Homework 3: Neural Networks with TensorFlow

Large-Scale Data Analysis

Christian Igel & Fabian Gieseke

Submission Deadline:
28 May 2019, 10:00

1 Standard Neural Network

Consider the following Keras model:

```
tf.keras.models.Sequential([
  tf.keras.layers.Dense(64, activation='sigmoid', input_shape=(1,)),
  tf.keras.layers.Dense(1, activation='linear')
])
```

1. **(10 pts.)** Report the number of trainable parameters of the model. Explain the number of parameters for each layer.

2. **(5 pts.)** Add a layer (`tf.keras.layers.Dense`) with 32 neurons before the finale layer. Show the code in your report.

3. **(10 pts.)** Rewrite the model using the Keras functional API. The notebook `MNIST with different Keras APIs.ipynb` gives an example (after "Functional interface:"). More details can be found in the online documentation (`https://www.tensorflow.org/alpha/guide/keras/functional`).

   Show the code in your report.

4. **(10 pts.)** Add a shortcut connection from the inputs to the last layer (the input and output dimensionality of the model must not change, i.e., it should still implement a mapping $\mathbb{R} \to \mathbb{R}$). You may need to concatenate the output of layers, which can be achieved using `tf.keras.layers.concatenate`.

   Report the number of trainable parameters of the resulting model. Show the code in your report.

2 Convolutional Neural Network

Consider the notebook `CIFAR_10_for_Assignment.ipynb` available on Absalon. It applies a convolutional neural network (CNN) to the CIFAR-10 image classification task, where real-world images have to be assigned to one of ten classes [4], see Fig. 1. The notebook is supposed to run with GPU support, for example, using *Colaboratory*. Even then, executing it will take some time. Thus, start with experimental work in good time before the deadline.

1. **(25 pts.)** Report the number of *trainable* parameters of the model. Explain the number of parameters for each layer.

   The batch normalization layer has trainable parameters that were not explained in the lecture. Explain these trainable parameters in the report. To find out about the

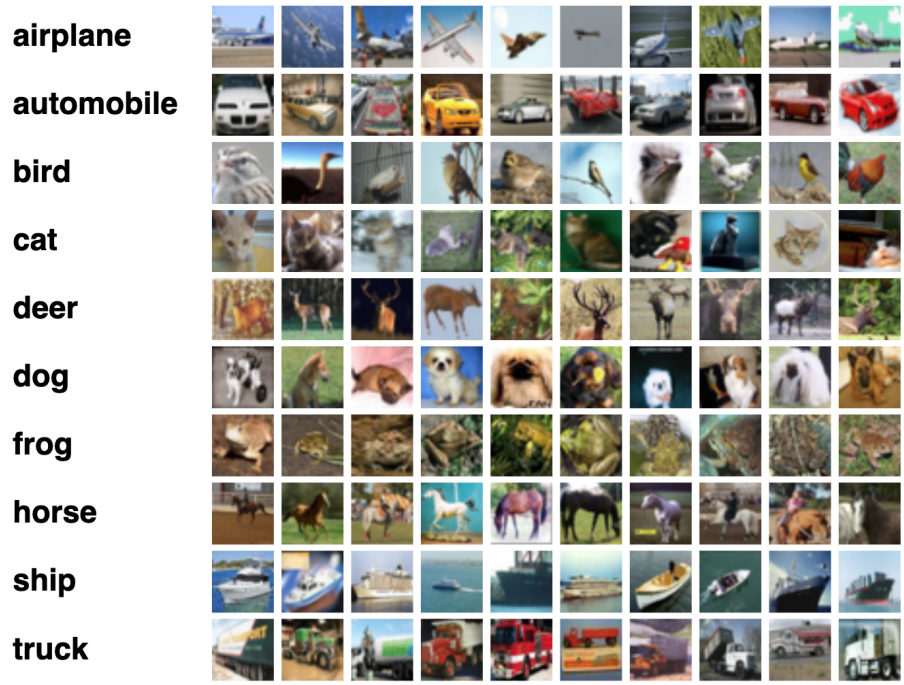

Figure 1: Ten random examples from CIFAR-10 for each of the ten classes.

trainable parameters, you are supposed to read the original paper "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [3], which you can find online.[1] You can ignore the non-trainable parameters.

2. **(10 pts.)** Convolutional neural networks are highly complex models. To reduce the risk of overfitting and in order to be able to learn difficult tasks with a high input variability, many training examples are needed. *Data augmentation* is a way to enlarge the training data set. The training data set is extended by artificial input generated from the available data. For example, a new training image can be generated by rotating, flipping, and/or shifting an available inout image. In the context of neural networks, this type of data augmentation can be traced back to [1]. It has been used successfully for CNNs, for instance already in the influential work by [5]

   Inspect the notebook `CIFAR 10 for Assignment.ipynb`. Which transformations are applied to the input images?

   To understand the flipping operation, you could have a look at the notebook `Keras Flipping.ipynb` available on Absalon. Would adding `vertical_flip=True` be helpful for CIFAR-10? Give a reason.

3. **(20 pts.)** Models and layers in Keras have a Boolean flag that can put them in training and testing mode by specifying `training=True` and `training=False`, respectively. The higher level functions, such as `tf.keras.Model.fit`, set these flags automatically.

   Please explain why two different modes are necessary

   (a) when using dropout (see the slides or section 7.12 in `https://www.deeplearningbook.org/contents/regularization.html` [2]), and

   (b) when using batch normalization [3] .

[1] Being able to extract information from an original deep learning research paper and official online documentation are learning goals of this assignment.

4. **(10 pts.)** In the notebook, the test data is used as validation/development data and the neural network is evaluated on these data during training. It allows to assess the performance when tuning the network architecture and the learning process. When doing so, is the performance on the test data an unbiased estimator of the generalization performance of the classifier? Give a reason.

5. **(optional)** Can you get a better performance by tuning the network architecture and the learning process? For example, you could try

   (a) to employ weight decay by adding `kernel_regularizer=tf.keras.regularizers.l2(weight_decay_parameter)` to layer definitions,

   (b) to add dropout by adding dropout layers, `model.add(tf.keras.layers.Dropout(dropout_rate)`,

   (c) to add neurons, layers or connections, and/or

   (d) to change the optimizer and its hyperparameters,

   Training a neural network is a random process (for several reasons – which?). Thus, single trials are not sufficient to establish that a modification generally improves the learning for a given task. Proper statistical evaluation is required. The least you can do is to repeat the training of the baseline and your alternative system a couple of times and compare the mean/median performance of the approaches.

   A validation accuracy over 90 % can be considered as a good result.

   If you find a solution that does not train considerably longer than the one in the notebook (say, it may take not more than 20 % more time if executed in *Colaboratory*) and reaches *consistently* more than 90 %, please submit your code. The winner will be announced, and her/his notebook will serve as the baseline for next year.

References

[1] H. S. Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[4] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.