Homework 04

IANNwTF

November 12, 2022

This week's deadline is *28.11., 23:59*.
Submit your homework via https://forms.gle/ApAZ5ubY8ewgNmJA9
Remember that you now have to review another group's homework. More on that can be found further down.

Contents

1 Reviews & How to do them

Welcome back to the fourth homework for IANNwTF. You have learned more about Optimization and Batching this week - in this homework we will not only apply this new knowledge, but also try and build our first slightly more involved network!

In addition to handing in your homework, you will have to review last week's homework of two groups and note down which group you reviewed on the homework submission form. This requires you to find two other groups and have a short (10 min) meeting where you go over each others homework submission, discuss it and then write a short summary of what you discussed on each others forum pages. We recommend using the Q&A timeslots for this purpose, but you can meet however and whenever you like. The main review part of this should be the discussion you have together. The written review in the forum should merely be a recap of the main points you discussed so that we can see that you did something and the reviewed group has access to a reminder of the feedback they received. **If there are any open questions regarding your or any other groups code afterwards, please feel invited to discuss this with us in the QnA sessions, so we can help you sort out how a perfect solution would have looked like!** Just to make this obvious: We will not penalize any submission (by you or any of the groups you reviewed) based on such questions. The purpose is helping you understand and code better only.

As to how a discussion could look like, you could for example have the group being reviewed walking the other two groups through their code. The other two groups should try to give feedback, e.g. "I like how you designed your data pipeline, looks very efficient.", "For this function you implemented there actually exists a method in e.g. Numpy or Tensorflow you could've used instead." or "Here you could've used a slightly different network architecture and higher learning rate to probably achieve better results.", and note down their main points.

Important! Not every member of every group has to be present for this. You could for example implement a rotation where every group member of yours only has to review every third homework.

2 Assignment: MNIST math

We are revisiting MNIST handwritten digits, but with a twist: We are not classifying handwritten digits, but rather asking our network to do some simple math on them: In subtask (1) we want to know whether $a + b \overset{?}{\geq} 5$, in subtask (2) we want to predict $y = a - b$, where a and b are respective MNIST digits!

In the following we ask you to write code (specifically your model and training data preparation) in a parameterized way, which allows you to reuse nearly all of the code for both subtasks, to minimize the effort required for this homework! We will try and give some hints on how to easily achieve this in the

following. If there are any open questions, make sure to ask away in the QnA sessions.

2.1 Preparing MNIST math dataset

Remember from last week:he MNIST dataset consists of seventy thousand labelled images, each depicting a single handwritten digit. This may sound like a lot of data, but as you'll see for yourselves in a bit, the images are rather small.

The MNIST dataset is included in TensorFlow, so you getting access to it is actually pretty easy. You can load it directly into your code like this:

```
import tensorflow_datasets as tfds
import tensorflow as tf

(train_ds, test_ds), ds_info = tfds.load('mnist', split=['train', '
    test'], as_supervised=True, with_info=True)
```

As we have worked with the data before, you are probably familiar with the structures already. If you need a recap, make sure to check last weeks homework again!

2.2 Two MNIST math datasets

Remember that for the following you want to create a preprocessing function. As subtask (1) an substask (2) require very similar inputs, we will create a parameterized preprocessing function - mapping a dataset and the 'subtraction' or 'larger than five' condition onto respective datasets. We will generally have **three different steps** to our preprocssing - general MNIST preprocessing (datatype, normalization, etc.), pairing data tuples onto respective parameterized targets, and finally batching and prefetching.

As a **first step**, for general preprocessing steps we follow last weeks ideas closely. You may follow the example code from the lecture (i.e. L03) again, but to summarize the steps:

The MNIST handwritten digits images come in uint8 datatype. This refers to unsigned 8-bit integers (think numbers 0-255). As the network requires float values (think continuous variables) as input rather than integers (whole numbers), we need to change the datatype: (**map**[1] in combination with **lambda** expressions[2] can be really useful here). In your first lambda mapping you want to change the datatype from uint8 to tf.float values [3]. To feed your network the 28x28 images also need to be flattened. Check out the reshape function [4], and if you want to minimize your work, try and understand how it interacts with size elements set to the value -1 (infering the remainder shape). In order to improve the performance you should also normalize your image values. Generally this

[1] https://www.tensorflow.org/api_docs/python/tf/data/Dataset#map

[2] https://docs.python.org/3/tutorial/controlflow.html?highlight=lambda#lambda-expressions

[3] https://www.tensorflow.org/api$_d$ocs/python/tf/cast

[4] https://www.tensorflow.org/api$_d$ocs/python/tf/reshape

means bringing the input close to the standart normal (gaussian) distribution with $\mu = 0$ and $\sigma = 1$, however we can make a quick approximation as that: Knowing the inputs are in the 0-255 interval, we can simply divide all numbers by 128 (bringing them between 0-2), and finally subtracting one (bringing them into -1 to 1 range).
Second Step is figuring out the parameterized targets: Think about the following: What kind of supervised learning task and respective loss-function and target type are required for our two subtasks respectively? It makes sense to work yourself through this backwards: First understand what loss function is required for each. Then check the respective TensorFlow documentation to find out how your targets should look. Once you know what target data-representation is needed for each, we can move to the second step: Each datapoint fed into the network requires two MNIST images as input, with one target being created from their combination. We will discuss many approaches to combining dataset elements - but the dataset zip functionality [5] probably is the most direct solution (as alternatives you might check out the respective window, scan or even batch functions!). After pairing two dataset elements, you can now make sure each dataset element gets two input mnist images, but you still have to calculate the respective target as the third element.

Before finalizing the dataset in the **third step** we recommend checking your dataset at this point - you expect each element to be a triple (tuple with three elements), the first two of which are images, the last one the respective target. Finally we shuffle and (mini-)batch the dataset, and can use the apply method to create datasets for both the $a + b \overset{?}{\geq} 5$ and $a - b = y$ problem!

3 Building shared weight models

In this step we have to build a network, that can solve the math task: This requires a network that takes two inputs (i.e. two input images), and outputs, i.e. predicts the math result. Technically you could just combine (e.g. concatenate) the two input images into one vector, however this would be rather inefficient. Instead we want you to learn about weight-sharing in a hands-on way: One intuitive way to solve this task would be to feed both inputs seperatedly into a layer, then combine (e.g. concatenate) the results from this layer[6]. But both of these input layers would have to solve essentially the same problem for their respective input images: Extract the information, what digit is depicted! As both are basically tasked with the same problem, and have the same input and output shapes, we can improve our network by **using the same layer for both inputs!**[7]. Finally you have to make sure to parameterize your model to use the

[5] https://www.tensorflow.org/api_docs/python/tf/data/Dataset#zip

[6] Think: Input one goes into input layer one and produces activation one. Input two goes into input layer two and produces activation two. Activation layer one and two are combined (concatenated) and fed into layer three, which outputs a scalar

[7] Think: Input one goes into input layer one and produces activation one. Input two also goes into input layer one and produces activation two. Activation one and activation two are

correct activation functions for subtasks (1) and subtask (2) respectively.

4 Training the networks

Create a training function (with an internal training loop function), which is able to run either of the two subtasks. The function takes two inputs: The subtask to solve and the optimizer to use for it, both specified via inputs: Write the function, such that it creates respective models and datasets based on the parameterized preprocessing function you have written before. Choose the correct loss-function for the task and run training.

5 Experiments

Run training with a classic SGD optimizer (without momentum) and an Adam Optimizer. For an outstanding submission, you are required to also add SGD with Momentum, RMSProp and AdaGrad optimizers and compare the training results by plotting them cleanly side-by-side.

combined (concatenated) and fed into layer three, which outputs a scalar