Image classification on CIFAR-10

using basic Feed-Forward Neural Networks

# Digit recognition example (lecture)

The example provided in the course material is a very simple neural network [NN], trained to recognize handwritten digits. The MNIST database is a subset of the much larger Special Databases 1 and 3 provided by NIST (The National Institute for Standards and Technology).

This example will be explained in detail in the following sections, starting with the tools that are being used. The code, copied from the lecture slides, is presented, and then explained.

## Requirements

This section discusses the tools and resources necessary to create the example NN. The contents of this section also apply to later sections discussing the creation of a simple NN using the same tools.

### Keras

The example uses the Keras API, a tool with the focus to let researchers quickly build deep learning models for fast experimentation.

Keras uses layers and models in an object-oriented manner, where the model is the object which contains the layers. Keras provides simple and more complex models to fit the needs of the researcher. Apart from the layers, which have their own set of parameters, it is also possible to adjust optimizer, loss function and hyper parameters.

Keras is a high-level API of TensorFlow 2 and can be run on many kinds of systems, from GPU clusters to browser or mobile devices.[[1]](#footnote-1)

### TensorFlow (2)

TensorFlow is a machine learning platform, developed and maintained by Google employees, and open source. It provides the infrastructure to calculate with multi-dimensional arrays, which are also called tensors. It automatically finds the gradient of any differentiable array. In contrast to other array calculation APIs, TensorFlow can take advantage of GPUs and distribute calculations over multiple machines.[[2]](#footnote-2)[[3]](#footnote-3)

### MNIST

As described in the beginning, MNIST is a database containing handwritten digits. It is a good benchmark for the performance of a machine learning algorithm (in OCR) and is small enough to be also manageable by weaker machines.

The images of the digits are 28 by 28 pixels, each having a grey scale value of 8 bit. This constitutes a 3-dimensial vector, two dimensions for the position of the pixel and one for the greyscale vector.

There are 70,000 of these images in the dataset, 60,000 of them are the training set and 10,000 are the test set. The writing style of the digits varies widely from clean office writing to messy writing of high-school members.[[4]](#footnote-4)

## Description

This section is meant to describe the programming aspect of the example exercise.

1. import tensorflow as tf
2. from tensorflow.keras import models
3. from tensorflow.keras import layers
4. from tensorflow.keras.datasets import mnist

The lines 1 to 4 ..

1. (train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()
2. network = models.Sequential() # linear stack of layers
3. network.add(layers.Dense(512, activation='relu', input\_shape=(28 \* 28,)))
4. network.add(layers.Dense(10, activation='softmax')) # digit probability
5. network.compile(optimizer='rmsprop', loss='categorical\_crossentropy',metrics=['accuracy'])
6. train\_images = train\_images.reshape((60000, 28 \* 28))
7. train\_images = train\_images.astype('float32') / 255
8. test\_images = test\_images.reshape((10000, 28 \* 28))
9. test\_images = test\_images.astype('float32') / 255
10. from tensorflow.keras.utils import to\_categorical
11. train\_labels = to\_categorical(train\_labels)
12. test\_labels = to\_categorical(test\_labels)
13. network.fit(train\_images, train\_labels, epochs=5, batch\_size=128)
14. test\_loss, test\_acc = network.evaluate(test\_images, test\_labels)
15. print(network.summary())

## Explanation

# The Cifar-10 dataset

## Description

## Loading and inspection

## (possible) Preprocessing

## Statistics and Graphs

# Implementation of basic NN

## Model design

* 2 categories
* 1 in, 1 out
* extendable

## Evaluation of first results

# Adaptations

## 5 Categories

## 10 Categories

## Evaluation of performances

# Improvements

## Adding and changing Layers

## Hyperparameter adjustments

## Explanation of most note-worthy improvements

# Final Performance evaluation

## 2 Categories

## 5 Categories

## 10 Categories

.

# References

# Annex

1. About Keras, <https://keras.io/about/> [↑](#footnote-ref-1)
2. <https://www.tensorflow.org/about> [↑](#footnote-ref-2)
3. <https://keras.io/getting_started/intro_to_keras_for_researchers/> [↑](#footnote-ref-3)
4. <http://yann.lecun.com/exdb/mnist/> [↑](#footnote-ref-4)