DD2430 - Project Course in Data Science
Attractor Memory and Sparse Representations for Multi-modal Clustering

Pierre Marza and Antoine Sueur
Department of Computer Science
KTH Royal Institute of Technology
Stockholm, SWEDEN
marza@kth.se - sueur@kth.se
Supervisor : Pawel HERMAN

Abstract

The human neocortex uses efficient learning architectures and data representations that allow for generalization and, particularly, multi-modality, i.e. semantic connections between different spaces or domains (language and vision for instance). The idea developed in this paper is to try to model this behavior through an autoassociative memory and feedforward representation learning networks inspired from the way neural information is processed in the neocortex. We also aim to prove the necessity of sparsifying information to gain generality and robustness, as it seems to be the case in some of the neural information processing pathways in the human brain. To this end, we produce a sparse representation of an image and its associated caption thanks to a sparse autoencoder and we send it to an attractor model to perform unsupervised learning on the fused representations. The challenge is to obtain basins of attractors which are wide enough to be able to perform clustering.

1 Introduction

Multi-modal learning is a promising field of research as it can be a powerful way to gain generalization capabilities at no cost in certain domains using prior knowledge acquired using other modalities. One can imagine that combining both image, audio or textual data might carry more information and allow for better results given certain tasks such as sentiment analysis or scene recognition in videos. One of the reasons why human neocortex is so efficient could be this ability to connect the different representations of a concept (vision, sound, language...) to a high-level idea. For instance linking the concept of what a cat looks like and how it is textually described to the idea of what a cat is.
Considering the fact that only a fraction of the neurons are active at the same time in the brain, it also seems like sparsity might play an important role in handling different sensory inputs and the possible absence of part of them. For a neural network, this means being able to associate data to specific clusters even when textual data or image data is missing.

Nowadays, state of art in multi-modal learning is achieved through Deep Neural Networks and gradient based learning using backpropagation ([11], [12], [13], [14]). This approach allows to obtain great results but on the other hand, suffers from limitations. Indeed, such algorithms need huge computation capacities and a vast amounts of training data. Our aim in this paper is to try to approach results we could have obtained using Deep Learning but in a more biologically plausible way, reducing the necessary computation and data.

2 Related work

One main inspiration for using sparse representations comes from the company Numenta and their work on distributed sparse representations[1] (SDRs). These representations rely on very sparse binary vectors built in a distributed way to achieve robustness and generalization on specific problems. A distributed representation is a way to encode information using general features. In this way, similar vectors will use the same features, allowing the creation of semantic links between them. We have also used the Bayesian Confidence Propagating Neural Network (BCPNN) architecture for our model, which was thoroughly detailled by in *Attractor Memory with Self-organizing input* (Johansson, Lansner, 2006)[2]. This architecture relies on the rules of associative or Hebbian learning to train a neural network based on the co-activations of neurons without ever relying on backpropagation[3]. Another idea introduced in this paper is to sparsify the input to get a new hidden representation that would later be sent to the BCPNN.

3 Problem description

We aim to make a brain inspired neural network model capable of learning an unsupervised representation of an image and its associated caption to perform clustering. The idea is to observe on which attributes of the data each cluster has been built to see if the modalities are of equal importance. Our final goal is to obtain representations that are sparse enough to be robust to a lack of information : after learning, we want to be able to associate a new image to the right cluster even if we do not have the caption and vice versa.
As a result, the final question we are trying to answer in this paper is : **Can a brain inspired attractor memory achieve results close to the ones of deep learning state-of-the-art multimodal architectures and is sparseness in representations a key to generalization and robustness ?**

4 Methodology and implementation

4.1 Architecture

First of all, we need to get an initial representation of the image and the caption. To this end, we perform feature extraction using a pretrained Residual Neural Network (Resnet-18) for the image and a Word2Vec embedding for the textual caption. The Convolutional Neural Network outputs a 1000-dimension representation for a given 3x224-dimension image and the word embedding allows us to get a 300-dimension vector representing the caption.

Then, we normalize the two representations and concatenate them to have a 1300-dimension vector containing all the information. We send the latter to a sparse autoencoder with a single hidden layer containing 2000 units using sigmoid as activation function. We train this overcomplete autoencoder using gradient descent on the reconstitution mean squared error and a regularization term. Indeed, we use here L1 regularization to sparsify the encoded representation by avoiding trivial mappings. Moreover, we have been able to tune the number of hidden units and the coefficient representing the importance of the regularization term inside the minimized error to try to get a very sparse representation without losing information when encoding the input.

The output of the autoencoder is discretized before being sent to a BCPNN. Indeed, this network is organized as a set of hypercolumns all containing the same number of units. Each unit in the network is connected to all units in the other hypercolumns but there is no interaction between units inside a hypercolumn. We chose to build 2000 hypercolmuns, i.e. one for each feature of the encoded vector, containing 4 units. As a result, we need to discretize the representation by converting each continuous dimension into a vector of 4 binary values, one being equal to 1 and the 3 others to 0.
BCPNN has a Hebbian type of learning rule based on the probabilities of co-activation of the different units connected together. Indeed, if we index presynaptic units with i and postsynaptic units with j, we compute the following probability estimates :

$$p_i = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu$$

$$p_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu$$

Where P denotes the number of training patterns and ξ_i^u is the i^{th} coordinate of the μ^{th} pattern.

Then, we deduce the weights between units and the biases as follows,

$$w_{ij} = \begin{cases} 0 & \text{if } p_i = 0 \text{ or } p_j = 0 \\ \frac{1}{P} & \text{else if } p_{ij} = 0 \\ \frac{p_{ij}}{p_i p_j} & \text{otherwise} \end{cases}$$

$$\beta_i = \begin{cases} \frac{1}{P^2} & \text{if } p_i = 0 \\ p_i & \text{otherwise} \end{cases}$$

The activation function that we use during the recall phase of the BCPNN is a softmax function inside each hypercolumn : by tuning the temperature of the softmax function, we succeeded in always having only one unit much more activated than the others inside a hypercolumn (to stay consistent with the way we discretize the output of the autoencoder to send it to the BCPNN).

We finally perform clustering using a simple k-means method on the output of the BCPNN after convergence.

Fig.1 gives a synthetic and more visual description of the architectures we are using.

4.2 Dataset

We created a simple dataset to evaluate the efficiency of our architecture. The data samples are images with a grey background and a colored circle in a certain position of the image. There are 9 positions possible for the circle : [Top left,Middle left,Bottom left, Top center, Middle center, Bottom center, Top right,Middle right, Bottom right] and 8 possible colors for the circle : [Black, White, Red, Green, Yellow, Cyan, Blue, Pink]. Then, we can have 9*8 = 72 different images which is small enough for us to visualize and interpret results but still complicated enough that it requires a sensible model. We added Gaussian noise to the position of the circle in the image to improve the generalisation capabilities of our network.

The caption associated is a simple sentence explaining the position of the circle in the image. First of all, we didn't specify the color of the circle in the caption but, then, tried to add it to observe the consequences. Fig.2 shows 2 examples of data samples.

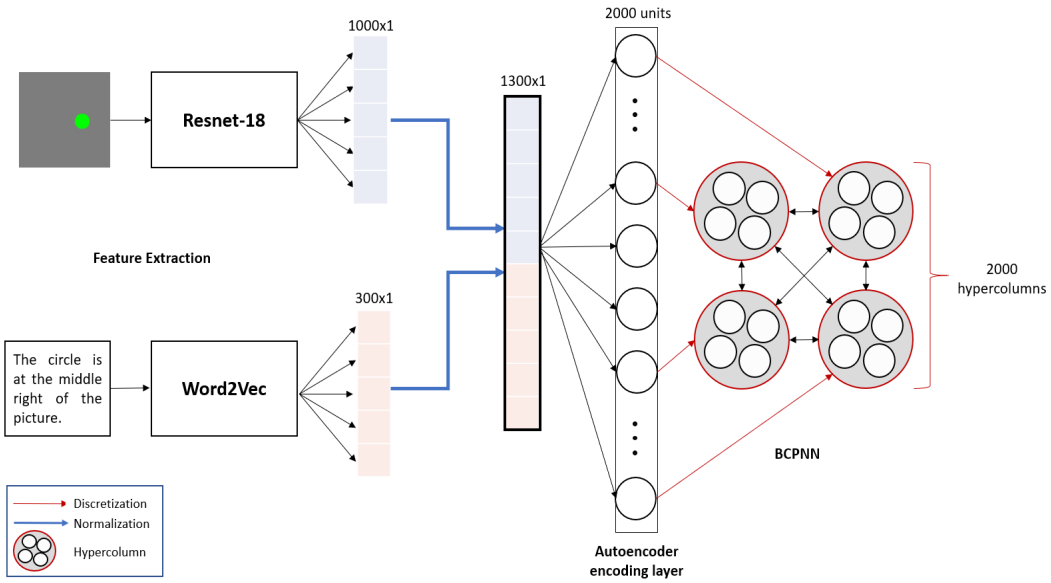
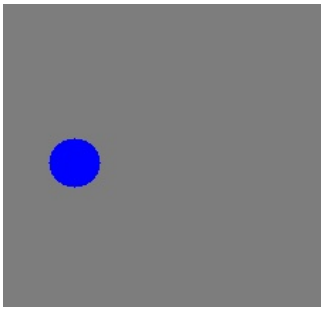


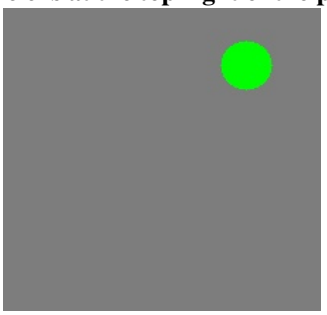
Figure 1: Schematic diagram of our current architecture. Features are first extracted for textual and image data using respectively Resnet-18 and Word2Vec. After being normalized and concatenated, an overcomplete autoencoder creates an intermediary representation that is then discretized and fed to the hypercolumns

The circle is at the middle left of the picture.



(a) Data sample with a blue circle at the middle left of the image (with Gaussian noise on the position) and the associated caption

The circle is at the top right of the picture.



(b) Data sample with a green circle at the top right of the image (with Gaussian noise on the position) and the associated caption

Figure 2: Two examples of training images with their associated captions

5 Experimental evaluation and results

5.1 Similarity measure after feature extraction

An essential consideration was first to verify that our problem was solvable, i.e. it was possible to separate vectors obtained by feature extraction (Resnet and Word2Vec) into distinct clusters using a similarity measure. To this end, we used the t-distributed stochastic neighbor embedding (t-SNE) algorithm[4]. It is a non-linear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space. In our case, we needed to go from the 1300 dimensions of the concatenated representations into a visualization in two dimensions. The huge interest of this algorithm is that it allows to preserve similarities from the input to the output space. In Fig.3, we can observe 9 principal clusters, corresponding to the 9 possible positions of the circle in the image. Moreover, inside each cluster, we have 8 sub-clusters representing the 8 colors

available in the dataset. As a result, we can conclude that we have here perfect initial conditions for the clustering task we want to achieve.

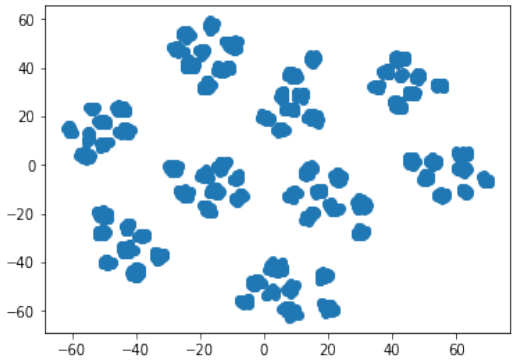
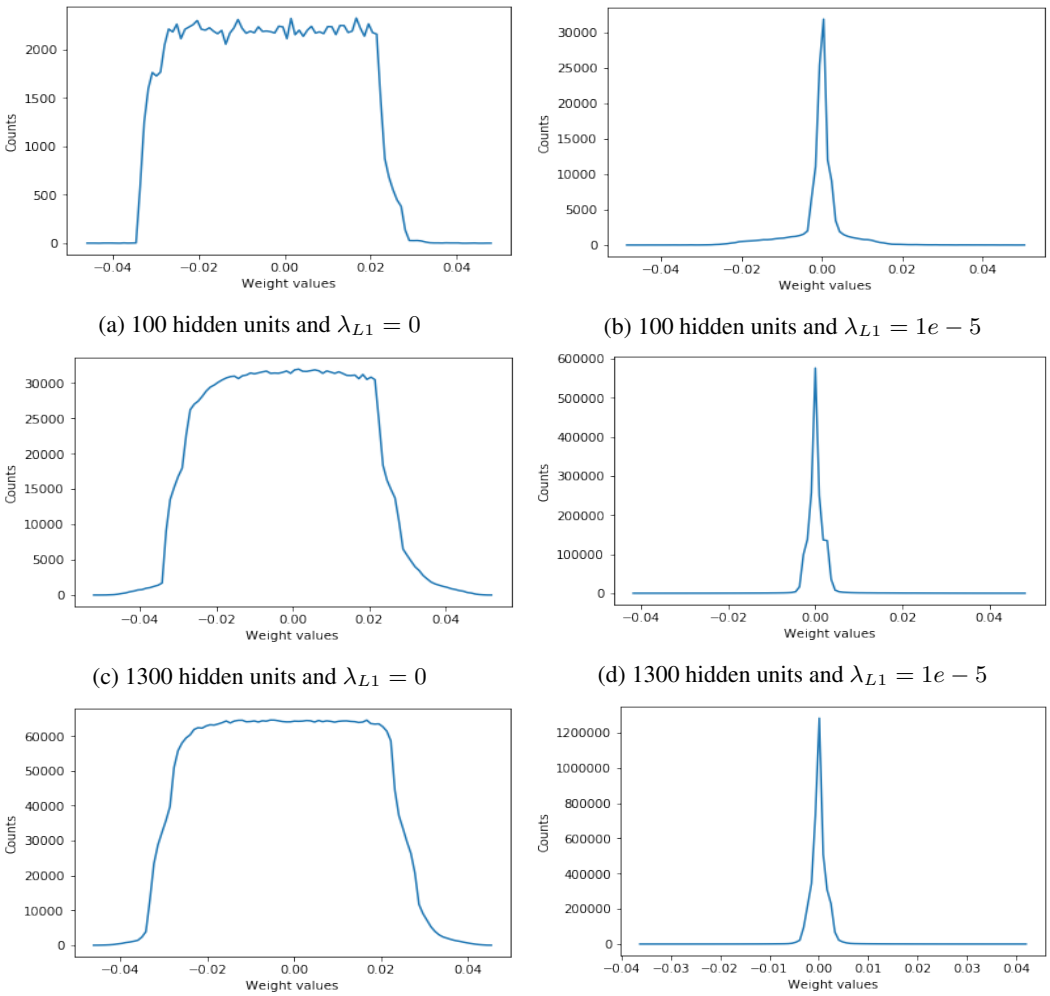


Figure 3: Visualization of concatenated image and text representations after feature extraction on training samples using t-SNE algorithm

5.2 Sparsity of the autoencoder hidden layer

We had to tune the parameters of the autoencoder to obtain representations as sparse as possible but without losing capabilities of reconstruction, i.e. quality of encoding. In particular, we could modify the number of hidden units to compare the efficiency of an undercomplete (less hidden units than inputs units) and overcomplete (diverging architecture) autoencoders. Moreover, we observed the effect of the L1 regularization on the sparsity. We compared the resulting sparsity by visualizing the distribution of the weights in the hidden layer. As we can see in Fig.4, first, using L1 regularization allows to have a density distribution much more peaked around 0, i.e. less uniform. This means that a lot of weights are equal to 0 and very few are different from 0. As a result, regularization seems essential to sparsify representations. On the other hand, we can not observe a real strong difference when modifying the number of hidden units when there is no regularization, except a more uniform distribution. This seems reasonable as, without regularization, the autoencoder only minimises the mean squared error, favoring trivial mappings. Thus, the network will not use new hidden units we could add. Though, with regularization, having more units seems to make the distribution a bit more peaked around 0 and, so, to increase the sparseness a bit. With a more complex dataset the difference in the number of hidden units would probably be more sensible. Indeed, in our case, the images we consider are so simple that the autoencoder needs very few units to encode the input.



(a) 100 hidden units and $\lambda_{L1} = 0$

(b) 100 hidden units and $\lambda_{L1} = 1e-5$

(c) 1300 hidden units and $\lambda_{L1} = 0$

(d) 1300 hidden units and $\lambda_{L1} = 1e-5$

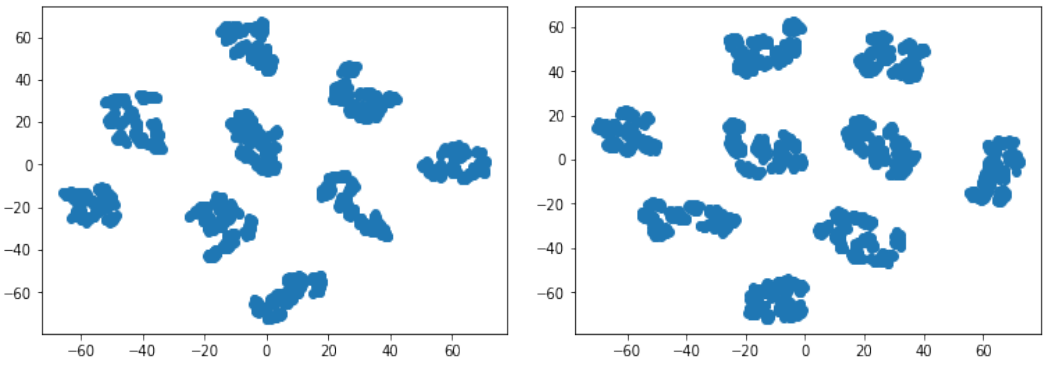
(e) 3000 hidden units and $\lambda_{L1} = 0$

(f) 3000 hidden units and $\lambda_{L1} = 1e-5$

Figure 4: Density distribution of the weigths in the hidden layer of the autoencoder depending on the number of hidden units and the value of the L1 coefficient

5.3 Similarity measure after the autoencoder

Then, we also had to verify if the sparse representations from the autoencoder could be separated into clusters. Once again, we reduced the dimensionality of the 2000-dimension vector to obtain points with 2 dimensions for the sake of visualization using the t-SNE algorithm. We did the same for the discretized representations that we will send to the BCPNN. Fig.5 summarizes these two visualizations. We notice that sparsifying the inputs preserves the similarities as we still have 9 clusters when visualizing the hidden representations. However, we have lost the sub-clusters we had before the autoencoder. This means that the latter considers the position of the circle in the image as a more relevant feature to encode the input than the color of the circle. This could be due to the fact that the information of the color is not present in the caption. Discretization doesn't affect the existing similarities, which is reasonable as similar vectors will be encoded in close ways.



(a) Encoded representation given by the autoencoder

(b) Discretized encoded representation

Figure 5: Visualization of the autoencoder hidden representation before and after discretization on training samples using t-SNE algorithm

5.4 BCPNN

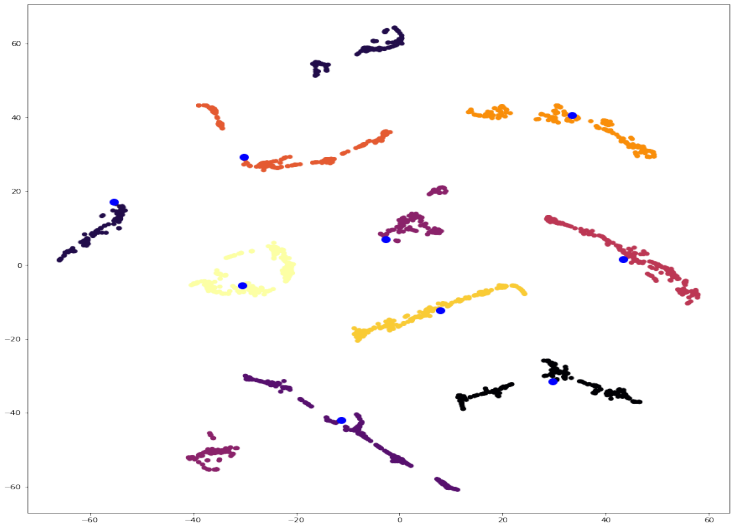


Figure 6: k-means clustering on BCPNN output - Cluster centroids are represented as blue points

On Fig.6, we can see that the model is able to accurately identify different patterns amongst the data. Each centroid is then used to perform prediction on the input vectors. To make sure of the validity and homogeneity of the clusters, we have predicted images from our test set and observed images from same clusters.

For instance, on two of the clusters, it seems that the model has learned to cluster data based on position and not color (see Fig.7). The first cluster is associated with objects in the bottom right corner and the second one in the top right hand corner. The fact that the color is not taken into account seems reasonable as the color information is only present in the image and not in the caption. As a result, we could even see it as noise when considering the whole representation of both modalities.

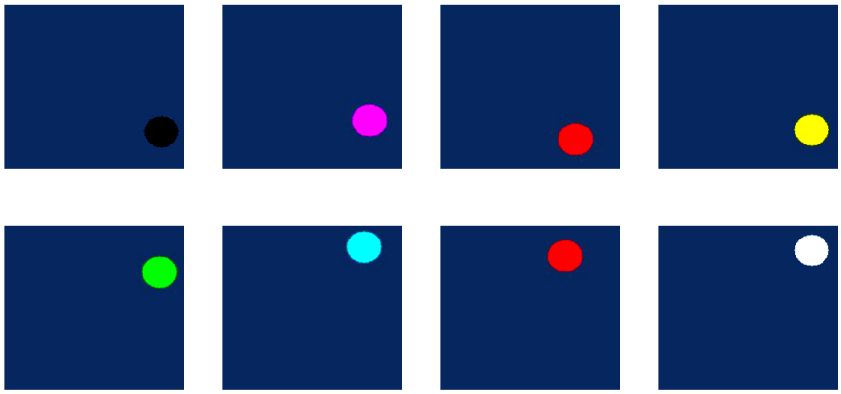
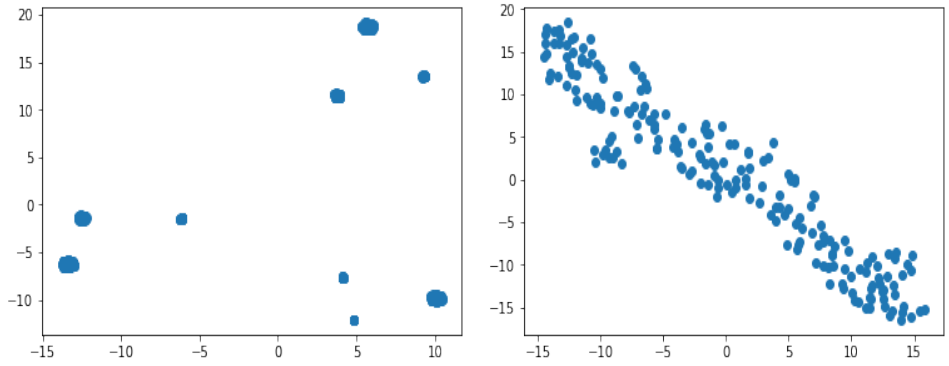


Figure 7: Test images associated to one cluster (first row) and to a second one (second row)

Then, we tried to evaluate the robustness of the model in regards to a possible lack of information, i.e. either missing image or caption in the input.



(a) Without image

(b) Without caption

Figure 8: Visualization of the BCPNN output with test samples missing either the image or the caption using t-SNE algorithm

Looking at the graph on the left of Fig.8, it seems that most of the valuable information that the autoencoder extracts comes from the caption data. When removing the information about the image, we can see distinct clusters corresponding to the possible positions of the object. This seems to indicate some robustness capabilities as the model is still able to perform clustering when removing a part of the information, even if this one is not the most relevant. Indeed, being able to extract important information and get rid of redundant data can be seen as robustness. Moreover, the right part of Fig.8 doesn't contradict this conclusion as if we remove too much information, it is reasonable to think that the model's clustering wouldn't be efficient.

6 Summary and discussion

In this project, we have shown evidence that using brain inspired architectures allows us to perform clustering on multi-modal data while being robust to missing modalities on a synthetic dataset. The issue remains however to test the model on a more complex one to make sure that the results are not linked to just overfitting on the training set. One such dataset could be the COCO dataset which contains a large amount of images and associated captions.

Using sparse representations has allowed us to have enough distinct basins of attractors in the BCPNN to perform clustering. Indeed, as in classical Hopfield networks, the number of maximum patterns storable by the network increases at the same time as the sparsity. This is due to the fact that sparse vectors are less likely to have the same features activated and thus become more orthogonal, making it easier for the network to recall patterns.

One possible modification that could be made to our model to make it more biologically plausible would be to avoid preprocessing the image input. It could be interesting to compare the results of the clustering when using the raw pixel input when compared to the features extracted from the Residual network. This would also have two other added benefits. First it would increase the performance as no extra features would be calculated for the representations. Secondly, by not using feature extractors our input could also become sparser, possibly increasing the generalization capability of the model.

One other possible enhancement to the model would be to use density based clustering. In our current approach, we specify the number of clusters in advance as we know how many we would like to see. On more complex datasets, this would probably not be as efficient and using clustering models such as DBSCAN would allow the model to automatically select similar clusters.

7 References

[1] *Encoding Data for HTM Systems*, Purdy

[2] *Attractor Memory with Self-organizing Input*, Johansson and Lansner

[3] *Learning representations by back-propagating errors*, Rumelhart, Hinton, Williams

[4] *Visualizing Data using t-SNE*, Hinton and Van der Maaten

[5] *Performance of a Computational Model of the Mammalian Olfactory System*, Benjaminsson, Herman and Lansner

[6] *From ANN to biomimetic information processing*, Lansner, Benjaminsson and Johansson

[7] *Multidimensional Scaling*, Steyvers

[8] *Odor Recognition in an Attractor Network Model of the Mammalian Olfactory Cortex*, Herman, Benjaminsson and Lansner

[9] *A Bayesian attractor network with incremental learning*, Sandberg, Lansner, Petersson and Ekeberg

[10] *An Attractor Memory of Neocortex*, Johansson

[11] *Multisensor data fusion: A review of the state-of-the-art*, Khaleghi, Khamis, Karray, Razavi

[12] *Multimodal deep learning*, Ngiam, Khosla, Kim, Nam, Lee, Ng

[13] *Multimodal learning with deep Boltzmann machines*, Srivastava, Salakhutdinov

[14] *Robust face recognition via multimodal deep face representation*, Ding, Tao