Homework 05

IANNWTF 20/21

Submission until 02 Dec 23:59 via https://forms.gle/H31ckx5251Qg3Ndm6

Welcome back to the fifth homework for IANNWTF. This week we have learned about further strategies to improve your network's perormance

In this homework, we want to build a CNN classifying pictures from the famous Cifar10 dataset. Your goal is to apply all optimization techniques you know and find useful to achieve a really good performance but to avoid overfitting! Have fun! [1]

This week we again try to keep the instructions as short as possible, so that you are challenged to try it yourself. Because the best way to learn something is to do it yourself. If you are lost, check the hints for advice or visit us during the Q&A Sessions (their purpose is to support you with any trouble you encounter, so really feel free to drop by!). Also, you can always refer to the code snippets and notebooks provided in courseware.

1 Data set

We will work with the Tensorflow Cifar10 dataset. https://www.cs.toronto.edu/%7Ekriz/cifar.html. It contains 60.000 coloured images of cells with equal sizes. Each images corresponds to one of 10 categories. The dataset is already implemented as a keras.dataset module and very convenient to load! Maybe try to find an load it yourself before referring to the hints. [2]

Print out some of the images together with their labels. The names of the label classes are: "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck". [3]
Perform other necessary or beneficial preprocessing steps.[4]

2 Model

Implement a Convolutional Neural Network to classify the pictures. This time, try to work in some of the optimization techniques you learned about (Data Augmentation is very optional here though).
[5] [6] [7] [8] [9]

3 Training

Then train your network. We would like you to just get started and try it yourself. In the hints we put some training hyperparameters for your orientation in case you're stuck. [10]

For this task, you should reach a test accuracy of around 60% to pass. In the outstanding homeworks, we would like to see at least 85% accuracy, but be aware that training will take some time.

4 Visualization

Visualize accuracy and loss for training and test data using matplotlib.

It might be useful to work in the visualization into your training loop, so you can check on your networks progress while training. It will save you time as you can see how your changes affect the training earlier during the process.

5 Outstanding Requirements

To receive an outstanding your solution should include the following things:

- >= 85% accuracy
- a proper Data Pipeline
- use multiple regularization techniques
- nice visualizations
- useful comments

Data Augmentation is a nice add-on, but optional.

Notes

[1]Remember, you learned about optimizers, weight penalties, dropout, batch normalization and data augmentation.

[2]You can load this dataset directly from keras `https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10` with the method load_data(). Remember, this way you directly get a tain- and a testdataset.

[3]You can put the label names in a list (in the correct order) and match the label classes to the list index when printing.

[4] Relevant preprocessing steps:

- Shuffling
- Batching, an orientation would be minibatches of size 64. Feel free to also try other batchsizes and see what happens.
- Normalize the images so your network can work with them better. Check Courseware/Image Representation for inspiration.
- One-hot-encode the labels. What should be the depth of the resulting labels?
- Feel free to apply Data Augmentation (rotation, zooming...). Check out Courseware/Regularization/Data augemantation for that. But be aware, it will slow down your training a lot!

[5]You can always start with the convolutional network you used to solve last week's homework and make some additions to it. Think about how the readout layer for this dataset should differ from the Malaria one! [1]

[6]If you want to use Dropout / Batch Normalization, remember to pass on the training flag in the call function. Also think about the order of these layers! [2]

[7]If you are using Batch Normalization layers, remember your previous layer should not have any activation function, but you have to apply an activation function manually after the Batch Normalization layer.

[8]Use a readout layer. You can decide to reshape your inputs before this read out layer (check out tf.keras.layers.Flatten()) or use global average pooling: tf.keras.layers.GlobalAveragePooling2D()

[9]6 convolutional layers should be enough to reach around 80% accuracy (if trained long enough). Feel free to build deeper networks but they need longer to train and are more prone to overfitting.

[10] Training Hyperparameters for orientation:

- epochs: around 30, depending on your architecture you can train even longer as long as the model is not overfitting
- learning rate: should be very small! Try something between 0.00001 and 0.0001.
- optimizer: Adam
- loss: BinaryCrossEntropy. Check `tf.keras.losses.BinaryCrossentropy()`
- accuracy: how many items in prediction and target are the same (in the batched vectors)? $\rightarrow$ take the mean of it

[1]Number of classes.
[2]First apply Batch Normalization, then the activation function and then Dropout.