Assignment 1: Let the Tensors Flow

Deadline: October 25th, 9am

In this assignment, you will implement and train your first deep model on a well-known image classification task. You will also familiarize yourself with the Tensorflow (https://www.tensorflow.org/) library we will be using for this course.

General Assignment Notes

- You will need to do some extra reading for these assignments. Sorry, but there is no way around this. This should be significantly reduced by attending the practical exercise sessions, but things are re-iterated here for people who missed them (that's why there's so much text ;)).
- Assignments are posed in a very open-ended manner. Often you only "need" to complete a rather basic task. However, you will get *far* more out of this class by going beyond these basics. Some suggestions for further explorations are usually contained in the assignment description. Ideally though, you should really see what interests *you* and explore those directions further. Share and discuss any interesting findings in class and on Mattermost!
- Please don't stop reading at "Bonus"; see above. Don't be intimidated by all the text; pick something that interests you/lies within your capabilities and *just spend some time on it*.

Setting Up

To work on your own machine

Install Python (https://www.python.org/) (3.x – depending on your OS you might need to install a not-so-recent version as the newest ones may be incompatible with TF) if you haven't done so, and install Tensorflow 2 (https://www.tensorflow.org/install/). This should be as simple as writing

```
pip install tensorflow
```

in your console. If you want GPU support (and have an appropriate GPU), you will need to follow extra steps (see the website). For now, there should be no need since you will usually use your own machine only for development and small tests.

If you want to do everything in Colab (see below), you don't need to install Tensorflow yourself.

Google Colab

Google Colab (https://colab.research.google.com) is a platform to facilitate teaching of machine learning/deep learning. There are tutorials available on-site. Essentially, it is a Jupyter notebook environment with GPU-supported Tensorflow available.

If you want to, you can develop your assignments within this environment. See below for some notes. Notebooks support Markup, so you can also write some text about what your code does, your observations etc. This is a really good idea!

Running code on Colab should be fairly straightforward; there are tutorials available in case you are not familiar with notebooks. There are just some caveats:

- You can check which TF version you are running via `tf.__version__`. Make sure this is 2.x!
- You will need to get external code (like `datasets.py`, see below) in there somehow. One option would be to simply copy and paste the code into the notebook so that you have it locally available. Another would be to run a cell with `from google.colab import files; files.upload()` and choose the corresponding file, this will load it "into the runtime" to allow you to e.g. import from it. Unfortunately you will need to redo this every time the runtime is restarted.
- Later you will need to make data available as well. Since the above method results in temporary files, the best option seems to be to upload them to Google Drive and use `from google.colab import drive; drive.mount('/content/drive')`. You might need to "authenticate" which can be a bit fiddly. After you succeed, you have your drive files available like a "normal" file system. If you find better ways to do this (or the above point), please share them with the class!
- In newer Colab versions, there is actually a button in the "Files" tab to mount the drive – should be a bit simpler than importing `drive`. The following should work now:
    1. Find the folder in your Google Drive where the notebook is stored, by default this should be `Colab Notebooks`.
    2. Put your data, code (like `datasets.py` linked in one of the tutorials further below) etc. into the same folder (feel free to employ a more sophisticated file structure, but this folder should be your "root").
    3. Mount the drive via the button, it should be mounted into `/content`.
    4. Your working directory should be `content`, verify this via `os.getcwd()`.
    5. Use `os.chdir` to change your working directory to where the notebook is (and the other files as well, see step 2), e.g. `/content/drive/My Drive/Colab Notebooks`.
    6. You should now be able to do stuff like `from datasets import MNISTDataset` in your notebook (see MNIST tutorial further below).

Tensorflow Basics

NOTE: The Tensorflow docs went through significant changes recently. In particular, most introductory articles were changed from using low-level interfaces to high-level ones (particularly Keras). We believe it's better to start with low-level interfaces that force you to program every step of building/training a model yourself. This way, you actually need to understand what is happening in the code. High-level interfaces do a lot of "magic" under the hood. We will proceed to these interfaces after you learn the basics.

Get started with Tensorflow. There are many tutorials on diverse topics on the website, as well as an API documentation (https://www.tensorflow.org/api_docs/python/tf). The following should suffice for now:

- Read up on tensors and operations (https://www.tensorflow.org/tutorials/customization/basics) to get a basic idea of the stuff we

are working with. For more info, check the guide (https://www.tensorflow.org/guide/tensor) (the stuff about “evaluating” and printing tensors is unfortunately outdated).
- A special and very important kind of tensors are variables (https://www.tensorflow.org/guide/variable) (you can ignore the stuff on Keras for now).
- Automatic differentiation (https://www.tensorflow.org/tutorials/customization/autodiff) is probably the most important concept in Tensorflow!
- See how to fit a simple linear model (https://www.tensorflow.org/guide/basic_training_loops). Once again, you may ignore the Keras stuff (second part).

Linear Model for MNIST

MNIST is a collection of handwritten digits and a popular (albeit by now trivialized) benchmark for image classification models. You will see it A LOT.

Go through this basic MNIST tutorial (http://blog.ai.ovgu.de/posts/jens/2019/002_tf20_basic_mnist/index.html) we wrote just for you! It’s a logistic (softmax) regression “walkthrough” both in terms of concepts and code. You will of course be tempted to just copy this code; please make sure you understand what each line does.

Play around with the example code snippets. Change them around and see if you can predict what’s going to happen. Make sure you understand what you’re dealing with!

Note: If you copy/paste this into Colab, make sure that the TF version is 2.x!

Building A Deep Model

If you followed the tutorial linked above, you have already built a linear classification model. Next, turn this into a *deep* model by adding a hidden layer between inputs and outputs. To do so, you will need to add an additional set of weights and biases (after having chosen a size for the layer) as well as an activation function.
There you go! You have created a Multilayer Perceptron. **Hint:** Initializing variables to 0 will not work for multilayer perceptrons. You need to initialize values randomly instead (e.g. random_uniform between -0.1 and 0.1). Why do you think this is the case?

Next, you should explore this model: Experiment with different hidden layer sizes, activation functions or weight initializations. See if you can make any observations on how changing these parameters affects the model’s performance. Going to extremes can be very instructive here. Make some plots!

Also, reflect on the Tensorflow interface: If you followed the tutorials you were asked to, you have been using a very low-level approach to defining models as well as their training and evaluation. Which of these parts do you think should be wrapped in higher-level interfaces? Do you feel like you are forced to provide any redundant information when defining your model? Any features you are missing so far?

Bonus

There are numerous ways to explore your model some more. For one, you could add more hidden layers and see how this affects the model. You could also try your hand at some basic visualization and model inspection: For example, visualize some of the images your model classifies incorrectly.

Can you find out *why* your model has trouble with these?

You may also have noticed that MNIST isn't a particularly interesting dataset – even very simple models can reach very high accuracy and there isn't much "going on" in the images. Luckily, Zalando Research has developed Fashion MNIST (https://github.com/zalandoresearch/fashion-mnist). This is a more interesting dataset with the *exact same structure* as MNIST, meaning you can use it without changing anything about your code. You can get it by simply using `tf.keras.datasets.fashion_mnist` instead of regular MNIST.
You can attempt pretty much all of the above suggestions for this dataset as well!

How to Hand In Your Assignment

In general, you should prepare a notebook (`.ipynb` file) with your solution. The notebook should contain sufficient outputs that show your code working, i.e. **we should not have to run your code to verify that you solved the task**. You can use markdown cells to add some text, e.g. to document interesting observations you made or problems you ran into.

- If you work on Colab, make sure to save your notebooks with outputs! Under Edit -> Notebook settings, make sure the box with "omit code cell output…" is **not** ticked.

Hand in your notebook via Moodle (https://elearning.ovgu.de/course/view.php?id=11018) until the deadline specified at the very beginning. Just upload your solution file(s) for the corresponding assignment.

You can form groups to do the assignments (up to three people). However, **each group member needs to upload the solution separately** (because the Moodle group feature is a little broken). At the very top of the notebook, include the names of all group members!

What to hand in this time

- No need to hand in any of the "Tensorflow basics"/linear model stuff, this is just for you to get familiar!
- *Do* hand in working code that defines, trains and evaluates an MLP on MNIST!
- Hand in any additional experiments you tried!
- You can answer the various "food for thought" questions in this assignment via Markdown cells.