FROM TEXT CLASSIFICATION TO IMAGE CLUSTERING, PROBLEMS LESS OPTIMIZED

by

Amirhossein Herandi

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN Computer Science

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2018

Acknowledgements

I would like to thank my hard-working professor and supervisor Dr. Heng Huang for all his help and guidance during my research.

April 10, 2018

Abstract

FROM TEXT CLASSIFICATION TO IMAGE CLUSTERING,

PROBLEMS LESS OPTIMIZED

Amirhossein Herandi, MS

The University of Texas at Arlington, 2018

Supervising Professor: Heng Huang

Machine Learning is thriving. Every industry is using its techniques in some way to improve their efficiency and revenue. However, the focus on research is not divided equally between all of the different areas and problems that this field can tackle and analyze. Currently, Computer Vision is the one area that is being focused very extensively by researchers and companies alike, and as a result has seen an amazing boost in the recent years. This ranges from the well-known problems of classification that use discriminative models all the way to more novel problems that use generative models such as style transfer, super resolution, and description generation. Yet, some other problems have not been worked on nearly as much as of now. These problems include some Natural Language Processing tasks like Sentence Classification and even Computer Vision problems such as Image Clustering. Each of these tasks has their own set of difficulties and obstructions that need to be tackled before they can be researched properly and used in the industry which is a great driving force for research. Specifically, the case of clustering seems to be interesting to look into as more and more lable-less and unknown data is being generated every day without means to process and analyze them efficiently. We will discuss these problems that have been focused on less throughout the recent years.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

With the advent of the internet and more recently, digitization of data, the need to process and interpret data has become a prevalent and interesting problem. There are massive amounts of data available publicly and privately today, a large portion of which has not been utilized yet, mostly because the techniques currently available are not sufficiently robust to interpret them in their present state.

Current data is noisy and unstructured, it is unlabeled and vague, making most available algorithms unable to process them without spending significant time and effort for structuring and cleaning the data.

On the other hand, when the data is sufficiently organized, with the emergence of a multitude of new ways of looking at data and interpreting them, it has been shown that not only there are various methods to approach any data related problem, e.g. fully or semi-supervised classification as well as unsupervised learning, but also using the same data more novel and groundbreaking tasks could be accomplished, e.g. segmentation and image captioning.

However, as mentioned before, almost all the current tasks need sufficiently clean data with correct labeling to be usable in real world tasks. While some huge datasets like ImageNet (Deng, Dong et al. 2009) have been successfully and acceptably labeled using crowd-sourcing websites like Amazon Turk (Buhrmester, Kwang et al. 2011), more difficult data like medical images or even some text reports cannot be correctly cleaned or labeled by ordinary people on these websites. This is one of the biggest problems that still holds the Machine Learning and Data Mining back from fully exploiting the huge amounts of unused data available online, and alleviating this problem could save a lot of money for companies. It is for this reason that Unsupervised Learning will probably be very successful in the near future as it does not have a need for labeling.

Typically, classic Machine Learning methods perform well with moderate numbers of data, while Deep Learning methods tend to work well when large amounts of data are available. The fact that companies and websites accumulate more and more data as we progress through the years makes these Deep Learning algorithms even more promising in dealing with these data efficiently as the databases grow.

Most current methods are offline algorithms, meaning that by adding any new data points the entirety of the learning process needs to be redone to improve the model based on all the available data. While sometimes this might be possible and may even be necessary in order to achieve the best performance possible, it wouldn't be feasible for gigantic datasets from companies like Netflix or Amazon to retrain their algorithms every time they add new data points, new customers or products. This emphasizes the need for novel and effective online deep learning algorithms which can handle scaling datasets, while maintaining their robust data analyzing power to an acceptable extent.

Chapter 2

Text Classification

Introduction

Classification of images has been extensively researched in the last couple of decades in the field of Machine Learning, and specially now with Deep Learning, however, text classification has not been pursued to that extent yet. There could be several reasons for this phenomenon. One could argue that most companies are investing more in Computer Vision right now and its applications compared to Natural Language Processing which focuses on text processing and manipulation.

Most current models use word embeddings, which were initially introduced by (Bengio, Ducharme et al. 2003) and later with the introduction of the very popular word2vec word embeddings (Mikolov, Sutskever et al. 2013) became widely used, and for most word based NLP tasks is the standard word representation model. Not unlike computer vision, it is believed that the model will learn hierarchical features from the text, n-grams, phrases, sentences are essentially combinations of several words.

However, when analyzing sentences on the word level, inputs, which are sentences, can become relatively small. These small networks would usually prohibit the use of deep networks. To alleviate this problem some models have been reproduced that work on the character level (Zhang, Zhao et al. 2015) making the inputs much longer and enabling the use of very deep networks (Conneau, Schwenk et al. 2016), but these methods cannot use word embeddings. Currently there is no consensus on which approach, word level or character level models, is better, and there are advantages and disadvantages for both.

Related Works

As with most areas in machine learning, classification, text classification here specifically, is an important part of Natural Language processing as well, and just like any other area in machine learning researchers have started exploiting machine learning models and techniques in the past few years for text classification like convolutional networks (Kim 2014, Yosinski, Clune et al. 2014, Conneau, Schwenk et al. 2016, Le, Cerisara et al. 2017) as well as recurrent neural networks (Xiao and Cho 2016, Yogatama, Dyer et al. 2017). However, traditional methods typically use linear classifiers for text classification (Joachims 1998, McCallum and Nigam 1998, Fan, Chang et al. 2008). It has been shown that with a good rank constraint and fast loss approximation they could be scaled to large datasets rapidly as well (Joulin, Grave et al. 2016).

Machine Translation has been one of the hottest topics in the field of NLP. Recent works have been using novel neural machine translation techniques like encoder decoder models for human-like translations (Gehring, Auli et al. 2016, Gehring, Auli et al. 2017, Klein, Kim et al. 2017). The source sentences are fed through the encoder, the decoder gets the last hidden state from the encoder as input and generates the translation word by word to the target language. This method is at the core of the Google Translation service.

In Natural Language Processing Convolutional Neural Networks were first used by (Collobert and Weston 2008, Collobert, Weston et al. 2011). As an alternative to the local max pooling layer in the original LeNet model (LeCun, Bottou et al. 1998) they have used a new global max pooling layer which is shown to be effective for text. Furthermore, by co-training several deep models on various tasks, they proposed to transfer task specific information. With slight modification to (Collobert and Weston 2008) a simpler model was proposed by (Mikolov, Sutskever et al. 2013) which uses fixed, or in some cases fine-tuned, word embeddings, word2vec, and its combination as multi-channel. Their work demonstrates that even this simple model could achieve state-of-the-art performance on several small datasets. In order to handle variable length inputs as is common with text input, dynamic *k*-max pooling was proposed by (Kalchbrenner, Grefenstette et al. 2014), which is a generalization of the max pooling operator with *k* is dynamically set as part of the network.

All of the research mentioned previously uses word embeddings as their basis, which was introduced by (Bengio, Ducharme et al. 2003) in order to alleviate the curse of dimensionality by using distributed representations. The problem with these methods is that each sentence or even paragraphs only has a few number of words, thus preventing models from becoming very deep as input size is a limiting factor. It has been shown that word representation based inputs might not be inclusive for many inputs specially in social media as typos, hashtags, and other non-conventional writing habits are often seen (Severyn and Moschitti 2015). Hence, they proposed a convolutional model that would work on the character level which avoids the need for any word preprocessing or tokenization. Later on, (Conneau, Schwenk et al. 2016) introduced a much simpler model which could be made much deeper as a results.

Method

Here, we have focused on experimenting with several well-known Deep Learning architectures and methods for text classification such as different types of Recursive Neural Networks like LSTMs (Hochreiter and Schmidhuber 1997) and GRUs (Cho, Van Merriënboer et al. 2014, Chung, Gulcehre et al. 2014) as well as traditional CNNs (LeCun, Haffner et al. 1999) that have recently been shown to work well with text as well.

LSTMs and GRUS

LSTMs (Long Short-Term Memory) and GRUs (Gated Recurrent Unit) are both replacements for conventional RNN models that avoid the vanishing and exploding gradient problems. They also have another advantage, all recurrent models deal with sequential data it is better to have mechanisms available to control how much past elements from the sequence or past states can effect cell and hidden state updates. Both are widely used right now but they have a few differences that sets them apart. LSTMs use four gates called input, forget, cell, and output gates as well as two states by the names of cell and hidden states. The input gate decides on how of the new cell state should be kept and the forget gate decides how much of the current memory should be forgotten. The output gate oversees how much of the cell state should be given to the next layer, and the cell gate is a candidate for the hidden state which is calculated using a combination of the current input and the previous hidden state. GRUs similarly have reset, Input, and new gates and only the hidden state. Unlike LSTMs, GRUs don't have any internal memory or a cell state and neither do they have an output gate. The reset gate determines how the previous memory should be combined with the new input, and the update gate decides how much of the previous memory should be used.

- LSTMs have more gates and usually produce with more consistent results.
- GRUs have 1 fewer gate compared to LSTMs and are somewhat faster to train and mostly achieve better performance on smaller datasets.

LSTM

$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$	input gate
$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$	forget gate
$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$	cell gate
$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$	output gate
$c_t = f_t c_{(t-1)} + i_t g_t$	cell state
$h_t = o_t \tanh(c_t)$	hidden state

GRU

$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr})$	reset gate
$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz})$	input gate
$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t(W_{hn}h_{(t-1)} + b_{hn})\right)$	new gate
$h_t = (1 - z_t)n_t + z_t h_{(t-1)}$	hidden state

As you can see, GRUs have fewer parameters compared to LSTMs so during training they will finish training faster. Even though GRUs are more recent that LSTMs they are not necessarily better, and while they surpass LSTMs in some cases they still cannot consistently outperform them.

CNNs

LSTMs and GRUs are the standard in most sequential tasks like text, however, there is a big problem that they cannot be parallelized very well which in practice makes them much slower to use compared to parallelizable models like convolutional neural network (CNNs). In our work we are using a simple one-dimensional CNN model as one of our methods in the ensemble learning that we are doing. The CNN network actually performs really well as we will see in the experiments section.

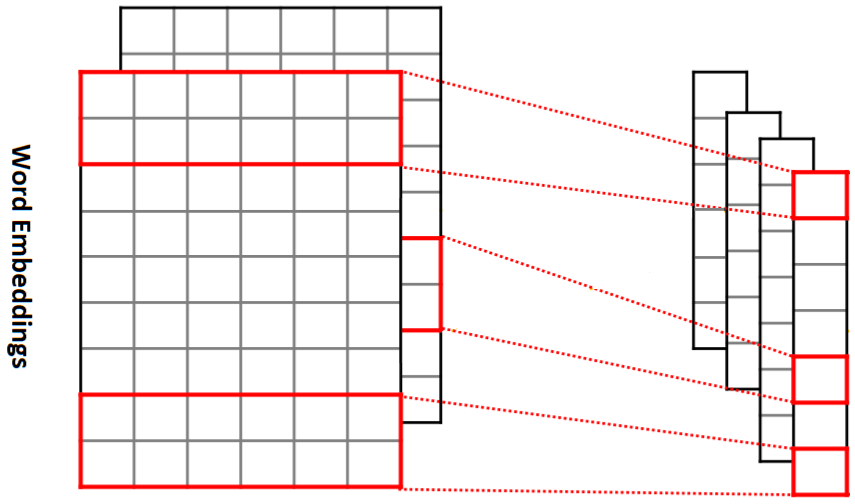


Figure 1 One dimensional Convolutional Neural Network on word embeddings.

Experiments

The key here is that while each individual network works well by its own, using an ensemble (Hansen and Salamon 1990) created by the output probability of all of the networks always gives us a probability that is as good as the best and often better than each one individually.

To be able to compare our results with previous work, we have used standard text classification datasets available at (http://goo.gl/JyCnZq), which we will explain in more detail later on.

As you know, most Machine Learning algorithms can only process numbers and not actual raw text. There are several methods to convert text input into a numeral format and then feed them to the algorithm for learning and analysis. One way would be to do this at the character level and look at each character as an individual entity or feature in the data. Another approach would be to take a word level approach, in which each word would have its own unique representation instead. As mentioned by (Le, Cerisara et al. 2017) word level implementations seem to be the most effective between these two models. Moreover, character level modelling means that each letter is considered one feature instead of a word, and as a result, each sentence is seen as a much larger input from the model’s point of view, because one sentence contains far more letters than words. This drastic increase in the number of features consequently requires the model to be deeper with more parameters to learn, which would not be practical.

Furthermore, the use of word level classification enables us the use of pretrained word embeddings, like GloVe (Pennington, Socher et al. 2014) or Word2Vec (Mikolov, Chen et al. 2013, Mikolov, Sutskever et al. 2013) instead of learning them from scratch. We have decided to use the Global Vectors for Word Representation (GloVe) for our word vector representations of words, these embeddings are one of the more recent ones trained on a very large dataset that was collected by Common Crawl[1] which maintains a corpus of web crawl data. We use the embeddings which have 300 dimensions.

One of the advantages of using word embedding vectors is that being pretrained on a gigantic dataset, they contain a lot of information, meaning, as well as context for each word, which would make the whole training process much faster and more efficient. In our experiments we have seen using these embeddings and fixing them throughout the training process give us the best results with the least overfitting. However, finetuning them by training them further using our own dataset during training makes our models significantly slower and prone to overfitting.

The CNN model has 3 one dimensional convolutional layers each followed by a one dimensional max pooling layer. We have used similar models for our BiLSTM and BiGRU models. We are using two layers stacked on top of each other and use only the last output from each direction from the second layer for our inference, because we are

[1] http://commoncrawl.org/

doing sentence classification we only need the last output from the LSTM or GRU and as we are using the Bidirectional models we need one output per direction. We have used a window size of 72, hidden dimension size 48 for LSTM and GRU and 196 for CNN, learning rate of 5e-4, regularization value of 2e-6 with a batchsize of 256 over 250 epochs to train all 3 models and for all 3 datasets. We decided to use Self Normalizing Linear Units (Klambauer, Unterthiner et al. 2017) for all of our activations other than the softmax layers at the end of the models which are used for classification. The convolving done in CNNs slides a filter window across the input which in this case with text it is sequentially going over the words. Depending on the filter size it would be similar to sequentially feeding the networks n-grams with n being the filter size.

Alternative Models: We compare our ensemble model with various state of the art as well as baseline models and report the results in Table 2. These models consist of:

- Naïve Bayes, which uses a simple count based unigram language model (Yogatama, Dyer et al. 2017)
- Kneser–Ney Bayes, similar to Naïve Bayes but uses a more sophisticated method that uses trigrams and Kneser-ney smoothing (Yogatama, Dyer et al. 2017)
- MLP Naïve Bayes, a version similar to Naïve Bayes that is an extension of the Naïve Bayes by using a feed forward neural network (Yogatama, Dyer et al. 2017)
- Bag of words, which uses most frequent words from training data (Zhang and Wallace 2015)
- Ngrams, similar to Bag of words but instead of words it uses most frequent ngrams (pairs of n words that appear together) (Zhang and Wallace 2015)

- Ngrams TFIDF, which is similar to the Ngrams model but instead uses TFIDF features (Text Frequency Inverse Document Frequency) (Zhang and Wallace 2015)
- Discriminative LSTM, the top performing model from (Yogatama, Dyer et al. 2017) that uses logistic regression on top of a normal LSTM model.
- Fasttext, very simply model that trains its own word embeddings which are the only parameters in the model that are trained for the actual classification, it has a rank constraint and a fast loss approximation (Joulin, Grave et al. 2016)
- Word-DenseNet, a word-level implementation of the Densenet architecture (Huang, Liu et al. 2017) from (Le, Cerisara et al. 2017), specifically chose one with most of the highest values for this comparison, specifically the Word-DenseNet N_b= (4−4−4−4) Global Average-Pooling model.
- Word shallow-and-wide CNN, the model with the best performance from (Le, Cerisara et al. 2017) in which 3 convolutional layers with varying filter sizes are used on the input and their outputs are concatenated together to be fed to a fully connected layer.

Dataset	# Training Samples	# Test Samples	# Classes
AG News	120,000	7,600	4
Yelp Review Polarity	560,000	7,600	2
DBPedia	560,000	60,000	14

Table 1 Text Classification Datasets

Datasets: We have used several baseline text classification datasets for out comparisons which are explained below:

- AG News, news articles from the internet consisting of titles as well as descriptions from 4 different classes, 120,000 training and 7,600 test samples (Del Corso, Gulli et al. 2005)
- Yelp Review Polarity, (Yelp Bin) From the Yelp Dataset challenge in 2015 that consists of 560,000 training and 7,600 test samples from 2 classes.
- DBPedia, 14 different classes have been chosen from DBPedia 2014 (Wikipedia), with 560,000 training and 60,000 test samples.

Model	AGNews	Yelp Bin	DBPedia
Naïve Bayes	90.0	86.0	96.0
Kneser–Ney Bayes	89.3	81.8	95.4
MLP Naïve Bayes	89.9	73.6	87.2
Discriminative LSTM	92.1	92.6	**98.7**
Fasttext	92.5	**95.7**	98.6
Word-DenseNet	91.7	95.8	**98.7**
Word shallow-and-wide CNN	92.2	95.8	**98.7**
CNN	92.9	94.0	98.6
BiLSTM	93.6	92.7	98.6
BiGRU	93.3	92.5	98.5
Ensemble	**93.8**	93.7	**98.7**

Table 2 Text Classification performance comparison on AG News, Yelp Binary, and DBPedia Datasets.

Moreover, Because the nature of our experiment is to boost our performance using ensembles of different models we have done several things to take these ensemblings even further. We have tried the ensemble of the best models based on the best validation error with the model at the last epoch, calling these best and final respectively. Furthermore, as we are using dropout layers in all of our models, because of the stochastic nature of dropout and its randomness we decided that at inference, aside from using the clean network without any dropout to get the probabilities for classification decision, we also used corrupted paths that still apply dropouts. We passed the test data once through the clean network and 19 times with dropouts and take their average and we call this method Noisy Ensemble, you can see that in the BiLSTM and BiGRU models this actually produces a significant boost in accuracy, this is done for both best models and final models, and finally for the last model we have averaged all of these probabilities for our final ensemble which we call Best+Final Noisy Ensemble and mostly produces the best overall result. These results are detailed in Table 3.

Model	Inference Method	AGNews	Yelp Bin	DBPedia
CNN	Final	92.4	93.9	98.5
	Best	92.7	**94.0**	98.5
	Best+Final	92.7	**94.0**	98.6
	Final Noisy Ensemble	92.9	93.9	98.6
	Best Noisy Ensemble	92.8	93.9	98.5
	Best+Final Noisy Ensemble	92.9	**94.0**	98.6
BiLSTM	Final	92.2	75.0	98.3
	Best	92.5	89.9	98.5
	Best+Final	92.5	87.1	98.5
	Final Noisy Ensemble	93.5	93.3	98.5
	Best Noisy Ensemble	93.3	90.3	98.5
	Best+Final Noisy Ensemble	93.6	92.7	98.6
BiGRU	Final	92.3	88.0	97.9
	Best	92.6	90.3	98.3
	Best+Final	92.5	90.3	98.3
	Final Noisy Ensemble	93.3	92.5	98.5
	Best Noisy Ensemble	93.0	91.6	98.4
	Best+Final Noisy Ensemble	93.3	92.5	98.5
Ensemble of all	Final	93.6	93.6	**98.7**
	Best	93.6	93.7	**98.7**
	Best+Final	93.7	93.8	**98.7**
	Final Noisy Ensemble	93.8	93.9	**98.7**
	Best Noisy Ensemble	93.6	93.3	98.6
	Best+Final Noisy Ensemble	**93.8**	93.7	**98.7**

Table 3 Classification performance comparison between our own models with different inference methods.

Chapter 3

Image Clustering

In this chapter we will be mostly going through our work published at ICCV 2017 (Dizaji, Herandi et al. 2017). As mentioned previously, clustering is one of the problems in Machine Learning and Computer vision that has been focused on less in the recent years. One of the main reasons for this is that in the practical case, clustering and more generally unsupervised learning means that the amount of information available on available data is far less than supervised learning and classification.

Deep Learning is preforming extremely well with supervised learning, but with deep learning and the current large complicated and complicated models, each one of these models has several hyperparameters the tuning of which could alter the end results significantly. In supervised problems this matter is extremely helpful and produces results much higher than anything before them. These hyper parameters range from the depth and the width of the networks to learning rates and regularization values. These hyperparameter tuning need to be done for each dataset individually to ensure best results. Even though, these tunings help supervised learning to a great extent, it is actually an obstacle for unsupervised learning.

In unsupervised learning, at least in real world problems, there is no label to be used for optimizing and selecting the best hyperparameters. This makes it important to create universal models that would work well with as many datasets as possible, meaning models as small and as simple as possible with the least number of hyperparameters to optimize. Before Deep Learning, methods like K-means and Agglomerative Clustering where the baselines for unsupervised learning, but with the need for methods that work efficiently with larger datasets while producing results usable in the industry, scalable algorithm like one's based on Deep Learning are becoming more and more desirable.

In this chapter we will be mostly going through our work published at ICCV 2017 (Dizaji, Herandi et al. 2017). As mentioned previously, clustering is one of the problems in Machine Learning and Computer vision that has been focused on less in the recent years. One of the main reasons for this is that in the practical case, clustering and more generally unsupervised learning means that the amount of information available on available data is far less than supervised learning and classification.

Deep Learning is preforming extremely well with supervised learning, but with deep learning and the current large complicated and complicated models, each one of these models has several hyperparameters the tuning of which could alter the end results significantly. In supervised problems this matter is extremely helpful and produces results much higher than anything before them. These hyper parameters range from the depth and the width of the networks to learning rates and regularization values. These hyperparameter tuning need to be done for each dataset individually to ensure best results. Even though, these tunings help supervised learning to a great extent, it is actually an obstacle for unsupervised learning.

In unsupervised learning, at least in real world problems, there is no label to be used for optimizing and selecting the best hyperparameters. This makes it important to create universal models that would work well with as many datasets as possible, meaning models as small and as simple as possible with the least number of hyperparameters to optimize. Before Deep Learning, methods like K-means and Agglomerative Clustering where the baselines for unsupervised learning, but with the need for methods that work efficiently with larger datasets while producing results usable in the industry, scalable algorithm like one's based on Deep Learning are becoming more and more desirable.

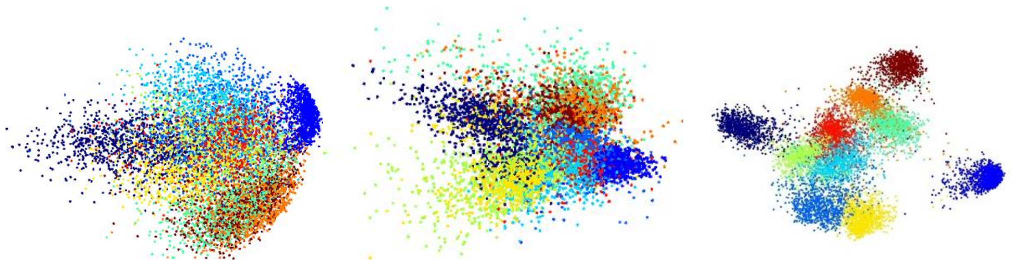
Here I will include our complete paper published in ICCV 2017 titled "Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization".

Authors:

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, Heng Huang

Introduction

Clustering is one of the fundamental topics in machine learning and computer vision research, and it has gained significant attention for discriminative representation of data points without any need for supervisory signals. The clustering problem has been extensively studied in various applications; however, the performance of standard clustering algorithms is adversely affected when dealing with high-dimensional data, and their time complexity dramatically increases when working with large-scale datasets. Tackling the curse of dimensionality, previous studies often initially project data into a low-dimensional manifold, and then cluster the embedded data in this new subspace (Roth and Lange 2004, Tian, Gao et al. 2014, Wang, Chang et al. 2016)}. Handling large-scale datasets, there are also several studies which select only a subset of data points to accelerate the clustering process (Shinnou and Sasaki 2008, Chen and Cai 2011).



(a) Raw data (b) NonJoint *DEPICT* (c) Joint *DEPICT*

Figure 2 Visualization to show the discriminative capability of embedding subspaces using *MNIST-test* data. (a) The space of raw data. (b) The embedding subspace of non-joint *DEPICT* using standard stacked denoising autoencoder (SdA). (c) The embedding subspace of joint *DEPICT* using our joint learning approach (MdA).

However, dealing with real-world image data, existing clustering algorithms suffer from different issues: 1) Using inflexible hand-crafted features (e.g. SIFT, HOG), which do not depend on the input data distribution; 2) Using shallow and linear embedding functions, which are not able to capture the non-linear nature of data; 3) Non-joint embedding and clustering processes, which do not result in an optimal embedding subspace for clustering; 4) Complicated clustering algorithms that require tuning the hyper-parameters using labeled data, which is not feasible in real-world clustering tasks.

To address the mentioned challenging issues, we propose a new clustering algorithm, called deep embedded regularized clustering (*DEPICT*), which exploits the advantages of both discriminative clustering methods and deep embedding models. *DEPICT* generally consists of two main parts, a multinomial logistic regression (soft-max) layer stacked on top of a multi-layer convolutional autoencoder. The soft-max layer along with the encoder pathway can be considered as a discriminative clustering model, which is trained using the relative entropy (*KL* divergence) minimization. We further add a regularization term to this objective based on a prior distribution for the frequency of cluster assignments. This regularization term penalizes unbalanced cluster assignments and prevents allocating clusters to outlier samples.

Although this deep clustering model is flexible enough to discriminate the complex real-world input data, it can easily get stuck in non-optimal local minima during training and result in undesirable cluster assignments. In order to avoid overfitting the deep clustering model to spurious data correlations, we utilize the reconstruction loss function of autoencoder models as a data-dependent regularization term for training parameters.

In order to benefit from a joint learning framework for embedding and clustering, we introduce a unified objective function including our clustering and auxiliary reconstruction loss functions. We then employ an alternating approach to efficiently update the parameters and estimate the cluster assignments. It is worth mentioning that in the standard learning approach for training a multi-layer autoencoder, the encoder and decoder parameters are first pretrained layer-wise using the reconstruction loss, and the encoder parameters are then fine-tuned using the objective function of the main task (Vincent, Larochelle et al. 2010). However, it has been argued that the non-joint fine-tuning step may overwrite the encoder parameters entirely and consequently cancel out the benefit of the layer-wise pretraining step (Zhao, Mathieu et al. 2015). To avoid this problem and achieve optimal joint learning results, we simultaneously train all of the encoder and decoder layers together along with the soft-max layer. To do so, we sum up the squared error reconstruction loss functions between the decoder and their corresponding (clean) encoder layers and add them to the clustering loss function.

Figure 1 demonstrates the importance of our joint learning strategy by comparing different data representations of *MNIST-test* data points (LeCun, Bottou et al. 1998) using principle component analysis (PCA) visualization. The first figure indicates the raw data representation; The second one shows the data points in the embedding subspace of non-joint *DEPICT*, in which the model is trained using the standard layer-wise stacked denoising autoencoder (SdA); The third one visualizes the data points in the embedding subspace of joint *DEPICT*, in which the model is trained using our multi-layer denoising autoencoder learning approach (MdA). As shown, joint *DEPICT* using MdA learning approach provides a significantly more discriminative embedding subspace compared to non-joint *DEPICT* using standard SdA learning approach.

Moreover, experimental results show that *DEPICT* achieves superior or competitive results compared to the state-of-the-art algorithms on the image benchmark datasets while having faster running times. In addition, we compared different learning strategies for *DEPICT*, and confirm that our joint learning approach has the best results. It should also be noted that *DEPICT* does not require any hyper-parameter tuning using supervisory signals, and consequently is a better candidate for the real-world clustering tasks.

Thus, we summarize the advantages of *DEPICT* as:

- Providing a discriminative non-linear embedding subspace via the deep convolutional autoencoder;
- Introducing an end-to-end joint learning approach, which unifies the clustering and embedding tasks, and avoids layer-wise pretraining;
- Achieving superior or competitive clustering results on high-dimensional and large-scale datasets with no need for hyper-parameter tuning using labeled data.

Related Works

There is a large number of clustering algorithms in literature, which can be grouped into different perspectives, such as hierarchical (Williams 2000, Heller and Ghahramani 2005, Zhang, Wang et al. 2012) centroid-based (Lloyd 1982, Bezdek, Ehrlich et al. 1984, Bahmani, Moseley et al. 2012, Nie, Wang et al. 2014),graph-based (Shi and Malik 2000, Nie, Wang et al. 2016, Nie, Wang et al. 2016, Wang, Nie et al. 2016), sequential (temporal) (Keogh, Chu et al. 2001, Sargin, Yemez et al. 2008, Zhou, De la Torre et al. 2013, Sadoughi and Busso 2015, Sadoughi, Liu et al. 2015) and subspace clustering models (Agrawal, Gehrke et al. 1998, Kailing, Kriegel et al. 2004, Gao, Nie et al. 2015, Nie and Huang 2016). In another sense, they are generally divided into two subcategories, generative and discriminative clustering algorithms. The generative algorithms like *K-means* and Gaussian mixture model (Biernacki, Celeux et al.

2000) explicitly represent the clusters using geometric properties of the feature space, and model the categories via the statistical distributions of input data. Unlike the generative clustering algorithms, the discriminative methods directly identify the categories using their separating hyperplanes regardless of data distribution. Information theoretic (Li, Zhang et al. 2004, Barber and Agakov 2006, Krause, Perona et al. 2010)\, max-margin (Xu, Neufeld et al. 2005, Zhao, Wang et al. 2008), and spectral graph (Ng, Jordan et al. 2002) algorithms are examples of discriminative clustering models. Generally it has been argued that the discriminative models often have better results compared to their generative counterparts, since they have fewer assumptions about the data distribution and directly separate the clusters, but their training can suffer from overfitting or getting stuck in undesirable local minima (Ng and Jordan 2002, Raina, Shen et al. 2004, Krause, Perona et al. 2010). Our *DEPICT* algorithm is also a discriminative clustering model, but it benefits from the auxiliary reconstruction task of autoencoder to alleviate this issue in training of our discriminative clustering algorithm.

There are also several studies regarding the combination of clustering with feature embedding learning. Ye et al. introduced a kernelized *K-means* algorithm, denoted by *DisKmeans*, where embedding to a lower dimensional subspace via linear discriminant analysis (*LDA*) is jointly learned with *K-means* cluster assignments (Ye, Zhao et al. 2008). Combination of linear embedding with spectral clustering is also presented in (Yang, Xu et al. 2010, Nie, Zeng et al. 2011). But these models all suffer from having shallow and linear embedding functions, which cannot represent the non-linearity of real-world data.

A joint learning framework for updating code books and estimating image clusters was proposed in (Xie and Xing 2015) while *SIFT* features are used as input data.

A deep structure, named *TAGnet* was introduced in (Wang, Chang et al. 2016), where two layers of sparse coding followed by a clustering algorithm are trained with an alternating learning approach. Similar work is presented in (Wang, Yang et al. 2015) that formulates a joint optimization framework for discriminative clustering and feature extraction using sparse coding. However, the inference complexity of sparse coding forces the model in (Wang, Yang et al. 2015) to reduce the dimension of input data with *PCA* and the model in (Wang, Chang et al. 2016) to use an approximate solution. Hand-crafted features and dimension reduction techniques degrade the clustering performance by neglecting the distribution of input data.

Tian et al. learned a non-linear embedding of the affinity graph using a stacked autoencoder, and then obtained the clusters in the embedding subspace via *K-means* (Tian, Gao et al. 2014). Trigeorgis et al. extended semi non-negative matrix factorization (*semi-NMF*) to stacked multi-layer (deep) *semi-NMF* to capture the abstract information in the top layer. Afterwards, they run *K-means* over the embedding subspace for cluster assignments (Trigeorgis, Bousmalis et al. 2014). More recently, Xie et al. employed denoising stacked autoencoder learning approach, and first pretrained the model layer-wise and then fine-tuned the encoder pathway stacked by a clustering algorithm using Kullback-Leibler divergence minimization (Xie, Girshick et al. 2016). Unlike these models that require layer-wise pretraining as well as non-joint embedding and clustering learning, *DEPICT* utilizes an end-to-end optimization for training all network layers simultaneously using the unified clustering and reconstruction loss functions.

Yang et al. introduced a new clustering model, named *JULE*, based on a recurrent framework, where data is represented via a convolutional neural network and embedded data is iteratively clustered using an agglomerative clustering algorithm (Yang, Parikh et al. 2016). They derived a unified loss function consisting of the merging process for agglomerative clustering and updating the parameters of the deep representation. While *JULE* achieved good results using the joint learning approach, it requires tuning of a large number of hyper-parameters, which is not practical in real-world clustering tasks. In contrast, our model does not need any supervisory signals for hyper-parameter tuning.

Deep Embedded Regularized Clustering

In this section, we first introduce the clustering objective function and the corresponding optimization algorithm, which alternates between estimating the cluster assignments and updating model parameters. Afterwards, we show the architecture of *DEPICT* and provide the joint learning framework to simultaneously train all network layers using the unified clustering and reconstruction loss functions.

DEPICT Algorithm

Let's consider the clustering task of N samples, $\mathrm{X} = [\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}]$, into K categories, where each sample $\mathrm{x}_\mathrm{i} \in \mathrm{R}^{\mathrm{d}_\mathrm{x}}$. Using the embedding function, $\varphi_\mathrm{W}: \mathrm{X} \rightarrow \mathrm{Z}$, we are able to map raw samples into the embedding subspace $\mathrm{Z} = [\mathrm{z}_1, \dots, \mathrm{z}_\mathrm{n}]$, where each $\mathrm{z}_\mathrm{i} \in \mathrm{R}^{\mathrm{d}_\mathrm{z}}$ has a much lower dimension compared to the input data (i.e. $\mathrm{d}_\mathrm{z} \ll \mathrm{d}_\mathrm{x}$). Given the embedded features, we use a multinomial logistic regression (soft-max) function $\mathrm{f}_\theta: \mathrm{Z} \rightarrow \mathrm{Y}$ to predict the probabilistic cluster assignments as follows.

$$p_{ik} = P(y_i = k|\mathbf{z_i}, \boldsymbol{\Theta}) = \frac{exp\left(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{T}}\mathbf{z_i}\right)}{\sum_{k'=1}^{K} exp\left(\boldsymbol{\theta}_{\boldsymbol{k'}}^{\boldsymbol{T}}\mathbf{z_i}\right)}$$

1

where $\Theta = [\theta_1, \dots, \theta_\mathrm{k}] \in \mathrm{R}^{\mathrm{d}_\mathrm{z} \times \mathrm{K}}$ are the soft-max function parameters, and p_{ik} indicates the probability of the i-th sample belonging to the k-th cluster.

In order to define our clustering objective function, we employ an auxiliary target variable $\mathbf{Q}$ to refine the model predictions iteratively. To do so, we first use Kullback-Leibler ($\mathbf{KL}$) divergence to decrease the distance between the model prediction P and the target variable Q.

$$\mathcal{L} = \mathrm{KL}(\mathrm{Q}|\mathrm{P}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{q_{ik}} \log\frac{\mathrm{q_{ik}}}{\mathrm{p_{ik}}}$$

2

In order to avoid degenerate solutions, which allocate most of the samples to a few clusters or assign a cluster to outlier samples, we aim to impose a regularization term to the target variable. To this end, we first define the empirical label distribution of target variables as:

$$\mathrm{f_k} = \mathrm{P}(\mathrm{y} = \mathrm{k}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}} \mathrm{q_{ik}}$$

3

where f_k can be considered as the soft frequency of cluster assignments in the target distribution. Using this empirical distribution, we are able to enforce our preference for having balanced assignments by adding the following *KL* divergence to the loss function.

$$\begin{gathered}\mathcal{L} = \mathrm{KL}(\mathrm{Q}||\mathrm{P}) + \mathrm{KL}(\mathrm{f}||\mathrm{u}) \\ = \left[\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{q_{ik}} \log\frac{\mathrm{q_{ik}}}{\mathrm{p_{ik}}}\right] + \left[\frac{1}{\mathrm{N}}\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{f_k} \log\frac{\mathrm{f_k}}{\mathrm{u_k}}\right] \\ = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{q_{ik}} \log\frac{\mathrm{q_{ik}}}{\mathrm{p_{ik}}} + \mathrm{q_{ik}} \log\frac{\mathrm{f_k}}{\mathrm{u_k}}\end{gathered}$$

4

where u is the uniform prior for the empirical label distribution. While the first term in the objective minimizes the distance between the target and model prediction distributions, the second term balances the frequency of clusters in the target variables.

Utilizing the balanced target variables, we can force the model to have more balanced predictions (cluster assignments) P indirectly. It is also simple to change the prior from the uniform distribution to any arbitrary distribution in the objective function if there is any extra knowledge about the frequency of clusters.

An alternating learning approach is utilized to optimize the objective function. Using this approach, we estimate the target variables Q via fixed parameters (expectation step), and update the parameters while the target variables Q are assumed to be known (maximization step). The problem to infer the target variable Q has the following objective:

$$\min_{\mathrm{Q}} \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{q_{ik}} \log \frac{\mathrm{q_{ik}}}{\mathrm{p_{ik}}} + \mathrm{q_{ik}} \log \frac{\mathrm{f_k}}{\mathrm{u_k}}$$

5

where the target variables are constrained to $\sum_{\mathrm{k}} \mathrm{q_{ik}} = 1$. This problem can be solved using first order methods, such as gradient descent, projected gradient descent, and Nesterov optimal method (Nesterov 2013), which only require the objective function value and its (sub)gradient at each iteration. In the following equation, we show the partial derivative of the objective function with respect to the target variables.

$$\frac{\partial \mathcal{L}}{\partial \mathrm{q_{ik}}} \propto \log\left(\frac{\mathrm{q_{ik} f_k}}{\mathrm{p_{ik}}}\right) + \frac{\mathrm{q_{ik}}}{\sum_{\mathrm{i'}=1}^{\mathrm{N}} \mathrm{q_{i'k}}} + 1$$

6

Investigating this problem more carefully, we approximate the gradient in Eq. 6 by removing the second term, since the number of samples N is often big enough to ignore the second term. Setting the gradient equal to zero, we are now able to compute the closed form solution for Q accordingly.

$$q_{ik} = \frac{p_{ik}/(\sum_{i'} p_{i'k})^{\frac{1}{2}}}{\sum_{k'} p_{ik'}/(\sum_{i'} p_{i'k'})^{\frac{1}{2}}}$$

7

For the maximization step, we update the network parameters $\boldsymbol{\psi} = (\boldsymbol{\Theta}, \mathbf{W})$ using the estimated target variables with the following objective function.

$$\min_{\psi} -\frac{1}{N}\sum_{i=1}^{N}\sum 1_{k=1}^{K} q_{ik} \log p_{ik}$$

8

Interestingly, this problem can be considered as a standard cross entropy loss function for classification tasks, and the parameters of soft-max layer Θ and embedding function W can be efficiently updated by backpropagating the error.

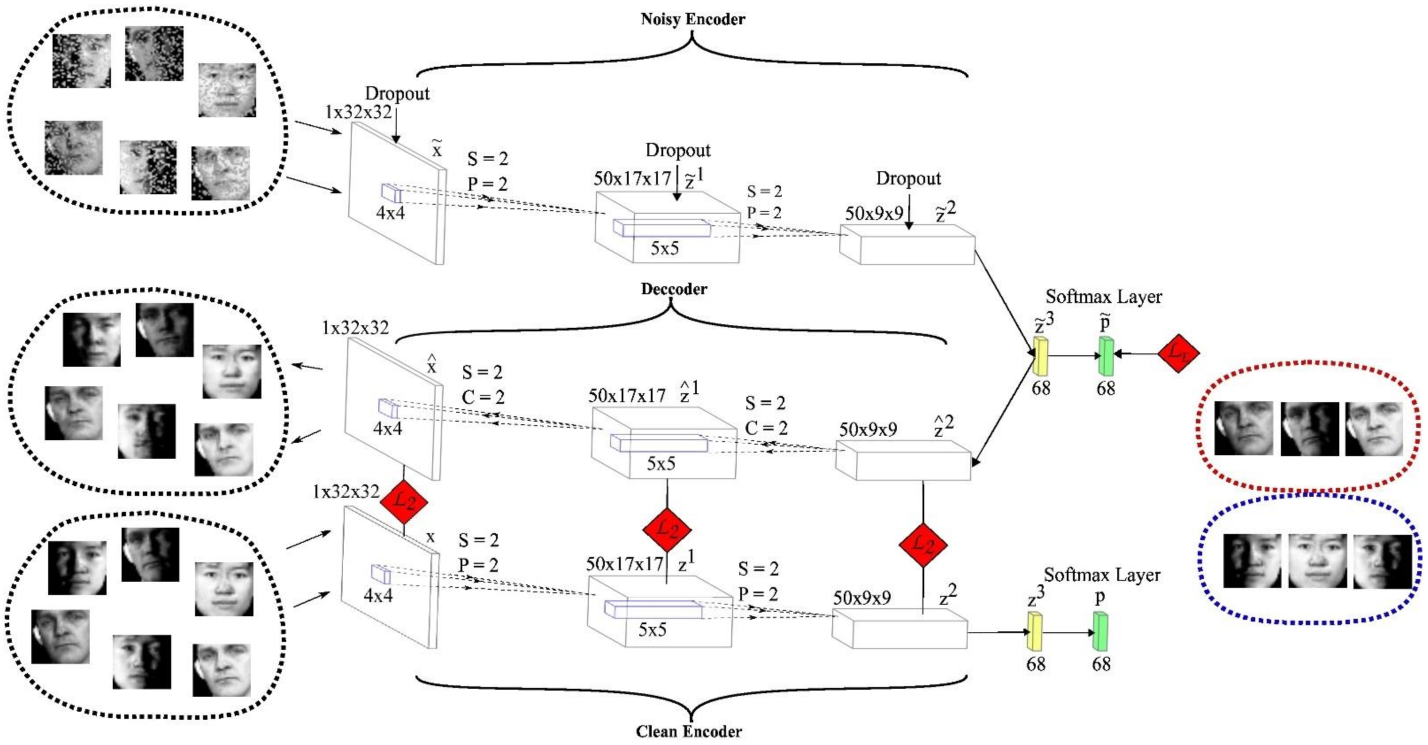


Figure 3 Architecture of *DEPICT* for *CMU-PIE* dataset. *DEPICT* consists of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. In order to illustrate the joint learning framework, we consider the following four pathways for *DEPICT*: Noisy (corrupted) encoder, Decoder, Clean encoder and Soft-max layer. The clustering loss function, L_E, is applied on the noisy pathway, and the reconstruction loss functions, L_2, are between the decoder and clean encoder layers. The output size of convolutional layers, kernel sizes, strides (S), paddings (P) and crops (C) are also shown.

DEPICT Architecture

In this section, we extend our general clustering loss function using a denoising autoencoder. The deep embedding function is useful for capturing the non-linear nature of input data; However, it may overfit to spurious data correlations and get stuck in undesirable local minima during training. To avoid this overfitting, we employ autoencoder structures and use the reconstruction loss function as a data-dependent regularization for training the parameters. Therefore, we design *DEPICT* to consist of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. Due to the promising performance of strided convolutional layers in (Radford, Metz et al. 2015, Yeh, Chen et al. 2016), we employ convolutional layers in our encoder and strided convolutional layers in the decoder pathways, and avoid deterministic spatial pooling layers (like max-pooling). Strided convolutional layers allow the network to learn its own spatial upsampling, providing a better generation capability.

Unlike the standard learning approach for denoising autoencoders, which contains layer-wise pretraining and then fine-tuning, we simultaneously learn all of the autoencoder and soft-max layers. As shown in Figure 3, *DEPICT* consists of the following components:

1. Corrupted feedforward (encoder) pathway maps the noisy input data into the embedding subspace using a few convolutional layers followed by a fully connected layer. The following equation indicates the output of each layer in the noisy encoder pathway.

$$\widetilde{z^{l}} = \mathrm{Dropout}\left[g\left(W_e^l \widetilde{z^{l-1}}\right)\right] \quad 9$$

where $\widetilde{z^{l}}$ are the noisy features of the l-th layer, $\mathrm{Dropout}$ is a stochastic mask function that randomly sets a subset of its inputs to zero (Srivastava, Hinton et al. 2014), g is the activation function of convolutional or fully connected layers, and W_e^l indicates the weights of the l-th layer in the encoder. Note that the first layer features, $\widetilde{\mathrm{z}^0}$, are equal to the noisy input data, $\tilde{\mathrm{x}}$.

2. Followed by the corrupted encoder, the decoder pathway reconstructs the input data through a fully connected and multiple strided convolutional layers as follows,

$$\widehat{\mathrm{z}^{\mathrm{l}-1}} = \mathrm{g}\left(\mathrm{W}_{\mathrm{d}}^{\mathrm{l}}\widehat{\mathrm{z}^{\mathrm{l}}}\right)$$

10

where $\widehat{\mathrm{z}^{\mathrm{l}}}$ is the l-th reconstruction layer output, and $\mathrm{W}_{\mathrm{d}}^{\mathrm{l}}$ shows the weights for the l-th layer of the decoder. Note that input reconstruction, $\hat{\mathrm{x}}$, is equal to $\widehat{\mathrm{z}^0}$.

3. Clean feedforward (encoder) pathway shares its weights with the corrupted encoder, and infers the clean embedded features. The following equation shows the outputs of the clean encoder, which are used in the reconstruction loss functions and obtaining the final cluster assignments.

$$\mathrm{z}^{\mathrm{l}} = \mathrm{g}\left(\mathrm{W}_{\mathrm{e}}^{\mathrm{l}}\mathrm{z}^{\mathrm{l}-1}\right)$$

11

where z^{l} is the clean output of the l-th layer in the encoder. Consider the first layer features z^0 equal to input data x.

4. Given the top layer of the corrupted and clean encoder pathways as the embedding subspace, the soft-max layer obtains the cluster assignments using eq. 1.

Note that we compute target variables Q using the clean pathway, and model prediction $\widetilde{\mathrm{P}}$ via the corrupted pathway. Hence, the clustering loss function $\mathrm{KL}\left(\mathrm{Q}||\widetilde{\mathrm{P}}\right)$ forces the model to have invariant features with respect to noise. In other words, the model is assumed to have a dual role: a clean model, which is used to compute the more accurate target variables; and a noisy model, which is trained to achieve noise-invariant predictions.

As a crucial point, *DEPICT* algorithm provides a joint learning framework that optimizes the soft-max and autoencoder parameters together.

$$\min_{\psi} -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik} \log \widetilde{p_{ik}} + \frac{1}{N}\sum_{i=1}^{N}\sum_{l=0}^{L-1}\frac{1}{|z_i^l|}||\, z_i^l - \widehat{z_i^l}||_2^2$$

12

where $|z_i^l|$ is the output size of the l-th hidden layer (input for $l = 0$), and L is the depth of the autoencoder model.

The benefit of joint learning frameworks for training multi-layer autoencoders is also reported in semi-supervised classification tasks (Rasmus, Berglund et al. 2015, Zhao, Mathieu et al. 2015). However, *DEPICT* is different from previous studies, since it is designed for the unsupervised clustering task, it also does not require max-pooling switches used in stacked what-where autoencoder (SWWAE) (Zhao, Mathieu et al. 2015), and lateral (skip) connections between encoder and decoder layers used in ladder network (Rasmus, Berglund et al. 2015). **Algorithm 1** shows a brief description of *DEPICT* algorithm.

Algorithm 1 *DEPICT* Algorithm

```
Initialize $\mathbf{Q}$ using a clustering algorithm
While not converged do
    $\min_{\psi} -\frac{1}{N}\sum_{ik} q_{ik} \log \widetilde{p_{ik}} + \frac{1}{N}\sum_{il} \frac{1}{|z_i^l|} |z_i^l - \widehat{z_i^l}|_2^2$
    $p_{ik}^{(t)} \propto \exp(\theta_k^T z_i^L)$
    $q_{ik}^{(t)} \propto p_{ik} / \left(\sum_{i'} p_{i'k}\right)^{\frac{1}{2}}$
end
```

Experiments

In this section, we first evaluate *DEPICT*[2] in comparison with state-of-the-art clustering methods on several benchmark image datasets. Then, the running speed of the best clustering models are compared. Moreover, we examine different learning approaches for training *DEPICT*. Finally, we analyze the performance of *DEPICT* model on semi-supervised classification tasks.

Datasets:

In order to show that *DEPICT* works well with various kinds of datasets, we have chosen the following handwritten digit and face image datasets. Considering that clustering tasks are fully unsupervised, we concatenate the training and testing samples when applicable.

MNIST-full: A dataset containing a total of 70,000 handwritten digits with 60,000 training and 10,000 testing samples, each being a 32 by 32 monochrome image (LeCun, Bottou et al. 1998).

MNIST-test: A dataset which only consists of the testing part of *MNIST-full* data.

[2] Our code is available in https://github.com/herandy/DEPICT

USPS: It is a handwritten digits dataset from the *USPS* postal service, containing 11,000 samples of 16 by 16 images.

CMU-PIE: A dataset including 32 by 32 face images of 68 people with 4 different expressions (Sim, Baker et al. 2002).

Youtube-Face (YTF): Following (Yang, Parikh et al. 2016), we choose the first 41 subjects of YTF dataset. Faces inside images are first cropped and then resized to 55 by 55 sizes (Wolf, Hassner et al. 2011).

FRGC: Using the 20 random selected subjects in (Yang, Parikh et al. 2016) from the original dataset, we collect 2,462 face images. Similarly, we first crop the face regions and resize them into 32 by 32 images.

Dataset	# Samples	# Classes	# Dimensions
MNIST-full	70,000	10	1×28×28
MNIST-test	10,000	10	1×28×28
USPS	11,000	10	1×16×16
FRGC	2,462	20	3×32×32
YTF	10,000	41	3×55×55
CMU-PIE	2,856	68	1×32×32

Table 4 Clustering Dataset Descriptions

Clustering Metrics: We have used 2 of the most popular evaluation criteria widely used for clustering algorithms, accuracy (ACC) and normalized mutual information (NMI). The best mapping between cluster assignments and true labels is computed using the Hungarian algorithm (Kuhn 1955) to measure accuracy. NMI calculates the normalized measure of similarity between two labels of the same data (Xu, Liu et al. 2003). Results of NMI do not change by permutations of clusters (classes), and they are normalized to have $[0,1]$ range, with 0 meaning no correlation and 1 exhibiting perfect correlation.

Evaluation of Clustering Algorithms

Alternative Models: We compare our clustering model, *DEPICT*, with several baseline and state-of-the-art clustering algorithms, including K*-means*, normalized cuts (*N-Cuts*) (Shi and Malik 2000), self-tuning spectral clustering (*SC-ST*) (Zelnik-Manor and Perona 2005), large-scale spectral clustering (*SC-LS*) (Chen and Cai 2011), graph degree linkage-based agglomerative clustering (*AC-GDL*) (Zhang, Wang et al. 2012), agglomerative clustering via path integral (*AC-PIC*) (Zhang, Zhao et al. 2013), spectral embedded clustering (*SEC*) (Nie, Zeng et al. 2011), local discriminant models and global integration (*LDMGI*) (Yang, Xu et al. 2010), *NMF* with deep model (*NMF-D*) (Trigeorgis, Bousmalis et al. 2014), task-specific clustering with deep model (*TSC-D*) (Wang, Chang et al. 2016), deep embedded clustering (*DEC*) (Xie, Girshick et al. 2016), and joint unsupervised learning (*JULE*) (Yang, Parikh et al. 2016).

Implementation Details: We use a common architecture for *DEPICT* and avoid tuning any hyper-parameters using the labeled data in order to provide a practical algorithm for real-world clustering tasks. For all datasets, we consider two convolutional layers followed by a fully connected layer in encoder and decoder pathways. While for all convolutional layers, the feature map size is 50 and the kernel size is about 5×5, the dimension of the embedding subspace is set equal to the number of clusters in each dataset. We also pick the proper stride, padding and crop to have an output size of about 10×10 in the second convolutional layer. Inspired by (Radford, Metz et al. 2015), we consider leaky rectified (leaky RELU) non-linearity (Maas, Hannun et al. 2013) as the activation function of convolutional and fully connected layers, except in the last layer of encoder and first layer of decoder, which have Tanh non-linearity functions. Consequently, we normalize the image intensities to be in the range of $[-1, 1]$. Moreover, we set the learning rate and dropout to 10^{-4} and 0.1 respectively, adopt adam as our optimization method with the default hyper-parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\mathrm{e}-08$ (Kingma and Ba 2014). The weights of convolutional and fully connected layers are all initialized by Xavier approach (Glorot and Bengio 2010). Since the clustering assignments in the first iterations are random and not reliable for clustering loss, we first train *DEPICT* without clustering loss function for a while, then initialize the clustering assignment q_{ik} by clustering the embedding subspace features via simple algorithms like K*-means* or *AC-PIC*.

Quantitative Comparison: We run *DEPICT* and other clustering methods on each dataset. We followed the implementation details for *DEPICT* and report the average results from 5 runs. For the rest, we present the best reported results either from their original papers or from (Yang, Parikh et al. 2016). For unreported results on specific datasets, we run the released code with hyper-parameters mentioned in the original papers, these results are marked by (*) on top. But, when the code is not publicly available, or running the released code is not practical, we put dash marks (-) instead of the corresponding results. Moreover, we mention the number of hyper-parameters that are tuned using supervisory signals (labeled data) for each algorithm. Note that this number only shows the quantity of hyper-parameters, which are set differently for various datasets for better performance.

Table 5 reports the clustering metrics, normalized mutual information (NMI) and accuracy (ACC), of the algorithms on the aforementioned datasets. As shown, *DEPICT* outperforms other algorithms on four datasets and achieves competitive results on the remaining two. It should be noted that we think hyper-parameter tuning using supervisory signals is not feasible in real-world clustering tasks, and hence *DEPICT* is a significantly better clustering algorithm compared to the alternative models in practice. For example, *DEC*, *SEC*, and *LDMGI* report their best results by tuning one hyper-parameter over nine different options, and *JULE-SF* and *JULE-RC* achieve their good performance by tweaking several hyper-parameters over various datasets. However, we do not tune any hyper-parameters for *DEPICT* using the labeled data and only report the result with the same (default) hyper-parameters for all datasets.

Dataset	*MNIST-full*		*MNIST-test*		*USPS*		*FRGC*		*YTF*		*CMU-PIE*		#Tuned HPs
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	
K-means	0.500*	0.534*	0.501*	0.547*	0.450*	0.460*	0.287*	0.243*	0.776*	0.601*	0.432*	0.223*	0
N-Cuts	0.411	0.327	0.753	0.304	0.675	0.314	0.285	0.235	0.742	0.536	0.411	0.155	0
SC-ST	0.416	0.311	0.756	0.454	0.726	0.308	0.431	0.358	0.620	0.290	0.581	0.293	0
SC-LS	0.706	0.714	0.756	0.740	0.681	0.659	0.550	0.407	0.759	0.544	0.788	0.549	0
AC-GDL	0.017	0.113	0.844	0.933	0.824	0.867	0.351	0.266	0.622	0.430	0.934	0.842	1
AC-PIC	0.017	0.115	0.853	0.920	0.840	0.855	0.415	0.320	0.697	0.472	0.902	0.797	0
SEC	0.779*	0.804*	0.790*	0.815*	0.511*	0.544*	-	-	-	-	-	-	1
LDMGI	0.802*	0.842*	0.811*	0.847*	0.563*	0.580*	-	-	-	-	-	-	1
NMF-D	0.152*	0.175*	0.241*	0.250*	0.287*	0.382*	0.259*	0.274*	0.562*	0.536*	0.920*	0.810*	0
TSC-D	0.651	0.692	-	-	-	-	-	-	-	-	-	-	2
DEC	0.816*	0.844*	0.827*	0.859*	0.586*	0.619*	0.505*	0.378*	0.446*	0.371*	0.924*	0.801*	1
JULE-SF	0.906	0.959	0.876	0.940	0.858	0.922	0.566	0.461	**0.848**	**0.684**	0.984	0.980	3
DEPICT	**0.917**	**0.965**	**0.915**	**0.963**	**0.927**	**0.964**	**0.610**	**0.470**	0.802	0.621	0.974	0.883	0

Table 5 Clustering performance of different algorithms on image datasets based on accuracy (ACC) and normalized mutual information (NMI). The numbers of tuned hyper-parameters (# tuned HPs) using the supervisory signals are also shown for each algorithm. The results of alternative models are reported from

original papers, except the ones marked by ($*$) on top, which are obtained by us running the released code. We put dash marks (-) for the results that are not practical to obtain.

Running Time Comparison

In order to evaluate the efficiency of our clustering algorithm in dealing with large-scale and high dimensional data, we compare the running speed of *DEPICT* with its competing algorithms, *JULE-SF* and *JULE-RC*. Moreover, the fast versions of *JULE-SF* and *JULE-RC* are also evaluated. Note that *JULE-SF*(fast) and *JULE-RC*(fast) both require tuning one extra hyper-parameter for each dataset to achieve results similar to the original *JULE* algorithms in Table 5 (Yang, Parikh et al. 2016). We run *DEPICT* and the released code for *JULE* algorithms[3] on a machine with one Titan X pascal GPU and a Xeon E5-2699 CPU.

[3] https://github.com/jwyang/JULE-Torch

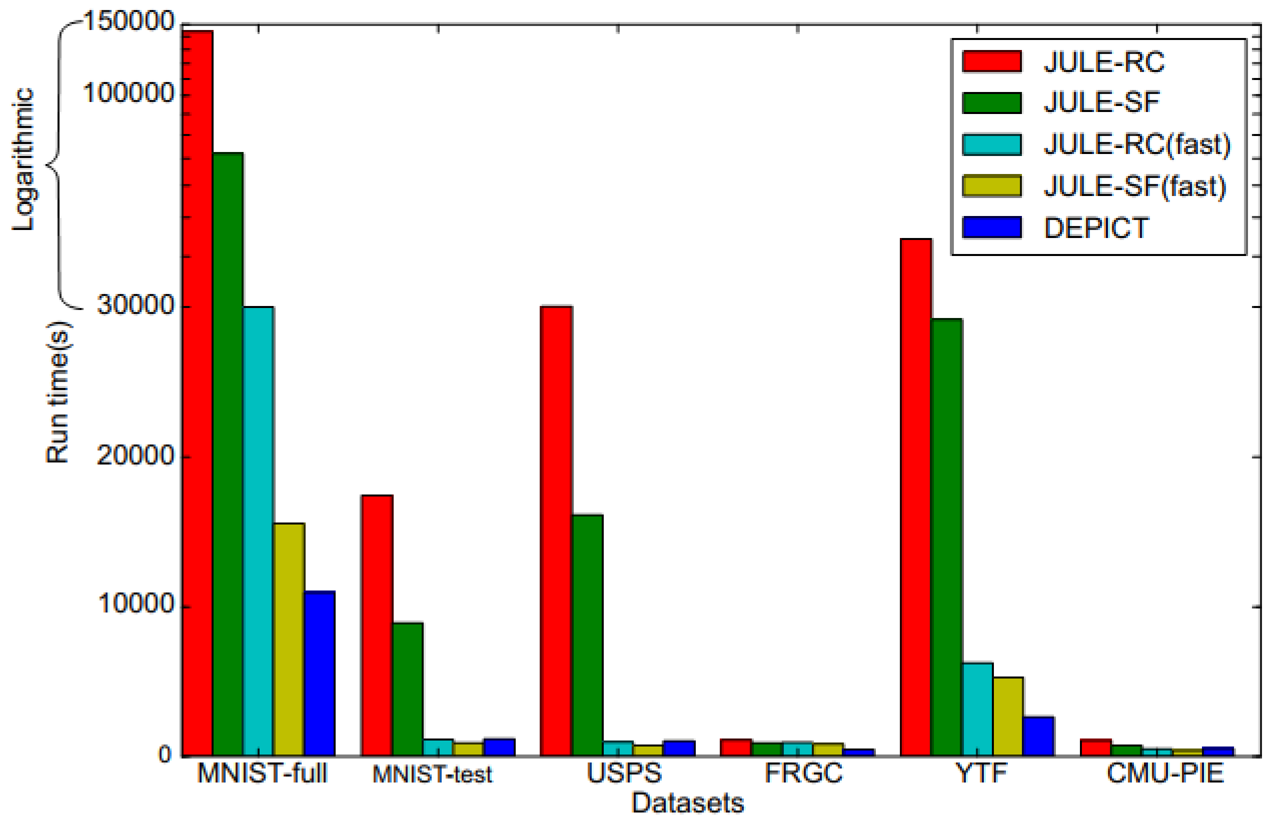


Figure 4 Running time comparison of DEPICT and JULE clustering algorithms on image datasets.

Figure 4 illustrates the running time for *DEPICT* and *JULE* algorithms on all datasets. Note that running times of *JULE-SF* and *JULE-RC* are shown linearly from 0 to 30,000 and logarithmically for larger values for the sake of readability. In total, *JULE-RC*, *JULE-SF*, *JULE-RC*(fast), *JULE-SF*(fast) and *DEPICT* take 66.1, 35.5, 11.0, 6.6 and 4.7 hours respectively to run over all datasets. While all algorithms have approximately similar running times on small datasets (*FRGC* and *CMU-PIE*), when dealing with the large-scale and high-dimensional datasets (*MNIST-full* and *YTF*), *DEPICT* almost shows a linear increase in the running time, but the running times of original JULE algorithms dramatically grow with the size and number of input data. This outcome again emphasizes the practicality of *DEPICT* for real-world clustering tasks.

Evaluation of Learning Approach

In order to evaluate our joint learning approach, we compare several strategies for training *DEPICT*. For training a multi-layer convolutional autoencoder, we analyze the following three approaches : 1) Standard stacked denoising autoencoder (SdA), in which the model is first pretrained using the reconstruction loss function in a layer-wise manner, and the encoder pathway is then fine-tuned using the clustering objective function (Vincent, Larochelle et al. 2010). 2) Another approach (RdA) is suggested in (Xie, Girshick et al. 2016) to improve the SdA learning approach, in which all of the autoencoder layers are retrained after the pretraining step, only using the reconstruction of input layer while data is not corrupted by noise. The fine-tuning step is also done after the retraining step. 3) Our learning approach (MdA), in which the whole model is trained simultaneously using the joint reconstruction loss functions from all layers along with the clustering objective function.

Furthermore, we also examine the effect of clustering loss (through error back-prop) in constructing the embedding subspace. To do so, we train a similar multi-layer convolutional autoencoder (*Deep-ConvAE*) only using the reconstruction loss function to generate the embedding subspace. Then, we run the best shallow clustering algorithm (*AC-PIC*) on the embedded data. Hence, this model (*Deep-ConvAE+AC-PIC*) differs from *DEPICT* in the sense that its embedding subspace is only constructed using the reconstruction loss and does not involve the clustering loss.

Dataset		MNIST-full		MNIST-test		USPS		FRGC		YTF		CMU-PIE	
		NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
Deep-ConvAE + AC-PIC	SdA	0.255	0.348	0.313	0.345	0.223	0.290	0.120	0.230	0.414	0.302	0.354	0.266
	RdA	0.615	0.455	0.859	0.900	0.886	0.866	0.443	0.363	0.597	0.425	0.912	0.817
	MdA	0.729	0.506	0.876	0.942	0.906	0.878	0.583	0.427	0.640	0.448	0.931	0.883
DEPICT	SdA	0.365	0.427	0.353	0.390	0.328	0.412	0.211	0.300	0.414	0.302	0.354	0.266
	RdA	0.808	0.677	0.899	0.950	0.901	0.923	0.551	0.444	0.652	0.450	0.951	0.926
	MdA	**0.917**	**0.965**	**0.915**	**0.963**	**0.927**	**0.964**	**0.610**	**0.470**	**0.802**	**0.621**	**0.974**	**0.883**

Table 6 Clustering performance of different learning approaches, including SdA, RdA and MdA, for training *DEPICT* and *Deep-ConvAE+AC-PIC* models.

Table 6 indicates the results of *DEPICT* and *Deep-ConvAE+AC-PIC* when using the different learning approaches. As expected, *DEPICT* trained by our joint learning approach (MdA) consistently outperforms the other alternatives on all datasets. Interestingly, MdA learning approach shows promising results for *Deep-ConvAE+ AC-PIC* model, where only reconstruction losses are used to train the embedding subspace. Thus, our learning approach is an efficient strategy for training autoencoder models due to its superior results and fast end-to-end training.

Semi-Supervised Classification Performance

Representation learning in an unsupervised manner or using a small number of labeled data has recently attracted great attention. Due to the potential of our model in learning a discriminative embedding subspace, we evaluate *DEPICT* in a semi-supervised classification task. Following the semi-supervised experiment settings (Rasmus, Berglund et al. 2015, Zhao, Mathieu et al. 2015), we train our model using a small random subset of *MNIST-training* dataset as labeled data and the remaining as unlabeled data. The classification error of *DEPICT* is then computed using the *MNIST-test* dataset, which is not seen during training. Compared to our unsupervised learning approach, we only utilize the clusters corresponding to each labeled data in training process. In particular, only for labeled data, the cluster labels (assignments) are set using the best map technique from the original classification labels once, and then they will be fixed during the training step.

Model	100	1000	3000
T-SVM (Vapnik 1999)	16.81	5.38	3.45
CAE (Rifai, Vincent et al. 2011)	13.47	4.77	3.22
MTC (Rifai, Dauphin et al. 2011)	12.03	3.64	2.57
PL-DAE (Lee 2013)	10.49	3.46	2.69
AtlasRBF (Pitelis, Russell et al.	8.10	3.68	-
M1+M2 (Kingma, Mohamed et	3.33±0.14	2.40±0.05	2.18±0.04
SWWAE (Zhao, Mathieu et al. 2015)	8.71±0.34	2.83±0.10	2.10±0.22
Ladder (Rasmus, Berglund et	1.06±0.37	0.84±0.08	-
DEPICT	2.65±0.35	2.10±0.11	1.91±0.06

Table 7 Comparison of *DEPICT* and several semi-supervised classification models in *MNIST* dataset with different numbers of labeled data.

Table 7 shows the error results for several semi-supervised classification models using different numbers of labeled data. Surprisingly, *DEPICT* achieves comparable results with the state-of-the-art, despite the fact that the semi-supervised classification models use 10,000 validation data to tune their hyper-parameters, *DEPICT* only employs the labeled training data (e.g. 100) and does not tune any hyper-parameters. Although *DEPICT* is not mainly designed for classification tasks, it outperforms several models including *SWWAE* (Zhao, Mathieu et al. 2015), *M1+M2* (Kingma, Mohamed et al. 2014), and *AtlasRBF* (Pitelis, Russell et al. 2014), and has comparable results with the complicated *Ladder network* (Rasmus, Berglund et al. 2015). These results further confirm the discriminative quality of the embedding features of *DEPICT*.

Chapter 4

Conclusion

We have utilized various ensembling methods with the use of well-known models for Natural Language Processing tasks to build a robust and efficient classifier for text data. We have used CNNs, LSTMs, GRUs, as well as the inherent ensembling nature of dropouts while making use of both best and last models to build this model.

We proposed a new deep clustering model, *DEPICT*, consisting of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. We employed a regularized relative entropy loss function for clustering, which leads to balanced cluster assignments. Adopting our autoencoder reconstruction loss function enhanced the embedding learning. Furthermore, a joint learning framework was introduced to train all network layers simultaneously and avoid layer-wise pretraining. Experimental results showed that *DEPICT* is a good candidate for real-world clustering tasks, since it achieved superior or competitive results compared to alternative methods while having faster running speed and not needing hyper-parameter tuning. Efficiency of our joint learning approach was also confirmed in clustering and semi-supervised classification tasks.

Chapter 5

Future Work

Although word embeddings like GloVe provide models with a lot of analysis power, these methods are by no means ideal. Even though the data used to train these embeddings was enormous and contained more than 1.9 million words, it still does not contain all of the words in every dataset, this is especially evident when working with specialized datasets like medical text data. This introduces the problem of Out Of Vocabulary (OOV) words. For now we have decided to overlook this flaw as the embeddings seem to work well even considering these out of vocabulary words. However, there can be many ways to tackle this problem and it is one of the problems being researched right now. In their current form there is 3 things that could happen when an out of vocabulary word is fed through the word embedding layer, it might provide a completely random set of vectors, a fixed vector set reserved for OOV words, or just the average of all the vectors in the embeddings.

One method worth experimenting with would be to train a small network on the vocabulary set so that given each word it would output the corresponding embedding. If the model achieves a sufficiently high accuracy to reproduce word vectors it might be usable instead of the word embedding layer at the base of the network. The advantage here would be that the word embedding process could be more deeply integrated into networks and more importantly, because this is an actual network trained on words to produce their word vectors even given Out Of Vocabulary words it should output vectors at least more usable than random or average over all vectors. Another advantage here would be space usage, currently the GloVe word embeddings are 5 GBs, by using a network that reproduces the same vectors through learning and referencing, this space requirement can be significantly reduced. Our early tests show that using a simple LSTM model can reduce the space requirement at least by 10 times.

References

Agrawal, R., et al. (1998). Automatic subspace clustering of high dimensional data for data mining applications, ACM.

Bahmani, B., et al. (2012). "Scalable k-means++." Proceedings of the VLDB Endowment **5**(7): 622-633.

Barber, D. and F. V. Agakov (2006). Kernelized infomax clustering. Advances in neural information processing systems.

Bengio, Y., et al. (2003). "A neural probabilistic language model." Journal of Machine Learning Research **3**(Feb): 1137-1155.

Bezdek, J. C., et al. (1984). "FCM: The fuzzy c-means clustering algorithm." Computers & Geosciences **10**(2-3): 191-203.

Biernacki, C., et al. (2000). "Assessing a mixture model for clustering with the integrated completed likelihood." IEEE transactions on pattern analysis and machine intelligence **22**(7): 719-725.

Buhrmester, M., et al. (2011). "Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data?" Perspectives on psychological science **6**(1): 3-5.

Chen, X. and D. Cai (2011). Large scale spectral clustering with landmark-based representation. AAAI.

Cho, K., et al. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259.

Chung, J., et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555.

Collobert, R. and J. Weston (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. Proceedings of the 25th international conference on Machine learning, ACM.

Collobert, R., et al. (2011). "Natural language processing (almost) from scratch." Journal of Machine Learning Research **12**(Aug): 2493-2537.

Conneau, A., et al. (2016). "Very deep convolutional networks for natural language processing." arXiv preprint arXiv:1606.01781.

Del Corso, G. M., et al. (2005). Ranking a stream of news. Proceedings of the 14th international conference on World Wide Web, ACM.

Deng, J., et al. (2009). Imagenet: A large-scale hierarchical image database. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE.

Fan, R.-E., et al. (2008). "LIBLINEAR: A library for large linear classification." Journal of Machine Learning Research **9**(Aug): 1871-1874.

Gao, H., et al. (2015). Multi-view subspace clustering. Proceedings of the IEEE International Conference on Computer Vision.

Gehring, J., et al. (2016). "A convolutional encoder model for neural machine translation." arXiv preprint arXiv:1611.02344.

Gehring, J., et al. (2017). "Convolutional sequence to sequence learning." arXiv preprint arXiv:1705.03122.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics.

Hansen, L. K. and P. Salamon (1990). "Neural network ensembles." IEEE transactions on pattern analysis and machine intelligence **12**(10): 993-1001.

Heller, K. A. and Z. Ghahramani (2005). Bayesian hierarchical clustering. Proceedings of the 22nd international conference on Machine learning, ACM.

Hochreiter, S. and J. Schmidhuber (1997). "Long short-term memory." Neural computation **9**(8): 1735-1780.

Huang, G., et al. (2017). Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. European conference on machine learning, Springer.

Joulin, A., et al. (2016). "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759.

Kailing, K., et al. (2004). Density-connected subspace clustering for high-dimensional data. Proceedings of the 2004 SIAM International Conference on Data Mining, SIAM.

Kalchbrenner, N., et al. (2014). "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188.

Keogh, E., et al. (2001). An online algorithm for segmenting time series. Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, IEEE.

Kim, Y. (2014). "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882.

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980.

Kingma, D. P., et al. (2014). Semi-supervised learning with deep generative models. Advances in Neural Information Processing Systems.

Klambauer, G., et al. (2017). Self-normalizing neural networks. Advances in Neural Information Processing Systems.

Klein, G., et al. (2017). "Opennmt: Open-source toolkit for neural machine translation." arXiv preprint arXiv:1701.02810.

Krause, A., et al. (2010). Discriminative clustering by regularized information maximization. Advances in neural information processing systems.

Kuhn, H. W. (1955). "The Hungarian method for the assignment problem." Naval Research Logistics (NRL) **2**(1-2): 83-97.

Le, H. T., et al. (2017). "Do Convolutional Networks need to be Deep for Text Classification?" arXiv preprint arXiv:1707.04108.

LeCun, Y., et al. (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE **86**(11): 2278-2324.

LeCun, Y., et al. (1999). Object recognition with gradient-based learning. Shape, contour and grouping in computer vision, Springer**:** 319-345.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. Workshop on Challenges in Representation Learning, ICML.

Li, H., et al. (2004). Minimum entropy clustering and applications to gene expression analysis. Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE, IEEE.

Lloyd, S. (1982). "Least squares quantization in PCM." IEEE transactions on information theory **28**(2): 129-137.

Maas, A. L., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. Proc. icml.

McCallum, A. and K. Nigam (1998). A comparison of event models for naive bayes text classification. AAAI-98 workshop on learning for text categorization, Citeseer.

Mikolov, T., et al. (2013). "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781.

Mikolov, T., et al. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems.

Nesterov, Y. (2013). Introductory lectures on convex optimization: A basic course, Springer Science & Business Media.

Ng, A. Y. and M. I. Jordan (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems.

Ng, A. Y., et al. (2002). On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems.

Nie, F. and H. Huang (2016). Subspace Clustering via New Low-Rank Model with Discrete Group Structure Constraint. IJCAI.

Nie, F., et al. (2016). New l1-Norm Relaxations and Optimizations for Graph Clustering. AAAI.

Nie, F., et al. (2014). Clustering and projected clustering with adaptive neighbors. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.

Nie, F., et al. (2016). The Constrained Laplacian Rank Algorithm for Graph-Based Clustering. AAAI.

Nie, F., et al. (2011). "Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering." IEEE Transactions on Neural Networks **22**(11): 1796-1808.

Pennington, J., et al. (2014). Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).

Pitelis, N., et al. (2014). Semi-supervised learning using an unsupervised atlas. Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer.

Radford, A., et al. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434.

Raina, R., et al. (2004). Classification with hybrid generative/discriminative models. Advances in neural information processing systems.

Rasmus, A., et al. (2015). Semi-supervised learning with ladder networks. Advances in Neural Information Processing Systems.

Rifai, S., et al. (2011). The manifold tangent classifier. Advances in Neural Information Processing Systems.

Rifai, S., et al. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. Proceedings of the 28th International Conference on International Conference on Machine Learning, Omnipress.

Roth, V. and T. Lange (2004). Feature selection in clustering problems. Advances in neural information processing systems.

Sadoughi, N. and C. Busso (2015). Retrieving target gestures toward speech driven animation with meaningful behaviors. Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ACM.

Sadoughi, N., et al. (2015). MSP-AVATAR corpus: Motion capture recordings to study the role of discourse functions in the design of intelligent virtual agents. Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on, IEEE.

Sargin, M. E., et al. (2008). "Analysis of head gesture and prosody patterns for prosody-driven head-gesture animation." IEEE transactions on pattern analysis and machine intelligence **30**(8): 1330-1345.

Severyn, A. and A. Moschitti (2015). Twitter sentiment analysis with deep convolutional neural networks. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM.

Shi, J. and J. Malik (2000). "Normalized cuts and image segmentation." IEEE transactions on pattern analysis and machine intelligence **22**(8): 888-905.

Shinnou, H. and M. Sasaki (2008). Spectral Clustering for a Large Data Set by Reducing the Similarity Matrix Size. LREC.

Sim, T., et al. (2002). The CMU pose, illumination, and expression (PIE) database. Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on, IEEE.

Srivastava, N., et al. (2014). "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research **15**(1): 1929-1958.

Tian, F., et al. (2014). Learning deep representations for graph clustering. AAAI.

Trigeorgis, G., et al. (2014). A deep semi-nmf model for learning hidden representations. International Conference on Machine Learning.

Vapnik, V. N. (1999). "An overview of statistical learning theory." IEEE Transactions on Neural Networks **10**(5): 988-999.

Vincent, P., et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of Machine Learning Research **11**(Dec): 3371-3408.

Wang, X., et al. (2016). Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM.

Wang, Z., et al. (2016). Learning a task-specific deep architecture for clustering. Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM.

Wang, Z., et al. (2015). A Joint Optimization Framework of Sparse Coding and Discriminative Clustering. IJCAI.

Williams, C. K. (2000). A MCMC approach to hierarchical mixture modelling. Advances in Neural Information Processing Systems.

Wolf, L., et al. (2011). Face recognition in unconstrained videos with matched background similarity. Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE.

Xiao, Y. and K. Cho (2016). "Efficient character-level document classification by combining convolution and recurrent layers." arXiv preprint arXiv:1602.00367.

Xie, J., et al. (2016). Unsupervised deep embedding for clustering analysis. International conference on machine learning.

Xie, P. and E. P. Xing (2015). Integrating Image Clustering and Codebook Learning. AAAI.

Xu, L., et al. (2005). Maximum margin clustering. Advances in neural information processing systems.

Xu, W., et al. (2003). Document clustering based on non-negative matrix factorization. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM.

Yang, J., et al. (2016). Joint unsupervised learning of deep representations and image clusters. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Yang, Y., et al. (2010). "Image clustering using local discriminant models and global integration." IEEE Transactions on Image Processing **19**(10): 2761-2773.

Ye, J., et al. (2008). Discriminative k-means for clustering. Advances in neural information processing systems.

Yeh, R., et al. (2016). "Semantic image inpainting with perceptual and contextual losses." arXiv preprint arXiv:1607.07539.

Yogatama, D., et al. (2017). "Generative and discriminative text classification with recurrent neural networks." arXiv preprint arXiv:1703.01898.

Yosinski, J., et al. (2014). How transferable are features in deep neural networks? Advances in neural information processing systems.

Zelnik-Manor, L. and P. Perona (2005). Self-tuning spectral clustering. Advances in neural information processing systems.

Zhang, W., et al. (2012). Graph degree linkage: Agglomerative clustering on a directed graph. European Conference on Computer Vision, Springer.

Zhang, W., et al. (2013). "Agglomerative clustering via maximum incremental path integral." Pattern Recognition **46**(11): 3056-3065.

Zhang, X., et al. (2015). Character-level convolutional networks for text classification. Advances in neural information processing systems.

Zhang, Y. and B. Wallace (2015). "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification." arXiv preprint arXiv:1510.03820.

Zhao, B., et al. (2008). Efficient multiclass maximum margin clustering. Proceedings of the 25th international conference on Machine learning, ACM.

Zhao, J. J., et al. (2015). "Stacked What-Where Auto-encoders." CoRR **abs/1506.02351**.

Zhou, F., et al. (2013). "Hierarchical aligned cluster analysis for temporal clustering of human motion." IEEE transactions on pattern analysis and machine intelligence **35**(3): 582-596.

Biographical Information

I have been doing research in the field of machine learning, both supervised as well as unsupervised learning. My research includes tasks from computer vision to natural language processing. I am striving to become a successful and influential data scientist.