparsity of natural images with respect to suitable dictionaries [12, 13]. MixMate comprises K different auto-encoders with an attention module, all of which are parameterized by a set of cluster-specific dictionaries. We train the network with a loss function from the classical Expectation-Maximization algorithm [14]. These properties enable MixMate to achieve superior clustering performance on benchmark image datasets with far fewer parameters. MixMate also exhibits other benefits, such as an interpretable architecture, a simpler initialization scheme, and the ability to cluster incomplete data.[1]

2. GENERATIVE MODEL

Given a collection of high-dimensional data, *sparse dictionary learning* posits that each data point $\boldsymbol{y} \in \mathbb{R}^M$ can be represented as a sparse combination $\boldsymbol{x} \in \mathbb{R}^D$ of global atoms $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_D \in \mathbb{R}^M$ arranged as columns of a *dictionary* $\boldsymbol{A} \in \mathbb{R}^{M \times D}$. Collectively, $\boldsymbol{A}$ and the sparsity of $\boldsymbol{x}$ define a highly constrained subspace to explain $\boldsymbol{y}$.

If a dataset can be partitioned into K natural clusters (e.g. $K = 10$ digit identities in a handwritten digits dataset), it is reasonable to assume that each cluster lies in a different subspace. In this context, we assume that objects belonging to cluster $k \in \{1, \ldots, K\}$ are generated by a cluster-specific dictionary $\boldsymbol{A}_k$. The relative size of each cluster is encoded into the prior $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K] \in [0, 1]^K$, with $\sum_{k=1}^K \pi_k = 1$. Putting these together, we employ the following *sparse dictionary learning mixture model* as the generative process for each element $\boldsymbol{y}$ in a dataset $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$,

$$\begin{aligned} z &\sim \text{Categorical}(\boldsymbol{\pi}), \\ \boldsymbol{x} &\sim \text{Laplace}(\lambda) \propto \exp(-\lambda \|\boldsymbol{x}\|_1), \\ \boldsymbol{y} \mid \boldsymbol{x}, z = k &\sim \mathcal{N}(\boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{I}) \propto \exp(-\|\boldsymbol{y} - \boldsymbol{A}_k \boldsymbol{x}\|_2^2). \end{aligned} \quad (1)$$

[1] Our code can be found at https://github.com/al5250/mixmate

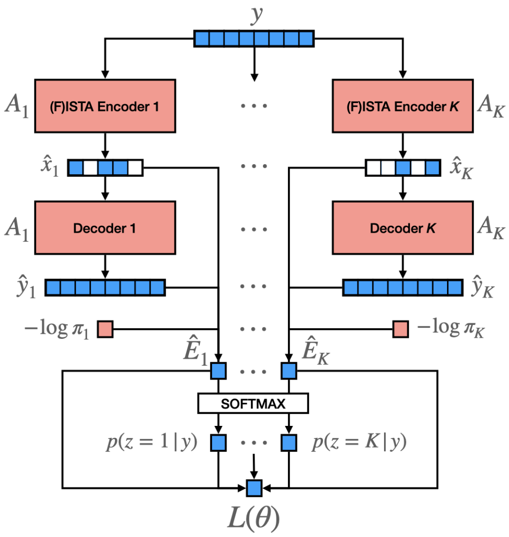

Fig. 1: A diagram of the MixMate architecture. Arrows outline the flow of information for the *forward* pass. Note the tying of weights between the encoders and decoders.

The Laplace prior on $\boldsymbol{x}$, which induces the ℓ_1 norm on $\boldsymbol{x}$, constrains $\boldsymbol{y}$ to be constructed from only a few columns of $\boldsymbol{A}_k$, with λ dictating the strength of the sparsity penalty.

2.1. Parameter Estimation & Inference for Clustering

With the generative model, we can cluster data by inferring the cluster identity z for each $\boldsymbol{y}$ through the posterior $p(z|\boldsymbol{y})$. This requires estimation of the latent code $\boldsymbol{x}$ and the model parameters $\theta = \{\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K, \boldsymbol{\pi}\}$. One effective method for accomplishing this goal is the expectation-maximization (EM) algorithm [14]. EM fits the model to $\mathcal{Y}$ by alternating between an E-Step and an M-Step, by minimizing the following loss function $L(\theta)$ based on Eq. (1) until convergence:

$$L(\theta) = -\mathbb{E}_{p(\boldsymbol{x},z\,|\,\boldsymbol{y})}[\log p(\boldsymbol{x}, z, \boldsymbol{y})] \tag{2}$$
$$= -\sum_{k=1}^{K} \underbrace{p(z=k\,|\,\boldsymbol{y})}_{\textit{Attention}} \cdot \underbrace{\mathbb{E}_{p(\boldsymbol{x}\,|\,\boldsymbol{y},z=k)}}_{\textit{Encoding}} \underbrace{[\log p(\boldsymbol{x}, z=k, \boldsymbol{y})]}_{\textit{Decoding}}.$$

The *E-Step* enables computation of $L(\theta)$ by solving for latent variables $\{\boldsymbol{x}, z\}$ given a current parameter estimate $\tilde{\theta}$, and the *M-Step* updates the model parameters to a new estimate $\tilde{\theta}^{\text{new}}$. In the next section, we show how we can use the model and the loss function to derive a deep clustering network.

3. MIXTURE MODEL AUTO-ENCODERS

We now introduce our clustering framework, *Mixture Model Auto-Encoders (MixMate)*. MixMate (depicted in Fig. 1) has three main parts – the encoding, decoding, and attention modules – each of which corresponds to a key term in Eq. (2).

3.1. Architecture

Encoder We observe that $p(\boldsymbol{x}\mid\boldsymbol{y}, z=k)$ acts as an *encoder* that maps the data $\boldsymbol{y}$ to a posterior distribution over the sparse code $\boldsymbol{x}$ using the k-th cluster's parameters. However, this posterior is not analytically tractable, due to non-conjugacy between the Laplace prior and the Gaussian likelihood, preventing the computation of the expectation in Eq. (2). To resolve this issue, we approximate the posterior by its *mode* $\hat{\boldsymbol{x}}_k$ [7],

$$\hat{\boldsymbol{x}}_k = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}_k\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1. \tag{3}$$

This choice has two consequences. First, it reduces the expectation of Eq. (2) to evaluation of a single term at $\hat{\boldsymbol{x}}_k$. Second, it turns posterior inference into a *sparse coding* problem (Eq. (3)). We run the fast iterative shrinkage-thresholding algorithm (FISTA) [15, 16] for L iterations to obtain $\hat{\boldsymbol{x}}_k = \boldsymbol{\alpha}_{(L)}$, via the following recurrence indexed by $\ell = 1, \ldots, L$:

$$\begin{aligned} &t_{(0)} = 1, \quad t_{(\ell)} = \tfrac{1}{2}\big(1 + \sqrt{1 + 4t_{(\ell-1)}^2}\big), \\ &\boldsymbol{\alpha}_{(0)} = \boldsymbol{0}, \quad \boldsymbol{\alpha}_{(\ell)} = f_{\eta\lambda}(\boldsymbol{\beta}_{(\ell-1)} + \eta\boldsymbol{A}_k^\top(\boldsymbol{y} - \boldsymbol{A}_k\boldsymbol{\beta}_{(\ell-1)})), \\ &\boldsymbol{\beta}_{(0)} = \boldsymbol{0}, \quad \boldsymbol{\beta}_{(\ell)} = \boldsymbol{\alpha}_{(\ell)} + \tfrac{t_{(\ell-1)}-1}{t_{(\ell)}}(\boldsymbol{\alpha}_{(\ell)} - \boldsymbol{\alpha}_{(\ell-1)}). \end{aligned} \tag{4}$$

Here, $\boldsymbol{0} \in \mathbb{R}^D$ is the zero vector, η is the step size, and $f_\gamma : \mathbb{R}^D \to \mathbb{R}^D$ is the soft-thresholding operator (the double-sided rectified linear unit) with threshold γ. Eq. (4) specifies a cascade of linear and nonlinear operations. Thus, it can be interpreted as an L-layer neural network encoder with parameters $\boldsymbol{A}_k$ that inputs $\boldsymbol{y}$ and outputs $\hat{\boldsymbol{x}}_k$. As there are K different dictionaries, the encoding module contains K parallel FISTA encoders, each returning a different code $\hat{\boldsymbol{x}}_k$ (Fig. 1).

Decoder We now use $\hat{\boldsymbol{x}}_k$ to evaluate $\log p(\boldsymbol{x}, z=k, \boldsymbol{y})$. We call the negation of this quantity the k-th *energy* $\hat{E}_k$ of $\boldsymbol{y}$,

$$\begin{aligned} \hat{E}_k &= -[\log p(\boldsymbol{y}\mid\hat{\boldsymbol{x}}_k, z=k) + \log p(\hat{\boldsymbol{x}}_k) + \log p(z=k)] \\ &= \underbrace{\|\boldsymbol{y} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_k\|_2^2}_{\textit{Reconstruction}} + \underbrace{\lambda\|\hat{\boldsymbol{x}}_k\|_1}_{\textit{Regularization}} \underbrace{-\log\pi_k}_{\textit{Bias}}. \end{aligned} \tag{5}$$

Observe that $\hat{E}_1, \ldots, \hat{E}_k$ are functions of the input data $\boldsymbol{y}$ and the model parameters $\theta = \{\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K, \pi\}$. In Eq. (5), the first term involves passing the sparse code $\hat{\boldsymbol{x}}_k$ through a *decoder* $\boldsymbol{A}_k$ to obtain the k-th cluster's reconstruction of the data $\hat{\boldsymbol{y}}_k = \boldsymbol{A}_k\hat{\boldsymbol{x}}_k$. Therefore, Eqs. (3) and (5) specify K parallel *auto-encoders* that map $\boldsymbol{y}$ to K different reconstructions. The two other terms capture the sparsity of $\hat{\boldsymbol{x}}_k$ and the bias of cluster k, which are also used to determine the cluster of $\boldsymbol{y}$.

Attention The component of Eq. (2) that is last computed by the network (Fig. 1) is

$$\begin{aligned} p(z=k\mid\boldsymbol{y}) &\propto \pi_k \cdot \int p(\boldsymbol{x})p(\boldsymbol{y}\mid\boldsymbol{x}, z=k)d\boldsymbol{x} \\ &\approx \pi_k \cdot \Big\{\max_{\boldsymbol{x}} p(\boldsymbol{x}) \cdot p(\boldsymbol{y}\mid\boldsymbol{x}, z=k)\Big\} = \exp(-\hat{E}_k), \end{aligned} \tag{6}$$

where we have again dealt with non-conjugacy in the integral using a mode approximation. Normalizing Eq. (6) yields

$$p(z = k \mid \boldsymbol{y}) \approx \frac{\exp(-\hat{E}_k)}{\sum_{k'=1}^{K} \exp(-\hat{E}_{k'})}, \tag{7}$$

which is equivalent to the *softmax* nonlinearity $\boldsymbol{\sigma} : \mathbb{R}^K \to \mathbb{R}^K$ applied to the negation of $\hat{\boldsymbol{E}} = [\hat{E}_1, \ldots, \hat{E}_K]$. Therefore, $\hat{E}_k$ measures how likely $\boldsymbol{y}$ belongs to the k-th cluster, with a lower value indicating higher likelihood. Consequently, the loss function of MixMate in Eq. (2) can be expressed as

$$L(\theta) \approx \hat{\boldsymbol{E}}^\top \boldsymbol{\sigma}(-\hat{\boldsymbol{E}}), \tag{8}$$

which reflects a form of *attention* [17]. Eq. (8) shows that the goal of MixMate is to minimize a weighted average of the energies across the K clusters. To gain some intuition for why this objective is suited for clustering data, consider a sample $\boldsymbol{y}$ with $\hat{\boldsymbol{E}}$ that is small in the k-th component, but large for all other components. Thus, $\boldsymbol{\sigma}(-\hat{\boldsymbol{E}}) \approx \boldsymbol{e}_k$ where $\boldsymbol{e}_k$ is the k-th unit vector, suggesting that MixMate is certain that $\boldsymbol{y}$ belongs to cluster k. The resulting loss $L(\theta) \approx \hat{E}_k$ implies that MixMate will focus on lowering the energy of cluster k for $\boldsymbol{y}$ and ignore the other clusters. Therefore, as MixMate is trained, each auto-encoder will attenuate to a different portion of the dataset, thereby inducing a natural clustering of the data.

3.2. Training and Initialization

Training As a typical deep learning framework, MixMate alternates between a *forward pass* and a *backward pass* over several epochs. Each step involves a mini-batch $\mathcal{B} \subseteq \mathcal{Y}$ of data points. The *forward pass* passes each $\boldsymbol{y}_i \in \mathcal{B}$ through MixMate to compute $p(z_i|\boldsymbol{y}_i)$ and $L_i(\theta)$, corresponding to the E-Step. The *backward pass* updates the parameters to θ^{new} via *backpropagation* of the mini-batch loss $\frac{1}{|\mathcal{B}|}\sum_i L_i(\theta)$ through the architecture, corresponding to the M-Step. After training, a forward pass infers the assigned cluster for each data point.
Initialization Prior works have dedicated substantial efforts to network initialization to ensure strong clustering performance. The majority of these require *pre-training* for the sub-components of the architecture [1, 2, 4, 5]. Although effective, pre-training complicates the application of clustering algorithms, since it requires a non-trivial amount of 1) computation time and 2) effort in tuning the hyperparameters.

For MixMate, we introduce a simple initialization procedure that avoids pre-training. We select D data points from the training set assumed to belong to cluster k as the initial columns of $\boldsymbol{A}_k$. This assumes that a data point is reasonably modeled as a linear combination of other points in the *same* cluster [18, 19]. These points are determined by applying an off-the-shelf clustering algorithm on a sampled subset $\mathcal{S} \subseteq \mathcal{Y}$ such that $|\mathcal{S}| \ll N$ to obtain partitions $\mathcal{S}_1, \ldots, \mathcal{S}_K$. Then, D points in each $\mathcal{S}_k$ are used as the initial columns of $\boldsymbol{A}_k$. We can use algorithms that would be expensive to run on the full dataset (e.g. spectral clustering, sparse subspace clustering).

3.3. Comparison to Other Frameworks

In contrast to other deep networks [1, 2, 4, 5], MixMate learns and backpropagates through *tied* weights between the encoding and decoding modules, as a direct consequence of the generative model. Thus, MixMate is often much smaller in size compared to other deep-clustering architectures.

Within the literature, Deep Auto-Encoder Mixture Clustering (DAMIC) [4] and K-Deep Auto-Encoder (K-DAE) [5] are most similar to MixMate, also using K different auto-encoders. The key difference is that their auto-encoders are not derived from a generative model and have black-box designs that sacrifice interpretability for architectural flexibility.

4. EXPERIMENTS

We assessed the clustering performance of MixMate on three datasets, all with $K = 10$ ground-truth clusters: (a) **MNIST**, $N = 70,000$ handwritten digits of size 28×28 ($M = 784$), (b) **FashionMNIST** [20], $N = 70,000$ fashion images of size 28×28 ($M = 784$), and (c) **USPS** [21], $N = 11,000$ images of size 16×16 ($M = 256$). We evaluated performance using three metrics [4, 5] – (a) normalized mutual information (**NMI**), (b) adjusted Rand index (**ARI**), and (c) clustering accuracy (**ACC**). Since MixMate returns soft cluster assignments, we took the maximum probability (lowest energy cluster) for each data point as its deterministic assignment.

In all experiments, we used MixMate with $K = 10$. We set $D = 50$ per auto-encoder for MNIST and FashionMNIST, and $D = 30$ for USPS, due to smaller dataset size. Each encoder ran the FISTA algorithm for $L = 15$ iterations with step size $\eta = 0.04$. For sparsity, MNIST and FashionMNIST used $\lambda = 0.75$, while USPS used $\lambda = 0.25$ since its images are 3× smaller. Since the datasets we consider are known to contain balanced classes, we fixed $\pi_k = 1/K$ for all k, though in general, our framework allows for these π_k to be learned. We applied sparse subspace clustering (SSC) with default parameters [18] to $|\mathcal{S}| = 2000$ randomly selected data points, to obtain the initial dictionaries. SSC ran in fewer than 5 seconds due to the small subset size $|\mathcal{S}| \ll N$.

We recorded the performance on the full dataset *after initialization* (INIT) (i.e. before learning dictionaries) and *after training* (TRAIN). To be consistent with prior work [5], we used the Adam optimizer [22], $T = 50$ epochs, a batch size of 256 examples, and a learning rate of 0.001. For simplicity, we used neither batch normalization nor early stopping.
Clustering Results Table 1 shows the results averaged over five trials. We also included results for other unsupervised deep clustering algorithms – Deep Embedded Clustering (DEC) [1], Deep Clustering Network (DCN) [2], DAMIC [4], and K-DAE [5]. We observed that the trained MixMate almost always outperforms the baselines. It achieved superior performance while being much smaller, e.g., up to *50× fewer parameters* for the MNIST dataset. The INIT column

Table 1: Clustering metrics for various algorithms (highest numbers for each row are in **bold**). Columns marked with $*$ indicate numbers from [5]. The metrics are calculated as the mean over five trials. We include "± standard deviation" over the five trials for the fully trained MixMate numbers.

	DEC*	DCN*	DAMIC*	K-DAE*	MixMate	
					INIT	TRAIN
MNIST						
NMI	0.80	0.81	**0.86**	**0.86**	0.75	**0.86** ± 0.03
ARI	0.75	0.75	0.82	0.82	0.72	**0.85** ± 0.04
ACC	0.84	0.83	0.88	0.88	0.84	**0.92** ± 0.04
Params	2.1 M	2.1 M	22.1 M	21.4 M	0.4 M	
Fashion						
NMI	0.54	0.55	0.65	0.65	0.60	**0.68** ± 0.02
ARI	0.40	0.42	0.48	0.48	0.44	**0.52** ± 0.01
ACC	0.51	0.50	0.60	0.60	0.57	**0.63** ± 0.01
Params	N/A	N/A	N/A	N/A	0.4 M	
USPS						
NMI	0.77	0.68	0.78	0.80	0.79	**0.82** ± 0.01
ARI	N/A	N/A	0.70	0.71	0.73	**0.76** ± 0.02
ACC	0.76	0.69	0.75	0.77	0.79	**0.81** ± 0.03
Params	N/A	N/A	N/A	N/A	0.08 M	

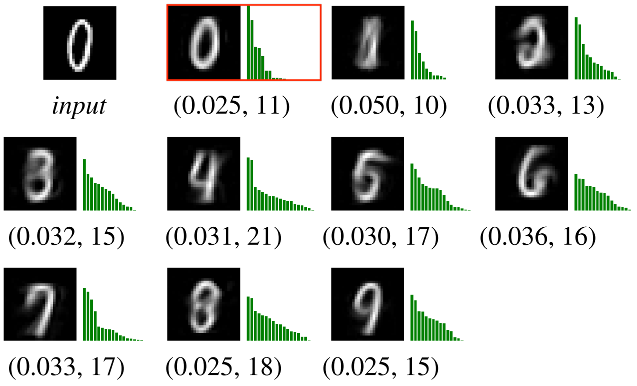
input
(0.025, 11)
(0.050, 10)
(0.033, 13)
(0.032, 15)
(0.031, 21)
(0.030, 17)
(0.036, 16)
(0.033, 17)
(0.025, 18)
(0.025, 15)

Fig. 2: Sample MNIST image $\boldsymbol{y}$, with MixMate's $K = 10$ reconstructions $\hat{\boldsymbol{y}}_k$ and sparse codes $\hat{\boldsymbol{x}}_k$ (green). Each image is labeled with (MSE, L0), corresponding to the mean-squared error of $\hat{\boldsymbol{y}}_k$ and the ℓ_0-norm of $\hat{\boldsymbol{x}}_k$, respectively. Observe that "0", "8", and "9" have similar MSE, but "0" has a sparser $\hat{\boldsymbol{x}}_k$. MixMate clusters the digit as a "0" (boxed in red).

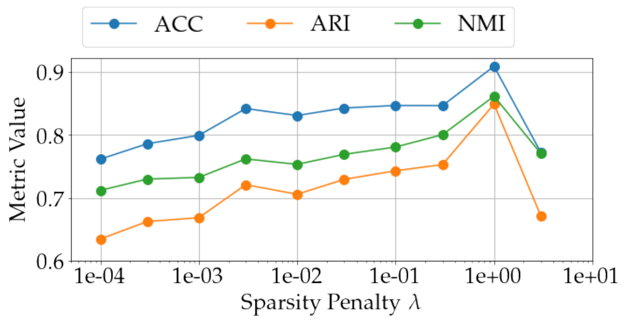

Fig. 3: The metrics ACC, ARI, and NMI as a function of the sparsity penalty λ (mean over five trials).

in Table 1 also shows that an *untrained* MixMate (with SSC initialized-dictionaries and a single forward pass to cluster the dataset) is competitive with other *fully trained* networks.

The Role of the Generative Model We discovered two patterns for how factors of $\hat{E}_k$ contributed towards clustering. In the first case, the reconstruction loss for a specific cluster was dominantly low and solely determined the cluster assignment. This was because each dictionary had learned columns for a particular part of the dataset (e.g. one of the digits in MNIST). In the second case, the reconstruction losses for several clusters were similar, but the "correct" cluster k had a sparser code, leading to a lower energy $\hat{E}_k$. An example of this scenario is depicted in Fig. 2. This shows that both reconstruction and latent sparsity are important factors for clustering.

The Effect of Varying $\boldsymbol{\lambda}$ We performed an ablation study by training MixMate on MNIST and varying λ (Fig. 3). We observed that clustering performance suffered when λ was too small or too large. With large λ, all auto-encoders poorly reconstructed the input, as the number and the amplitude of the latent codes were highly restricted. With small λ, every auto-encoder reconstructed the input similarly well, as the learned dictionaries were similar to each other and thus lost cluster specificity. In both cases, the latent codes were either too sparse (large λ) or too dense (small λ) and did not play a discriminant role. Therefore, both resulted in diffuse (rather than concentrated) cluster probabilities $\boldsymbol{\sigma}(-\hat{\boldsymbol{E}})$. This suggests that the right amount of sparsity plays an important role.

Clustering Incomplete Data Since MixMate's weights parameterize a generative model, we can adapt MixMate to handle missing data by adjusting the model. For an incomplete image $\boldsymbol{y}' \in \mathbb{R}^m$ (where $m \leq M$) with $M - m$ missing pixels, we can modify Eq. (1) as $\boldsymbol{y}' \mid \boldsymbol{x}, z = k \sim \mathcal{N}(\boldsymbol{\Psi}\boldsymbol{A}_k\boldsymbol{x}, \boldsymbol{I})$, where $\boldsymbol{\Psi} \in \{0,1\}^{m \times M}$ is a data-specific mask that extracts rows of $\boldsymbol{A}_k$ corresponding to the locations of the observed pixels of $\boldsymbol{y}'$. This simple change of replacing $\boldsymbol{A}_k$ with $\boldsymbol{\Psi}\boldsymbol{A}_k$ can be applied to the entire architecture (Sec. 3.1), enabling MixMate to cluster incomplete data. We trained the network on MNIST in which 90% of the images had 25% of the pixels missing uniformly at random. MixMate attained 0.86 ± 0.04 NMI, 0.85 ± 0.06 ARI, and 0.92 ± 0.04 ACC across five trials – similar to the scores for the clean data experiments. Such robustness can be useful for compressed sensing and remote sensing, where missing data is a common occurrence.

5. CONCLUSION

We introduced a novel architecture for deep clustering called *Mixture Model Auto-Encoders* (MixMate). MixMate combines the interpretability of the sparse dictionary learning model with the scalability of deep learning. The framework of MixMate can be extended to convolutional models, non-Gaussian data distributions [23], as well as the integration of other priors for the latent codes and the dictionaries, such as group sparsity [24] or smoothness [25].

6. REFERENCES

[1] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[2] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *International conference on machine learning*. PMLR, 2017, pp. 3861–3870.

[3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[4] S. E. Chazan, S. Gannot, and J. Goldberger, "Deep clustering based on a mixture of autoencoders," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing*, 2019, pp. 1–6.

[5] Y. Opochinsky, S. E. Chazan, S. Gannot, and J. Goldberger, "K-autoencoders deep clustering," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 4037–4041.

[6] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *arXiv preprint arXiv:2012.08405*, 2020.

[7] B. Tolooshams, S. Dey, and D. Ba, "Deep residual autoencoders for expectation maximization-inspired dictionary learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2415–2429, 2020.

[8] S. Wu, A. Dimakis, S. Sanghavi, F. Yu, D. Holtmann-Rice, D. Storcheus, A. Rostamizadeh, and S. Kumar, "Learning a compressed sensing measurement matrix via gradient unrolling," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6828–6839.

[9] B. Tolooshams, S. Dey, and D. Ba, "Scalable convolutional dictionary learning with constrained recurrent sparse auto-encoders," in *International Workshop on Machine Learning for Signal Processing*, 2018, pp. 1–6.

[10] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 2018.

[11] B. Tolooshams and D. Ba, "Pudle: Implicit acceleration of dictionary learning by backpropagation," *arXiv preprint arXiv:2106.00058*, 2021.

[12] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[13] J. Starck, F. Murtagh, and J. Fadili, *Sparse image and signal processing: Wavelets and related geometric multiscale analysis*, Cambridge university press, 2015.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[16] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of International Conference on Machine Learning*, 2010, pp. 399–406.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[19] P. Tankala, A. Tasissa, J.M. Murphy, and D. Ba, "K-deep simplex: Deep manifold learning via local dictionaries," *arXiv preprint arXiv:2012.02134*, 2021.

[20] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[21] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] B. Tolooshams, A. H. Song, S. Temereanca, and D. Ba, "Convolutional dictionary learning based auto-encoders for natural exponential-family distributions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9493–9503.

[24] E. Theodosis, B. Tolooshams, P. Tankala, A. Tasissa, and D. Ba, "On the convergence of group-sparse autoencoders," *arXiv preprint arXiv:2102.07003*, 2021.

[25] A. H. Song, B. Tolooshams, and D. Ba, "Gaussian process convolutional dictionary learning," *arXiv prepreint arXiv:2104.00530*, 2021.