# TensorFlow Ann(e)
# Lab rapport IT3105 - Module 3

Sigve Skaugvoll & Thomas Wold

September 2017

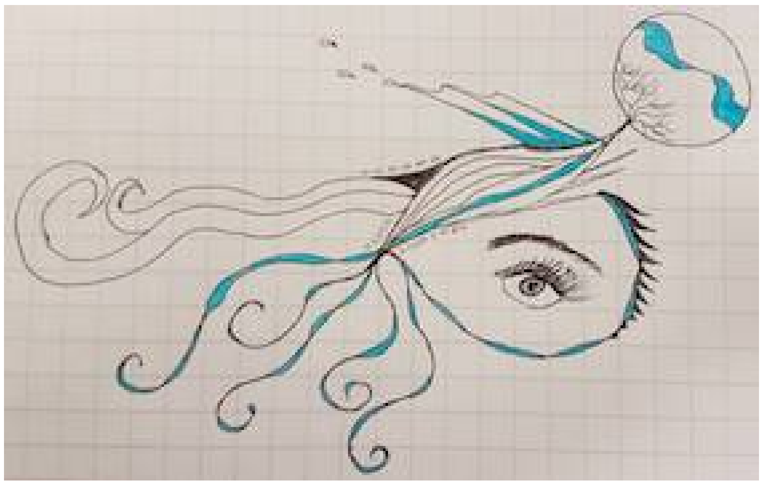


Figure 1: Drawn by Margit G. Schefte

## 1 Introduction

The assignments was to get a deep understanding of how TensorFlow works, by implementing a interface from scratch and tweak/further develop handed out TensorFlow code.

Use Tensorflow to classify data from a wide variety of sources, including the classic MNIST benchmark of digit images

A primary goal of this assignment is to streamline this trial-and-error process so that many different networks can be experimented with in a short period of time. To this end, you will build an interface to Tensorflow that facilitates these experiments

## 2 TensorFlow

"TensorFlow$^{TM}$ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well." - TensorFlow.com

# 3 Ann(e)

We choose to call our interface 'Anne', because it sounds cool.

## 3.1 Installation

### 3.1.1 Prerequisites

1. Python 3.5 or greater.
2. TensorFlow
3. mathplotlib

### 3.1.2 Installing TensorFlow

Optional: First activate your virtual environment $ pip install tensorflow

### 3.1.3 Installing Mathplotlib

Optional: First activate your virtual environment, which has TensorFlow installed

1. $ pip install mathplotlib
2. $ cd
3. $ cd .matplotlib/
4. $ vim matplotlibrc
5. Write: backend: TkAgg
6. Save and exit vim: hit 'esc', then type ':wq'

What this means is : There's a directory located at " (tilde)/" root called matplotlib. so cd into (tilde)/.matplotlib. Create a file called matplotlibrc in (tilde)/.matplotlib/matplotlibrc there and add the following code: backend: TkAgg

### 3.1.4 How to make TensorFlow Anne run

I'm using virtualenv and virtualenvwrapper for python on unix. So I installed tensorflow in my virtualenvironemnt. Thus I need to have the environment activated for tensorflow to be "installed" and working.

If i'm trying to run tensorflow code in a editor such as PyCharm and not from the terminal / command line. I need to let my IDEA know that I'm using a virtual environment. To let Pycharm know, do the following:

1. Hit the pyCharm on the statusline and then preferences or (cmd + ,)
2. Then on the left, in the dropdown menu, chose your project : in my case it's IT3105 - AI PROG
3. Then choose project interpreter.
4. In the dropdown at the top, scroll to see if your virtualenv is listed there, if so, choose it.
5. If its not listed, try hitting : show all, and see if it's there
6. If its not listed there either, we have to find it. click on the box with three dots. [...]
7. Choose local, and then navigate to where your virtual environemnt is "located" i.e : /users/¡username¿/.virtualenvs/¡virt
   env name¿

Now when we have selected our virtual env as our project interpreter, we have to wait for the IDE to update, index and do IDE stuff... Usually takes about a couple of minutes.

## 3.2 System requirements

1. System must be object-oriented and as modular as the classes Gann and Gannmodule in Tutor(ial)3.py
2. No god damn recompilation of code when reconfiguring the Ann.

## 3.3 Dataset requirements

1. One case per row. FORMAT: F0,...,Fn,T
2. Supported separators [',',';']
3. Features has to be Integers / Real numbers
4. Labels has to be Integers / Real numbers

## 3.4 Configuration Requirements

1. Abbreviation : Full name : Type
2. F : Number of Features : Int
3. L : Number of Classes / labels : Int
4. Nb : Nbits : Integer
5. Size : Number of Features : Int
6. Den : Density : tuple(Integer,Integer)
7. Dbl : Double : Boolean
8. Ran : Random : Boolean
9. Pt : Poptarg : Boolean
10. minS : minSegs : Integer
11. maxS : maxSegs : Integer

| Configuration Requirements | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | F | L | Nb | Size | Den | Dbl | Ran | Pt | MinS | MaxS |
| Autoencoder | $2^{Nb}$ | $2^{Nb}$ | int | - | - | - | - | - | - | - |
| Yeast | 8 | 10 | - | - | - | - | - | - | - | - |
| Glass | 9 | 7 | - | - | - | - | - | - | - | - |
| Wine | 11 | 6 | - | - | - | - | - | - | - | - |
| Parity | Nb | 2 | int | - | - | T | - | - | - | - |
| Dense | Size | Size | int | tup() | - | - | - | - | - | - |
| Bit | size | $size + 1$ | int | int | tup() | T/F | T/F | - | - | - |
| Segment | Nb | $MinS + MaxS$ | int | int | - | - | - | T/F | int | int |
| MNIST | 784 | 10 | - | - | - | - | - | - | - | - |
| Breast_cancer | 9 | 6 | - | - | - | - | - | - | - | - |

## 3.5 Performance Tests

### 3.5.1 Good configuration specifications

1. Abbreviation : Full name
2. P : Parameters
3. DS : Dataset
4. AutoenC : Autoencoder
5. B Cancer : Breast Cancer

| Good Configuration Parameters | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| P \ DS | AutoenC | Yeast | Glass | Wine | Parity | Bit | Segment | Mnist | B Cancer |
| Req: | – | 90% | 95% | 95% | 95% | 97.5% | 95% | 95% | – |
| Error | 0.05 | 0.055 | 0.0441 | 0.048 | 0.0225 | 0.017 | 0.0469 | 0.02749 | 0.01 |
| Dims | 8,30,8 | 8,90,10 | 9,133,7 | 11,11,6 | 10,120,2 | 15,120,16 | 25,142,9 | 784,28,28,10 | 9,10,6 |
| Epohs | 150 | 200 | 2400 | 30 | 2000 | 250 | 200 | 300 | 50 |
| LRate | 0.8 | 0.5 | 0.17 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.5 |
| Minibatch | 100 | 40 | 30 | 80 | 100 | 40 | 40 | 50 | 40 |
| Bounds | -1,1 | -1,1 | -0.3,0.3 | -0.5,0.5 | -0.5,0.5 | -0.5,0.5 | -0.5,0.5 | -1,1 | -1,1 |
| Cfrac | 1 | 1 | 1 | 1 | 1 | - | 1 | 0.1 | 1 |
| Vfrac | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Tfrac | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Vint | 20 | 40 | 80 | 10 | 40 | 40 | 40 | 40 | 50 |
| Show int | 10 | 10 | 10 | 30 | 50 | 20 | 10 | 10 | 10 |
| HAF | tanh | tanh | sigmoid | sigmoid | sigmoid | tanh | sigmoid | tanh | sigmoid |
| OAF | tanh | tanh | sigmoid | sigmoid | sigmoid | tanh | tahn | tanh | sigmoid |
| Cost Func | MSE | MSE | MSE | MSE | MSE | MSE | MSE | MSE | MSE |
| Softmax | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |

1. Case manager:

(a) Autoencoder
    i. Nbits = 3
(b) Parity
    i. Nbits = 10
    ii. Double = T
(c) Bit
    i. Nbits = 500
    ii. size = 15
    iii. Density = 0.7, 0.4
    iv. Poptarg = T
    v. Double = T
    vi. Random = T
(d) Segment
    i. Nbits = 25
    ii. size = 1000
    iii. Poptarg = T
    iv. MIN Segment = 0
    v. MAX Segemnt = 8

## 4 Detailed Explanation of Auto encoder

### 4.1 Anne configurations

For this task we are using Anne with eight inputs, one hidden layer, with 30 neurons and since input should equal output in auto coders, there are eight label neurons as well. The learning rate for Anne is 0.8, over a total of 150 epochs, each with mini batch size of 100 cases. The loss function is Means Square Error and activation functions are Hyperbolic Tangent.

### 4.2 Input layer

Since there are 8 different inputs / features each case generates a total of 8 different patterns. as can be seen in figure 2. Each row is one case. and column is the input value, all columns makes one pattern for a row

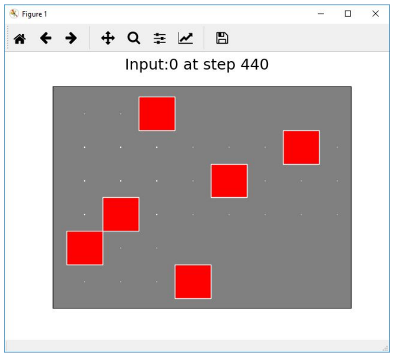


Figure 2: Input to Anne

### 4.3 Output layer

We want to train the network to output the same values / classification as what is put in. This involves the Hinton layer as well. To see if Anne knows how to do this, we can compare the two Hinton plots.

### 4.3.1 Input vs Output

One of many usages for auto encoders are compression. If we have 8 inputs, can train the network to reproduce, but compress the input features, if using less number of neurons in the hidden layer, than number of features / inputs.

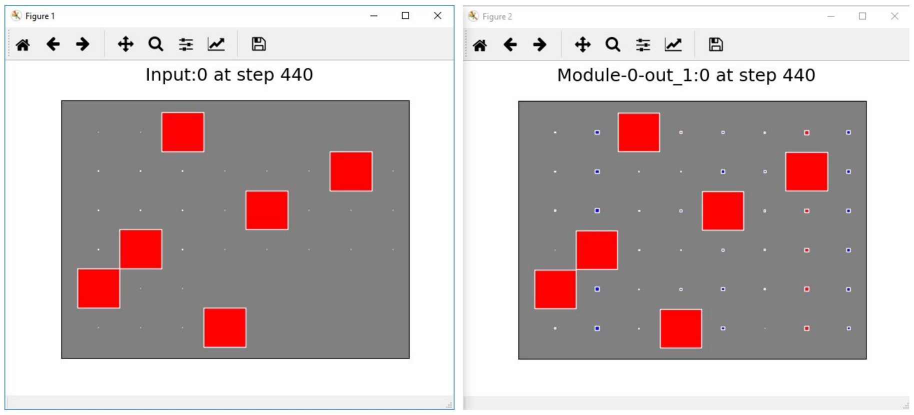


Figure 3: Input and wanted Output

## 4.4 Proper vs Poorly trained

Since we want input and out to be equal in auto encoders, we want the Hinton plotting for output to mimic the input Hinton plot. That indicates that Anne is proper trained. The figure to the left indicates a poorly trained Anne, she cant decide what pattern the input should be, for example case 2, in row 2 from top, we can see that Anne cannot decide whether the input should be column 2,3 or the correct, which is column 7, with much certainty. 3 is GTE to 7, thus indicating that 3 is the most likely one, but Anne is not absolutely sure, but with further training she will learn to recognize that the correct is 7. As she does in the figure to the right, row 2, we can see that Anne after some training, now recognizes the input and knows that the input should reproduce the Output pattern.

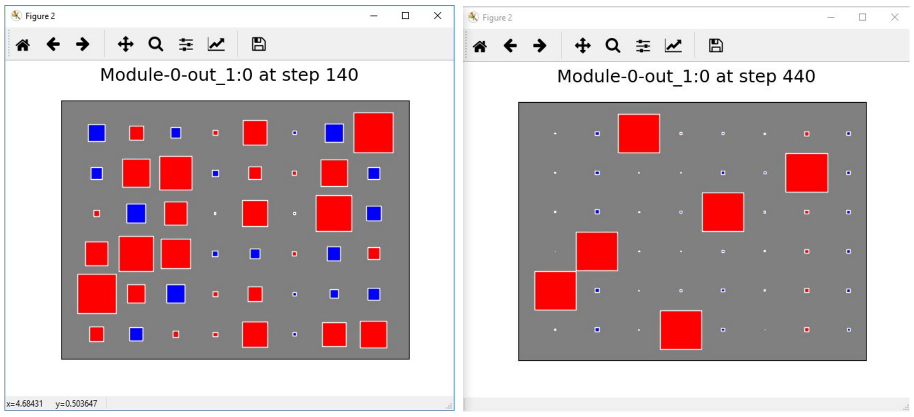


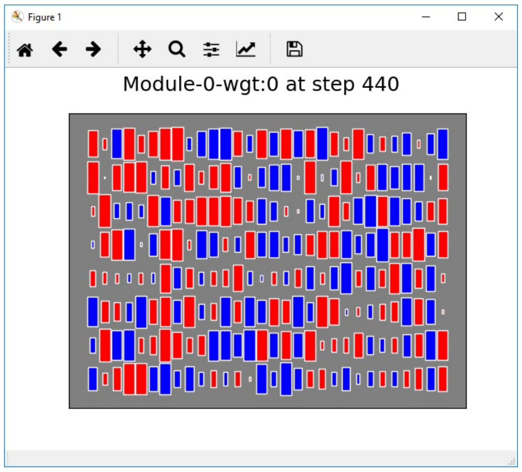
Figure 4: Poor vs Well

The learning / training is done by applying an activation function, tanh, to the input, which produces Anne's guess, than to check if it a correct guess, we check how wrong it is, by taking the expected value and subtract the guess. This error, is then back propagated through the network and applied to all weights. Thus teaching Anne what is correct and wrong.

## 4.5 Hidden layer's weights

If one of the hidden layers have fewer neurons that inputs, this is known as a bottleneck. This bottleneck is what causes compression. The network has to know how to remove something from the input, and reproduce it correctly for the output, if we make sure that it has to combine (using a bottleneck) we ensure that he network is trying to compress the input, but still get the correct output.

All of Anne's layers are fully connected, i.e. one feature are connected, through trainable weights to each of the next layers neurons. Since the layers are connected, we know that the first hidden layers input is the Input Layer. We also know that each hidden layers output is another layers input, this means that the last hidden layers output is actually the output layers input.



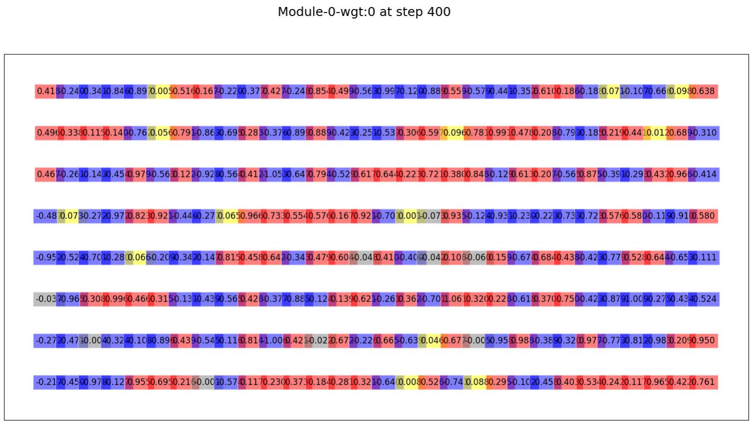


Figure 5: Hidden layers weight in Hinton and Display matrix - plots.

The columns are neurons in the hidden layer, and the rows are the values so row 1 column 1 is weight ones value. In the plots the color blue represents negative weights and red represents positives weights.

## 4.6 Mapping

The mapping is POST-training, and uses a small set of cases, known as map batch size, and is usually between 10 - 20 cases. The Mapping then shows a specified plot-function, of the x cases.
Each cases is one row in the plot.
Then if specified, the mapping shows a dendrogram for the specified layer output.

During mapping we have to grab the variables values we want to see, out of the net. Because of how TensorFlows works. TensorFlow first builds a graph, which just say how and what should be connected. But values are actually not shown until a TensorFlow run function is called, which then passes in the wanted values, and returns the specified wanted values, known as ‘grabvalues‘ which then the programmer is responsible for plotting. ‘Prob values‘ are values that tensorflow is responsible for showing in a TensorBoard.

## 4.7 Dendrogram

The dendrogram shows that Anne places get smalls distances between cases that produce the same output pattern / belongs to the same target. Thus the dendrogram gives a quick way to see if Anne is properly trained.
The way to see if this is the dendrogram recognizes patterns well is to have small distance low connections on the y-axis.

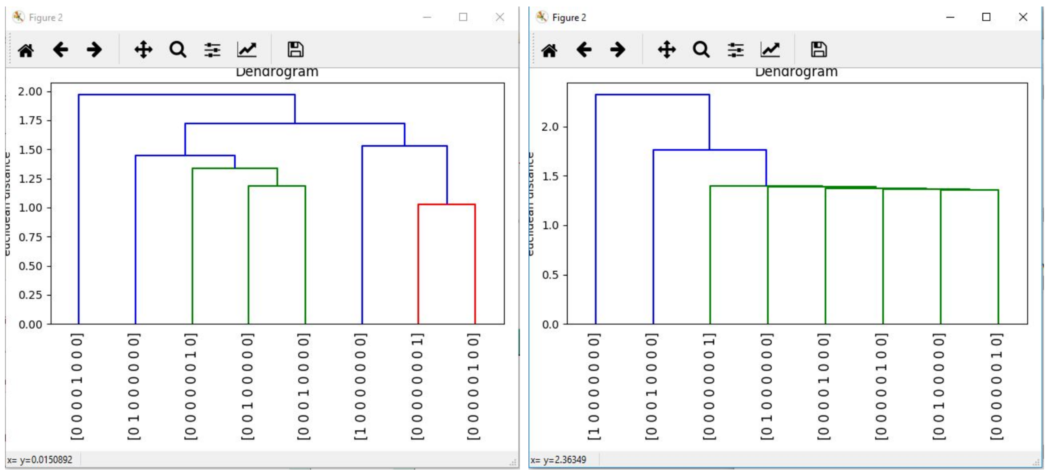


Figure 6: Poor-ish vs Well

## 5 Conclusion

Conclusion, Anne is very very good!

## References

1. TensorFlow : https://www.tensorflow.org/
2. Datasets : https://archive.ics.uci.edu/ml/index.php