CE888 Assignment 2

Ogulcan Ozer

***Abstract*—In this work, we explore autoencoders with a hidden softmax layer as a clustering method, by training an autoencoder and using the maximum of the output vector as our cluster assignment. We also examine traditioanl clustering algorithms and experiment with feature selection methods to see how they perform compared to each other.**

I. Introduction

UNSUPERVISED learning is a branch of machine learning that deals with unlabeled data to find patterns, groups and relations. This sub section of machine learning is useful to describe and understand unknown data, which also helps us understand the properties of future data samples. Right now we are collecting data much faster than we can process it. And most of the time the data we acquire is unlabeled, and these huge amounts of data are complex, noisy and it might include unnecessary information.

There are different ways to reduce the complexity of a given data and learn good features from it. In this work, we looked at PCA and autoencoders. Autoencoders are a part of representational learning, which can be trained with or without labels. They play a key role in deep learning applications. They are used in dimensionality reduction and feature extraction, and together with deep learning they gained a lot of traction after Hinton proposed a method to train deep belief networks one layer at a time [1]. Another addition to the autoencoders and PCA was assigning the data to clusters usnig the softmax activation function. Since softmax gives us the probability distribution over-the classes of our data- the output vector [2, pp. 178-181], we can use the output of our encoder for cluster assignment. The motivation for using autoencoders is, in what they do they are very similar to traditional feature selection algorithms, but they are also capable of learning good features from complex and high dimensional data. This aspect of autoencoders and unsupervised learning in general is important. As LeCun states in [3], people are not paying attention to unsupervised learning because of the success of the supervised learning in recent years. But it will be far more important in the future because of how living beings perceive and learn, they do not need labels to recognize the world around them.

In this paper In Section III, We explain the tools we used and mentioned the implementation steps we took for the experiments. Then we present the datasets that we used for the task and give detailed information about them.The detailed experiments and analyses for the task will be mentioned in Section IV. Finally, we talk about how we evaluated our models and we discuss the results.

II. Background

Clustering is used to naturally group the samples in order to extract information or make decisions [4]. Autoencoders were discovered in 1985 and presented in [5]. As the technology and our computational power improved immensely in the last decade, they became more and more feasible. After Hinton's solution [1] these simple network models became very popular, now they are used in the state-of-the-art machine learning models, breaking records and in some tasks performing better than humans [6]. But most of the success and interest to the autoencoders can be attributed to the studies on supervised learning.

In this study, first we looked at the well-known traditional models k-Means [7] and DBSCAN [8]. K-Means tries to divide the data into k clusters which are defined by k centroids. Each sample given to the algorithm is assigned to the cluster with the nearest mean. The k-Means algorithm is not deterministic and results in different clusters for different initializations. DBSCAN on the other hand separates the data by looking at it as highly populated areas that are separated by low population areas. The DBSCAN algorithm, as opposed to k-Means, is deterministic and results in same clusters.

After the traditional methods, we trained different autoencoders for dimensionality reduction and clustering. The model of our autoencoder networks is a simple end to end auto encoder with one hidden layer. First model uses linear

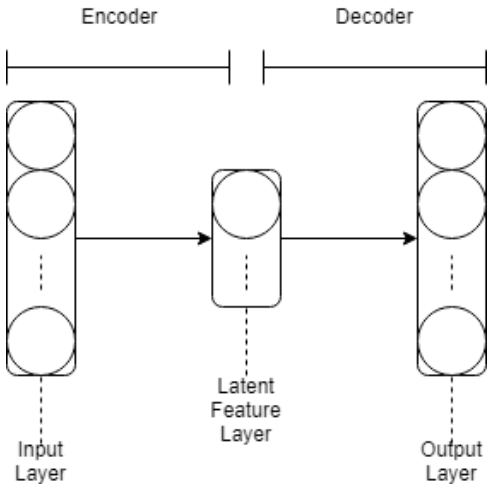

Fig. 1. Autoencoder model

activation functions, in order to achieve PCA-like behavior. Then our second autoencoder model uses non-linear sigmoid activation function, for the purpose of learning good features from high-dimensional data.

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{1}$$

The output of these two models are used as the inputs to the traditional clustering algorithms. Finally, our last model uses a softmax activation function in the latent feature layer. Softmax provides the probability distribution over the nodes of our latent feature layer. Which are then used to cluster the data

$$softmax_i(a) = \frac{\exp a_i}{\sum \exp a_i} \tag{2}$$

Even though the k-Means algorithm cannot cluster non-linear data, [9] and [10] show that traditional clustering methods can be improved significantly if a good feature selection method is applied before the clustering procedure. In [11] feature selection with autoencoders are compared to other feature selection methods like PCA , non-linear PCA etc. And it is stated that the autoencoders perform better than these techniques if they are initialized and trained carefully. Another study [12] learns overcomplete representations of features for data reconstructions. It is shown in their results that this sparse, energy based autoencoder with linear layers can reproduce the MINST dataset with 0.7% error, while not requiring any preprocessing. [13] also follows a similar approach, in which a deep sparse network trained in unsupervised fashion to recognize faces, humans and objects..

The most similar work [9] uses stacked denoising autoencoders to extract good features for clustering. When the pre-training of their SAE is over, the trained encoder model is extracted and merged with a custom layer which applies softmax activation to the resulting output vector from the encoder. After setting up the final model, they use k-Means to generate cluster centroids for the data. Then, these centroids are used to train/fine-tune the final model by minimizing the KL-divergence between the soft assignment of the network and the cluster centroids created by k-Means.

III. Methodology

In this study, we experimented with autoencoders using a softmax latent feature layer, to bin the data into the softmax nodes and use the maximum of this vector for our clustering assignment. But before testing our main objective, we first set up a base line specific for our datasets using the traditional clustering algorithms. We also looked into how feature selection procedures affect the performance of these clustering algorithms. First with PCA, then with linear autoencoders and finally with non-linear autoencoders. After creating the baseline and experimenting with feature selection, we proceeded to our main objective.

Implementation of the tasks were done using Python. Reading the data files and importing procedures were implemented using the Pandas library. The traditional clustering algorithms were implemented using the scikit-learn library which already includes the k-Means and DBSCAN algorithms. Also some data processing operations like normalization/scaling, train-test split were done using the functions provided by scikit-learn. For the tasks related to autoencoders, TensorFlow and Keras libraries were used. Then, a high level autoencoder class was implemented in order to easily create autoencoders with desired layers, nodes and activation functions. Finally, the data presentation and plotting the results were done using matplotlib and seaborn libraries.

The three datasets chosen to be clustered in this study are Modified NIST, Human Activity Recognition Using Smartphones (HAR) and High Time Resolution Universe Survey 2 (HTRU2).

MNIST is a image dataset of hand written digits. It is a modified version of the NIST dataset collected from Census Bureau employees and high-school students and processed by LeCun et.al..The original NIST set was normalized, so that each sample would be centered according to the weights of it's pixels in a 28x28 box. Images were also converted to grey-scale and shuffled to make it easy to use in machine learning applications [14]. The dataset consists of 60.000 training images and 10.000 test images. Since each sample is represented as a 28x28 pixel binary image, it can be said that samples have 784 features. MNIST dataset is frequently used in machine learning research, which makes it easier for researchers to compare their models and solutions to the others in the field.

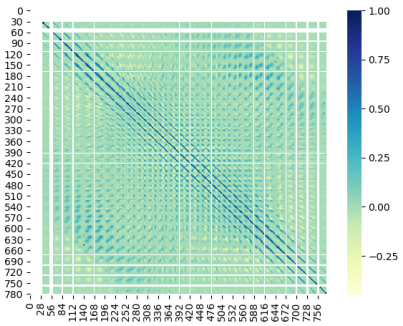

Fig. 2. Corraletion heatmap of MNIST features.

HAR is a dataset of multiple classes of human activities. It was created by recording different classes of activities of 30 volunteers by using the accelerometer and the gyroscope in a smartphone. These sensors were used to record the linear acceleration and angular velocity on three axes at 50 Hz. Volunteers were also recorded by a camera for the labeling of the activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) [15]. The dataset consists of 7352 training samples and 2947 test samples. Each sample has 561 features which are- extracted- estimated from the recorded accelerometer and gyroscope signals. These features are normalized, scaled and shifted to be within [-1,1]. Subject id labels for the features and mentioned raw accelerometer and gyroscope signals are also provided with the dataset. Dimensions of the subject ids are the same with their training and testing parts. On the other hand, the raw signals are separated into nine different files grouped under three axes, which are body acceleration (X,Y,Z), body gyroscope (X,Y,Z) and total acceleration (X,Y,Z).

Our last dataset HTRU2 was obtained by Bates [16] using ANNs to first filter out the bad candidates, then manually inspecting the outputs of the ANN. Each sample of HTRU2 is a set of features extracted from the data acquired by large radio

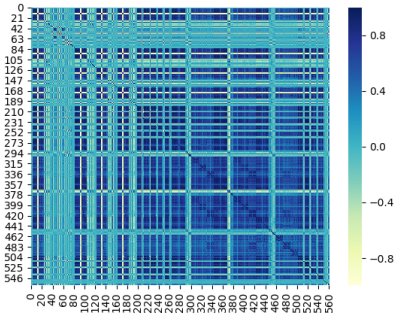

Fig. 3. Corraletion heatmap of HAR features.

telescopes during High Time Resolution Universe Survey. HTRU2 has 8 features and consists of 17,898 total samples. Only 1,639 of them are positive pulsar samples and the 16,259 of them are negative samples. The first four features are simple statistics of the pulsar profiles and rest of the features were obtained from DM-SNR curve [17].These compact features were obtained from 90.000 labeled pulsar candidates by using Pulsar Feature Lab [18].

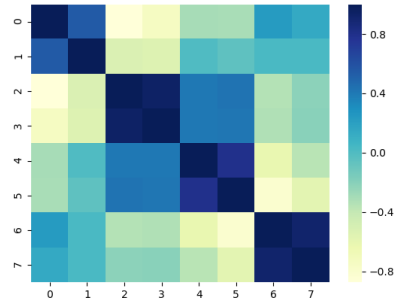

Fig. 4. Corraletion heatmap of htru features.

TABLE I
DATASET STATISTICS.

Datasets	Data Points	Features	Classes
MNIST	70.000	784	10
HAR	10.299	561	6
HTRU2	17,898	8	2

IV. EXPERIMENTS

The experiments done in this work can be separated into three sub-groups. First part is the clustering algorithms. The second part is the effect of feature selection methods. And the third part is the clustering using the maximum of the softmax output vector. Before running the experiments, for repeatability purposes, the randomness states were fixed.

A. Clustering

First, to have a base-line to compare our results we clustered our data using the k-Means algorithm. Since our datasets are labeled, we set the number of centroids k to the number of classes for each dataset, and we set the number of initializations to 10.

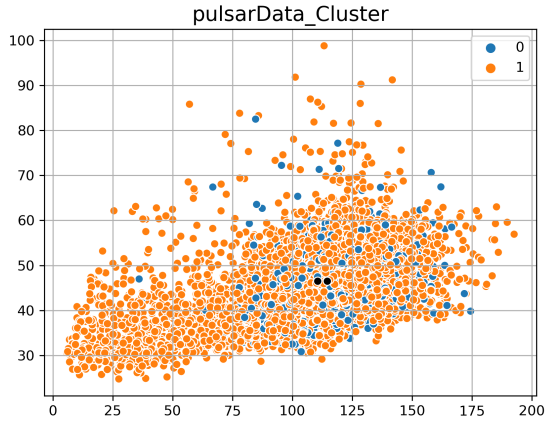

Fig. 5. Plot of first two features of the HTRU dataset, clustered and labeled by k-Means

TABLE II
K-MEANS CLUSTERING RESULTS.

Data - Method	Homogenity	Complete.	V-Measure	Accuracy
htru k-Means	0.031206	0.023027	0.026500	0.234949
har k-Means	0.570123	0.600019	0.584689	0.289853
mnist k-Means	0.486553	0.497699	0.492063	0.083400

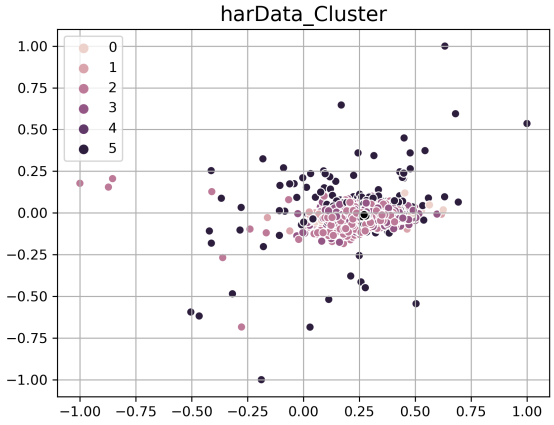

Fig. 6. Plot of first two features of the HAR dataset, clustered and labeled by k-Means

In addition to the k-Means, DBSCAN algorithm was also used to cluster our datasets. But the DBSCAN algorithm provided by the scikit-learn library labeled our datasets as noise and failed to cluster them. Which we think is because of the high dimensionality of our data. Thus, the results of the DBSCAN algorithm were discarded. Also because of the poor results of the MNIST, all the centroids were stacked at the origin and its k-Means plot was not included here.

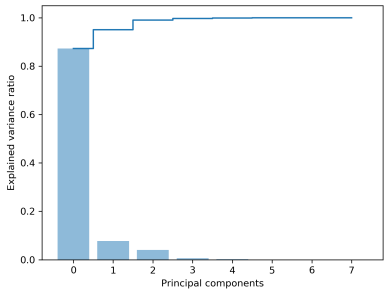

Fig. 7. HTRU Explained variance ratio.

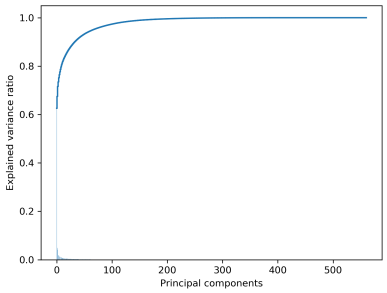

Fig. 8. HAR Explained variance ratio.

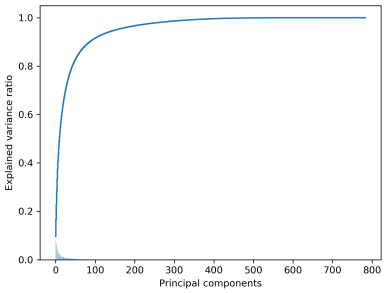

Fig. 9. MNIST Explained variance ratio.

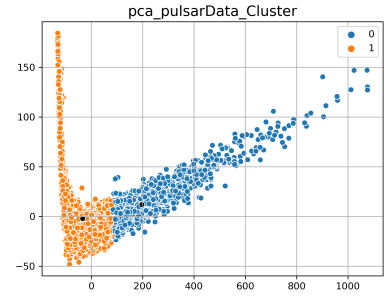

Fig. 10. HTRU clustering after PCA.

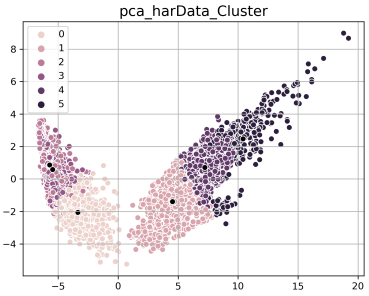

Fig. 11. HAR clustering after PCA

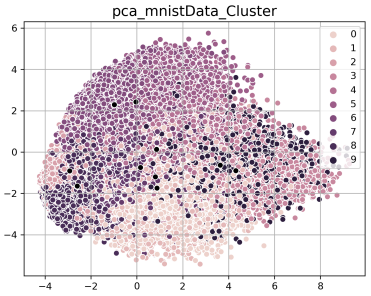

Fig. 12. MNIST clustering after PCA

TABLE III
K-MEANS CLUSTERING RESULTS AFTER PCA.

Data - Method	Num. Components	Homogenity	Completeness	V-Measure	Accuracy
htru k-Means w/PCA	4	0.031206	2.302700e-02	2.649971e-02	0.765051
har k-Means w/PCA	200	0.570146	5.996766e-01	5.845388e-01	0.318961
mnist k-Means w/PCA	400	0.486113	4.971782e-01	4.915831e-01	0.072782

B. Feature Selection and Clustering

1) PCA: For the feature selection, the first method we implemented was PCA. We transformed our data using the PCA function and retrieved the explained variance ratio to decide how many components we need. As it can be seen in figures 7,8 and 9, we choose the least amount of components representing almost all of the data.(≅0.99%) As a result, we used 4 components for HTRU, 200 for HAR and 400 for MNIST and clustered the PCA reduced data using the k-Means algorithm.

2) Linear AE: After clustering with PCA, we implemented an end-to-end autoencoder with linear activation functions. The autoencoder has the basic structure shown in Figure 1. The input and output layers were set to the number of features of the datasets, and the latent feature layers were set to the same numbers we used in the PCA reduction. 'Adam' was used as the optimizer, and 'mean squared error' was used as the loss function. Finally, the batch size was set as 128, number of training epochs were set to 50 and our test sets were assigned as the validation data.

3) Non-linear AE: For the last feature selection method, we implemented the same autoencoder but replaced the linear activation function in the encoder model with a non-linear activation function, which is in this case was sigmoid. We also increased the number of training epochs for the non-linear case to 100. Previous number of training epochs was not enough for the non-linear autoencoder, which ended up as an underfitted model.

C. Softmax Clustering

In our final experiment, since our objective is to 'bin' our data into the softmax nodes; We created a new auto encoder with a softmax latent feature layer that has N nodes where N is the number of classes the given dataset has. Then for the output layer we followed [9] and used Rectifier Linear Unit(ReLU) activation function, and we kept the rest of the network parameters same. After training the models, we separated each test set according to their class labels. Then we predicted each 'class-set' one by one to see which node played an active role in the clustering/binning of a given class sample. Finally, as it can be seen in the Table 5, if the classes of a given set was binned properly, we made the assumption that the model was trained correctly and evaluated the scores on all of the test set. The results presented here only show the nodes that best cluster a given class-set, excluding the HTRU set because

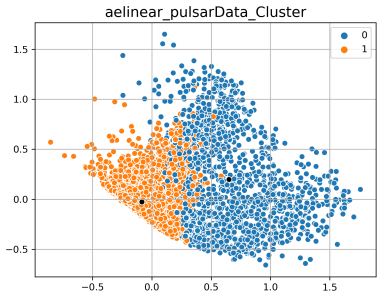

Fig. 13. HTRU Explained variance ratio.

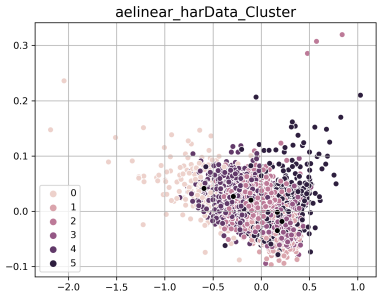

Fig. 14. HAR Explained variance ratio.

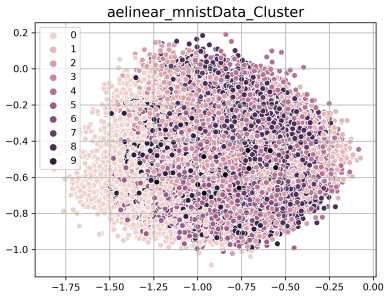

Fig. 15. MNIST Explained variance ratio.

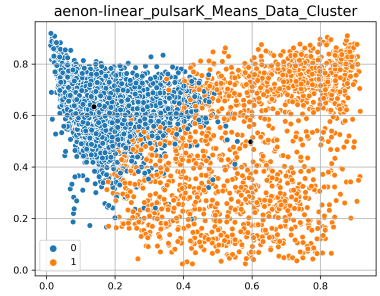

Fig. 16. HTRU clustering after PCA.

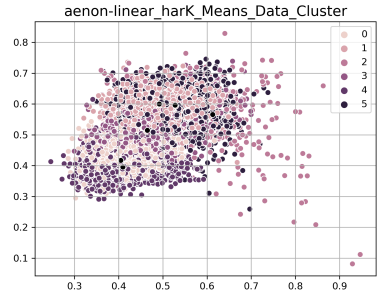

Fig. 17. HAR clustering after PCA

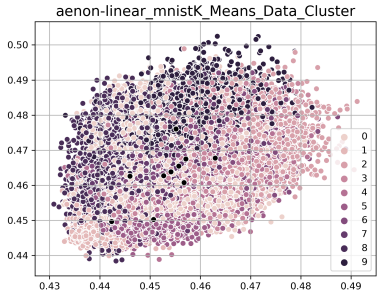

Fig. 18. MNIST clustering after PCA

TABLE IV
K-MEANS CLUSTERING RESULTS FOR LINEAR AND NON-LINEAR AUTOENCODERS.

Data - Method	Num. Latent Features	Homogenity	Completeness	V-Measure	Accuracy
htru k-Means w/LinAE	4	0.421208	3.674883e-01	3.925187e-01	0.935326
har k-Means w/LinAE	200	0.548367	5.951099e-01	5.707828e-01	0.180495
mnist k-Means w/LinAE	400	0.247627	2.516646e-01	2.496297e-01	0.087673
htru k-Means w/N-LinAE	4	0.361558	0.312014	0.334964	0.925059
har k-Means w/N-LinAE	200	0.542958	0.574462	0.558266	0.184576
mnist k-Means w/N-LinAE	400	0.493461	0.500976	0.497190	0.148200

it only has two classes. Complete class versus node accuracy tables can be found in the Appendix.

TABLE V
HTRU SOFTMAX RESULTS.

Class vs. Output Node Accuracy	Node 0	Node 1
Class 0	**0.9963**	0.00368
Class 1	0.26168	**0.73831**
HTRU Test Accuracy:		0.973184

TABLE VI
HAR SOFTMAX RESULTS.

Class vs. Output Node Accuracy	Node 3	Node 4	Node 5
Class 0	0.131	0.004	**0.77**
Class 1	0.00	**0.513**	0.377
Class 2	0.009	0.007	**0.704**
Class 3	**0.847**	0.0	0.0
Class 4	**0.874**	0.0	0.0
Class 5	**0.774**	0.0	0.0

V. DISCUSSION

Evaluation of the experiments in which we use k-Means, were done using scikit-learn metrics library. For each experiment completeness, homogeneity, v-measure and accuracy scores were calculated.

When we look at the results, we can clearly see that the k-Means algorithm with feature selection performs much better than our baseline clustering. But comparing the feature selection methods is complicated. If we look at PCA and linear autoencoder, their results were usually similar but differences could still be observed. And this point brings us back to [11]. Even though PCA is a robust algorithm, autoencoders are not and they can be difficult to train. Then, when we look at non-linear autoencoders, it is mostly an improvement over the other methods. Especially the performance increase on a complex dataset like MNIST is great. In the end we believe that if we

TABLE VII
HAR SOFTMAX RESULTS.

Class vs. Output Node Accuracy	Node 0
Class 0	**1.0**
Class 1	**1.0**
Class 2	**1.0**
Class 3	**1.0**
Class 4	**1.0**
Class 5	**1.0**
Class 6	**1.0**
Class 7	**1.0**
Class 8	**1.0**
Class 9	**1.0**

need to extract good features, we should start with simple and robust methods like PCA and try more complex methods like autoencoders if the former does not help.

For the softmax clustering part, evaluation was not as straightforward as the first part. Because when we try to cluster our data- even if we assume that it can cluster the data perfectly- we need to find out which output node clusters which class of the data. But this is a very strong assumption. Since we do not have any control over how the hidden softmax layer learns to represent the data while training, we can end up with endless combinations of possible representations. We can see this problem occuring in the results of the HAR and MNIST datasets. However, the results we got for the HTRU dataset was surprisingly good, and when we repeated the experiment our results were consistent most of the time. We believe the success of the HTRU dataset is related to its small amount of features and classes. In the future, we will definitely try different datasets that are a little bit larger than HTRU and compare it to our results.

The prediction and evaluation procedure we follow can be seen in Figure 19. After we predict the class-sets and apply argmax function to the resulting array, we end up with the indices of the nodes. Then, we calculate the accuracy score of the node index array compared to the class-set labels, which are just arrays filled with their own class labels. After calculating the accuracy for every class-set label, we end up with a distribution that shows how much of the predicted class-set is clustered by which output node.

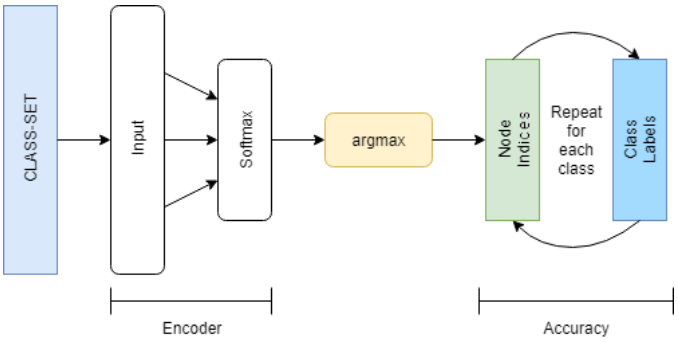

Fig. 19. Prediction and node-class accuracy calculation process.

VI. CONCLUSION

We briefly explained the purpose and the tasks of the project, and presented our results. While going through the literature, importance of unsupervised learning was understood. Also, it was realized that the field needs more attention. Later on, we described our model and the datasets we used. Then the experiments we have done were explained in detail. Finally, analyzed and discussed our results.

REFERENCES

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[7] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[9] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[10] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*, 2007, pp. 412–419.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[12] C. Poultney, S. Chopra, Y. L. Cun *et al.*, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, 2007, pp. 1137–1144.

[13] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," *arXiv preprint arXiv 1112.6209*, 2011.

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

[15] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[16] S. Bates, M. Bailes, B. Barsdell, N. Bhat, M. Burgay, S. Burke-Spolaor, D. Champion, P. Coster, N. D'Amico, A. Jameson *et al.*, "The high time resolution universe pulsar survey—vi. an artificial neural network and timing of 75 pulsars," *Monthly Notices of the Royal Astronomical Society*, vol. 427, no. 2, pp. 1052–1065, 2012.

[17] M. Keith, A. Jameson, W. Van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. Bates, N. Bhat, M. Burgay *et al.*, "The high time resolution universe pulsar survey–i. system configuration and initial discoveries," *Monthly Notices of the Royal Astronomical Society*, vol. 409, no. 2, pp. 619–627, 2010.

[18] R. Lyon, "Pulsar feature lab," 9 2015. [Online]. Available: https://figshare.com/articles/Pulsar_Feature_Lab/1536472

APPENDIX
FULL TABLES

See page 8.

TABLE VIII
HAR SOFTMAX CLUSTERING RESULTS.

Class vs. Output Node Accuracy	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5
Class 0	0.0	0.086	0.0	0.131	0.004	**0.77**
Class 1	0.0	0.006	0.10	0.00	**0.513**	0.377
Class 2	0.0	0.24	0.038	0.009	0.007	**0.704**
Class 3	0.0	0.0	0.152	**0.847**	0.0	0.0
Class 4	0.0	0.0	0.125	**0.874**	0.0	0.0
Class 5	0.0	0.0	0.225	**0.774**	0.0	0.0

TABLE IX
HAR SOFTMAX CLUSTERING RESULTS.

Class vs. Output Node Accuracy	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9
Class 0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 1	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 2	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 3	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 4	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 5	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 6	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 7	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 8	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Class 9	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0