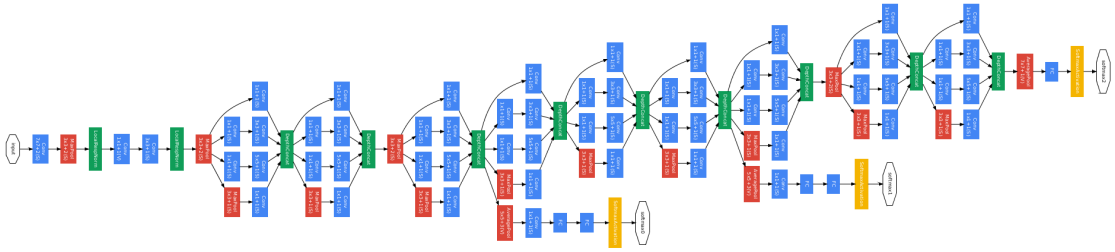
Deep Learning

Laboratory Course Manual

2018

Jan van Gemert, David M.J. Tax
Osman S. Kayhan, Tom J. Viering



Contents

Week 1

Introduction

Since this is the first time we are giving the Deep Learning course, we will update the lab manual (this document) each week. If anything is unclear regarding the exercises, please first try to ask the question on the forums on Brightspace, or ask your question during the lab sessions. If you have any comments how to improve the document, please drop us an email at tud.deeplearning@gmail.com.

Objectives When you have done the exercises for this week, you

- have a working installation of python, numpy, scipy, ipython and jupyter notebook.
- understand the basics of working with jupyter notebooks.
- have implemented kNN, SVM and/or a Softmax classifier in python.

1.1 Set up Python Environment

For the exercises in this course you will have to work on your personal laptop, the exercises will not work on TU Delft computers.

Exercise 1.1 **(a)** For windows users, install Anaconda for Python 3.6.

(b) For Mac users, The inbuilt system python installation can be used(which is typically python2). However, only versions 2.7, 3.5 and 3.6 is supported for this assignment. Irrespective of version, we recommend installation using Homebrew. After installing Homebrew, run the following commands in the terminal.

```
brew update
brew install python3 (or) brew install python2
```

The command `python3` typed in the terminal will then invoke the python3 interpreter. Please note that if you want to work with python2.7, you might need to edit your $PATH environment variable - placing `/usr/local/bin` ahead of `/usr/bin`. This can

be achieved by adding the line `export PATH='/usr/local/bin':$PATH` to the bash profile.

(c) For Linux users, you can use python 2.7, 3.5 and 3.6 for assignment of first week. Make sure you have installed python and pip. Besides, we recommend the usage of *Virtualenv*. In the console, use

```
cd <course file>
sudo pip install virtualenv
virtualenv -p python3 .env # directory
source .env/bin/activate
```

to create a new virtual environment using *Virtualenv*. We suggest you make an environment called `stanford_exercises` for week 1. If you are done with the lab practical, use the command `deactivate` in the console to deactivate the environment. In week 2 you can use the command

```
source .env/bin/activate
```

to activate the environment again.

To understand why you should use *Virtualenv*, see website1 and website2.

1.2 Jupyter Notebooks and Python

Exercise 1.2 (a) For Windows users: first download requirementswin.txt. Then open up the Anaconda console. Run the code

```
conda create -n exercises_stanford pip python=3.5
activate exercises_stanford
```

Now 'cd' to the directory where you put the requirements text file. Run the code

```
conda install --yes --file requirements_win.txt
```

For Mac and Linux users, the steps to accomplishing this can be found on this website: `http://cs231n.github.io/assignments2017/assignment1/` under 'Working locally'. During installation, use requirementslinux.txt file instead of the file given on the webpage above.

(b) Download the following IPython notebooks: Python Tutorial. Create a directory where you will put all your IPython notebooks and unzip this tutorial and put it in its own folder.

(c) Now we will start Jupyter Notebook. For Windows users, type

```
jupyter notebook --notebook-dir=E:\DL\
```

and replace `E:\DL\` by the directory path where you place your notebooks.

For Linux users, type

```
jupyter notebook
```

under your directory (`/home/user/DL/assignment1/`).

After these commands, a browser window will pop up with the Jupyter Notebook interface.

(d) Read Quickstart Ipython to understand how to use Jupyter Notebook.

(e) Open

```
Intro-to-Python\01.ipynb
```

in Jupyter Notebook. In the menu at the top, click 'Cell', 'All Output', 'Clear'. Go through the Notebook, executing cells by pressing 'Shift+Enter'. Observe the results and experiment a bit to get familiar with the Python programming language. If you get stuck in the cell with the command `help()`, this command requests user input. Type "q" in the text input box and press enter to escape.

(f) If you are unfamiliar with Python and / or Numpy, we recommend you to go through at least `01.ipynb`, `02.ipynb`, `03.ipynb`, `04.ipynb`, `08.ipynb`.

(g) Observe that 'File', 'Save and Checkpoint' saves the inputs and outputs of the IPython notebook. 'File', 'Close and Halt' closes a Jupyter Notebook and kills the process. Re-open the file, and observe that the input and output are the same as when you made the checkpoint. Note that when you re-open a file (after close and halt), a new process is started, so all variables are lost.

(h) When you double click on a text cell, you will see the source code. 'Shift-Enter' compiles the source into 'readable' text. This way you can learn how the Notebooks of the Python tutorial were created. You can add new cells to a Notebook by (at the top) clicking: 'Insert'.

Try to make a Jupyter Notebook yourself with some cells of your own. Try add a few cells with some headers, code, text (markdown), perhaps an equation (just type latex!) and plot.

See here for more info: Jupyter Notebook Tutorials, Markdown example.

(i) Before starting to Exercise 1.3, check also Python Numpy Tutorial.

1.3 Implement some classifiers in Python

Exercise 1.3 (a) Download the code of the exercise here Stanford Assignment 1. Download the CIFAR-10 dataset. Extract the file and put the data in the folder

```
cs231n/datasets/cifar-10-batches-py
```

(b) Open 'knn.ipynb' in Jupyter Notebook and complete the exercises. (You can find a reminder about knn here). Hints : Matrix broadcasting would be useful here, you could refer to this website. For the case of complete vectorization (without loops), the following hints can be helpful. The L2 distance / euclidean distance can be rewritten using $\sqrt{(x-y)^2} = \sqrt{x^2+y^2-2xy}$ in case x and y are numbers. What is the equation if x and y are vectors of length d? Finally, imagine that we have two data matrices X

and Y of size $n \times d$ and $m \times d$. How can we now write the distance matrix D of size $n \times m$ where entry D_{ij} is the L2 distance between x_i and y_j? You need to write the answer in terms of the matrices X and Y. Hint: the matrix XX^T is a $n \times n$ matrix of innerproducts between all objects in X.

(c) Unlike in the Pattern Recognition course, we will typically work with multiclass data, where we train several models at the same time (instead of using a one vs all approach). Read Linear Classifiers, Stanford to understand the basics of the multiclass SVM and the multiclass logistic regressor (softmax).

(d) Open 'svm.ipynb' and complete the exercises. (hints: It would be helpful to first calculate the derivative of the hinge loss when computing the gradient. For tuning hyperparameters part, some of you may encounter the numpy overflow problem with the learning rate and regularization strength you choose. Carefully change your parameters to avoid it.)

(e) Optional: open 'softmax.ipynb' and complete the exercises.

Week 2

Deep Learning Preliminaries

Objectives When you have done the exercises for this week, you

- have seen an example of data preprocessing in Python
- have learned basic terminology of Deep Learning,
- have trained a Logistic regression model in python using Scikitlearn,
- have implemented and have a better understanding of the Softmax function.

2.1 Udacity course

During this course we will use several exercises from the Udacity course called Deep Learning.

Exercise 2.1 **(a)** Register for Udacity (free).

(b) Enroll for the course Deep Learning.

(c) Go to Lecture 1 and complete items 1-20. Watch all the videos and be sure to complete the exercises.

Exercise 2.2 Now we will complete the `notMNIST` assignment of the Udacity course. This is item 21 of Lecture 1. For hints, go to Course Home, and click on Assignment 1 — these are additional videos with instructions. The Ipython notebook can be downloaded from here.

(a) Be sure to create a separate folder for this assignment, since to complete the exercise you will save several intermediate files to disk.

(b) For Conda (Windows) users it might be convenient to make a new environment for this exercise. For your convenience, follow instructions in zip file here.

(c) For Mac and Linux users, you can either follow the instructions in the assigment video (using Docker) or use the downloaded Ipython notebook and use Virtualenv (as

described in Week 1). You will need to install the following packages for the Udacity exercises:

```
libjpeg8-dev
scikit-learn
pyreadline
Pillow
imageio
jupyter
matplotlib
```

Exercise 2.3 Finish Lecture 1 of the Udacity course.

2.2 Softmax

Exercise 2.4 (a) Finish exercise 'softmax.ipynb' from last week.

Week 3

Introduction to Tensorflow

Objectives When you have done the exercises for this week, you have learned

- how to install Tensorflow,
- a general overview of Tensorflow,
- graph and session mechanism in Tensorflow,
- basic operation, tensor types and data importing in Tensorflow,
- how to use Tensorboard.

Exercise 3.1 **(a)** If you are on Windows, go to `https://www.tensorflow.org/install/install_windows` and follow the 'Anaconda' installation steps (CPU!). Make sure to use 'pip' to install Tensorflow as recommended in the guide, and validate your Tensorflow Installation.

(b) For MacOS users, go to `https://www.tensorflow.org/install/install_mac` and follow the 'Virtualenv' installation steps (CPU). Make sure to validate your installation.

(c) For Linux users, go to `https://www.tensorflow.org/install/install_linux` and follow the 'Virtualenv' installation steps (CPU). Make sure to validate your installation.

Exercise 3.2 **(a)** To become more familiar with TensorFlow, review TF Lecture 1 (slide 25 - 46). When following along the lecture, try writing all the codes in a Jupyter notebook and try to see if you can get the same results as in the slides.

(b) Slide 37: explain in your own words why the outcome is not 8. Write the answer down in your Jupyter notebook in a markdown cell (also with the other exercises).

(c) Slide 41: try to run the code in a Jupyter notebook.

(d) Why does Tensorflow use graphs? (Hint: Check slide 64)

(e) Make a graph with at least 4 operations, where the output of several operations are the inputs of other operations. Write the graph down with pen and paper, and compute the result using pen and paper. Now run the graph in TensorFlow, and make sure the result in tensorflow matches your result that you got using pen and paper.

Exercise 3.3 **(a)** To get more familiar with TensorFlow, review TF Lecture 2. This section mentions about basic operation, tensor types and data importing topics in Tensorflow.

(b) First, make sure you can run tensorboard. You will probably need to open up one console to start jupyter notebook (for writing your python code), and you will need a second console to start up Tensorboard. See slides 8 and 9.

(c) Slide 7 and 8: Create the graph in Jupyter notebook, run the code (saving the graph to the logfile) and visualize the graph in Tensorboard.

(d) To track your variables and nodes easily on Tensorboard, it is recommended to name them (Slide 13). Name your variables and check that you see the names in the graph in Tensorboard.

(e) Implement the broadcasting example, and see if you get the same results as in NumPy (Slide 17).

(f) Explain why it is not desirable to define a lot of / large constants in the graph. What should be used instead?

(g) Check the usage of `eval()` function and use it in a graph.

(h) How can you define a function without knowing value of its inputs? Write the answer down in your Jupyter notebook in a markdown cell.

(i) Make a graph that illustrates the differences between normal and lazy loading. Why is normal loading preferred?

Exercise 3.4 **(a)** Now you will experiment with some basic tensorflow operations. Complete the code of exercise — you might want to put it in a Jupyter notebook for your own convenience. In the code, the solution of first part is already given. You see `InteractiveSession` for the first time. To understand how it works, you can check the documentation. If you do not want use it, you can use normal `Session` as well. You might have to resort to the documentation on math operators to find the functions you need to apply.

(b) In the given part of example, did you try to print `x` and `y` as well? Could you obtain same result as `out`? If they are not matching, why?

Exercise 3.5 **(a)** Create a graph, with the following tensors: W shape (10, 784) x shape (784, 1) and y shape (10,1) and b shape (10,1). Declare these tensors as placeholders.

(b) Implement $y = \text{softmax}(Wx + b)$ in the tensorflow graph using tensor operations. Recall that the jth entry after the softmax is given by $\text{softmax}(x)_j = e^{x_j} / \sum_k e^{x_k}$.

(c) Store the predicted class in p (figure out the formula for p on your own — think about how you would go from softmax scores to your prediction).

(d) Visualize the graph in Tensorboard and verify it looks correct.

(e) What will we put in the placeholders? We will use our data and trained model from Week 2. Download this Ipython notebook (be sure you are using python 3), extract and run the code therein (this should be relatively fast). This will train a logistic regression model on the `not_mnist` dataset.

(f) Now, it is your task to feed in the x values of the test set, and the W and b obtained by logistic regression (use `sess.run` with `feed_dict` as in Slide 80 of TF Lecture 2).

Then by using the graph, compute the predicted class p. Compare the output of the graph (p) with the corresponding true label.

(g) Compute the accuracy of the logistic regressor by using the tensorflow graph to compute the prediction for all test samples (you can simply do a for loop over all samples — so feed all samples one by one). Make sure that the accuracy you get is equal to the accuracy reported in the IPython notebook.

(h) Can you also find a way to load a batch of data into the tensorflow graph at once, and that the graph actually computes directly the accuracy? Hint: an easy way to do this involves changing the shape of x and y.

Week 4

Tensorflow Examples

Objectives When you have done the exercises for this week, you have

- built and trained two linear models in tensorflow: linear and logistic regression
- implement and train a 'simple' 2 layer neural network in Tensorflow.

4.1 Linear Models in TensorFlow

First you will implement linear and logistic regression in Tensorflow. We will make use of the slides here. Before starting this section, please download the following files:

- Linear Regression example code
- Logistic Regression example code
- utils
- Data

Press `CTRL + S` to save the file to disk after opening it in your browser.

4.1.1 Linear Regression

You will follow the steps in the slides to design your linear regression model. Use the Linear Regression example code given above.

Exercise 4.1 **(a)** Follow along the steps in slides 11 - 40.

(b) Fill the missing parts of the code using the advice of the slides. For now use the MSE (mean squared error), do not yet try the Huber loss.

(c) Visualize the graph of your model.

(d) Visualize the training procedure of your model (for example, plot the MSE during training in a graph). Make sure your loss is going down and converges.

(e) Plot `real data` — `predicted data` by uncommenting the last few lines in the given code sample. What do you observe?

Exercise 4.2 (a) As detailed in the slides, try to use the Huber loss. What is the difference compared to using the MSE? Compare the MSE of both approaches on a test set.

(b) Check the slides: 40-59. In the example code, you used `placeholder`. Now try using `tf.data`. Are there any differences?

(c) Check the slides: 59 - 67. Instead of `GradientDecentOptimizer`, try another optimizer of your choice. Do you observe any differences? Note that the optimizers also admit hyperparameters. Try to experiment with a few parameters and also visualize the corresponding training curves.

4.1.2 Logistic Regression

The remaining slides explain how to use logistic regression. Again follow the steps in the slides and fill in the missing code in the logistic regression example code.

Exercise 4.3 (a) Fill the missing parts in the example code.

(b) Train your model with the Adam Optimizer and use the cross-entropy as loss function.

(c) Visualize the graph of the model.

(d) Train your model and visualize the training loss curve with tensorboard. Is the loss decreasing and converging?

(e) Evaluate the model on the test set. What is the accuracy?

4.2 2 Layer Neural Network

Now you have seen two examples of linear models, we are ready to move on to more complex models. Now you will design a simple Neural Network with two layers. This is again part of the Udacity Deep Learning course.

Exercise 4.4 (a) Watch videos from lesson 3 (videos no. 1- 9) here.

(b) What is the reason for using an activation function? What is/are the advantage(s) of that?

(c) Watch video 11. Which one is better, a deeper or a wider network? Explain why.

(d) Now we will complete exercise 2 (item 10). Download the code sample and complete item 10 by filling in the missing code. You can use your own computer or Google Colaboratory IPython Notebook (if you want to know how to use it, check the announcement). Use the RELU activation and 1024 hidden units.

(e) Visualize the graph of the model. Does it look good?

(f) Train the model, and visualize the loss during training. Does the loss go down and converge?

(g) What is the performance on the test set? How does it compare with our previously trained logistic regression model on the `notMNIST` dataset?

(h) Try several numbers of hidden units (for example: 20, 100, 500, 1000, 2000). How do the accuracies change on a testset?

Week 5

Convolutional Neural Networks

Objectives When you have done the exercises for this week, you have learned

- What a Convolutional Neural Network (CNN) is,
- How to implement CNNs with Tensorflow.

Exercise 5.1 This week we will use the Udacity material again.

(a) Important: Before you begin this week, make sure you have finished exercises **4.1**, **4.2** and **4.4** from last week. We will build upon this knowledge in this week.

(b) Watch videos 1 - 10 of the "Lesson 4: ConvNets".

(c) In a CNN, after we apply convolutions to an image, you obtain a feature map which has 3 dimensional shape. What do all dimensions of this feature map represent?

(d) Explain padding and stride. Why or when can padding be important for CNNs?

(e) Which one is better in terms of keeping information more, bigger stride or pooling? What are the advantages and disadvantages of using it?

Exercise 5.2 (a) Watch video 12. This will introduce the assignment.

(b) We have prepared a IPython notebook that is different from the notebook provided by the Udacity course. Download the IPython Notebook here.

(c) First we will take a good look at the code to see what is going on. The idea of this exercise is to fill out the missing parts of Table 5.1. The answers below marked with (TABLE) should be filled in the table.

(d) What is the input to the model? (TABLE) From now on we assume we set `only_visualize_model` to `True`. Write down the shape of the input tensor (TABLE). What is the shape of `layer1_weights` and `layer1_biases`? (TABLE) **Note: when you are asked to compute shapes or numbers, do so by hand with pen and paper and a calculator, do not write print statements in the code.**

(e) Look at the first application of `conv2d` in the definition of the model. What are the meaning of the input arguments? Use the documentation of tensorflow.

(f) How many filters are there in the first layer? (TABLE) What is the filter size?

Table 5.1: Overview of the convnet. The exercise will help you fill in this table.

	input	operation	output
layer 1	.....	.....	.....
	.....	.....	.....
shapes 1			
layer 2	.....	.....	.....
	.....	.....	.....
shapes 2			
layer 3	.....	.....	.....
shapes 3			
layer 4	.....	.....	.....
shapes 4			

(TABLE) Does the filter have a 3rd dimension? Why or why not? What stride and padding is used? Compute the shape of `conv1` (TABLE).

(g) What happens when we go from the variable `conv1` to the variable `hidden1`? (TABLE) What is the shape of `hidden1`? (TABLE)

(h) What is the input to the second layer? Write down the name and shape of the tensor (TABLE).

(i) How many filters are there in the second layer? (TABLE) What is the filtersize? (TABLE) Does the filter have a 3rd dimension? Why or why not? What stride and padding is used? Compute the shape of `conv2`.

(j) What happens when we go from the variable `conv2` to the variable `hidden2`? (TABLE) How many biases are there? (TABLE) Compute the shape of `hidden2` (TABLE).

(k) What is the input to the third layer? (TABLE) In the third layer, why do we first reshape?

(l) What is the operation in the third layer? (TABLE) How many parameters (weights and biases) do we need here? (TABLE)

(m) What is the output of the third layer? (TABLE) Write down the shape. (TABLE)

(n) What is the input to the fourth layer? (TABLE)

(o) What is the operation in the fourth layer? (TABLE) How many parameters (weights and biases) do we need here? (TABLE)

(p) What is the output of the fourth layer? (TABLE)

(q) The output of the function `model(data)` cannot directly be used. What operation is used first to obtain posterior probabilities?

(r) Compute the amount of trainable parameters of each layer. What is the total amount of trainable parameters? Which layer has most parameters?

(s) Set `only_visualize_model` to true. Visualize the graph with tensorboard. Confirm that the shapes of the tensors: `hidden1`, `hidden2`, `hidden3`, `output` that you computed by hand are correct.

(t) Set `only_visualize_model` to false and train. What test error do you obtain?

Exercise 5.3 **(a)** Now set all the strides to 1 of both convolutional filters. In order to still reduce dimensionality, you will add a max pooling operation (`nn.max_pool()`). For the max pool layer use a stride of 2 and a kernel size of 2. Retrain your model and compare the result with convolution with stride 2.

(b) Try to improve the performance of your model. You may get some inspiration from the architecture of LeNet5. For sure try different optimization settings (learning rate, amount of epochs), or a different optimizer, this will likely already make a big difference. If you have tried this, you may try adding more layers, more hidden units or more filters.

(c) Optional: Explain why we cannot set `b2` and `num` to fixed values in case we set `only_visualize_model` to `False`.

(d) Optional: in the code we have used `with tf.name_scope('compute_loss'):` to group together several tensorflow operations. Use this approach to build a very nice graph and visualize it. For example, make a scope for each layer, make a scope for the training data, etc.

(e) To gain a deeper understanding of convnets and their architecture, we recommend you to read the material of item 11 of Udacity (Readings). Please take the time to do this at home.