Unsupervised Clustering for Deep Learning: A tutorial survey

Artúr István Károly[1,2], Róbert Fullér[3,4], Péter Galambos[1]

[1] Antal Bejczy Center for Intelligent Robotics
Óbuda University, Bécsi út 96/B. H-1034 Budapest, Hungary
info@irob.uni-obuda.hu

[2] Doctoral School of Applied Informatics and Applied Mathematics
Óbuda University, Bécsi út 96/B. H-1034 Budapest, Hungary

[3] Institute for Applied Mathematics
Óbuda University, Bécsi út 96/B. H-1034 Budapest, Hungary

[4] Department of Informatics,
Széchenyi István University, Egyetem tér 1, H-9026, Győr, Hungary
e-mail: rfuller@sze.hu

Abstract: Unsupervised learning methods play an essential role in many deep learning approaches because the training of complex models with several parameters is an extremely data-hungry process. The execution of such a training process in a fully supervised manner requires numerous labeled examples. Since the labeling of the training samples is very time-consuming, learning approaches that require less or no labeled examples are sought. Unsupervised learning can be used to extract meaningful information on the structure and hierarchies in the data, relying only on the data samples without any ground truth provided. The extracted knowledge representation can be used as a basis for a deep model that requires less labeled examples, as it already has a good understanding of the hidden nature of the data and should be only fine-tuned for the specific task. The trend for deep learning applications most likely leads to substituting as much portion of supervised learning methods with unsupervised learning as possible. Regarding this consideration, our survey aims to give a brief description of the unsupervised clustering methods that can be leveraged in case of deep learning applications.

Keywords: Unsupervised learning; Clustering; Deep learning

1 Introduction

The three primary methods for learning-based systems are supervised, unsupervised and reinforcement learning. Reinforcement learning is applied in fields when an agent takes actions in an environment, and a suitable policy for acting has to be learned [1]. The other two learning methods are used when the output of the system does not influence the inputs in any way. In case of supervised learning the training samples are provided with correct labels, so a ground truth is available. Meanwhile, in unsupervised learning, no such a-priory knowledge of the data is required. Models that are trained in an unsupervised manner only require the collected data for training.

Deep learning is a widely researched topic currently. The majority of deep learning approaches utilize supervised learning [2]. However, given the vast number of trainable parameters of such models, the training process requires numerous labeled examples in order to achieve good generalization [2]. The labeling of the samples is a very resource-intensive and time-consuming process, and usually, the labeling can only be done manually [3]. So naturally arises a need for methodologies that enable the training of such models with less or no labeled examples. This is usually done by applying unsupervised learning first, and then fine-tuning the model with the help of labeled samples and supervised learning [2, 4]. Among others, the future of deep learning is expected to be driven by the development of more sophisticated and accurate unsupervised learning methods [2].

Supervised learning methods always have a well-defined objective, like classifying the inputs into one of the formerly known classes, or the regression of a function between a set of inputs and inspected outputs. In this case, the output features are formerly known, just like the class labels in classification. In unsupervised learning, however, the aim is to discover unknown structures, similarities, and grouping in the data [5].

Clustering is the process when the objective is to create groups of data samples (clusters) based on some kind of similarity measure between the data samples [5]. The difference between classification and clustering is that clustering is carried out in an unsupervised manner, so no class labels are provided, and sometimes even the number of clusters is not known a-priory.

In this survey, we provide a brief introduction of the most significant unsupervised clustering methods and their applicability in the field of deep learning. We aim to give a summary of such clustering techniques that can also be leveraged in deep learning applications, in the aspect of the expected trends of the future development in this field of study [2]. Previous approaches focused on either clustering methods or unsupervised deep learning. A detailed and general description of unsupervised clustering methods can be found in the work of Xu and Wunch, who provide an in-depth survey on clustering algorithms in [5]. They introduce the general description of a clustering procedure, provide multiple measures for similarity that are used

for clustering, and give a detailed explanation of several clustering algorithms and their applicability. Other works that provide a detailed introduction of clustering algorithms are [6] and [7]. Bengio et al. give an exhaustive survey on unsupervised learning in deep learning in [4].

2 Unsupervised clustering methods

In case of unsupervised learning, the task is to uncover hidden relationships, structures, associations or hierarchies based on the data samples provided to the system [5]. This kind of information gives us a better understanding of the data and the underlying processes generating it. The clusters can be constructed based on similarity or distance measures, so similar samples belong to the same cluster. The similarity measure can also be described as a distance measure because the similarity of two samples can be interpreted as the distance between the two samples in the feature space [5].

One approach for unsupervised clustering is to use these similarity measures and construct the regions of the feature space corresponding to the different clusters based on the computed distances between the training samples [5].

Another approach is to extract features with high discrimination power or to find principal directions of the feature space along which the data can be better separated. These features can be any subset of the input data or they can also be constructed by a network architecture [4].

Often, it is more convenient to use the computation mechanism of a supervised learning algorithm, but work it around, so no true labeling is needed. An example of this is anomaly detection, which can be handled as a clustering problem. In case of anomaly detection, one can expect two clusters, one for the normal samples and one for the anomalies. In case of an unsupervised training process for anomaly detection, one can extract the training data from only one cluster, the cluster of normal samples. As the training data, in this case, is expected to come from only one cluster, it can be automatically labeled as belonging to the same class, and the calculation for a supervised method can be used to train a system to classify similar samples as ones that belong to this cluster [8]. However, when an anomaly appears, that is not similar to the examples seen during the training process, then it is classified as a sample not belonging to the cluster of normal samples; thus it is classified as an anomaly.

The automatic generation of multiple labels is also beneficial. However, this method is no longer referred to as unsupervised learning. In the case of simulator-based labeling, for example, there is no need for the time consuming manual labeling of the data [9]. This enables large and very diverse datasets to be annotated. However, these samples are still provided with ground truth, even though they are not manually labeled. In case of the anomaly detection, the ground truth is the same for all of the

samples, so it is more of an assumption on the nature of the data rather than real labeling. That is why we call the use of only one label unsupervised learning, and not the automatic generation of multiple labels.

In case of unsupervised learning for deep learning models, there are two major approaches. Both rely on formulating an output for the neural network that can be used to train it in the usual way with the help of gradient descent [4].

The first one is to try to reconstruct the input of the network on its output. The loss function is computed based on the reconstruction error between the input and the output of the network [10, 4]. This method is expected to extract a meaningful compressed representation of the input from which it can be reconstructed with minimal error. This requires the compressed features to represent features of high discrimination power among the presented training samples. This way, the unsupervised training can be carried out on the whole network or layer-by-layer. After such training, the network is usually trained further with labeled examples, but it requires much less of them because thanks to the unsupervised pre-training a good representation of the input data is already available [2].

The second one is to use two networks in parallel. One of these networks that are called generator is used to generate data that is as similar to the input data of the other network (the discriminator) as possible [11]. The discriminator's objective is to discriminate the generated samples from the real ones. Both networks are trained in parallel. The generator is trained to produce data that can fool the discriminator even better and the discriminator is trained to be able to differentiate between synthetic and real data more accurately. The model itself appends the label for the inputs of the discriminator as a synthetic sample or real sample because it knows which samples come from the generator, while the update of the generator is based on the output response of the discriminator. So the system does not require a ground truth annotation. During the process of training, the discriminator has to develop an understanding of the features of the training dataset and later it can be used for classification as well (in a similar way like the anomaly detectors) [11].

3 Clustering algorithms

In this section we provide a brief description of the clustering algorithms which are especially suitable for deep learning applications. In table 1, we draw a straightforward categorization of the mentioned unsupervised clustering methods 1.

First, we discuss the approaches that are based on a distance measure. These methods define the similarity of data samples by the distance of the samples in the feature space. Because of this property, the simpler variants of these methods fail to cluster data that is not linearly separable. However, with the creative formulation of the similarity measure, or with proper pre-processing of the data, even these techniques can be applied for nonlinear clustering tasks [12].

Class	Methods	Section
Distance measure based	K-means clustering	3.1
	Hierarchical clustering	3.2
	Fuzzy clustering	3.3
	Support vector machines	3.4
	Spectral clustering	3.5
	Decision trees	3.6
Statistical	Expectation maximization algorithm	3.7
Neural networks	Self organizing maps (Kohonen networks)	3.8
	Adaptive resonance theory	3.9
	Autoencoders	3.10
	Co-localization	3.11
	Generative models	3.12

Table 1
Classification of unsupervised clustering methods

3.1 K-means clustering

The k-means algorithm is an iterative (learning) method to discover k number of clusters in the input space. The number k is defined a-priory [13, 14, 15]. Each cluster is represented with a cluster centroid in the feature space. The similarity measure of samples is a simple distance measure. The distance between the samples and the cluster centroids can be computed, and a sample is associated with the cluster with the closest centroid. All of the samples are associated with one and only one cluster, so the clusters cover the whole feature space, and they are disjoint. The iteration process consists of two stages: In the first stage, all the training samples are associated with one of the clusters. Then in the second stage, the location of the cluster centroids is updated. These two steps are repeated until a stopping condition is met. The stopping criteria can be that no further change in the classification of training samples happened after the last update of the cluster centroids, or the distance between the centroids before and after the update is smaller than a specified value (ϵ) [13, 14, 15].

Before the iteration process, there are some preliminary considerations to make that affect the result of the clustering. These considerations are the number of clusters k, the starting positions for the centroids of the clusters and the stopping condition.

A simple measure of the distance between the samples and the centroids is the Euclidean distance. The Euclidean distance of the two points is the Euclidean norm

of the vector pointing from one of the points to the other. The computation of the Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^n$ can be seen in equation 1.

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2} \tag{1}$$

The Euclidean distance of a sample $\mathbf{x} \in \mathbb{R}^n$ and a centroid $\mathbf{c} \in \mathbb{R}^n$ is $\|\mathbf{x} - \mathbf{c}\|$. Apart from the Euclidean distance, other metrics can be used like the Manhattan or the Mahalanobis distance [5].

The update of the centroids of clusters is carried out after each element of the training set has been associated with one of the clusters. The new cluster centroids can be computed as like a center of mass for all the samples associated with that given cluster.

The starting location of the cluster centroids can be randomly placed in the feature space, or random samples can be selected as initial cluster centroids to ensure that they are located in the feature space from where the data is drawn.

An algorithm for the k-means clustering method is presented in algorithm 1. In the algorithm, the squared Euclidean distance is used for distance measurement, in line 13.

Algorithm 1 : K-means clustering

1: **Definitions:**
2: Let k be the number of clusters,
3: $\mathbf{X}$, the set of training samples,
4: $\mathbf{x}_i \in \mathbf{X}$, the i^{th} sample, $i \in \{0, 1, \dots n\}$,
5: $\mathbf{x}_i \Rightarrow j^{th}$ cluster, means that the i^th sample belongs to the j^{th} cluster,
6: $\mathbf{c}_j$, the j^{th} cluster centroid, $j = \{1, 2 \dots k\}$,
7: s_j, the number of samples associated to the j^{th} cluster
8: **Initialization:**
9: Let $\mathbf{c}_j^+$ be a random sample drawn from $\mathbf{X}$, $\forall\, j$
10: **Iteration:**
11: **repeat**
12: $\mathbf{c_j} = \mathbf{c_j^+}$, $\forall\, j$
13: $\forall\, i$, **find** j for which $\|\mathbf{x}_i - \mathbf{c}_j\|^2$ is minimal $\forall j$ and set $\mathbf{x}_i \Rightarrow j^{th}$ cluster
14: $\forall\, j$, $\mathbf{c_j^+} = \frac{1}{s_j} \sum_{k=1}^{s_j} \mathbf{x}_k$, where $\mathbf{x}_k \Rightarrow j^{th}$ cluster
15: **until** $\mathbf{c_j^+} - \mathbf{c_j} < \epsilon$, or no samples are re-associated to an other cluster

It can be seen that the k-means algorithm is heavily constrained. Its performance is profoundly affected by the proper selection of preliminary parameters, like the number of clusters or the initial location for the centroids. There are methods for determining a good set of parameters for a given training set [16]; however these methods usually require the construction of several clustering systems with different

parameters and evaluating their results. It is computationally intensive to calculate the distance of each sample and each centroid, so for a large number of samples, the basic algorithm has to be altered [17]. Also, if the number of clusters has to be changed, the whole iteration process has to be done from scratch, so methods for enabling on-line k-mean clustering were also developed [18, 19]. The basic method also fails to reveal appropriate clusters for non-linearly separable data. This can be worked through by formulation a similarity measure that operates in a transformed space (which is usually higher dimensional) where the samples are linearly separable [12].

3.2 Hierarchical clustering

Unlike the k-means algorithm, hierarchical clustering does not propose disjoint clusters. It builds a hierarchical structure of clusters, that can be represented as a dendrogram [5]. The leaves of the dendrogram structure are the samples themselves (each belonging to its own class), and the root of the structure is the cluster that includes all of the samples. Thus cutting the dendrogram at different levels of hierarchy results in a different number of clusters. Unlike the k-means clustering algorithm, hierarchical clustering does not require the a-priory declaration of the number of clusters, however doing so can serve as a stopping condition, resulting in faster computation for the proposed clusters. The dendrogram can be built from the leaves to the root (agglomerative method) or from the root toward the leaves (divisive method) [5].

Both agglomerative and divisive methods are based on distance measures like the Euclidean distance to compute the similarity of clusters. This measure in the context of hierarchical clustering is called dissimilarity measure, which is the basis of the four major clustering strategies [5, 20].

Single linkage clustering defines the similarity of two clusters with the help of the minimum of all pairwise dissimilarities between the elements of the two clusters [20]. The complete linkage clustering strategy defines the similarity of two clusters as the maximum of the pairwise dissimilarities between the elements of the two clusters [20]. If the similarity between the clusters is defined by the average of the pairwise dissimilarities of the samples in the two clusters, then it is group-average clustering [20]. Finally, the clusters can also be given centroids (computed from the samples belonging to the clusters), just like the clusters in the k-means algorithm. The centroid clustering strategy defines cluster similarity with the dissimilarity measure between the centroids of the clusters [20].

Agglomerative methods start with assigning a cluster for all samples. Then a cluster for the two most similar clusters is created. This process is repeated until all samples belong to a single cluster (root) or until a certain amount of clusters (*k)* is discovered [5, 20].

Divisive methods start from one cluster (the root of the dendrogram) that holds

all the samples. The cluster is divided into two sub-clusters, but due to the large number of possible splits, the evaluation of all of the possible splits would be too computationally expensive. Usually, a good split is done by finding the two elements of the cluster with the highest dissimilarity and grouping the other samples to the element of the selected two, that is more similar to the given sample [20]. The created clusters can also be split into two, while all the resulted clusters contain only one sample, or a formerly given number of clusters is discovered.

The divisive method is harder to implement, but it can extract more meaningful clusters that the agglomerative approach because the latter tends to construct clusters based on local similarities without the knowledge of the global distribution of the data, while divisive methods have global information from the beginning [20].

3.3 Fuzzy clustering

The previously introduced methods of k-means and hierarchical clustering consider clusters with hard margins, meaning that a sample either belongs to a given cluster or not. In case of fuzzy clustering methods, a sample has a degree of membership in a given cluster, which is a continuous value rather than a binary. As clustering has a close relation to set theory, most of the clustering algorithms have a fuzzy implementation. In this section, we introduce a fuzzy equivalent of the k-means algorithm, the fuzzy c-means algorithm [21, 22, 23].

The fuzzy c-means algorithm is very similar to the k-means algorithm. The applied objective function to be minimized can be seen in equation 2. Where N is the number of samples, C is the number of clusters, $\mathbf{x}_i$ is the i^{th} sample, $i \in \{1, 2 \dots N\}$, $\mathbf{c}_j$ is the j^{th} cluster centroid, $j \in \{1, 2 \dots C\}$, μ_{ij} is the degree of membership of $\mathbf{x}_i$ in cluster j, $\|.\|$ is any norm for measuring distance (like the Euclidean norm) and m is a coefficient to control fuzziness $1 \leq m \leq \infty$ [21, 22].

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij}^m \|\mathbf{x}_i - \mathbf{c}_j\|^2 \tag{2}$$

The function J_m is minimized with an iteration process, during which the degrees of membership for each sample and each clusters are updated. After the update, the new centroids for the clusters are computed. The degrees of membership can be determined according to the equation 3 [21, 22].

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}} \tag{3}$$

$$i \in \{1, 2 \dots N\}, \; j \in \{1, 2 \dots C\}$$

The centroids of the clusters are calculated like in equation 4 [21, 22].

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m \cdot \mathbf{x}_i}{\sum_{i=1}^{N} \mu_{ij}^m} \tag{4}$$

$$j \in \{1, 2 \dots C\}$$

The preliminary steps before the iterative algorithm are to define the number of clusters (C), the coefficient for fuzziness (which is usually set to 2) and to assign an initial degree of membership for all training sample for each cluster. This is usually done by filling a matrix U of size $N \times C$ with random values for μ_{ij}. The stopping condition can be formulated exactly like in the k-mean algorithm [21, 22].

After these preliminary steps, the cluster centroids are computed with the help of equation 4 based on the training samples and the given U matrix containing the degree of membership of each training sample in each cluster. Then the elements of the matrix U are modified according to equation 3. These two steps are repeated until the stopping condition is met.

It can be seen from equation 3 that in the marginal case, if m is set to be $m = 1$, the degrees of membership converge to either zero or one, making it a crisp clustering method, like k-means.

According to this approach, most of the clustering methods can be fuzzified by assigning a degree of membership to the samples.

3.4 Support vector machines

The SVM-based clustering is usually referred to as support vector clustering described in detail in [24]. The idea behind the support vector clustering method is based on the work of Schölkopf et al. [25] and Tax and Dunin [26], who introduced methods to carry out the support vector description [27] of data structures in a high dimensional space with the help of kernel functions [12].

Rather than separating the data samples from each other directly in feature space, the kernel function enables to formulate a distance measure of a higher dimensional space called feature space and use this kernel function to design the separation of the data samples [12]. Such a separation can lead to highly nonlinear, complex decision boundaries in the data space.

The complete mathematical description of the support vector clustering (SVC) method can be found in [24]. In this survey, we only explain the core idea behind this technique.

In case of SVC, the training samples are mapped to a high dimensional feature space utilizing a Gaussian kernel function. The data in feature space is enclosed in a hypersphere of center $\mathbf{a}$ and a radius of R. A penalty parameter is added to control the allowed number of outliers. An outlier $\mathbf{x}$ is a sample in data space for which $\|\Phi(\mathbf{x}) - \mathbf{a}\|_2^2 > R^2 + \xi$. Where $\Phi(.)$ is the kernel function that maps the sample $\mathbf{x}$ from the data space to the feature space, and ξ is the slack variable to enable soft margin [24].

The contour of the hypersphere forms boundaries in the data space that separates points of the data space that are inside and those that are outside of the hypersphere when mapped to the feature space, with the given kernel function. These boundaries can be non-convex and can even form disjoint sets of points in the data space. The shape of the decision boundary depends on the parameters of the kernel function and the penalty coefficient for outliers [24]. The proper tuning of these parameters depends on the noise and overlap of structures in the provided data, and it is detailed in [24]. If the parameters are all set to suitable values, then smooth disjoint boundaries should form in the data space.

The clusters are marked by the disjoint sets in the data space [24]. So two samples in the data space $\mathbf{x}_1$ and $\mathbf{x}_2$ are said to belong to different clusters if any path that connects these two points in the data space exits the hypersphere in the feature space. In [24] this criterion is checked numerically for twenty points of a connecting line between $\mathbf{x}_1$ and $\mathbf{x}_2$.

A more straightforward approach for unsupervised clustering with support vector machines is to use the one-class support vector machine (OCSVM) [25, 26]. The OCSVM method operates as the basis of the SVC algorithm. If only two clusters are expected, like in anomaly detection [8], there is no need for the cluster assignment method proposed in [24] so the system can be simplified.

3.5 Spectral clustering

Spectral clustering is used for graph partitioning [28] by analyzing graphs with methods of linear algebra. The spectral clustering algorithm is also based on a similarity measure. The training data can be represented as a similarity graph, which is an undirected graph, with the training samples as the vertexes and the edges associated with a weight of the similarity between the two vertexes they connect. From the similarity graph, the graph Laplacian is computed. The different kinds of similarity graphs and graph Laplacians can be found in [28].

The graph Laplacian matrix is used to split the data into clusters. Given a required number of clusters noted by *k*, the first *k* eigenvectors with the largest corresponding eigenvalues of the graph Laplacian are selected [28]. These eigenvectors are used as centroids for the clusters. The data samples are then associated with one of the clusters with the help of the k-means method.

The implementation and interpretation of the spectral clustering method are de-

scribed from several aspects in [28].

3.6 Decision trees

A decision tree is a tree-like graph structure. In order to assign a sample to a cluster, the data is fed to the root node of the structure. At the nodes of the structure, the data is inspected according to one of its given features and decided in which branch the given sample should be propagated. So a splitting node has branches for all possible values of the given feature towards the leaves. The leaves of the structure correspond to different clusters, meaning that data with similar features fall to the same cluster [29].

The structure is constructed based on a training set. During the training, the objective is to find the best way to split the data, so to select an appropriate inspection feature for all of the nodes. This method is called a greedy algorithm to find the splitting feature [29]. The number of clusters can be controlled by the branching factor of the structure [29].

The appropriate splitting feature is selected based on a measure of their discrimination power. A split based on a feature that can discriminate the data will result in sets that are more homogeneous than that before the split. So often, the homogeneity measure is used that can also be computed based on the distance metric between the samples [30].

Basak and Krishnapuram proposed a method for unsupervised clustering with decision trees [30]. They introduced two homogeneity measures in their paper that are based on the distance metric between the samples. During the construction of the structure, at each node, a clustering of the training set is carried out based on a single or a group of features [30].

The appropriate feature to split by in a given node can be selected by removing features from the feature space and computing its effect on the homogeneity of the data, or by simply observing the homogeneity of the data along each feature separately [30].

Let the number of data points be N. If the similarity of two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed like μ_{ij} $i, j \in \{1, 2 \dots N\}$ in equation 5, where d_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ by a distance measure that is not necessarily the Euclidean distance and d_{max} is the maximum distance of all pairwise distances between the data points [30].

$$\mu_{ij} = g\left(1 - \frac{d_{ij}}{d_{max}}\right) \tag{5}$$

$$i, j \in \{1, 2 \dots N\}$$

The function g is computed according to equation 6 [30].

$$g(x) = \begin{cases} x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The discrimination power of a feature can be calculated according to equation 7, if the discrimination power is measured by removing the feature from the feature set, and according to equation 8, if the discrimination power is measured exclusively for a given feature [30].

$$H_f = -\left(\sum_{i,j} \mu_{ij}(1 - \mu_{ij}^f) + \mu_{ij}^f(1 - \mu_{ij}) \right) \tag{7}$$

$$\hat{H}_f = -\sum_{i,j} \hat{\mu}_{ij}^f(1 - \hat{\mu}_{ij}^f) \tag{8}$$

$$i, j \in \{1, 2 \dots N\}$$

The expression μ_{ij}^f is the similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$ with the feature f removed from the feature set and $\hat{\mu}_{ij}^f$ is the similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$ only along the feature f. In both cases the feature with the highest discrimination power is selected for splitting feature at a given node [30].

Basak and Krishnapuram found that with this method, a well interpretable hierarchical clustering of the data can be made, which can also be translated into clustering rules [30]. They also found that it is better to select a single feature as a splitting criterion than a set of features.

3.7 Expectation maximization algorithm

The expectation maximization algorithm is an iterative process to find the parameters of statistical models that describe the probability distribution of the data [31]. In case of clustering, we suppose that the data can be distributed into *k* different clusters and data in each cluster have its own probability distribution with its given parameters. The type of the distribution of the clusters is based on assumption [31].

If it is known which data points belong to which clusters, the estimation for the parameters of their probability distribution can be computed. If the parameters of the probability distributions are given, the probability for each data sample coming from the given distribution, belonging to a specific cluster can be computed. This boils down to a k-means-like iteration process, where the initial parameters for the probability distribution of the clusters are initialized, the probability of samples belonging to one cluster thus can be computed, and the parameters of the probability

distributions can be refined based on these calculated probabilities [31]. Then this process is repeated until a specified stopping condition is met.

So the expectation maximization algorithm is very much like the fuzzy c-means clustering, but with a stochastic aspect [31]. Instead of the degree of membership of the samples in each cluster, the probability of the samples of belonging to the clusters is used and the parameters to be updated are the parameters of the assumed statistical model [31].

3.8 Self organizing maps (Kohonen network)

The Kohonen network is a fully connected artificial neural network with only an input layer and an output layer [32, 33]. The input vectors presented to the network are associated with one of the *k* different clusters. In the output layer of the network, there is an output neuron for every cluster. So the output layer has *k* number of neurons. The neuron with the highest activation decides the cluster a sample belongs to [32, 33].

The network can be trained with the winner-take-all method [32, 33]. Let the weight vectors of the output neurons be $\mathbf{w}_i$ where $i \in \{1, 2, \dots k\}$ and the input vectors to be $\mathbf{x}$. The j^{th} output neuron is selected as the winner neuron, and the sample is associated with its cluster, if

$$\|\mathbf{x} - \mathbf{w}_j\| = \min_{i=1\dots k} \|\mathbf{x} - \mathbf{w}_i\|$$

The weight vector with the minimum distance from the sample can also be found by finding the maximum of the scalar products of the input vector and the weight vectors if the weight vectors are all normalized [32, 33]. This can be seen in equation 9.

$$< \mathbf{w}_j, \mathbf{x} >= \max_{i=1\dots k} < \mathbf{w}_i, \mathbf{x} > \tag{9}$$

Equation 9 only holds if the weight vectors of the network are normalized, so $\|\mathbf{w}_i\| = 1 \ \forall i$. The scalar product of the input vector and the weight vectors is the activation of the output neurons. This is why the winning neuron is selected as the one with the highest activation.

The interpretation of the scalar product is the projection of x in the direction of $\mathbf{w}_i$. So if the projection is greater in a given direction, that means the input vector is more similar to the normal weight vector pointing to that direction.

In case of the winner-take-all method, only the weight vector of the winner neuron is modified during the training process [32, 33]. The objective is to minimize the squared distance between the input vector and the winning weight vector. This can be done by computing the gradient of the objective function and use gradient descent

to modify the weights of the winning neuron [32, 33].

The gradient of the objective function with respect to the weights can be seen in equation 10.

$$\frac{d\|\mathbf{x} - \mathbf{w}_j\|^2}{d\mathbf{w}_j} = -2(\mathbf{x} - \mathbf{w}_j) \tag{10}$$

According to equation 10 the weight vector $\mathbf{w}_j$ should be modified in the direction of $\mathbf{x} - \mathbf{w}_j$. The modified weights can be calculated according to equation 11.

$$\mathbf{w}_j := \mathbf{w}_j + \eta(\mathbf{x} - \mathbf{w}_j) \tag{11}$$

After the weight update, the weight vector $\mathbf{w}_j$ has to be normalized again.

If the training data is linearly separable, the weight vectors of the Kohonen network will converge to point to the center of mass of the clusters [32, 33]. The number of output neurons must be larger than the number of clusters even if the number of clusters is not known a-priory. The appropriate neurons to cluster by then can be selected, by inspecting the direction of their weight vectors during and after training and the ones that are not necessary can be omitted [32, 33].

3.9 Adaptive resonance theory

The adaptive resonance theory (ART) [34] model for neural networks is very similar to the Kohonen network, but it includes some other functionality [35]. The structure of the ART model is the same as the Kohonen network with the exception that the output neurons also implement lateral inhibition. So the activation of an output neuron decreases the activation of the others. The objective is the same, to find a weight vector that is similar to the input vector. After the classification of the input vector, the output activations are compared to a vigilance parameter [35]. If the activation of the winning neuron is higher than the vigilance parameter, the training continues like it was a Kohonen network. However, if the vigilance parameter is larger than the winning neuron's activation, it means that the presented input vector is out of an expected range around the weight vector. In this case, the winning neuron is turned off, and the prediction is made again. This is done until one of the weight vectors overcome the vigilance parameter. If it is not overcome in any trials, then such a neuron is selected that does not represent a cluster yet, and its weights are modified towards the input vector [35].

With the tuning of the vigilance parameter, ART models can control the smoothness of classification [35]. High values of the vigilance parameter result in fine clusters and a lower value of the vigilance parameter results more general, smooth clusters [35].

3.10 Autoencoders

Autoencoders are artificial neural network structures that try to reconstruct their input on their output [36]. The loss function is computed from the reconstruction error of the network. As only the inputs and the computed outputs are used for the loss function, there is no need for labeling, and thus the network can be trained in an unsupervised manner [36].

The training of the autoencoder structure is carried out with the error backpropagation algorithm. A simple autoencoder structure can be seen in Figure 1.

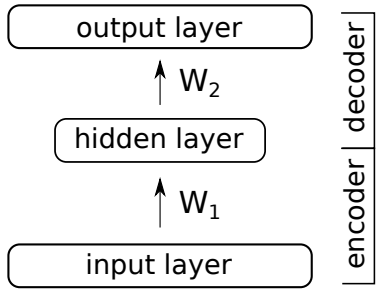

Figure 1
Simple autoencoder structure

In the structure in Figure 1, the network has fully connected layers. Let the input vector be $\mathbf{x} \in \mathbb{R}^n$, so the number of neurons in the input layer is n. The objective is to reconstruct the input in the output layer, so the output layer also consists of n number of neurons. The number of neurons in the hidden layer is chosen to be m where $m < n$. The matrix W_1 is the weight matrix between the input and the hidden layers. Each row of the W_1 matrix is a weight vector of a neuron in the hidden layer. So W_1 is a matrix of $\mathbb{R}^{m \times n}$. The rows of W_2 are the weight vectors of the output neurons, so $W_2 \in \mathbb{R}^{n \times m}$.

The hidden layer activations are computed according to equation 12, where $f(.)$ is the activation function of the hidden layer neurons, and it is applied element-wise to the result of the matrix-vector multiplication. It is important to note that the bias weights are not treated separately in this equation.

$$\mathbf{h} = f(W_1 \mathbf{x}) \tag{12}$$

The output of the network can be computed similarly to the hidden layer activations.

$$\mathbf{y} = f(W_2 \mathbf{h})$$

As the hidden layer has fewer neurons than the input layer and the objective of the

network is to reconstruct the input on its output, the encoder part is responsible for compression of the data [36]. The compressed representation of the data must be informative about the input so it can be decoded with high accuracy. So the encoder part of the network tries to extract features from the input that describe the data well. These features can be used for clustering because they represent directions in the feature space in which the input data can be well-separated [36].

The training of the structure can be accomplished by forming a loss function from the difference of the input and the output, such as $\|\mathbf{x} - \mathbf{y}\|^2$ and minimizing this function with respect of the weights of the network [36]. The gradient of the loss function with respect of the weights can be computed, so the error-back-propagation algorithm can be used to train the network without the need for labeled examples [36].

3.11 Co-localization

Image co-localization is an unsupervised method for object discovery in images. A set of images that all contain a common object are provided as training samples [37]. The task of image co-localization is to find a bounding box for the common object across all images [37].

Wei et al. proposed an image co-localization method, DDT (deep descriptor transformation) that leverages the features extracted by pre-trained convolutional neural networks [37]. A deep descriptor is a component vector of the output volume of a convolutional layer. If the output volume has the dimensions of w for width, h for height and d for depth, then a deep descriptor at the index i, j is the vector $\mathbf{x}_{i,j} \in \mathbb{R}^d,\ i \in \{1, \dots h\}$ and $j \in \{1, \dots w\}$.

The pre-trained model is presented with N number of images that has a common object on them to be localized. The deep descriptors for each image are collected $X^n = \{\mathbf{x}^n_{i,j} \in \mathbb{R}^d\}$, where $n \in \{1, \dots N\}$ and the mean vector for all of the deep descriptors is calculated as in equation 13 [37].

For equations 13,14 and 15, $i \in \{1, \dots h\}$, $j \in \{1, \dots w\}$ and $K = w \cdot h \cdot N$.

$$\bar{\mathbf{x}} = \frac{1}{K} \sum_n \sum_{i,j} \mathbf{x}^n_{i,j} \tag{13}$$

After calculating the mean vector, Wei et al. compute the covariance matrix according to equation 14 [37].

$$\mathrm{Cov}(\mathbf{x}) = \frac{1}{K} \sum_n \sum_{i,j} (\mathbf{x}^n_{i,j} - \bar{\mathbf{x}})(\mathbf{x}^n_{i,j} - \bar{\mathbf{x}})^T \tag{14}$$

The eigenvector of $\mathrm{Cov}(\mathbf{x})$ that corresponds to the largest eigenvalue is noted as ξ_1

and the first principal component of a deep descriptor at an index of i, j for a given image is described as in equation 15 [37].

$$p^1_{i,j} = \xi_1^T(\mathbf{x}_{i,j} - \bar{\mathbf{x}}) \tag{15}$$

The first principal component can be calculated for all values of i and j, and the result can be organized into a matrix $P^1 \in \mathbb{R}^{h \times w}$. The elements of P^1 for a given image with positive value represent positive correlation across all the N number of images for that descriptor, so it is likely to belong to the common object [37]. Thus the matrix P^1 is thresholded at zero for all of the images and the location of the largest connected positive region is sought. As the dimension of the P^1 matrix is the same as the feature map of the convolutional layer ($w \times h$), the location in the P^1 matrix can be reflected the image. So a region on the image can be found that correlates across all of the images [37]. A minimal enclosing bounding box can be formed for the proposed regions, thus solving the task of image co-localization. Also, if the P^1 matrix does not contain any element with a positive value, it means that the image does not contain the common object [37].

3.12 Generative models

In generative adversarial networks (GANs), there are two networks trained at the same time [11], a generator network and a discriminator network. The generator network generates data from random vectors, and the discriminator network tries to tell apart the generated and synthetic data samples. The objective of the generator network is to fool the discriminator. So it develops its internal parameters in a way that it can generate data seemingly coming from the same domain as the real training samples. The discriminator has to develop an understanding of the essential features of the data in order to be able to discriminate between the synthetic ones [11].

After both the networks are trained, the transfer of the output of the generator network can be examined by interpolating in the input space [11]. The results show that the transfer between two input vectors with input space interpolation is smooth. This implies that the discriminator network also possesses a smooth continuous representation of the feature space, which means that such a model can be used for extracting robust general low-level features even in case of training sets that are discontinuous in the feature space [11].

Mathematical operations with the input vectors also show that the generator deals with the features in the sense of similarity [38]. In case of image generation for human faces, for example, let an input vector $\mathbf{x}_{sf}$ yield an output of a smiling female face, $\mathbf{x}_{nf}$ a neutral female face and $\mathbf{x}_{nm}$ result in a neutral male face. The input vector $\mathbf{x}_{sm} = \mathbf{x}_{sf} - \mathbf{x}_{nf} + \mathbf{x}_{nm}$ will result in a smiling male face. This also implies, that the discriminator also has a sense of similarity, like smiling faces are similar, female faces are similar, male faces are similar etc. moreover, this knowledge can be utilized for clustering as well [38].

4 Applications

In this section, we introduce some examples of how the different unsupervised clustering techniques can be leveraged in deep learning applications.

The k-means clustering can be used to learn low-level filters for convolutional neural networks [39, 40, 41]. Socher et al. introduced a convolutional-recursive deep learning structure for 3D object recognition from RGB-D data [39]. A single convolutional layer first processed both the RGB and the depth modalities. They proposed an unsupervised method to build the filters based on the k-means clustering algorithm. They compared their proposed method to other models introduced in [42, 43, 44, 45]. Their experiments show, that their model, with an accuracy of 86.8 ± 3.3, was able to outperform all other methods except for the one introduced in [45], which had a 0.7% higher accuracy, but required five times more memory.

Coates and Ng introduced an unsupervised method for learning deep hierarchical features with the help of k-means clustering [46]. In their paper they describe the main considerations and limitations to perform multiple layers hierarchical feature representation with k-means clustering. They also show that this method, with an accuracy of 82%, can achieve the performance of the state-of-the-art unsupervised deep learning methods such as vector quantization (81.5% accuracy) and convolutional DBN (78.9% accuracy) on the full CIFCAR-10 dataset, but with easier implementation (only one hyperparameter 'k') and better scalability. A similar approach can be seen in [47].

Reducing the dimensionality of the data is an essential task for both visualizing high dimensional data for better understanding and for clustering, as the distance measures become simpler in reduced dimensional spaces. Yang et al. proposed a method to optimize the dimensionality reduction and the clustering method together, to construct a meaningful representation of the data in a reduced dimensional space [48]. They used a deep neural network as the dimensionality reduction system and trained it in respect to the clustering method (k-means clustering). In order to avoid trivial solutions where the network maps any input to such latent space that it can be trivially separated, a loss for the reconstruction of the input was also introduced, like in the autoencoder structure. This way the network was able to create a latent representation of the input with well separable clusters that are evenly scattered around the cluster centroids.

For graph partitioning Tian et al. showed that the reconstruction of the similarity matrix with autoencoders is a suitable alternative for the traditional matrix calculations in case of spectral clustering, in large-scale clustering problems, where the input space is very high dimensional [49]. The hidden layer activations can be used for the k-means clustering directly, instead of calculating the eigenvectors of the graph Laplacian to place cluster centroids. Based on this result Vilcek proposed an autoencoder structure for unsupervised detection of communities in social networks [50].

The connections between two neural network layers decide which features of the first layers affect which features of the second layer. In case of fully connected neural networks, all the neurons in the first layer can affect the activation of each neuron in the second layer. It is decided during training, which connections are neglected and which are of greater importance, by tuning the weights associated with the connections. Connections get neglected because not all first layer features are necessary for the computation of a given second layer feature [51, 41]. For unsupervised learning, the tuning of the weights this way is not always possible, and it is also computationally ineffective. Unsupervised clustering can be used to design the connections between neural network layers [51, 41].

Bruna et al. proposed a generalization of deep convolutional architectures (locally connected networks) based on analogies with graph theory and introduced a hierarchical and a spectral construction method for convolutional structures [52]. Experiments were carried out on a downsampled MNIST dataset, where the proposed method was able to achieve equal or lower classification error than a fully connected network that had more than twice the amount of parameters. In [53] another study on the topic of spectral methods for deep learning is presented.

Fuzzy rules can be extracted from the collected data with the help of deep learning [54]. In [54] a method is proposed for extracting fuzzy rules from the data by feeding it to a restricted Boltzmann machine and applying a probability based clustering method (similar to the expectation maximization algorithm) to form the fuzzy rules.

DFuzzy is a deep learning based fuzzy clustering approach for graph partitioning [55]. DFuzzy enables vertexes to belong to multiple clusters with different degree. An autoencoder structure is used to create graph partitions that can be mapped to vertexes by the decoder. An initial clustering of the graph is performed with the PageRank algorithm [56].

The NDT (neural decision tree) is a hybrid architecture of a decision three and multilayer neural networks [57]. At each node in the decision tree, the splitting is implemented by a neural network. Describing the structure as a whole and assuming shared weights, enable the optimization of the whole architecture globally. The authors compared the test set accuracy of the NDT, a decision tree and a neural network on 14 different datasets, and found that none of these methods have a significant advantage over the other, but the NDT model accuracy is in the top two on 13 of the 14 datasets.

Patel et al. proposed a probabilistic framework for deep learning [58]. Their proposed model, the Deep Rendering Mixture Model (DRMM) can be optimized with the expectation maximization algorithm. The method was introduced as an alternative for deep convolutional neural networks and their optimization with backpropagation. The model can be trained in an unsupervised and semi-supervised manner. The experiments show that the best performing DRMM architectures were able to achieve a test error rate of 0.57%, 1.56%, 1.67% and 0.91% in a semi-supervised scenario for the MNIST dataset with 100, 600, 1000 and 3000 labeled examples respectively.

Most of the convolutional and generative models for comparison had error rates that are nearly two times greater than these.

In [59] a spatial mixture model was proposed for the unsupervised identification of entities in the input. In the mixture model, each entity is described by a neural network with a given set of parameters. Based on the expectation maximization algorithm, an unsupervised clustering method is introduced, that enables optimization by differentiation. The basic idea behind this approach is similar to the neural decision tree [57], but instead of a decision tree, a mixture model is created.

A detailed description of the autoencoder structure, its role in unsupervised learning and place in deep learning, along with different types of autoencoders can be found in [36].

Radford et al. introduced the Deep Convolutional Generative Adversarial Network structure (DCGAN) [60]. In their work, they showed that deep convolutional models, as generative networks, can extract useful features from the presented images in an unsupervised manner. Their result shows that both the generator and the discriminator network can be trained to extract general features and thus they can be used for other purposes as well, for example as feature extractors.

Summary

The current results show that deep learning can benefit a lot from unsupervised clustering methods. The applications that utilize unsupervised learning in the process of deep learning, perform well in many cases [39, 41, 46, 48, 52, 57, 58] and can have other advantages, like fast training and inference, smaller memory needs and easy implementation due to the lack of labeling. This paper promotes the use of unsupervised techniques in the field of deep learning and argues that a significant aspect of deep learning research should be to find ways to exploit the information provided by the sheer data better, rather than acquiring more and more data in order to build even more complex models to enhance performance.

Acknowledgement

Róbert Fullér has been partially supported by FIEK program (Center for Cooperation between Higher Education and the Industries at the Széchenyi István University, GINOP-2.3.4-15-2016-00003) and by the EFOP-3.6.1-16-2016-00010 project.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[3] Allan Hanbury. A survey of methods for image annotation. *Journal of Visual Languages & Computing*, 19(5):617–627, 2008.

[4] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1:2012, 2012.

[5] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[6] P. Berkhin. A Survey of Clustering Data Mining Techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[7] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[8] A. I. Károly, J. Kuti, and P. Galambos. Unsupervised real-time classification of cycle stages in collaborative robot applications. In *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 97–102, Feb 2018.

[9] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

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

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Bernhard Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pages 301–307, 2001.

[13] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[14] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[15] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm:

Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[16] Catherine A Sugar and Gareth M James. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[17] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[18] Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. *CoRR*, abs/1412.5721, 2014.

[19] Isis Bonet, Adriana Escobar, Andrea Mesa-Múnera, and Juan Fernando Alzate. Clustering of Metagenomic Data by Combining Different Distance Functions. *Acta Polytechnica Hungarica*, 14(3), October 2017.

[20] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008.

[21] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[22] Wang Peizhuang. Pattern recognition with fuzzy objective function algorithms (james c. bezdek). *SIAM Review*, 25(3):442, 1983.

[23] Ernesto Moya-Albor, Hiram Ponce, and Jorge Brieva. An Edge Detection Method using a Fuzzy Ensemble Approach. *Acta Polytechnica Hungarica*, 14(3):20, 2017.

[24] Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[25] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[26] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[28] Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.

[29] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[30] Jayanta Basak and Raghu Krishnapuram. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE transactions on knowledge and data engineering*, 17(1):121–132, 2005.

[31] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[32] Teuvo Kohonen and Timo Honkela. Kohonen network. *Scholarpedia*, 2(1):1568, 2007.

[33] Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

[34] Stephen Grossberg. Adaptive resonance theory. Technical report, Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems, 2000.

[35] Gail A Carpenter, Stephen Grossberg, and David B Rosen. Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural networks*, 4(4):493–504, 1991.

[36] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012.

[37] Xiu-Shen Wei, Chen-Lin Zhang, Jianxin Wu, Chunhua Shen, and Zhi-Hua Zhou. Unsupervised object discovery and co-localization by deep descriptor transforming. *CoRR*, abs/1707.06397, 2017.

[38] Tom White. Sampling generative networks: Notes on a few effective techniques. *CoRR*, abs/1609.04468, 2016.

[39] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 656–664, 2012.

[40] Eugenio Culurciello, Jordan Bates, Aysegul Dundar, José Antonio Pérez-Carrasco, and Clément Farabet. Clustering learning for robotic vision. *CoRR*, abs/1301.2820, 2013.

[41] Aysegul Dundar, Jonghoon Jin, and Eugenio Culurciello. Convolutional clustering for unsupervised learning. *CoRR*, abs/1511.06241, 2015.

[42] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824, May 2011.

[43] L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 821–826, Sept 2011.

[44] M. Blum, Jost Tobias Springenberg, J. Wülfing, and M. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *2012 IEEE International Conference on Robotics and Automation*, pages 1298–1303, May 2012.

[45] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[46] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012.

[47] M. Dundar, Q. Kou, B. Zhang, Y. He, and B. Rajwa. Simplicity of kmeans versus deepness of deep learning: A case of unsupervised feature learning with limited data. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 883–888, Dec 2015.

[48] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *CoRR*, abs/1610.04794, 2016.

[49] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[50] Alexandre Vilcek. Deep learning with k-means applied to community detection in network. Project Report CS224W-31, Stanford University Center for Professional Development, 2014.

[51] Eugenio Culurciello, Jonghoon Jin, Aysegul Dundar, and Jordan Bates. An analysis of the connections between layers of deep neural networks. *CoRR*, abs/1306.0152, 2013.

[52] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[53] L. Shao, D. Wu, and X. Li. Learning deep and wide: A spectral method for learning deep networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(12):2303–2308, Dec 2014.

[54] Erick De la Rosa and Wen Yu. Data-driven fuzzy modeling using deep learning. *CoRR*, abs/1702.07076, 2017.

[55] Vandana Bhatia and Rinkle Rani. Dfuzzy: a deep learning-based fuzzy clustering model for large graphs. *Knowledge and Information Systems*, pages 1–23, 2018.

[56] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[57] Randall Balestriero. Neural decision trees. *CoRR*, abs/1702.07360, 2017.

[58] Ankit B Patel, Minh Tan Nguyen, and Richard Baraniuk. A probabilistic framework for deep learning. In *Advances in neural information processing systems*, pages 2558–2566, 2016.

[59] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. *CoRR*, abs/1708.03498, 2017.

[60] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.