Clustering and Unsupervised Anomaly Detection with l_2 Normalized Deep Auto-Encoder Representations

Caglar Aytekin, Xingyang Ni, Francesco Cricri and Emre Aksu
Nokia Technologies, Tampere, Finland
Corresponding Author e-mail: caglar.aytekin@nokia.com

***Abstract*—Clustering is essential to many tasks in pattern recognition and computer vision. With the advent of deep learning, there is an increasing interest in learning deep unsupervised representations for clustering analysis. Many works on this domain rely on variants of auto-encoders and use the encoder outputs as representations/features for clustering. In this paper, we show that an l_2 normalization constraint on these representations during auto-encoder training, makes the representations more separable and compact in the Euclidean space after training. This greatly improves the clustering accuracy when k-means clustering is employed on the representations. We also propose a clustering based unsupervised anomaly detection method using l_2 normalized deep auto-encoder representations. We show the effect of l_2 normalization on anomaly detection accuracy. We further show that the proposed anomaly detection method greatly improves accuracy compared to previously proposed deep methods such as reconstruction error based anomaly detection.**

I. Introduction

Cluster analysis is essential to many applications in computer vision and pattern recognition. Given this fact and the recent advent of deep learning, there is an increasing interest in learning deep unsupervised representations for clustering analysis [1], [2], [3], [4]. Most of the methods that perform clustering on deep representations, make use of auto-encoder representations (output of the encoder part) and define clustering losses on them. The focus of previous works have been on the choice of the auto-encoder type and architecture and the clustering loss. In DEC [1], first a dense auto-encoder is trained with minimizing reconstruction error. Then, as a clustering optimization stage, the method iterates between computing an auxiliary target distribution from auto-encoder representations and minimizing the Kullback-Leibler divergence to it. In IDEC [2], it is argued that the clustering loss of DEC corrupts the feature space, therefore IDEC proposes to jointly optimize the clustering loss and reconstruction loss of the auto-encoder. DCEC [4] argues the inefficiency of using dense auto-encoders for image clustering, therefore adopts a convolutional auto-encoder and shows that it improves the clustering accuracy of DEC and IDEC. GMVAE [3] adopts a variational auto-encoder in order to learn unsupervised representations and simply applies K-means clustering on representations.

In this manuscript, we show that regardless of the auto-encoder type (dense or convolutional), constraining the auto-encoder representations to be on the unit-ball, i.e. to be l_2 normalized, during auto-encoder training, greatly improves the clustering accuracy. We show that a simple k-means clustering on the auto-encoder representations trained with our constraint already gives improved accuracy with a large margin compared to baselines with or without additional clustering losses. Motivated by the high performance of our clustering method on deep representations, we propose an unsupervised anomaly detection method based on this clustering. We show that our anomaly detection method greatly improves on other deep anomaly detection strategies such as reconstruction error based ones. We also investigate the effect of l_2 normalization constraint during training on the anomaly detection accuracy and show that it leads to superior results compared to not applying the constraint.

II. Related Work

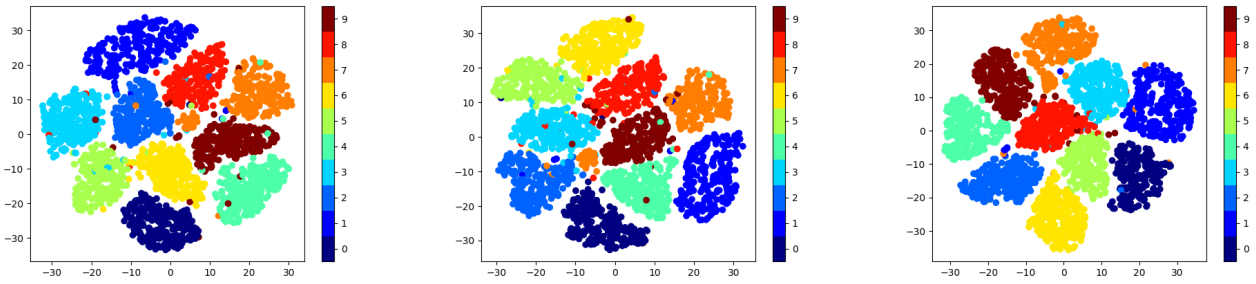
A. Deep Unsupervised Anomaly Detection

Unsupervised anomaly detection tries to find anomalies in the data without using any annotation [7]. Recently, deep learning methods have also been used for this task [5], [6]. These works train auto-encoders on the entire data and use reconstruction loss as an indicator of anomaly. DRAE [5] trains auto-encoders and uses reconstruction error as an anomaly indicator. Moreover, DRAE proposes a method to make the reconstruction error distributions of the normal and abnormal classes even more separable so that it is easier to detect anomalies. AVAE [6] trains both conventional and variational auto-encoders and use reconstruction error as an anomaly indicator.

The general assumption of the above works is that since the anomaly data is smaller in ratio than the normal data, the auto-encoder would not learn to reconstruct it accurately.

The above assumption seems to work in a specific definition of anomaly where the normal samples are drawn from a single class only and anomaly classes have been selected from many other classes [5]. However, the assumption fails in another anomaly type where the normal samples are drawn from multiple classes and anomaly class is sampled from a specific class [6].

In this paper we propose an unsupervised anomaly detection method based on clustering on deep auto-encoder representations and show that it gives a superior performance than reconstruction error based anomaly.



(a) No normalization (b) l_2 normalization after training (c) l_2 normalization during training

Fig. 1: Illustration of t-SNE encoding of auto-encoder representations for MNIST dataset to two dimensions. Best viewed in color.

B. Regularization and Normalization in Neural Networks

Due to the high number of parameters, neural networks have a risk of over-fitting to the training data. This sometimes reduces the generalization ability of the learned network. In order to deal with over-fitting, mostly regularization methods are employed. One of the most widely used regularization technique is weight norm regularization. Here the aim is to add an additional regularization loss to the neural network error, which gives high penalty to weights that have high norms. Both l_1 and l_2 norm can be exploited.

Recently some normalization techniques for neural networks emerged such as [8], [9]. Batch normalization [8], aims to find a statistical mean and variance for the activations which are calculated and updated according to batch statistics. The activations are normalized according to these statistics. In layer normalization [9], the mean and variance are computed from all of the summed inputs to the neurons on a layer on a single training sample. This overcomes the batch-size dependency drawback of batch-normalization. Although these methods were mainly proposed as tricks to make the neural network training faster by conditioning each layer's input, it is argued that they may also have a regularization effect due to their varying estimations of parameters for standardization at each epoch.

In our proposed method, the unit ball constraint that we put on activations is a normalization technique. However, unlike layer or batch normalization, the unit ball constraint is parameter-free as it simply sets the norm of each activation vector to 1. Therefore, it is free from the parameter estimation stochasticity. Yet, it may still act as a regularization method due to its hard constraint on some activations to be of fixed norm. This slightly resembles the l_2 norm regularization. A key difference is that in l_2 norm regularization, the norm is of the weights, but in our case it is applied on the activations. Another key difference is that we fix the activation norms to 1, whereas l_2 norm regularization penalizes to large weight norms and does not fix the norms to any value.

III. Proposed Method

A. Clustering on l_2 Normalized Deep Auto-Encoder Representations

We represent the auto-encoder representations for the input I as $E(I)$ and the reconstructed input as the $D(E(I))$. The representations are generally obtained via several dense or convolutional layers applied on the input I. In each layer, usually there are filtering and an activation operations, and optionally pooling operations. Let f_i be the computations applied to the input at layer i, then the encoded representations for an n-layer encoder are obtained as in Eq. 1.

$$E(I) = f_n(f_{n-1}(...f_1(I))) \tag{1}$$

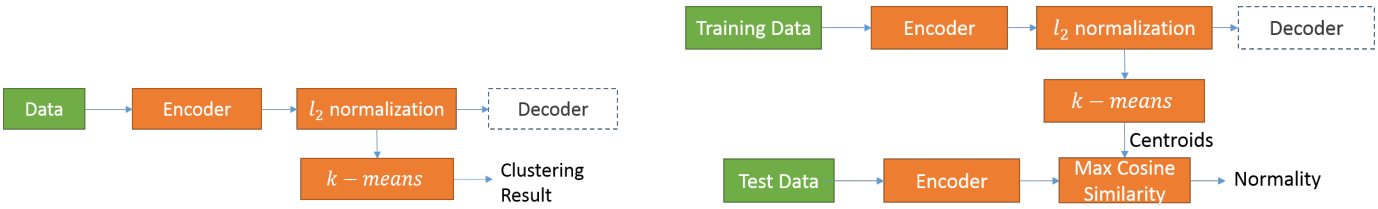
The reconstruction part of the auto-encoder applies on $E(I)$ and is obtained via several dense or deconvolutional layers. In each layer, usually there are filtering and activation operations and optionally un-pooling or up-sampling operations. Let g_i be the computations applied to the auto-encoder representations, then the reconstructed signal for an m-layer decoder is obtained as as in Eq. 2.

$$D(E(I)) = g_m(g_{m-1}(...g_1(E(I)))) \tag{2}$$

The auto-encoder training is conducted in order to reduce the reconstruction error (loss) given in Eq. 3.

$$L = \frac{1}{|J|}\sum_{j \in J}(I_j - D(E(I_j)))^2 \tag{3}$$

Here, we propose an additional step conducted on the auto-encoder representations $E(I)$. In particular, we apply l_2 normalization on $E(I)$. This corresponds to adding a hard constraint on the representations to be on the unit ball. The loss function with our introduced constraint can then be written as in Eq. 4, where L_c and E_c are loss and encoded representations with our introduced constraint.



(a) Clustering Method

(b) Normality Score Method

Fig. 2: Illustration of proposed methods in inference phase.

$$L_c = \frac{1}{|J|}\sum_{j\in J}(I_j - D(E_c(I_j)))^2, \qquad E_c(I) = \frac{E(I)}{\|E(I)\|_2} \tag{4}$$

We believe that l_2 normalized features (representations) are more suitable for clustering purposes, especially for the methods that use Euclidean distance such as conventional k-means. This is because the distances between the vectors would be independent of their length, and would instead depend on the angle between the vectors. As a positive side effect, enforcing the unit norm on representations would act as a regularization for the entire auto-encoder.

In Fig. 1, we illustrate t-SNE [10] encoding (to 2 dimensions) of the auto-encoder representations of networks with same architecture. All auto-encoders were trained on MNIST [11] dataset training split. The representations corresponding to Fig. 1a are from the auto-encoder that was trained by the loss in Eq. 3. The same representations with l_2 normalization applied after training are illustrated in Fig. 1b. Finally, the representations with l_2 constraint during training , i.e. training with loss Eq. 4, are illustrated in Fig. 1c.

It is observed from the figures that the l_2 normalization during training (Fig. 1c) results into more separable clusters. One example is the distributions of digit 7 in MNIST dataset. Note that the numbers are indicated with color codes where the color bar is available in Fig. 1. It is clearly observed that with no normalization during training (Fig. 1a), digit 7 is divided into 2 parts where the small part is surrounded by 8,9,6 and 2 digits. With normalization applied after training (Fig. 1b) this effect becomes even more evident. So, we clearly observe here that applying normalization after training does not help at all. But, with l_2 normalization constraint during training (Fig. 1c), we see a clear separation of digit 7 as a single cluster from the rest of the numbers. Moreover, it can be observed that the clusters are more compact in Fig. 1c compared to others.

After training the auto-encoder with the loss function in Eq. 4, the clustering is simply performed by k-means algorithm. No more clustering loss is applied. Our clustering method is illustrated in Fig. 2a.

B. Unsupervised Anomaly Detection using l_2 Normalized Deep Auto-Encoder Representations

Here, we propose a clustering based unsupervised anomaly detection. We train an auto-encoder on the entire dataset including normal and abnormal samples and no annotation or supervision is used. The auto-encoder is simply trained with the loss in Eq. 4. After training, the l_2 normalized auto-encoder representations are clustered with k-means algorithm. We assume the anomaly cases to be considerably smaller in number than any normal clusters. Note that this assumption does not put any constraint on the dataset, but it simply follows the general definition of anomaly. Therefore, the centroids obtained by the k-means method can be considered to be representations of normal clusters by some errors that are caused by anomaly samples in the dataset. To each sample i, we assign the normality score v_i in Eq. 5.

$$v_i = \max_j (E_c(I_i) \cdot \frac{C_j}{\|C_j\|_2}) \tag{5}$$

In Eq. 5, C_j is a cluster centroid and $\cdot$ is the dot product operator. Notice that we l_2 normalize the cluster centroids. Since representations $E_c(I_i)$ are already l_2 normalized, $v_i \in [0, 1]$ holds. The measure in Eq. 5 is intuitive considering that we expect high similarities of normal samples to normal classes. Our normality scoring method is illustrated in Fig. 2b.

The normality score can be used for anomaly detection in a straightforward manner. Simply, the abnormal samples can be detected as the ones having $v_i < \tau$, where $\tau \in [0, 1]$ is a threshold.

IV. Experimental Results

A. Clustering

Evaluation Metrics: We use the widely used evaluation for unsupervised clustering accuracy [1] given in Eq. 6.

$$acc = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n} \tag{6}$$

In Eq. 6, l_i is the ground truth labeling of sample i, c_i is the cluster assignment according to the one-to-one mapping defined by m, where m ranges over all possible one-to-one mappings between clusters generated by the algorithm and the ground truth labels. The maximization in Eq. 6 can be performed by Hungarian algorithm [12].

We compare the clustering accuracy of auto-encoder representations with and without l_2 normalization constraint. We make this comparison in dense and convolutional auto-encoders. For dense auto-encoder, we use MNIST [11] and for convolutional auto-encoders, we use MNIST [11] and USPS [15] datasets. This is due to the availability of results in the works that use dense and convolutional auto-encoders.

For dense auto-encoder, we use the network structure which is used both in DEC [1] and IDEC [2]. In encoding part there are 4 layers with $500-500-2000-10$ hidden neurons and in decoding part there are $2000-500-500-d$ neurons, where d is the dimension of the input. We re-implement the auto-encoder training and with leaky relu [13] activations after each hidden layer except for the last one and trained the auto-encoder end to end for 100 epochs. We select the best model with lowest reconstruction error. As it can be observed from Table I, we obtain a very similar clustering accuracy when we apply k-means on auto-encoder representations compared to the original paper of DEC [1]. Note here that results indicated with * corresponds to our own implementation. Other results for baselines are borrowed from original papers. Table I shows that when we train the auto-encoder with our l_2 normalization constraint on the representations, we achieve a much better clustering accuracy when we apply k-means on the representations. We denote our method as AE-l_2 which stands for auto-encoder with l_2 normalization. Moreover, our clustering accuracy is even better than the methods that define a separately designed clustering loss on the representations (DEC and IDEC).

Next, we make experiments for the convolutional auto-encoder. For this, we make use of the model structure introduced in DCEC [4]. This model consists of 5x5, 5x5 and 3x3 convolutional filters in the encoding layers respectively. There are 32,64 and 128 filters in encoding layers respectively. Convolutions are applied with 2x2 strides and with relu [14] activations. After the convolutional layers, the activations are flattened and there is a dense layer of dimension 10. This is followed by another dense layer and reshaping. Decoder part consists of 64,32 and 1 deconvolutional filters of size 3x3, 5x5 and 5x5 respectively. Relu activations were applied after each convolution, except for the last one. The network was trained for 200 epochs as in the original paper of DCEC [4]. In Table II, we show clustering accuracy of k-means applied on convolutional autoencoder representations. We were able to obtain similar results as in the original paper (DCEC). Note here that results indicated with * corresponds to our own implementation. Other results for baselines are borrowed from original papers. It can be observed from Table II that when we train the convolutional autoencoder with our l_2 normalization constraint on representations, we achieve a much better performance. We denote our method as CAE-l_2 which stands for convolutional auto-encoder with l_2 normalization. Our performance is superior to DCEC which introduces additional clustering loss.

TABLE I: Clustering on Dense Auto-Encoder Representations

	AE* k-means	AE k-means	DEC	IDEC	AE-l_2 k-means
MNIST	81.43	81.82	86.55	88.06	**90.20**

TABLE II: Clustering on Convolutional Auto-Encoder Representations

	CAE* k-means	CAE k-means	DCEC	CAE-l_2 k-means
MNIST	84.83	84.90	88.97	**95.11**
USPS	73.521	74.15	79.00	**91.35**

TABLE III: Comparison of Normalization Methods

	batch-norm	layer-norm	l_2-norm
MNIST	70.67	70.83	**95.11**
USPS	74.95	75.263	**91.35**

l_2 versus Batch and Layer Normalization: Due to l_2 normalization step in our clustering method, we compare it with applying other normalization techniques training. In particular we train two separate networks by using batch [8] and layer [9] normalization, instead of l_2 normalization. All other setup for the experiments are the same. Batch size of 256 is used for all methods in order to have a large enough batch for batch normalization. Our method performs superior to both baselines by a large margin, as the accuracies in Table III indicate. More importantly it is noticed that neither batch nor layer normalization provides a noticeable accuracy increase over the baseline (CAE+k-means). Moreover in MNIST dataset, layer and batch normalization results into a significant accuracy decrease. This is an important indicator showing that the performance upgrade of our method is not a result of a input conditioning, but it is a result of the specific normalization type that is more fit for clustering in Euclidean space.

B. Anomaly Detection

Evaluation Metrics: An anomaly detection method often generates an anomaly score, not a hard classification result. Therefore, a common evaluation strategy in anomaly detection is to threshold this anomaly score and form a receiver operating curve where each point is the true positive and false positive rate of the anomaly detection result corresponding to a threshold. Then, the area under the curve (AUC) of RoC curve is used as an evaluation of the anomaly detection method [7].

Here, we evaluate our method introduced in Section III-B. The evaluation setup and implementation of our method are as follows. In MNIST training dataset, we select a digit class as anomaly class and keep a random 10% of that class in the dataset while the remaining 90% is ignored. We leave the rest of the classes as is. Then, we use the convolutional autoencoder structure in DCEC [4] and train it with our l_2 normalization constraint on representations. Finally, we apply k-means clustering on the representations and keep the centroids. In our experiments we use k=9 for k-means, since we assume that we know the number of normal

classes in the data. For MNIST test dataset, we calculate the auto-encoder representations. As a normality measure for each sample, we calculate the corresponding representation's maximum similarity to pre-calculated cluster centroids as in 5. It should be noted that we repeat the above procedure by choosing a different class to be anomaly class, for all possible classes.

We also evaluate two baselines. In the first baseline, we exactly repeat the above procedure, but without l_2 normalization constraint on representations. In the second baseline, again we train the auto-encoder with l_2 normalization constraint on representations. Then, on the test set, we calculate the reconstruction error per sample and define that as anomaly score. Using reconstruction error based anomaly detection follows the works in AVAE [6] and DRAE [5]. The setups in AVAE and DRAE are different than ours. In AVAE, the training is only conducted on normal data, so the method is not entirely unsupervised in that sense. In DRAE, the anomaly definition is different: only a single class is kept as normal and samples from other classes are treated as anomaly. That setup presents a much easier case and therefore reconstruction error based anomaly detection produces acceptable results. Next, we show that in our setup this is not the case.

For each method we plot a RoC curve via thresholding the normality (or anomaly) score with multiple thresholds. Then, we evaluate the area under the RoC curve (AUC) for measuring the anomaly detection performance. The training and test datasets for all methods are the same. Due to the random selection of 10% of the anomaly class to be kept, performance can change according to the partition that is randomly selected. Therefore, we run the method 10 times for different random partitions and report the mean AUC.

It can be observed from Table IV that our clustering based anomaly detection method drastically outperforms the reconstruction error based anomaly detection for CAE neural network structure.

It is worth noting here an interesting observation from Table IV: for digits 1, 7 and 9, reconstruction error based anomaly detection gets a very inferior performance. This is most evident in digit 1. The reason for this is that the digit 1 is very easy to reconstruct (only 1 stroke) and even though an auto-encoder is trained on much less examples of this digit, it can reconstruct it quite well. This shows a clear drawback of the reconstruction error based anomaly detection. However, in our clustering based method, we achieve a very high accuracy in all the digits.

The effect of our proposed l_2 normalization constraint on representations during training can also be observed from Table IV. In $9/10$ cases, i.e. digits selected as anomaly, anomaly detection with the network trained with l_2 normalization constraint on representations performs much better than the one without. Only in digit 9, we observe an inferior accuracy of our method. Compared to other digits, we observe less performance for digits 4 and 9. We argue that this might be happening due to very similar appearance of these digits in some handwritings. Therefore, the method may confuse these numbers with each other during clustering.

TABLE IV: Anomaly Detection with Auto-Encoder Representations

Anom. Digit	CAE. (recons)	CAE (cluster)	CAE-l_2 (cluster)
0	0.7025	0.7998	**0.9615**
1	0.0782	0.8871	**0.9673**
2	0.879	0.7512	**0.9790**
3	0.8324	0.8449	**0.9382**
4	0.7149	0.4988	**0.7825**
5	0.8359	0.7635	**0.9136**
6	0.6925	0.7896	**0.9497**
7	0.5767	0.7421	**0.9100**
8	0.8912	0.9200	**0.9237**
9	0.514	**0.8944**	0.7495

TABLE V: Anomaly Detection with Auto-Encoder Representations

Anom. (Digit)	AE. (recons.)	VAE. (recons.)	CAE-l_2 cluster
0	0.825	0.917	**0.9615**
1	0.135	0.136	**0.9673**
2	0.874	0.921	**0.9790**
3	0.761	0.781	**0.9382**
4	0.727	**0.808**	0.7825
5	0.792	0.862	**0.9136**
6	0.812	0.848	**0.9497**
7	0.508	0.596	**0.9100**
8	0.869	0.895	**0.9237**
9	0.548	0.545	**0.7495**

In Table V, we compare our method to another method [6] that performs reconstruction error based anomaly detection, but using dense auto-encoders. There is also a variational auto-encoder based version of the method. It should be noted that this method trains auto-encoders only on normal data. This presents a much easier task compared to our case where we also include anomalous samples during training. Thus our case is entirely unsupervised. Still, $9/10$ cases, our method outperforms both variants of the method with a large margin. Only in digit 4, we observe an inferior performance of our method compared to VAE method.

V. Conclusion

In this paper, we have applied a l_2 normalization constraint to deep autoencoder representations during autoencoder training and observed that the representations obtained in this way clusters well in Euclidean space. Therefore, applying a simple k-means clustering on these representations gives high clustering accuracies and works better than methods defining additional clustering losses. We have also shown that the high performance is not due to any conditioning applied on the representations but it is due to selection of a particular normalization that leads to more separable clusters in Euclidean space. We have proposed an unsupervised anomaly detection method on l_2 normalized deep auto-encoder representations.

We have shown that the proposed l_2 normalization constraint drastically increases the anomaly detection method's performance. Finally, we have shown that the commonly adopted deep anomaly detection method based on the reconstruction error performs weak in a definition of anomaly, whereas our method performs superior.

REFERENCES

[1] J. Xie, R. Girshick and A. Farhadi, *Unsupervised deep embedding for clustering analysis*, International Conference on Machine Learning, pp. 478-487, June, 2016.

[2] X. Guo, L. Gao, X. Liu and J. Yin, *Improved deep embedded clustering with local structure preservation*, International Joint Conference on Artificial Intelligence, pp. 1753-1759, June, 2017.

[3] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M- C- Lee, H. Salimbeni, K- Arulkumaran and M. Shanahan, *Deep unsupervised clustering with gaussian mixture variational autoencoders*, arXiv preprint arXiv:1611.02648, 2016.

[4] X. Huo, X. Liu, E. Zheand J. Yin, *Deep Clustering with Convolutional Autoencoders*, International Conference on Neural Information Processing, pp. 373-382, 2017.

[5] Y. Xia, X. Cao, F. Wen, G. Hua and J. Sun, *Learning discriminative reconstructions for unsupervised outlier removal*, Proceedings of the IEEE International Conference on Computer Vision, pp. 1511-1519, 2015.

[6] J. An and S. Cho, *Variational autoencoder based anomaly detection using reconstruction probability*, SNU Data Mining Center, Tech. Rep., 2015.

[7] M. Goldstein and S. Uchida, *A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data*, PloS one, vol 11, no. 4, 2016.

[8] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, International conference on machine learning, pp. 448-456, 2015.

[9] J. L. Ba, J. R. Kiros and G. E. Hinton, *Layer normalization*, arXiv preprint arXiv:1607.06450, 2016.

[10] L. V. D. Maaten and G. Hinton, *Visualizing data using t-SNE*, Journal of machine learning research, pp. 2579-2605, 2008.

[11] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[12] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Research Logistics, 2(1-2), pp. 83-97, 1955.

[13] V. Nair and G. E. Hinton, *Empirical evaluation of rectified activations in convolutional network*, arXiv preprint arXiv:1505.00853, 2015.

[14] B. Xu, N. Wang, T. Chen and M. Li, *Rectified linear units improve restricted boltzmann machines*, International Conference on Machine Learning, pp. 807-814, 2010.

[15] Y. LeCun, O. Matan, B. Boser, J. D. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket and H. S. Baird, *Handwritten zip code recognition with multilayer networks*. International Conference on Pattern Recognition, vol. 2, pp. 35-40, 1990.