ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

Machine Learning on Microcontrollers

227-0155-00G

Exercise 1

Conda environments and blinking LEDs

Michele Magno, PhD
Marco Giordano
Pietro Bonazzi

1 Introduction

Welcome to the laboratory of Machine Learning on Microcontrollers.

In this exercise, you will learn to install Anaconda and to create a conda environment that we will use in our future exercises.

2 Notation

Student Task: Parts of the exercise that require you to complete a task will be explained in a shaded box like this.

Note: You find notes and remarks in boxes like this one.

3 Preparation

Python is an extremely agile programming language for creating both programs and scripts, and it comes with a myriad of packages that allow you to do almost everything. Packages boost your productivity enriching the programming interface with new abstracted features, and let you focus on what is important.

Developers write packages using different programming languages, and they (luckily) update them on a regular basis. This variability creates non-trivial challenges to who works on data science projects (e.g. Machine Learning) and needs a smooth interaction between different versions of different packages.

A possible solution to this problem is to use a Package Manager to *automagically* install and keep track of the status of the available packages. There is more than one valid Package Manager out there, but during this exercise we will focus on the Python ecosystem Anaconda.

4 Anaconda

Anaconda is an open-source Python distribution for scientific tasks that comes with hundreds of already-installed packages, a Graphical User Interface (GUI), and the Conda engine to manage its packages.

Some Package Managers help the user in finding new packages, installing them and maintaining the already installed ones; Conda, in addition, takes also care of the inter-dependencies between the various packages, avoiding to break previously working environments, and allows to maintain different environments for different applications and needs.

A Python environment is an isolated set of Python packages that exist locally within that precise environment. For example, if you are working on two different projects that require different versions of Python and different versions of a package, you can create two different environments. Each environment keeps information about its own Python version, its own package references, etc., and it is possible to switch from one environment to the other depending on the project you are working on.

Working with environments improves code repeatability and reproducibility, and reduces the conflicts that can arise due to the intrinsic nature of the packages.

4.1 Installation

Even if in this exercise we will install Anaconda on a Linux system, you can install it also on Windows and macOS [1].

First of all, download the main installer from Anaconda's official website:

`https://www.anaconda.com/products/individual`

To do so, follow the link and, under *Linux*, select *"64-Bit (x86) Installer (550 MB)"*.

Open a new terminal and change the working directory to the one in which you downloaded the installer. If this folder is called INSTALLER_DIRECTORY, type:

```
$ cd INSTALLER_DIRECTORY
```

Before running the installer, we need to add executable permissions with the command *chmod*.

```
$ chmod +x ./Anaconda3-2020.07-Linux-x86_64.sh
```

Now, we can install Anaconda executing the installer and following the instructions prompted on the screen.

```
$ ./Anaconda3-2020.07-Linux-x86_64.sh
```

Be sure to install Anaconda in a directory in which you have write accesses.

When asked `Do you wish the installer to initialize Anaconda3 by running conda init?` Type `Yes`. This will put the Anaconda-related commands for your shell in the `~/.bashrc` file. Alternatively you can run `conda init` yourself from the command line.

4.2 Walkthrough

Once the installation is over, close the current terminal window and open a new one to update the shell's internal variables. You can launch Anaconda Navigator, the GUI of Anaconda, with the command [2]:

```
$ anaconda-navigator
```

After a while, you will see the main window appear. On the left, we have the main panel with four different sub-menus:

- **Home** Useful applications that can be installed and managed by Anaconda.
- **Environments** You can create and manage different environments, each with different packages of different versions.
- **Learning** Learning material, with tutorials and references about Python and its ecosystem.
- **Community** References to community events, forums, etc.

[1] For more information, visit Anaconda documentation `https://docs.anaconda.com/anaconda/install/`

[2] On certain systems, an UnboundLocalError occurs when launching Anaconda Navigator. In this case, run `$ conda update anaconda-navigator` before opening the GUI

From `Environments→Create` it's possible to create a new environment, linked to a specific Python version. Then, selecting the new environment, it will be possible to manage the related packages.

Everything you see on Anaconda Navigator can be accessed also via a terminal. For example, you can create a new empty environment typing:

```
$ conda create -n ENV_NAME
```

You can also initialize the environment with specific packages and versions. The following command creates a new environment called ENV_NAME, in which you can use Python 3.8 with PKG_0 (version 3.4) and PKG_1.

```
$ conda create -n ENV_NAME python=3.8 PKG_0=3.4 PKG_1
```

A more reliable way to create environments is to use environment .yml files. If you are interested, you can look at the documentation `https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html` to know more about this. Keeping environmental information on a file make the sharing of the environment with other people easier.

When you are happy with your environments, you can switch to one of them by typing:

```
$ conda activate ENV_NAME
```

As you can see, the name of the current environment is displayed on the left of the shell's active line. To list all the available environments and see which is active if it's not clear, type:

```
$ conda env list
```

The active environment is marked with a star. If you want to leave the current environment, just execute:

```
$ conda deactivate
```

Installing further packages to your conda environment can be done using Anaconda Navigator, under `Home→Install`, or from the command line using:

```
$ conda install PKG_NAME
```

A version specifier can be added by simply writing `=VERSION` directly after the package name.

If conda does not support the package or version you want, pip can also be used. To avoid conflicts between environments and to ensure that the environment's pip is used, instead of the system's pip, we suggest you use: `/path_to_env/bin/pip install package_name`. Additionally, other channels can be added to conda to support more packages. To do this in the command line, simply type:

```
$ conda config --add channels CHANNEL_NAME
```

Student Task 1 (Conda Setup): Using the instructions above, install anaconda for your operating system (if not already installed).

1. Create a new conda environment with python 3.8. Call this environment `mfcc_test` and activate it.
2. Add the channel `conda-forge` to your anaconda installation
3. Install the packages `jupyter` and `librosa` in the `mfcc_test` environment.

5 STM32 CUBE IDE

In the following weeks you will learn how to configure and program the STM Microcontroller (MCU) that you will use throughout the first half of this course. The development board for this course is the B-L475E-IOT01A2 featuring an STM32L475VG low-power MCU with a maximum clock frequency of 80 MHz, 128 KB of RAM and 1 MB of flash memory. The processor core is based on the ARM Cortex M4F, featuring advanced Digital Signal Processor (DSP) operation capabilities and a floating-point unit.

Today, we will simply test the integrity of our MCU and verify the correctness of the STM32 CUBE IDE installation.

Student Task 2 (Blinking LED):

1. Download the Blinky project available on thelecture page.
2. Import the project in the STM32 CUBE IDE and program the board. What is the behaviour of the MCU?

$\mathcal{E}$ **Congratulations! You have reached the end of the exercise.**
If you are unsure of your results, discuss with an assistant. $\mathcal{E}$