HIGH SCHOOL EDITION
Journal of Student Research

Volume 11 Issue 2 (2022)

Deep Clustering with Robust Autoencoder (DCRA)

Connor Lee[1], Albert Wang[1] and Stefano Rizzo[#]

[1]Saratoga High School, Saratoga, CA, USA
[#]Advisor

ABSTRACT

Accordingly to Science Daily, 90 percent of all the data in the world has been generated in the last two years. However, the world is analyzing less than 1 percent of its data so far. With the advancement of high-performance computing, deep learning methods are readily applied to analyze large-scale high dimensional datasets. These machine learning methods have achieved significantly efficient training and inferencing as well as producing much more accurate predicted results. Clustering is an unsupervised machine learning method of identifying and grouping similar data points into the same cluster. Clustering plays a fundamental role in the data mining and machine learning community for grouping data into structures so that similar data points are assigned to similar groups. Furthermore, to process these huge amounts of high-dimensional data, deep learning becomes a key technique to learn and perform feature representation of data in latent space for many real world applications. In this paper, we propose deep clustering with robust autoencoder (DCRA), which jointly utilizes robust auto-encoder and deep clustering to perform feature representation and cluster assignments simultaneously. Multiple experiments using open public datasets have been conducted to evaluate our model's performance. Our results show DCRA is capable of generating high quality clusters with high clustering accuracy of 90% above in high dimensional datasets. The decreasing training and test loss with increasing number of epochs also validates our results.

Introduction

Nowadays, with the huge amount of high-dimensional data available, deep learning [1], one type of machine learning method based on neural networks with representation learning, becomes more and more popular due to its ability to process large numbers of features and deal with unstructured data. A basic neural network [2] is made up of three components: input layer, hidden layer and output layer. The nonlinearity activation functions make neural networks learn more complex input-output mappings and improve the prediction accuracy in many applications.

To learn the key features of given high-dimensional data, autoencoder [3], a special type of neural network, is designed to learn a lower-dimensional feature representation (encoding) of the given data. One of the applications of autoencoders is to discover clusters of similar data points in an unsupervised setting. The problem statement of clustering can be formed as: given a dataset and the number of clusters (k), algorithm needs to split data points into k clusters so that similar data points are in the same cluster.

Clustering [4] performs a key role in many real-world applications, such as customer segmentation, pattern recognition, recommendation systems, natural language processing, etc. Most traditional clustering methods, such as K-means and DBscan, are applicable for low-dimensional data. The deep learning based clustering techniques are different from traditional clustering techniques as they cluster the data-points by finding complex patterns rather than using simple pre-defined metrics like euclidean distance between data points. Similar to deep learning in a supervised approach, deep clustering has a great impact on many applications. The objective function of deep clustering algorithms is generally a linear combination of unsupervised representation learning loss (or called reconstruction loss) and a clustering oriented loss. There have been some great works on deep clustering. For instance, deep embedded clustering (DEC) [5] is the first model to use deep autoencoders for clustering. The training has two stages. In the first stage the autoencoder is run by training it the usual way, without clustering. In the second stage, the decoder is ignored, and the encoder is refined to produce better clusters with a "cluster hardening" procedure. Another work called Deep Clustering Network (DCN) [6] pre-trains the autoencoder, and then jointly optimizes the reconstruction loss and K-means clustering loss with alternating cluster assignments.

Despite the significant progress described above, most of the deep neural networks (DNNs) approaches don't take much consideration of the robustness of the output. DNNs have been shown to be vulnerable to subtle input perturbations [7, 8] which could entirely alter the DNN's output. To design a robust DNN, the output should not only be accurate, but also resistant against input perturbations.

In this paper, we propose DCRA, Deep Clustering with Robust Autoencoder, by performing joint learning to minimize the reconstruction loss and clustering loss as well as introducing robustness to the framework. At the beginning, we add noise to the original input data, then the autoencoder is trained to reconstruct the original input, finally K-means clustering is performed on latent feature representation by assigning these two losses with different weights in a joint learning way. We conduct experiments on different groups of image datasets. The experimental results show that our approach could effectively boost performance of the clustering algorithm on a variety of datasets.

Background

Autoencoder

An autoencoder [3] is a type of artificial neural network used to learn efficient data encoding in an unsupervised manner. It converts high-dimensional data to low-dimensional data. Therefore, it is useful in noise removal, feature extraction, compression, clustering tasks, etc.

In general, there are two parts in an autoencoder: the encoder and the decoder. The encoder is used to generate a reduced feature representation from an initial input *x*, which maps *x* to a lower-dimensional feature vector *z*. The decoder is used to reconstruct *x'* from *z* by minimizing the loss function. We train the model by comparing *x* to *x'* and optimizing the parameters to increase the similarity between *x* and *x'*.

- Encoder network: it translates the original high-dimensional input into the latent low-dimensional vector. The input size is larger than the output size.
- Decoder network: it recovers the data from the latent vector with larger output layers.

See Figure 1 below for an illustration of the autoencoder framework:

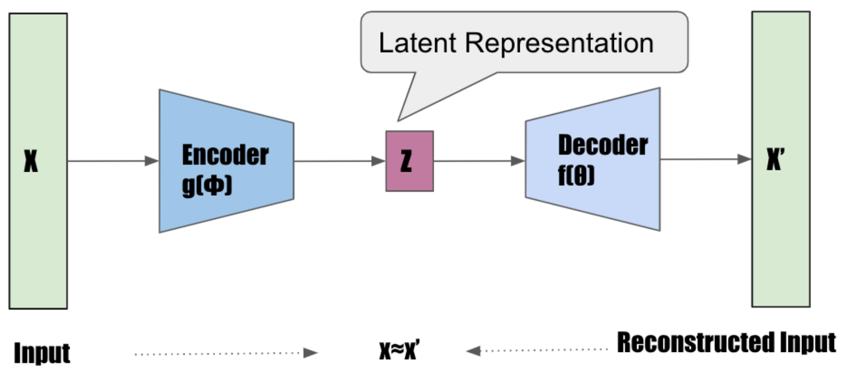


Figure 1. Vanilla autoencoder framework

The model contains an encoder function $g(.)$ parameterized by ϕ and a decoder function $f(.)$ parameterized by θ. The low-dimensional code learned for input x in the latent vector layer is $z = g_{\phi}(x)$ and the reconstructed input is $x' = f_{\theta}(g_{\Phi}(x))$.

The parameters (θ, ϕ) are learned together to output a reconstructed data sample same as the original input, $x \approx f_{\theta}(g_{\Phi}(x))$, or in other words, to learn an identity function. There are various metrics to quantify the difference between two vectors, such as cross entropy when the activation function is sigmoid, or MSE (mean square error) loss.

Deep Clustering

A deep neural network has been successfully applied in various supervised learning tasks [9, 10]. The essential representations of complex data structure is learnt by constructing a network with multiple hidden layers and then training the network with a large amount of data [11]. The success of deep neural networks in supervised learning environments inspires the recent development of unsupervised deep learning methods for data clustering. These methods are called deep clustering. Deep clustering combines deep learning and clustering. It performs feature representation and cluster assignments simultaneously. Its' clustering capability outperforms traditional clustering algorithms. Most deep clustering studies are two-stage training schemes based on an autoencoder. First, an autoencoder is trained to reduce the data dimension. Then, the encoder acts as a latent feature extractor and uses a clustering algorithm to train it simultaneously. The two-stage clustering methods have been successfully applied in many research works [12, 13]. Later on, one-stage clustering methods that jointly accomplish feature transformation and clustering is developed as an alternative approach. For instance, Deep adaptive image clustering (DAC) is a typical one-stage image clustering algorithm [14]. It proposes a self-learning approach to perform image clustering. The objective function is used by selecting highly confident image pairs. And then the cluster assignment is incorporated into classification labels.

Presently, most of the deep clustering algorithms don't take the factor of robustness into consideration. In this work, we incorporate robustness concepts into our proposed deep clustering model development and performance evaluation.

Robustness

Robustness is useful for both algorithm selection and decision making. For instance, in high dimensional data, some noisy features can make clustering very difficult [15]. Robustness provides a simple and intuitive measure of the stability and predictability of an algorithm. Robust clustering methods are designed to be resistant to small perturbations of the original data and the inclusion of irrelevant features. Similarly, robust autoencoders can help to reduce the model reconstruction error and generate latent features which resists the noisy input. Robustness should be integrated to clustering and autoencoder when actual data collected are prone to unwanted noise.

Methods

Architecture

The proposed deep clustering architecture is shown in Figure 2. It consists of three components: encoder, decoder and clustering component. This new architecture differs from the vanilla autoencoder framework shown in Figure 1 in a few aspects: (1) Robust Training: Robust autoencoder is an extension of the traditional autoencoder architecture. An autoencoder neural network aims at reconstructing high dimensional data from hidden latent space. To construct a robust autoencoder, we add some noise to the clean data points being introduced. The robust autoencoder network learns the noisy data points to reconstruct the clean data. Before training, original input x is perturbed by adding noise, we use Gaussian noise (μ, σ) in our work. The noise added data x' becomes the input to encoder. (2) Clustering Capability: To generate clusters, latent features are not only used for decoder input, but also are used for clustering. (3) Joint Learning: Joint loss is used for clustering and decoding. It is a linear combination of reconstruction loss and clustering loss.

Detailed flow is described below:

(1) To overcome overfitting and improve robustness, the original input x is corrupted by adding some noise $x'^{(i)} \sim f_{noise}(x'^{(i)}|x^{(i)})$, where f_{noise} is chosen as adding random noise following gaussian distribution to input x

(2) Corrupted input x' is trained by an encoder and decoder. Model target is to recover the original input x and minimize the reconstruction loss (let's denote as L_1), where $L_1 = \frac{1}{n}\sum_{i=1}^{n} \quad (x^i - f_\theta(g_\Phi(x'^i)) \quad)^2$.

(3) Encoder generates compressed low-dimensional latent features of the input, denoted as Z. Since classical clustering methods fits for low-dimensional data, in this work, K-means algorithm is used for generate clusters to minimize the cluster assignment loss for each data point (let's denote as L_2), where $L_2 = \frac{1}{n}\sum_{i=1}^{n} \quad (x^i - o_k)^2, (k = 1,2,..K)\, , i = (1,2,...,n)$.

(4) The entire loss function L is the linear combination of L_1 and L_2 , where $L = \lambda L_1 + (1\text{-}\lambda) L_2$ and λ is used for balancing the impact of L_1 and L_2.

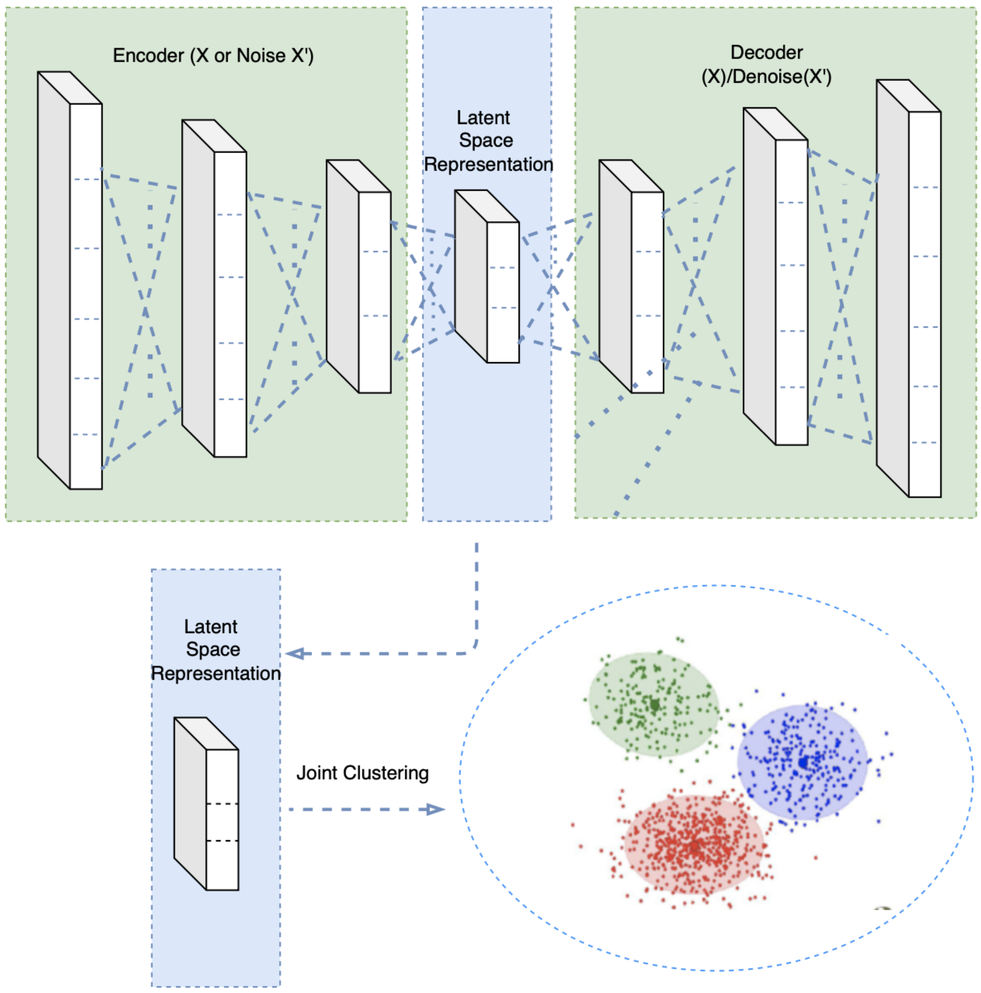


Figure 2. Proposed Model: Deep Clustering with Robust Autoencoder (DCRA)

Algorithms

The Deep Clustering with Robust Autoencoder (DCRA) algorithm includes robustness algorithms and K-means clustering algorithms. They work together to learn the input data.

Deep Robust Autoencoder Algorithm

```
Algorithm 1: Deep Robust Autoencoder (DRA)
1  Input: source data $X$, corrupted data $\bar{X}$
2  Initialization: Initialize the parameters of the encoder $Q_\phi$, decoder $G^\theta$
3  Output: reconstructed data $\hat{X}$
4  Execute each batch:
5  while $(\phi, \theta)$ not converged do
6      Sample $\{x_1, \dots, x_n\} \in X$ for $i = 1, \dots, n$ from training set
7      Add noise and generate corrupted data $\{\bar{x}_1, \dots, \bar{x}_n\}$ from $\{x_1, \dots, x_n\} \in X$
8      Compute MSE loss:
           $L = \frac{1}{n}\sum_{i=1}^{n} \|x_i - G_\theta(Q_\phi(\bar{x}_i))\|^2$
9      Update $Q_\phi$, $G_\theta$ by minimizing $L$
10 end
```

Deep Clustering Algorithm K-Means

```
Algorithm 2: Deep Latent KMeans Clustering (DLRC)
1  Input: latent representation data $\hat{X}$ , No. of cluster K
2  Initialization: Initialize centroids $o_1, \dots, o_K \in O$
3  Output: updated centroids $o_1, \dots, o_k$
4  while $(o_1, \dots, o_K)$ not converged do
5      For each , find closet centroid $o_k \in O \leftarrow argmin_k \|\hat{x} - o_k\|^2$, where $k \in (1, \dots, K)$
6      $o_k \leftarrow mean(x_n)$, where $x_n \in cluster_k$
7      update $(o_1, \dots, o_K)$
8      end
```

DRA+DLRC Joint Learning

For joint learning purposes, the loss function is to minimize the reconstruction loss of original input and the cluster assignment loss. The entire loss function L is the linear combination of L_1 and L_2 , where $L = \lambda L_1 + (1-\lambda)L_2$ and λ is used for balancing the impact of L_1 and L_2.

Data Sets Used for This Analysis

- MNIST [16]: containing 60k training samples and 10k test samples from 10 digit classes. Each digit is a 28×28 grayscale image. We choose the digit 0 as the inlier class and the others as outliers.
- Fashion MNIST [17]: consisting of 60k training samples and $10k$ test samples from 10 classes. Each sample is a 28×28 grayscale image in a clothes category. We use the class 0 as inliers (Xi), and the others as outliers.
- BSDS 300 [18]: a dataset used frequently for image denoising and super-resolution. The dataset is composed of a large variety of images ranging from natural images to object-specific such as plants, people, food etc. The images are divided into a training set of 200 images, and a test set of 100 images.

Performance Metrics

- MSE Loss: mean square error (MSE) metric is used to measure and evaluate model's training and inference.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \|y_i - \hat{y}_i\|$$

There are two parts to calculate the loss. For the autoencoder part, MSE is the reconstruction loss between reconstructed and original data points. For the k-means clustering part, MSE is the cluster assignment loss between data points to the assigned cluster centroid.

- PSNR (peak signal-to-noise ratio): PSNR computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.
- Clustering Accuracy: (ACC): ACC is the unsupervised equivalent of classification accuracy. ACC is used to map between the cluster assignment output of the algorithm with the ground truth. This mapping is required because an unsupervised algorithm may use a different label than the actual ground truth label to represent the same cluster.

$$ACC = max_C \frac{\sum_{i=1}^{n} \quad 1(y = C(c_i)}{n}$$

Model Parameters

Table 1 lists out the hyper-parameters and correspondent values used in the model for training purposes. They will be tuned in order to achieve optimal performance.

Table 1. Model Parameters Description:

Parameters	Value	Description
dim_h	40	the size of hidden layers
µ	0	noise mean
sigma	1.0	noise variance
lambda	0.01	hyperparameter for balancing losses
lr	0.0002	learning rate for the Adam optimizer
epochs	25	number of epochs completed by the model
batch_size	256	batch size for stochastic gradient descent

Results

MNIST Dataset

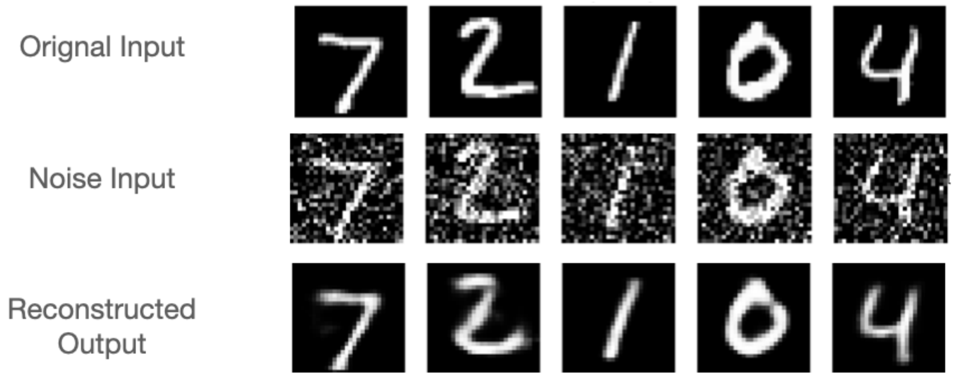
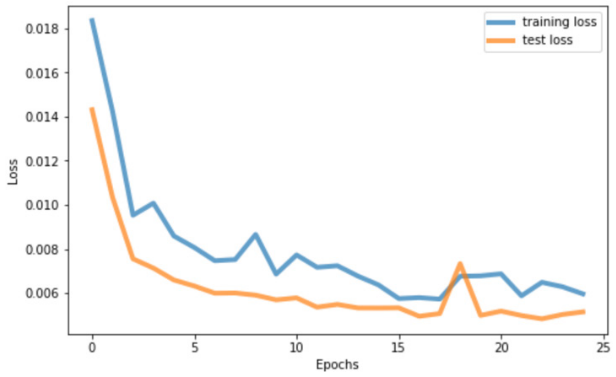


Figure 3. MINIST (Row 1: Original Input; Row 2: Reconstructed Output)

Training Loss/Test Loss



Clustering Visualization

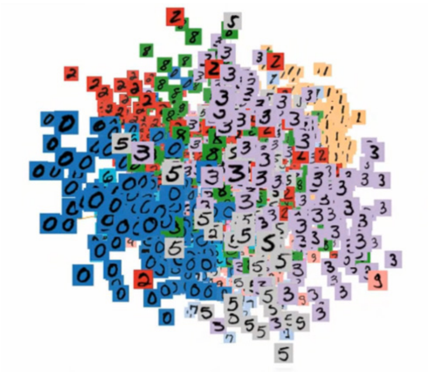


Figure 4. MINIST Train/Test loss; Clustering Visualization

Analysis I: for MNIST dataset, we visualize the model performance by randomly selecting some original digits and corresponding reconstructed digits. The model training loss and test loss are also plotted out. We can see our model perform well on training and testing sets. There is no overfitting issue. To validate the clustering performance, the test sets are executed by our model and latent features are projected onto the tensorboard. In this experiment, the ACC achieves 95.06%.

Fashion MNIST DataSet

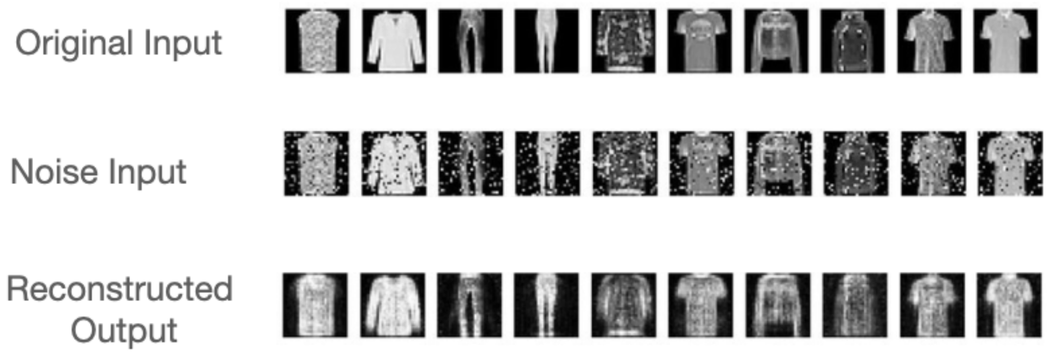
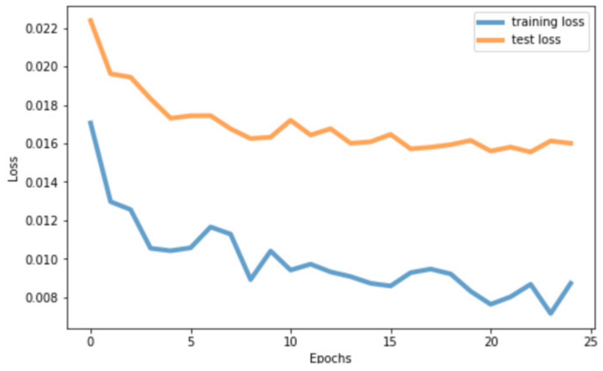


Figure 5. DCRA outputs on Fashion MNIST

Training Loss/Test Loss



Clustering Visualization

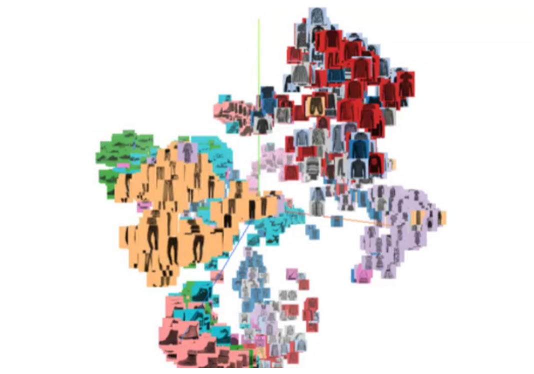
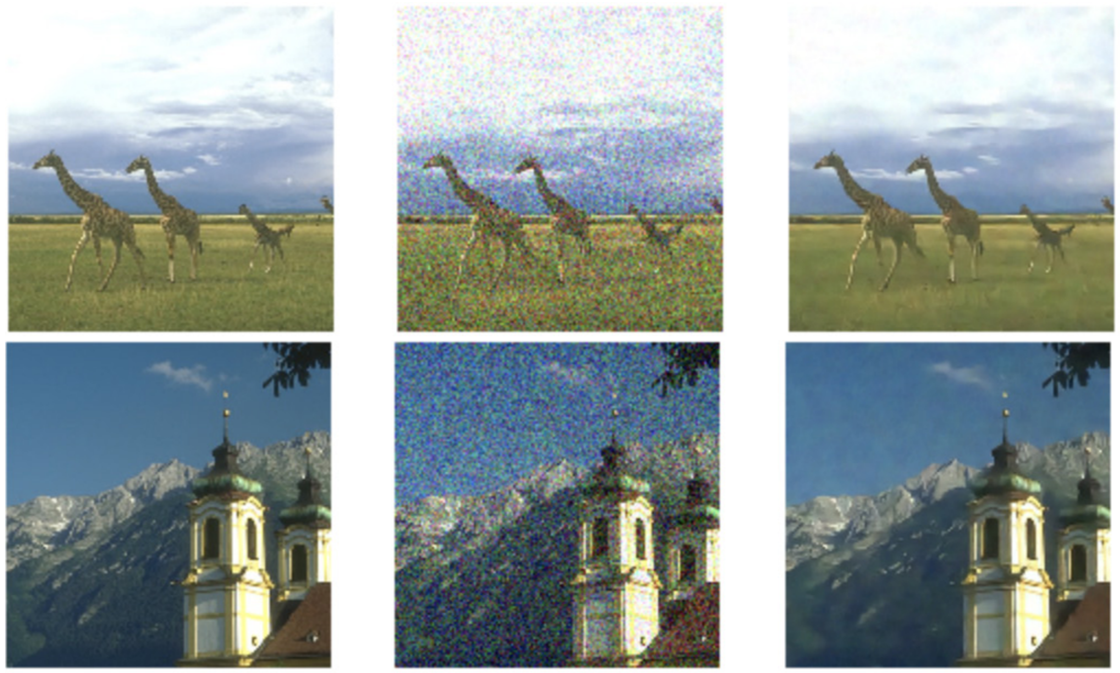


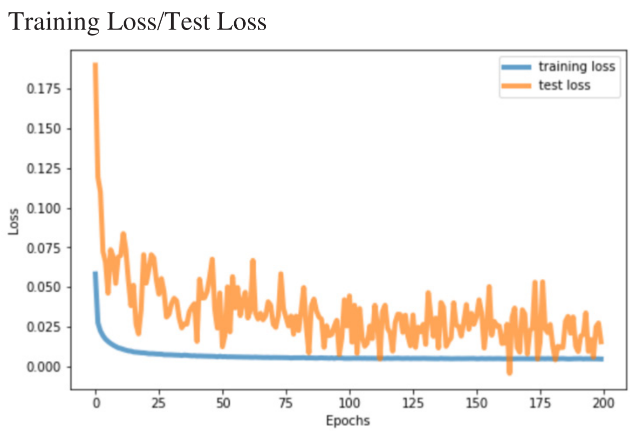
Figure 6. Fashion MINIST Train/Test loss; Clustering Visualization

Analysis II: Similarly, for Fashion MINIST dataset, we visualize the model performance by randomly selecting some original fashion clothes and corresponding reconstructed fashion clothes. The model training loss and test loss are also plotted out. We can see our model performs well on training but not very well on the testing set. Although there doesn't have overfitting issues, apparently we need to tune the model or add regularization techniques to get best performance. To validate the clustering performance, the test sets are executed by our model and latent features are projected onto the tensorboard. In this experiment, the ACC achieves 93.48%.

BSDS DataSet

Clean Noise Input Reconstructed Output





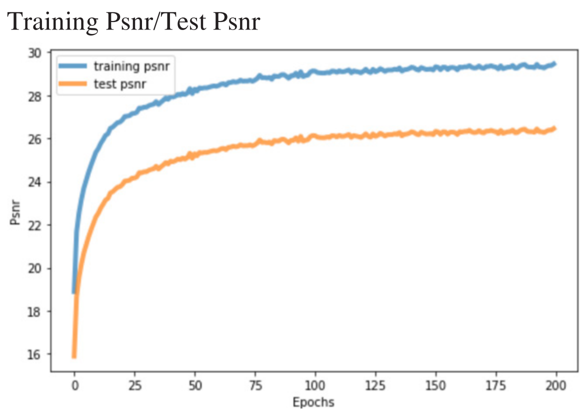


Figure 7. DCRA outputs on BSDS dataset

Analysis III: Similarly, for the BSDS dataset, we visualize the model performance by randomly selecting some original pictures and corresponding reconstructed pictures. The model training loss and test loss are also plotted out. We can see our model perform well on training and testing sets. The higher PSNR shows the algorithm has the capability to reconstruct images well.

Discussion

Our Deep Clustering with Robust Autoencoder performs joint learning to minimize the reconstruction loss and clustering loss as well as introducing robustness to the framework. From the experimental results, the clustering accuracy for MNIST, Fashion MNIST can achieve more than 90%, it shows our joinit learning can generate high quality clusters using learned latent vectors. For BSDS data, we used PSNR to evaluate the reconstruction quality, the training PSNR and test PSNR can achieve greater than 20. However, we can see our model performs well on MNIST and BSDS sets but doesn't perform very well on FASHION MNIST.

For MNIST and BSDS dataset, the model's training loss and test loss decreases with the increased number of epochs. This means our model doesn't overfit to the training data and generalizes to unseen data (i.e. test data). On the other hand, the test loss doesn't decrease too much for FashionMNIST data. One possible reason is FashionMNIST data has more complicated features and model overfits and doesn't generalize well to the unseen data. To further improve the performance for different datasets, we will investigate our model architecture to figure out the most suitable one, for instance, the number of network layers, the hyper-parameter values, drop out etc. We would like to integrate the pytorch tuning framework to perform hyper-parameter tunings and identify optimal parameters for our model to fit for different data sets.

Conclusion

We proposed a joint learning model by utilizing the clustering and the robust autoencoder framework. The model is evaluated using three real image datasets. Our experiments show that the model is effective to reconstruct the original input and generate meaningful clusters using latent key features.

As a next step, we plan to deploy this Deep Clustering with Robust Autoencoder (DCRA) model to applications such as natural language processing and recommendation systems. For instance, this model could be used to help the public school educational systems to automatically identify and correct any wrong typing from tens of thousands' student input forms. This model could also be applied to generate useful cluster information for further investigation of environment issues such as western state wildfires.

Acknowledgments

We would like to thank our mentor Dr Stefano Giovanni Rizzo for the guidance of literature research, brainstorming, open source data research and results discussion.

References

1. https://en.wikipedia.org/wiki/Deep_learning
2. https://en.wikipedia.org/wiki/Artificial_neural_network
3. Dor Bank, Noam Koenigstein, Raja Giryes, Autoencoders, https://arxiv.org/abs/2003.05991
4. https://en.wikipedia.org/wiki/Cluster_analysis
5. Junyuan Xie, Ross Girshick, Ali Farhadi, Deep Embedding Clustering, ICML'16: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, June 2016 Pages 478–487
6. Xifeng Guo, Xinwang Liu, Jianping Yin , Deep Clustering with Convolutional Autoencoders, ICONIP 14 November 2017
7. Zhihao Zheng, Pengyu Hong, Robust Detection of Adversarial Attacks by Modeling the Intrinsic Properties of Deep Neural Networks, Advances in Neural Information Processing Systems 31 (NeurIPS 2018)
8. Kui Ren, Tianhang Zheng, Zhan Qin ,Xue Liu, Adversarial Attacks and Defenses in Deep Learning, https://www.sciencedirect.com/science/article/pii/S209580991930503X#!
9. Iqbal H Sarker , Machine Learning: Algorithms, Real-World Applications and Research Directions, DOI: 10.1007/s42979-021-00592-x
10. Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald & Edin Muharemagic , Deep learning applications and challenges in big data analytics, Journal of Big Data volume 2, Article number: 1 (2015)
11. Jeff Heaton, Applications of Deep Neural Networks, https://arxiv.org/abs/2009.05673
12. Jung-Hua Wang, Jen-Da Rau and Wen-Jeng Liu, Two-stage clustering via neural networks, IEEE Transactions on Neural Networks 14(3):606-15
13. Yazhou Ren, Ni Wang, Mingxia Li, Zenglin Xu , Deep Density-based Image Clustering, https://arxiv.org/abs/1812.04287
14. Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, Deep Adaptive Image Clustering, 2017 IEEE International Conference on Computer Vision (ICCV)
15. Stephan Zheng, Yang Song, Thomas Leung, Ian Goodfellow, Improving the Robustness of Deep Neural Networks via Stability Training, CVPR 2016, https://doi.org/10.48550/arXiv.1604.04326
16. Tommaso Dreossi, Shromona Ghosh, Alberto Sangiovanni-Vincentelli, Sanjit A. Seshia, A Formalization of Robustness for Deep Neural Networks, https://doi.org/10.48550/arXiv.1903.10033
17. MNIST, https://en.wikipedia.org/wiki/MNIST_database
18. FashionMNIST: https://paperswithcode.com/dataset/fashion-mnist
19. BSD Dataset, https://paperswithcode.com/dataset/bsd
20. Parsons L, Haque E, Liu H: Subspace Clustering for High Dimensional Data: a Review. SIGKDD Explor Newsl. 2004, 6: 90-105. 10.1145/1007730.1007731.

21. Jörnsten R, Vardi Y, Zhang CH: A Robust Clustering Method and Visualization Tool Based on Data Depth. 2002, Basel: Birkhäuser
22. Junyuan Xie, Ross Girshick, Ali Farhadi, Unsupervised deep embedding for clustering analysis, ICML'16: Proceedings of the 33rd International Conference on International Conference on Machine Learning, Volume 48