Cognition & Computation Project

Federico Zanotti[†]

University of Padua

February 17, 2022

1 Introduction

In this project I tested different configuration of Restriced Boltzmann Machine (RBM) and its deep version, Deep Belief Network, composed by stacked layer of RBMs. These models are called generative model and they try to capture the manifold of the data by learning the probability distribution of the input in an unsupervised way.
The generative models enable to automatically uncover underlying features of the data that humans don't are aware of them. We don't know how we recognize a digit or a letter, we just know from our experience without even thinking about it. The RBMs instead have no experience and so they learn that images of digits and letters are composed by edges, that can be combined in different way in order to create more different examples.
In this report I will present this way of learning with different models on a popular dataset for Machine Learning: EMNIST.

2 Dataset and Methodology

EMNIST dataset is composed by $814,255$ examples of handwritten digits between 0 and 9 and the alphabet's letters in upper case and lower case that together form 62 classes. For semplicity of computation, I used the balanced version with $112,800$ examples and 47 label (digits, letters from A to Z and $a, b, d, e, f, g, h, n, q, r, t$).

In the project I used PyTorch, a Python's library, and I divided the dataset in training set with $78,960$ examples and the test set with $33,840$. Moreover I rotated the images by 90 degree for a good visualization and I normalized them. For the simulations I followed the code used in the labs, modifying it for my purposes.

3 Models

In order to see how the RBM learns I tested a simple one and a DBN with different parameter configurations. For every hidden layer of these models

- Weights learned are plotted, showing which features the RBMs are learning
- Hierarchical Clustering is calculated
- Accuracy is computed by adding a Linear Classifier

The standard parameters for the DBN are:

[†]federico.zanotti@studenti.unipd.it (ID: 2007716)

Parameters	Values
Hidden units	[500]
Learning Rate	0.1
Learning rate decay	False
Initial Momentum	0.5
Final Momentum	0.9
Weight Decay	0.00002
Xavier init	False
Increase to cd K	False
K	1

Moreover I did a comparison with a simple Neural Network composed by Convolutional layers.

3.1 Simple RBM

This first model is a RBM with 500 neurons and the parameter configuration is standard. The training is done for 50 epochs with batch size of 100.
In the figure below a subset of weights learned by the hidden layer is plotted. Each square is one output of the first hidden layer and this last one is connected to the visible units, so to the pixels of the image. From the plot can be seen that the RBM is learning the edges of the images, and also the dot-like weights mean that the neuron is sensitive to specific points in image.
In the Figure 1 we can see the dendogram and the hierarchical relationship between the labels.

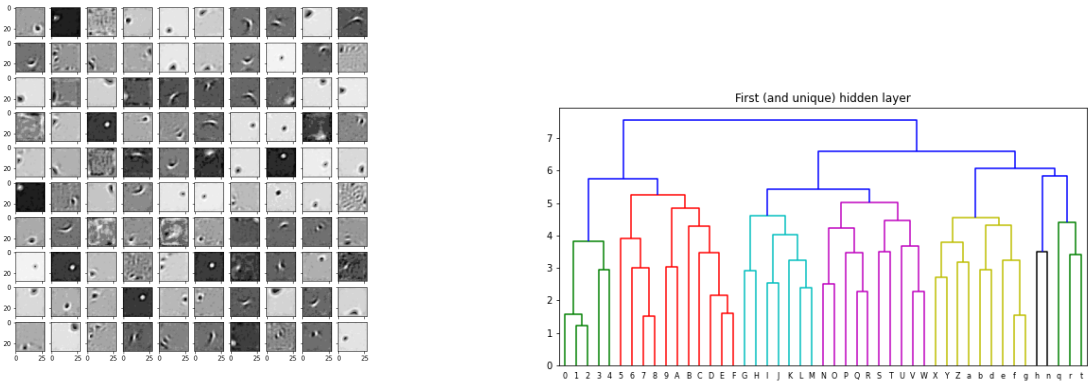


Figure 1 *(a) Weights Plot. (b) Dendogram.*

In the Dendogram we can see a difference between the clusters composed by all the digits and upper case letters from A to F and the others labels, so in some way the RBM hidden representation is defining the similar labels.

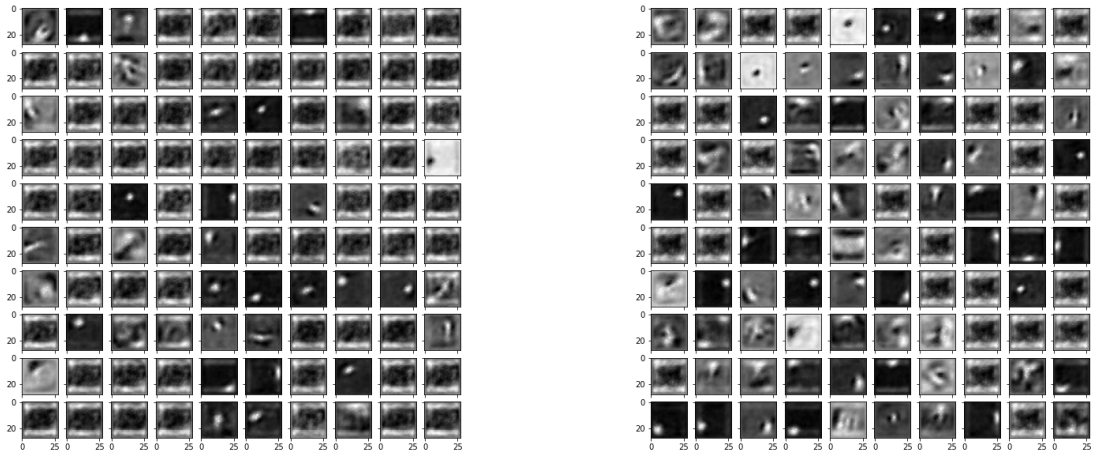
Then I trained the representations obtained by the RBM with a simple linear classifier with SGD optimizer for 1000 epochs. I obtained $64,43\%$ of accuracy

3.2 DBM

In this section I will present different plots of weights corresponding to different configuration of the parameters. In order to see how the hidden representation changes with stacked RBM, for every parameter configuration I always keep fixed the number of neurons and the number of hidden layers. In particular I used 4 stacked RBM and 500 neurons for the first three RBM, while for the last one I used 2000 neurons.

3.3 Standard Parameters

In the Figure 2(a) there is the weights plot of the last layer of the DBM with standard parameters of previous table. Despite being the last level the plot is still not very clear, but there is a more specialization, because we can see more specialized curvilinear dot and curvilinear edges.



(a) *Weights of the 4th DBN layer*

(b) *Weights of the 4th DBN layer with Xavier*

Figure 2.*Weights plots*

3.4 Different Parameters Configurations

In all the different configurations the weights plots are not very clear, because the learning did not converged, but they are still meaningful plots.
The first parameter I changed is the learning rate, but the weights plot are not significant, because the visualization is equal to a random connection.
The second parameter I tested is the weight decay, and I changed it from standard Initialization of 0.0002 to 0.2.
In Figure 3 there are the weights plot of the first RBM (on the left) and the second RBM (on the right).

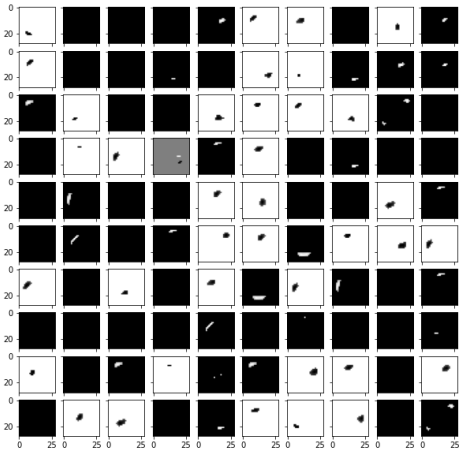
From the left plot we can cleary see the strong of the connection between the nuerons in the first hidden layer with the input elements, that is pixel in the image. Infact there is a clear distinction of white and black squares with black and white dots or line. In this case the learning is very specified because we have total black or total white squares indicating a negative or positive connection.

In the second plot on the right the hidden representation becomes more defined with circles and shapes with respect to the previous plots described.
The plots related to the remaining hidden layers are not reported because they are very similar to fig tot b.

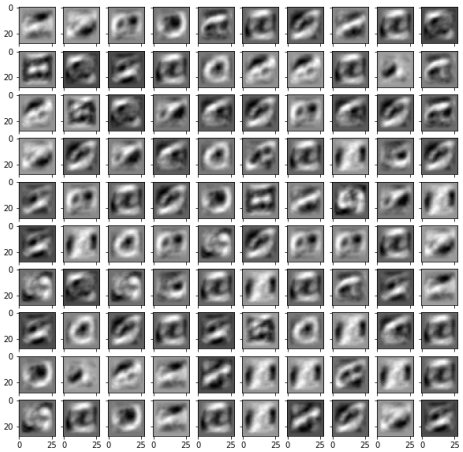
The last significant change in the paramater is related to the initialization of the weights. Instead of using a random initialization, Xavier Initizialization is adopted. In Figure 2(b) the weights plotted are related to the last hidden layer and they are similar to the standard parameter plot, but there are new shapes encoded, showing that learning is improved a little bit.

3.5 CNN

In contrast to the RBMs in this section it's presented the feature maps related to the convolutional layers of a simple Neural Network. This is trained for 5 epochs with a batch size of 1024. Here below there is the structure of the model.
In order to get the feature maps of an image, the model is tested on a simple digit image (an image of 9)



(a) *First Layer*

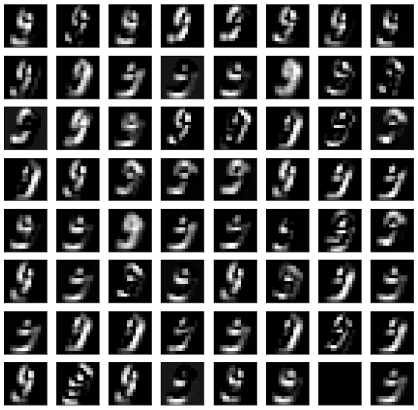


(b) *Second Layer*

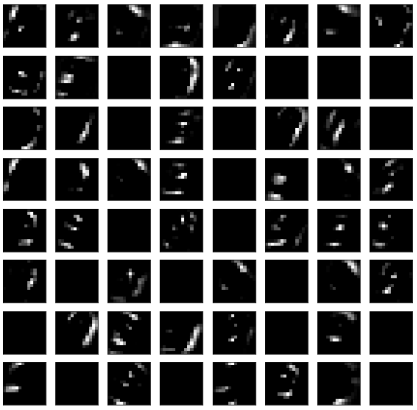
Figure 3.*Weights plot with weight decay*

Layer Type	Size
Convolution + ReLU	5x5x512
Max Pooling	2x2
Convolution + ReLU	5x5x256
Max Pooling	2x2
Convolution + ReLU	5x5x128
Max Pooling	2x2
Convolution + ReLU	3x3x64
Max Pooling	2x2
Flatten	-
Fully Connected + ReLU + Dropout	128
Softmax	47

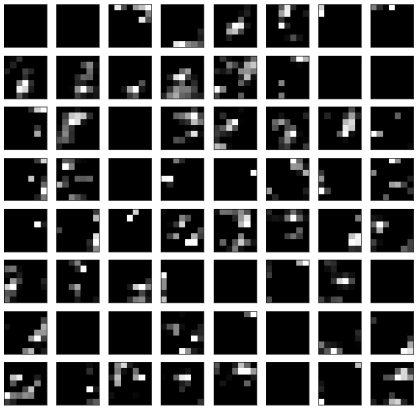
and in the next plots (Figure 4) there are the feature maps (the output of all the convolutional layers). It's clearly more visible how the model learns, infact every neurons learn a specific part of the same image. In particular the weights plotted in Figure 4(b) are very similar to the hidden representations of the RBM.



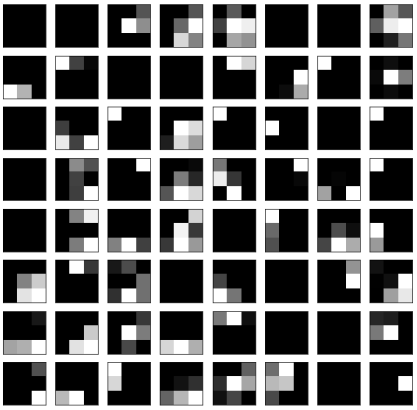
(a) *First Layer*



(b) *Second Layer*



(c) *Third Layer*



(d) *Fourth Layer*

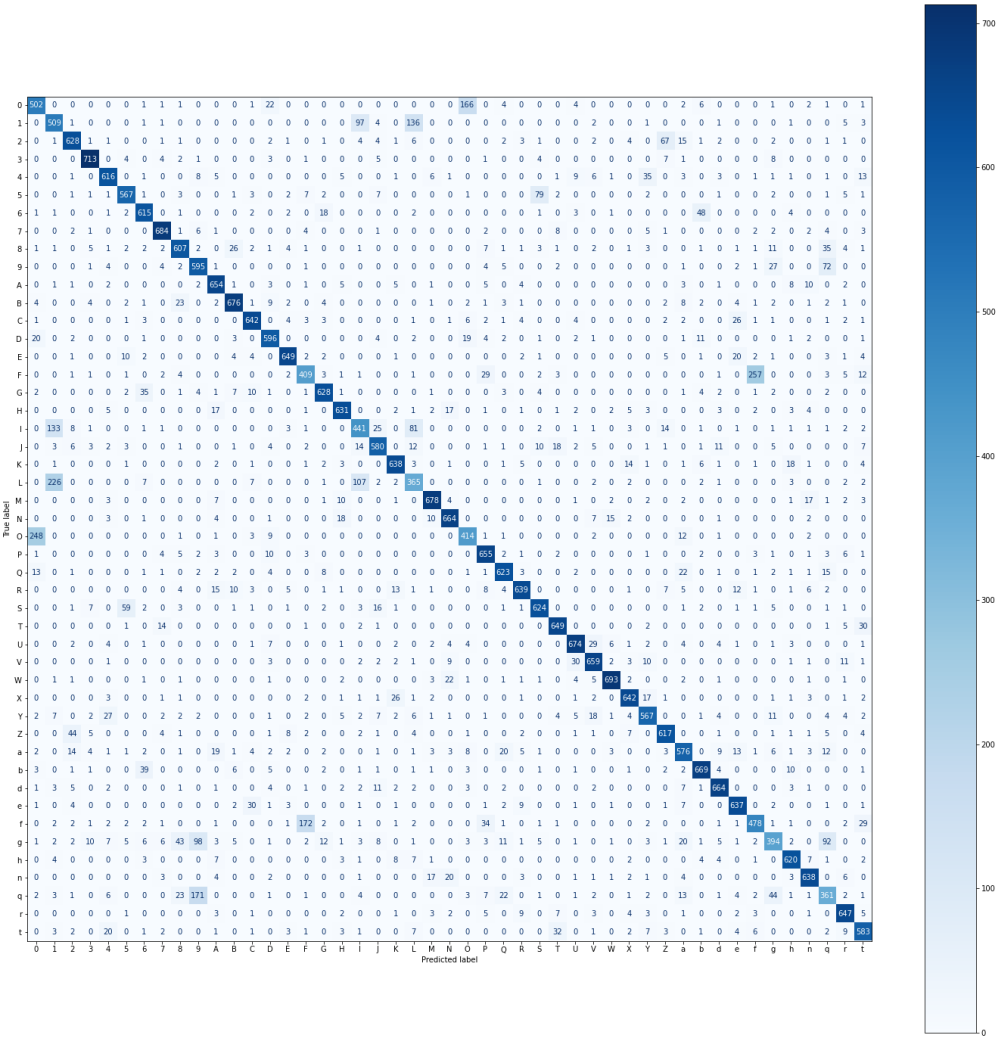
Figure 4. *Feature maps CNN*

4 Results and Discussion

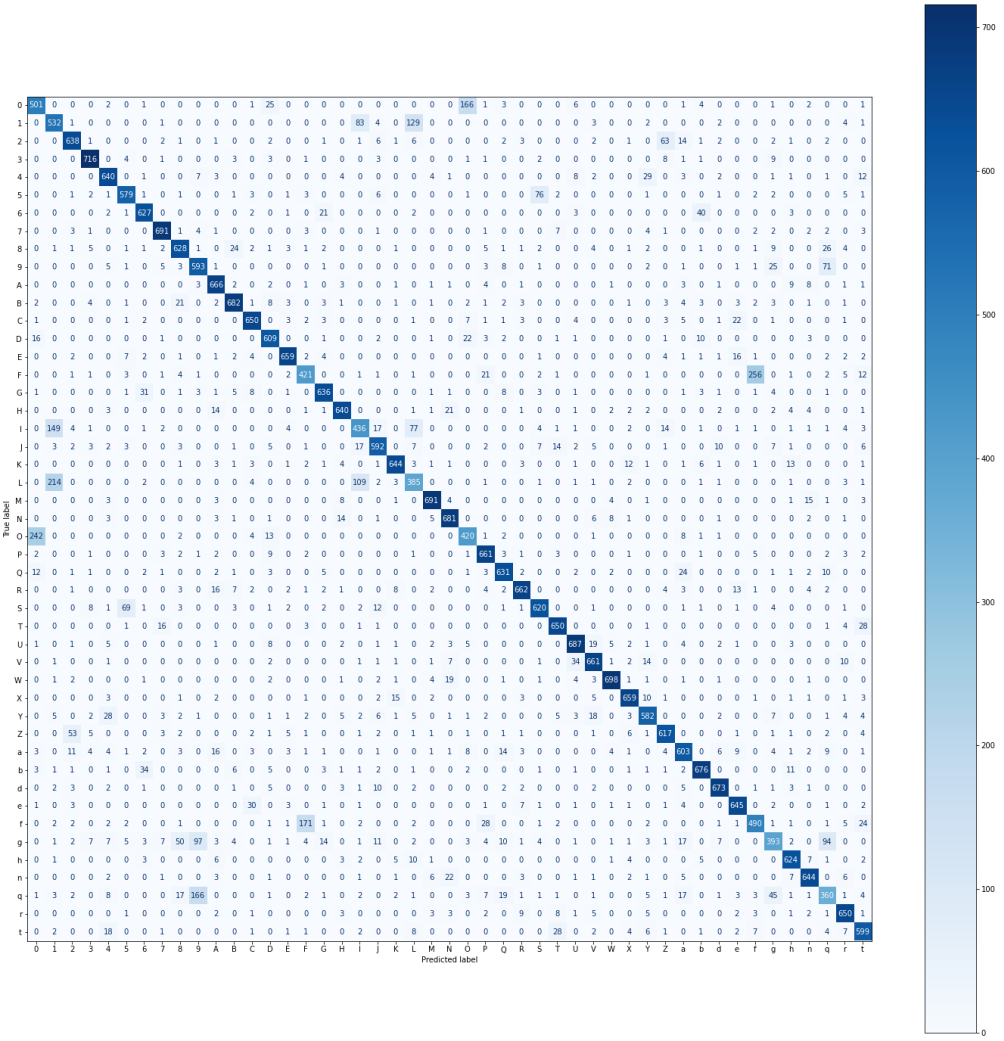
In the next Table there are final accuracy scores reported and in the Appendix there is a visualization of the confusion matrix of the Standard and with Xavier Initialization DBM. The diagonal represents the good predictions, while the other values are images missclassified. It's interesting notice that there are many errors when the label is O, 0, L, F, f and q. Infact it's reasonable from the RBMs point of view have similar hidden representations from O with 0, and this is true even for humans. At the same time L is interpreted similar to 1, F is confused with his counter part f and viceversa and q is confused with 9, infact they are very similar because they are composed by a circle followed by a line.

Parameters Changed/Model used	Accuracy (%)
Standard	82,76
Learning Rate	55,75
Weight Decay	22,83
Xavier Init	84,20
CNN	86,76

A Appendix



Confusion matrix of the standard DBN



Confusion matrix of the DBN with Xavier Initialization