

## Semi-Supervised Learning with Ladder Networks


### CSE 5290 Artificial Intelligence

#### Group 2

#### 1. Introduction

In this modern era of autonomous cars and deep learning, pure supervised learning is widely popular and trending. Supervised learning is when an agent is fed with several examples (training data) to learn features which are already labeled and after the learning process, test data is given to the agent to examine how accurate the agent is. The **key point** here is that our data needs to be **labeled** for the agent to make sense out of it and storing useful knowledge from it which can further be used to solve the challenges presented to an agent. In general, supervised learning is used mostly for the traditional classification problem. However, there are some learning approaches that does not or partially require labelling like the unsupervised learning.

Unsupervised learning, on the other hand is when an agent is not provided with labeled data set unlike the supervised learning and the training data is completely unlabeled. The agent evolves by finding patterns and assigning weights to more probable outcomes and storing the desired knowledge for the test data challenge. Unsupervised learning is mostly used for clustering problems where data is bunched into similar groups. Unsupervised learning is widely used for machine learning, pattern recognition, image analysis, computer graphics and much more. Unsupervised learning is not much efficient when used for classification as it needs to label the groups after clustering.

These above-mentioned approaches either work well given a lot of labelled data or given nothing but solving clustering problems. However, for supervised learning, labelling data is expensive to collect and this is one of the biggest drawbacks of supervised learning. Deep learning was inspired by the human brain[1] where several million neurons combine to form a single thought in our mind. As it is said “*Practice makes you perfect*” is true as repetition of one task several times would make that certain million neurons fire at the same time, starting from a cluttered, unorganized manner but performing it over and over would fire those specific neurons so many times that the electric charge between those specific neurons increase to an extent where we can perform the same task with so much ease that it can be completed within seconds. Deep learning tries to mimic this aspect of our brain and then neural networks are born. The only problem with supervised learning is that humans have not been evolved to use supervised learning. In our olden times, we used to discover new things and discovery is not associated with supervised learning. Discovery falls somewhere between supervised and semi-supervised learning which is semi-supervised learning That’s why, Hint  says it,

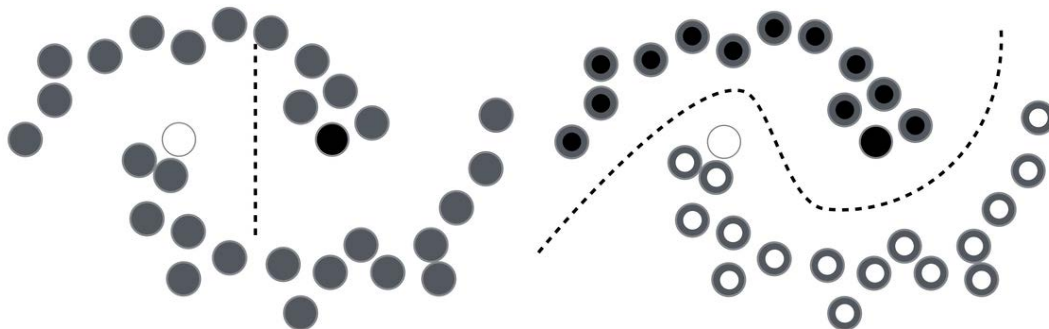
*“We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.”[2]*

In semi-supervised learning, the agent is provided with a very few labeled data in addition to a large amount of unlabeled data as training dataset and then it tries to classify from the test data. This approach excels supervised learning from the point of labelling where we only must label a few

hundreds of data rather than labelling millions of them. Semi-supervised learning gets the best of both worlds: Supervised and Unsupervised learning; solving the classification problem. Semi-supervised can be combined with several approaches and in this project, we combine semi-supervised learning with ladder networks. Valpola (2015) proposed ladder network where it combines the supervised and unsupervised learning and both are used to train simultaneously rather than training in a traditional way where unsupervised is used for pre-training followed by pure supervised learning [3]. This minimizes the sum of unsupervised and supervised cost functions using back-propagation which avoids the need for pre-training. In this project, we follow the footsteps of the authors of the paper “Semi-supervised learning with ladder network”[3] and we prove how efficient semi-supervised learning is compared to supervised learning. We also study the implementation of this paper in python.

## 2. Overview

As stated above, semi-supervised learning uses a very small amount of labeled data and a very huge amount of unlabeled data for training. But how does semi-supervised learning use unlabeled data for classification? Let’s look at an example to clarify that.



**Figure 2.1 Example 1 – Classification from semi-supervised learning (Before and After)**

Here, let’s assume that we have only two labeled data; A black and a white dot that is clearly visible in Figure 2.1 (Before). Now as we can see that there are a lot of other grey dots which are nothing but unlabeled data which we will use for training. If we consider smoothness assumption, we conclude that labels are homogeneous in densely populated space, that is data points that are near to each other can be considered to be in the same group. Now if we take this assumption and iterate this over and over, we finally train our classifier to be almost perfect as shown in Figure 2.1 (After). If we use this assumption, then we can get higher accuracy even though we use plentiful of unlabeled data. In real world, there are a lot of unlabeled data and very scarce amount of labeled data as labelling is expensive.

Ladder network use the above approach and combines supervised and unsupervised network to get even more accuracy and a minute loss. Ladder network combines the supervised and unsupervised learning and both of them are used for training simultaneously rather than training in a traditional way where unsupervised is used for pre-training followed by pure supervised learning [3].

We used MNIST dataset for our testing and achieved remarkable accuracy after the full training. While ladder networks are astonishing, there are three key factors for ladder networks:

### 1. Compatibility

Ladder networks are compatible with any supervised learning methods and can be added to any existing feedforward neural networks. It can also be extended to any recurrent neural networks.

### 2. Scalability resulting from local learning

Ladder networks can be scaled for some very deep neural networks as this model has unsupervised learning targets on each layer besides the traditional supervised learning targets.

### 3. Computational efficiency

Including the decoder triples the computation, however it does not affect the training time.

## 3. Algorithm

For the basics, let's break down the algorithm into four basic steps:

- a. As we are working with a traditional recurrent neural network, we use the encode-decoder functions for input-output sequence mapping. First let's take a feedforward neural network which aids supervised learning as an encoder. We use two encoder paths – One clean and One Corrupted.
- b. Next, we add a decoder that inverts the mapping on each layer of the encoder as the decoder uses a denoising function (Denoising Source Separation) for removing noise. As it is also used for reconstruction, the difference between the reconstructed output and the clean source is the **unsupervised cost**.
- c. Then, we calculate the **supervised cost** from the difference of target output and corrupted encoder output. The **final cost** is the summation of the supervised and unsupervised costs.

$$final\ cost = supervised\ cost + unsupervised\ cost$$

- d. The last step is to train the whole network using standard optimization techniques like stochastic gradient descent to minimize the cost.

### Let's dive in more into the technical implementation of the algorithm.

- First let's represent the labeled and unlabeled data. Note that labeled data and unlabeled data are in two groups (x and y) and the labeled data is scarce and unlabeled data is plentiful.
  - Labelled data:  $\{x_t, y_t\} \ 1 \leq t \leq N$
  - Unlabelled data:  $\{x_t\} \ N + 1 \leq t \leq M$
  - Often labelled data is scarce, unlabelled data is plentiful:  $N \ll M$

- Then we corrupt the input  $x_t$  by adding some Gaussian noise and we get  $\tilde{x}_t$ 
  - Encoder function  $f: x \rightarrow h$
  - Decoder function  $g: h \rightarrow \hat{x}$
  - Reconstruction  $\hat{x}$

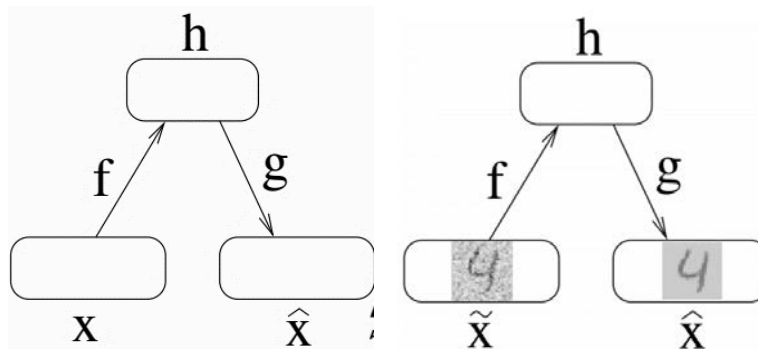


Figure 3.1: EncoderDecoder functions

- Then we use the Denoising AutoEncoder (DAE) for reconstruction and denoising function. As shown in figure 3.1 (b), we can see the when the corrupted input is fed, it generates the output which is robust to the noise and reconstructs as precise as the input.

Encoder:  $b = f(\tilde{x}) = \phi(W^{(1)}\tilde{x} + b^{(1)})$

Decoder:  $\hat{x} = g(h) = W^{(2)}h + b^{(2)}$

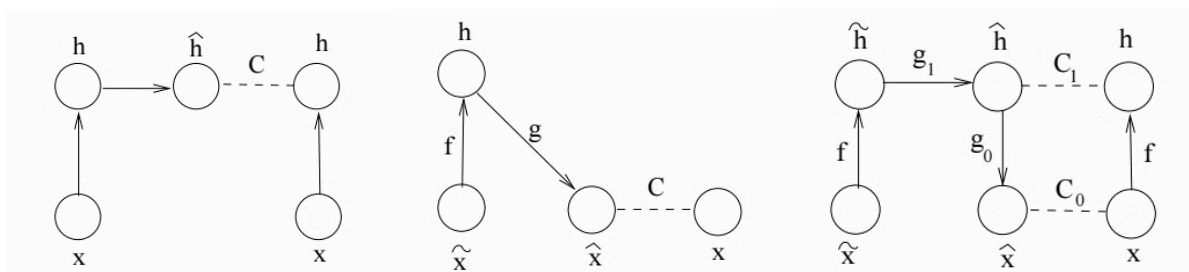


Figure 3.2: Combining DSS and DAE

- In figure 3.2, we combine the Denoising Source Separation (DSS) and Denoising AutoEncoder (DAE) which forms a ladder network of degree 1.  $C_x$  is the checking function and it is not going to be used for training. It will be used in the testing phase.

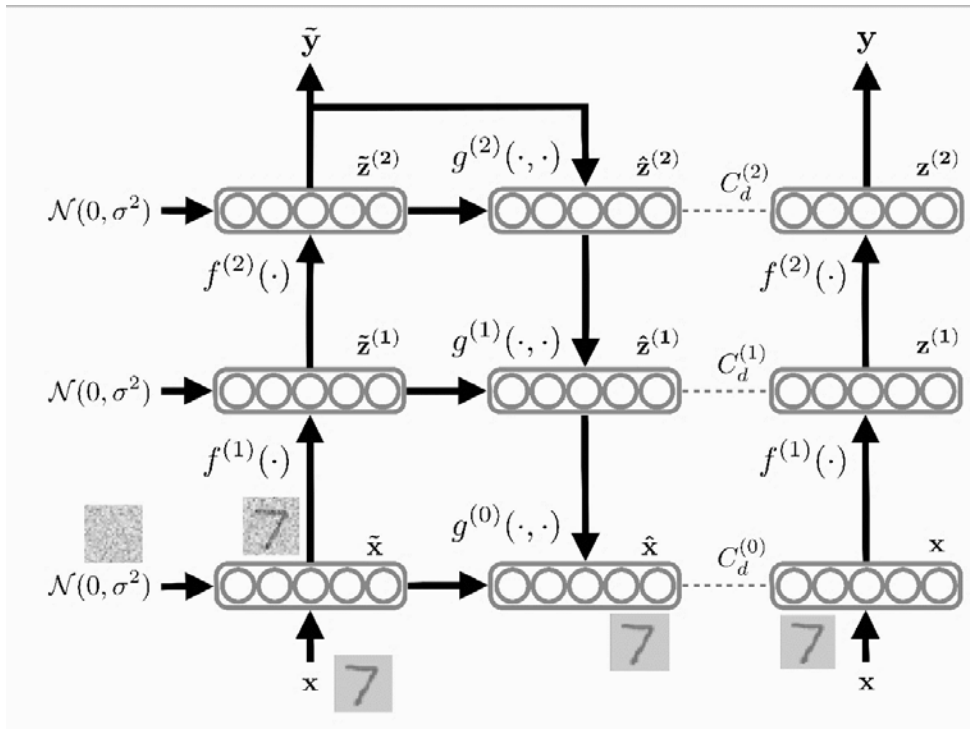


Figure 3.3: Ladder network architecture

- In figure 3.3, we see the whole degree 3 Ladder network. Here the image of numeric digit “7” is passed as input and then Gaussian noise ( $N$ ) is added to the input. Then we have the encoder function for climbing up ladder and then the decoder function comes while stepping down the ladder. Everything is parallelly checked with the function  $C$  for testing and increasing accuracy.

#### 4. Results

We have tested the code with the MNIST database and we got 98.9% maximum accuracy. However, we also tested on various machines and found some unique results and patterns. Below, we have attached some code snippets.

```
1 import tensorflow as tf
2 import input_data
3 import math
4 import os
5 import csv
6 from tqdm import tqdm
7
8 layer_sizes = [784, 1000, 500, 250, 250, 250, 10]
9
10 L = len(layer_sizes) - 1 # number of layers
11
12 num_examples = 60000
13 num_epochs = 150
14 num_labeled = 100
15
16 starter_learning_rate = 0.02
17
18 decay_after = 15 # epoch after which to begin learning rate decay
19
20 batch_size = 100
21 num_iter = (num_examples/batch_size) * num_epochs # number of loop iterations
22
23 inputs = tf.placeholder(tf.float32, shape=(None, layer_sizes[0]))
24 outputs = tf.placeholder(tf.float32)
25
```

Figure 4.1: Code snippet: Initialization

- In figure 4.1, we see that we are using Tensorflow api from Google. For passing the labeled data for encoder, we are only using 100 images out of 60,000 images (MNIST dataset).
- We are also applying batch normalization for normalizing our data and also defining several layer sizes.
- For the output, we started with approximately 11% accuracy and got 98.55% in CPU as seen in Figure 4.2.

```
=== Training ===
Initial Accuracy: 11.66 %
100%|██████████| 89999/90000 [7:09:29<00:00, 3.35it/s]Final Accuracy: 100%|██████████| 90000/90000 [7:09:32<00:00, 1.03s/it]
98.55 %
Process finished with exit code 0
```

Figure 4.2: Output results

- In figure 4.3, we can see the different comparisons with different machines and the one with the most powerful GPU performed in least amount of time with the maximum accuracy. The CPU took almost 7 hours to train while the base GPU took around 2 hours and the best GPU (NVIDIA 770) took us only 45 minutes. The time was reduced 10 times compared to the CPU.

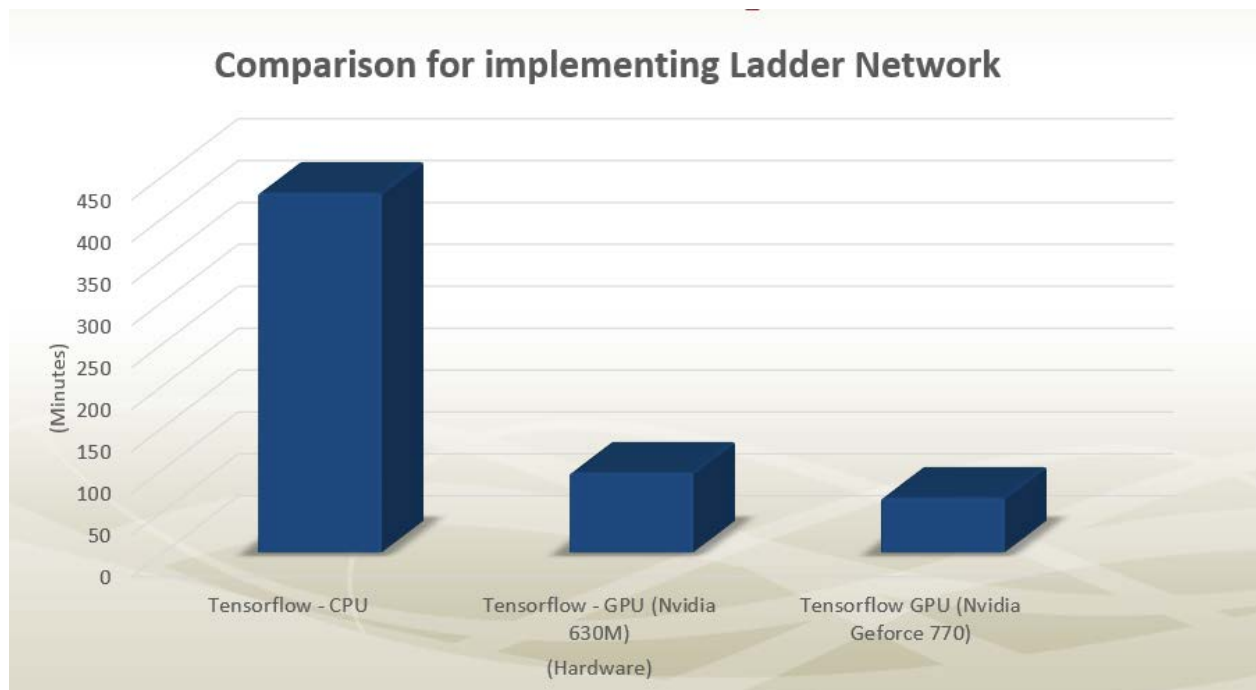



Figure 4.3: Results: Time Comparison

## 5. Conclusion

To put everything in a nut shell, supervised learning is a classification problem, in which the training set is given and the agent hypothesizes a function that maps from input to output based on observation of some input-out pairs. In unsupervised learning which is a clustering problem, given a completely unknown set, the agent tries to learn. If we divide the inputs into groups based on parameters the learning process is going to be a classification.

In the implementation of Ladder Network, we found that the performance of this method is very impressive. This method is simple and easy to implement with many architectures like feed-forward networks and recurrent neural networks. The training is based on back propagation from a simple cost function. This method is compatible with supervised methods, moreover, it has local unsupervised learning target on every layer which provides adaptability for very deep neural networks. Using the

decoder also triples the computation during training so the result can be achieved faster through the better utilization 

## 6. References

- [1] Plunkett, K., and J. L. Elman. "Exercises in rethinking innateness: A handbook for connectionist experiments." (1997).
- [2] LeCun, Bengio, Hinton, Nature 2015
- [3] Rasmus, Antti, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. "Semi-supervised learning with ladder networks." In *Advances in Neural Information Processing Systems*, pp. 3546-3554. 2015.