RWTH Aachen University
Faculty of Mathematics, Computer Science and Natural Sciences
Media Informatics

Master's Thesis

Development of a Source Library for Transfer Learning: Leveraging Clustering and Contrastive Learning Techniques

Sneha Banerjee
Matriculation Number: 430069

Wednesday 21st February, 2024

Examiner: **Prof. Dr. Christian Bauckhage**
Supervisor: **Daniel Schiller**

Declaration of Originality

Name	Sneha Banerjee
Matriculation number	430069
Address	Allmandring 8A,70569 Stuttgart
Title	*Development of a Source Library for Transfer Learning: Leveraging Clustering and Contrastive Learning Techniques*

I now declare,

- that I wrote this work independently,
- that no sources other than those stated are used and that all statements taken from other works—directly or figuratively—are marked as such,
- that the work submitted was not the subject of any other examination procedure, either in its entirety or in substantial parts,
- that I have not published the work in whole or in part, and
- that my work does not violate any rights of third parties and that I exempt the University against any claims of third parties.

Abstract

Contents

1 Introduction

1.1 Motivation

1.2 Task and Goal

1.3 Approach

1.4 Contribution

2 Related Work

3 Background

3.1 Bin Picking

3.1.1 Definition and Application

3.1.2 Gripper Types

3.1.3 Model-Based Approaches

3.1.4 Model-Free Approaches

3.1.5 Robot System

3.2 Deep learning-based 3D Image processing

3.2.1 Convolution Neural Network

Inspired from the visual cortex of humans, convolutional neural network(Convolutional Neural Network (CNN) or ConvNet) is a type of deep neural network designed for processing data which appear in a grid like manner like images, videos, etc. It was introduced by Yann LeCun et. al in [13] in 1998. It gets it name because of the usage of a special kind of linear mathematical operation called the convolution instead of using matrix multiplication as prevalent in the pre-existing neural networks. The key components of a CNN are - Convolution layer, activation function, pooling layer, loss function, output layer. The principle building block if a CNN is the convolution layer. It consists of a number of learnable filters (kernels) which can be visualised like a cubic block. The success of CNNs can be attributed to three major concepts: sparse interactions, parameter sharing and equivariant representations.

Sparse interactions In traditional fully connected network, matrix multiplication is performed which involves a parameter matrix. The interaction between input and output units are captured by a distinct parameter in the parameter matrix. But CNNs have sparse interactions, i.e. only a subset of units or neurons in a layer is connected to a local region in the preceding layer. This is done by using kernels that are significantly smaller than the input. This implies, less number of parameters need to be stored which reduces the memory consumption and also it is computationally efficient because it has to perform fewer operations. For example, if there are m inputs and n outputs, the fully connected network would need to store m x n parameters and have a runtime of $O(m \text{ x } n)$ time complexity per input. On the other hand, if we restrict the number of connections for each unit to be k, then we would have k x n and have a runtime of $O(k \text{ x } n)$ time complexity per input, where k is quite some fold lesser in magnitude as compared to m. Moreover, since convolution layers are stacked upon one another in a deep convolution network, units in the deeper layers have a larger receptive field, because of it's indirect interaction with a larger region in the input.

Parameter sharing Another important focus behind using the convolution layer was to reduce the number of parameters in a Fully Connected Network (FCN). For example in a 1024*1024 image, a FCN would have over 1 million hidden units, which means it would have over 1 trillion trainable parameters. But the pixels in an image are only locally correlated. So, CNNs make use of the kernels to limit the focus on smaller regions on the image at a time known as the receptive field. This significantly reduces the number of trainable parameters, thus reducing the memory consumption. [2]. These layers perform a convolution operation between the input (eg. image) I and the kernel K to produce an output S known as the feature map as in Eq.1.

Equivariant Representations A function is said to be equivariant if the input to a function is changed, then the output changes in the similar way. Because of the parameter sharing mechanism, convolutions operations are translation equivariant. When the kernels the applied on an input image, the convolution layers generate a 2D map of where the particular feature occurs in the particular image. Furthermore, a pixel is related to it's neighboring pixels to form a meaningful context(create a feature e.g. an edge in an image) but it is not limited to where it can occur throughout the image. Thus it creates multiple filters each

of which look for the same feature throughout the image. But it is also to be kept in mind that convolution is not equivariant to other geometric and affine transformations like rotation, scaling, etc. Since convolution operations are commutative in nature, Eq. 1 can also be written as Eq.2. Typically, Eq. 2 is easier to incorporate into a machine learning library since values for both 'm' and 'n' varies within a small range.[6]

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{1}$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m,j-n)K(m,n) \tag{2}$$

Activation function The next step in a CNN is to apply an activation function. the purpose of using a non linear activation function is to capture the non-linearity in the data. Moreover if non-linearity is not used in between the multiple layers of a neural network, the network is effectively only one layer deep which is not capable even to capture the non-linearity in real world datasets. Rectified Linear Unit (ReLU) is the most commonly used non-linear activation function used in CNNs because unlike sigmoid function or tanh function it does not penalise "too correct" data points. Another remarkable benefit of using ReLU is it's ability to propagate gradient through deep networks with a constant factor. Also it is more memory efficient to use ReLU as it doesn't require to store the ReLU outputs separately as compared to tanh outputs.

Pooling The third step of a CNN is to use the pooling layer. The main purpose of this operation is to make the detection of the features robust to the exact location of the eye (i.e. invariant to small translations). The different types of pooling operations are - max pooling and average pooling. It also helps in reducing the dimensionality of the input without losing too much information. This is done to make the computations faster down the deeper layers of the network. If the pooling operations are performed after every k pixels, then the next layer processes inputs that are k times lesser. Since the number of parameters in a layer are dependent on the size of the input to the layer, it significantly reduces the computational overhead on using pooling operations.

Complete CNN

3.2.2 Clustering Algorithms

When the dataset does not have any labelled data, then it is said to be unsupervised learning. In this case, there is no ground truth available to measure the correctness of the outputs generated by the Machine Learning (ML) models. The primary focus of unsupervised learning is to find hidden and interesting pattern in the data. Unsupervised learning is of utmost importance in the Artificial Intelligence (AI) world as several real world datasets do not have available annotations, which requires a lot of human effort. Unsupervised learning algortihms can be broadly categorized into the following domains-clustering, dimensionality reduction, and association analysis. Clustering algorithms aim at grouping unlabelled data into groups or clusters based on how similar or dissimilar the datapoints are to one another. It can reveil underlaying hidden pattern in the data and is used in applications like image segmentation, fraud detection, etc. Dimensionality reduction is reduces the number of irrelavant features or dimensions of the dataset. The inclusion of more features does help in the better representation of the dataset but it also significantly increases the memory consumption and complexity to work with it. Also it is often difficult to visualise real-life datasets with too many features. Association analysisis a rule-based unsupervised learning method that reveals the relationship between attributes in the dataset. It is used in applications like market analysis, intrusion detection, etc.

Clustering algorithms can be broadly classified into the following categories: density-based, distribution-based, and hierarchical-based. In density-based clustering, the algorithm looks for areas of high concentration of the datapoints and groups them as a cluster. The benefit of this algorithm is that the shape a cluster can be is not limited and hence doesn't necessarily have to be convex in nature as show in Fig. 1. Moreover, these algorithms are also very robust to outliers as they do not force the outliers to belong to any category but are rather ignored as it is unlikely that outliers can form an area of high concentration of datapoints. I have carried on the experiments on this thesis on two such density-based clustering algortihms called DBSCAN[5] and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithms[14]. The available scikit-learn implementations were used for this purpose[17]. The DBSCAN[5] algortihm does not require the users to define the number of clusters to be

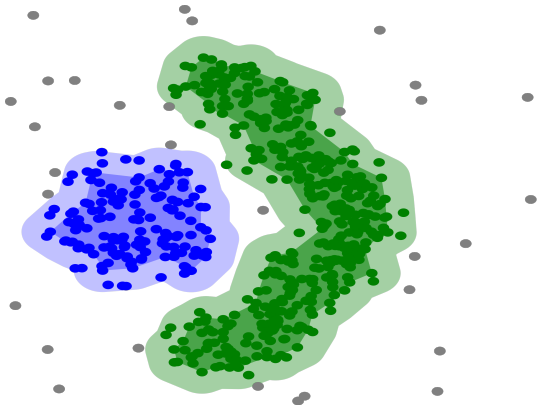

Figure 1: Density-Based Spatial Clustering of Applications with Noise (DBSCAN) can find non-linearly separable non-convex clusters.[3]

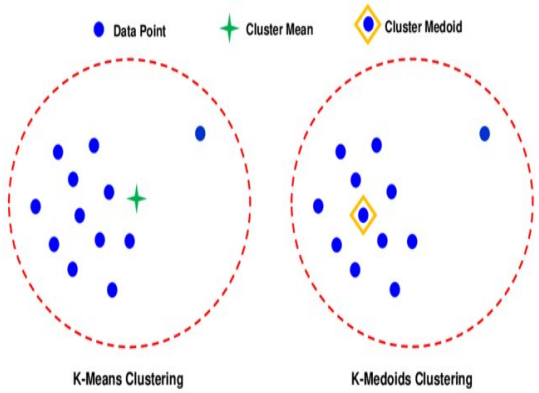

Figure 2: Difference between K-Means and K-Medoids.[4]

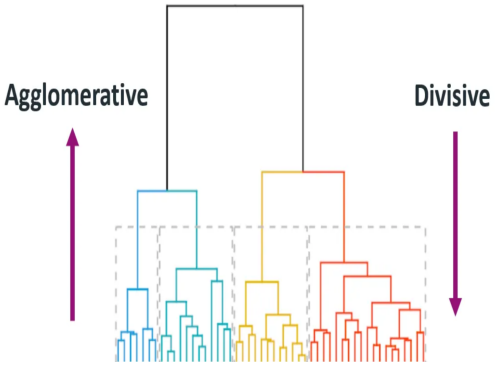

Figure 3: A dendogram in hierarchical clustering.[7]

generated which doesn't force dissimilar datapoints to belong to the same cluster. However, this algorithm was still sensitive to two parameters ε, the maximum distance betwwen the datapoints to be considered in the same cluster and the minimum number of samples in the cluster which the user needs to define [17]. But finding an optimum value for these parameters often require domain expertise and are dependent on the data. The HDBSCAN[14] algorithm mitigates this issue and thus does not require the user to define these two parameters. It is a hierarchical density-based clustering that performs the DBSCAN algorithm over multiple ε values to find the most stable result. In distribution-based algorithms, a datapoint is said to be a member of the cluster depending on the probability of it's membership to the cluster. The more the distance of a point increases from the centre of the cluster, the less is it's probability of belonging to that cluster. Centroid-based algorithms groups the datapoints based on some initial cluster centres. Once all the dataspoints are softly assigned to some cluster membership, the cluster centres are recalculated and this process is iterated until convergence. These algorithms are sensitive to the initial parameters like the cluster centres chosen in the first step. Another major disadvantage of these clustering algorithms are that they always form spherical clusters. The user also needs to define the number of clusters the dataset is to be grouped in, which makes it sensitive to outliers. However, these algorithms can be executed very fast and we have used two such algorithms in our experiments- K-Means[9]and K-Medoids[10]. In K-Means, the mean of the datapoints of the clusters is assigned as the cluster-centroid. It might not be an actual datapoint in the dataset, rather a blurred, noisy average of a datapoints in the cluster. On the contrary, the K-Medoids algortihm assigns an actual datapoint of the dataset, that is most centrally located as the cluster centroid as shown in Fig.2. Thus K-Medoids is more robust to outliers and noises as compared to K-Means.
Hierarchical-based clustering algorithm that form a hierarchy of clusters. Datapoints in a cluster are more

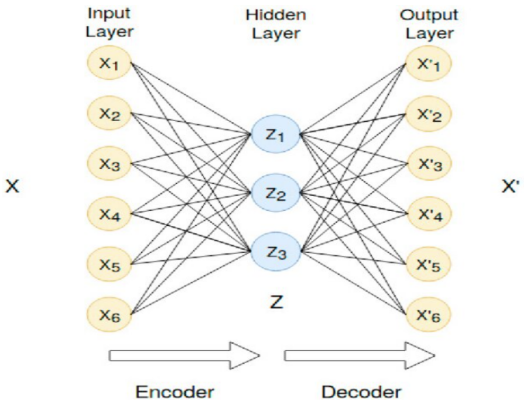

Figure 4: Architecture of an undercomplete autoencoder.[15]

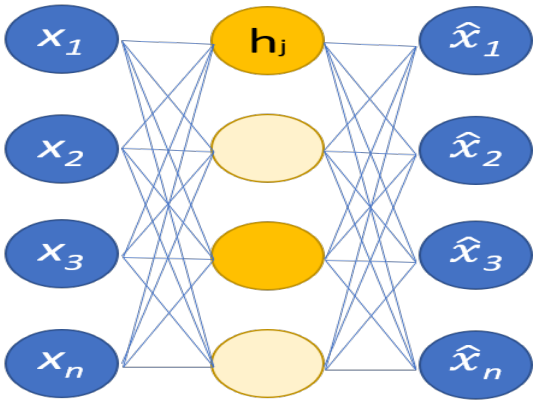

Figure 5: Architecture of a single layer sparse encoder. The hidden nodes in bright yellow are active, while the ones in light yellow are set to zero, hence inactive[1]

similar to each other as compared to other groups. this hierarchy of clusters is visualised by a hierarchy tree called dendograms. hierarchical clustering algorithms can be of two types - agglomerative and divisive. In agglomerative clustering, each datapointis considered as a separate cluster in the first step. Then these clusters are merged into one another until only one cluster remains. Thus at the end, the last level cluster consists of all teh datapoints in the dataset. The divisive method is the reverse procedure of the aggomerative method. In the beginning, all the datapoints are considered to be in a single cluster and gradually the cluster is broken into smaller clusters, untill each cluster consists of only one datapoint. A visual representation of a dendogram has been shown in Fig.3

3.2.3 Autoencoders

Autoencoders are a special type of feedforward neural network in which the output tries to reconstruct the input. It is predominantly used in unsupervised learning for the tasks if dimensionality reduction, learning feature representations, and for data compression.It tries the encode the input data into a more compact representation with lower dimensions called the "code"[6] or "latent space". The idea is that this latent space representation should capture the most vital aspects pf the input data. This is done by the first part of the network called the encoder. The second part of the network, the decoder then tries to decode this compressed representation of the input data to reconstruct the original input data as accurately as possible. during the training of an autoencoder, the goal is to minimise this reconstruction loss, i.e. the difference between the original input data and the output of the decoder. There are different types of autoencoders - undercomplete autoencoders, convolutional autoencoders, regularized autoencoders, and variational autoencoders,

Undercomplete autoencoders ensures that the dimension of the latent space representation is less than the dimension of the original input data as show in Fig.4. Because, the output of the decoder to be the exact copy of the input is of no use. Rather, if the dimension of the code is less than that of the input, then it ensures that the autoencoder learns those representative features of the input data which are most salient[6].

Convolutional autoencoders are an extension of traditional autoencoders where convolution layers are used as building blocks in both the encoder and the decoder part of the autoencoder. After training the network, the encoder part is used for ectracting the features of the data and the decoder part is used for the reconstruction of the input data.

Regularized autoencoders are a special type of autoencoders that employs one or more regularisation techniques to prevent the model from overfitting. The different types of regularisations used are - L1 or L2 Regularisation which adds a penalty to the loss function depending on L1 or L2 norm of the model weights As a result of this, the model is capable of learning sparse representation of the training data where many weights are set to a very small value or zero as shown in Fig5. Dropout is also used as a regularization tech-

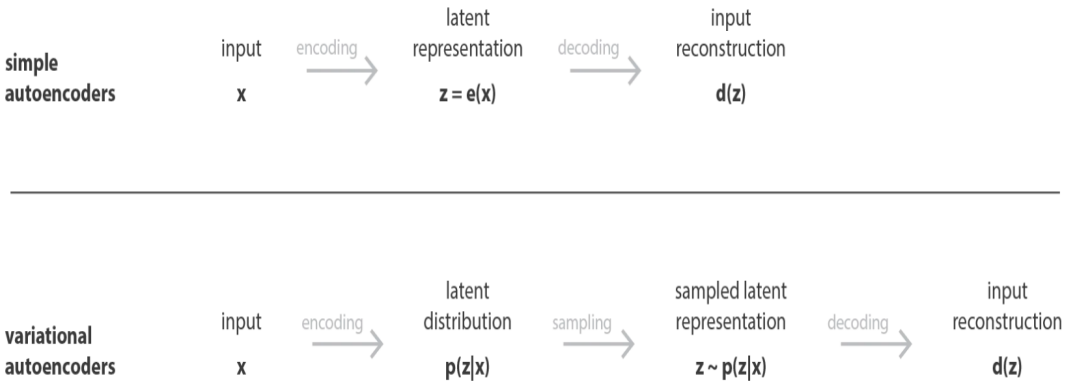

Figure 6: Difference between a vanilla autoencoder and a VAE.[19]

nique where a percentage of the model's nodes are set to zero during training so that the model doesn't rely heavily on any the weight of any particular node, thus preventing overfitting and improving generalisation on unseen data. The previously mentioned regularisation techniques give rise to a variation of regularised autoencoders called the sparse autpencoders[16]. Sometimes noise is also added to the training data or the hidden layers of the model to increase the robustness the model giving rise to denoising autoencoders[20]. One more regularisation technique if to apply a contractive regularization term based on Frobenius norm of the Jacobian matrix of the model's hidden layer activations with respect to the input data[18, 1]. This ensures that a small neighborhood of the input data corresponds to a small neighborhood in the latent space representation, which means small perturbations in the input data leads to small or zero variation in the latent space representation[18, 1]. By doing so, it makes the model more robust to small changes in the input data, thus preventing overfitting.
Variational Autoencoders (VAE)s are a distinct type of autoencoders which produces latent space representations that are continuous, which allows random sampling and interpolation for the generation of new datapoints. The difference between a vanilla autoencoder and a VAE is shown in Fig.6. Instead of producing a single vector for the latent space representation, it generates two vectors: a vector of means μ and a vector of standard deviations γ for all datapoints. An encoding is then sampled from a distribution with mean μ and standard deviation γ. Thus even for the same input, i.e. when the mean and the standard deviation are the same, the sampled encoding would vary because of the involvement of the sampling procedure. The mean vector controls the position where the encoding of the input should have it's centre and the standard deviation controls the area over which the sampled encoding is allowed to vary from the mean. As a result of this, the decoder learns to decode not just the encoding of a single point but also the points in the neighborhood and hence a continuous latent space representation is obtained. In order to ensure that the latent space representation satisfies that the nearby encoding are similar to each other on a local scale while also facilitating interpolation on a global scale VAEs jointly optimizes the reconstruction loss and the Kullback–Leibler divergence (KL)[12] loss.[11, 8]

3.2.4 PointNet Autoencoder

3.2.5 Siamese Network

3.3 Transfer learning

3.3.1 Definition

3.3.2 Categories

3.3.3 Relevant Algorithms

3.4 Network generation process with PQ-Net++

4 Method

4.1 Suitable Algorithm and Adjustments

4.2 Whatever new Concept we propose

4.3 Architecture

4.4 Source and Target Objects

4.5 Data Generation

4.6 Important Aspects

5 Application of algorithms

6 Experiments

7 Implementation in a Bin Picking Cell

8 Conclusions

8.1 Discussions

8.2 Limitations

8.3 Future Scope

9 Summary

List of Acronyms

CNN Convolutional Neural Network

FCN Fully Connected Network

ReLU Rectified Linear Unit

ML Machine Learning

AI Artificial Intelligence

DBSCAN Density-Based Spatial Clustering of Applications with Noise

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise

VAE Variational Autoencoders

KL Kullback–Leibler divergence

List of Figures

List of Tables

List of Symbols

Bibliography

[1] *Autoencoder*. https://en.wikipedia.org/wiki/Autoencoder.

[2] *Convolutional Neural Networks, Explained*. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[3] *DBSCAN Wikipedia*. https://en.wikipedia.org/wiki/DBSCAN.

[4] Alireza Entezami, Hassan Sarmadi, and Behzad Saeedi Razavi. "An innovative hybrid strategy for structural health monitoring by modal flexibility and clustering methods". In: *Journal of Civil Structural Health Monitoring* 10.5 (2020), pp. 845–859.

[5] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] *Hierarchical Clustering*. https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe656b.

[8] *Intuitively Understanding Variational Autoencoders*. https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf.

[9] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[10] Xin Jin and Jiawei Han. "K-Medoids Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 564–565. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_426. URL: https://doi.org/10.1007/978-0-387-30164-8_426.

[11] Diederik P Kingma, Max Welling, et al. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[12] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[13] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] Claudia Malzer and Marcus Baum. "A hybrid approach to hierarchical density-based cluster selection". In: *2020 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI)*. IEEE. 2020, pp. 223–228.

[15] Sumit Misra et al. "An autoencoder based model for detecting fraudulent credit card transaction". In: *Procedia Computer Science* 167 (2020), pp. 254–262.

[16] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[17] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[18] Salah Rifai et al. "Contractive auto-encoders: Explicit invariance during feature extraction". In: *Proceedings of the 28th international conference on international conference on machine learning*. 2011, pp. 833–840.

[19] *Understanding Variational Autoencoders(VAEs)*. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[20] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.