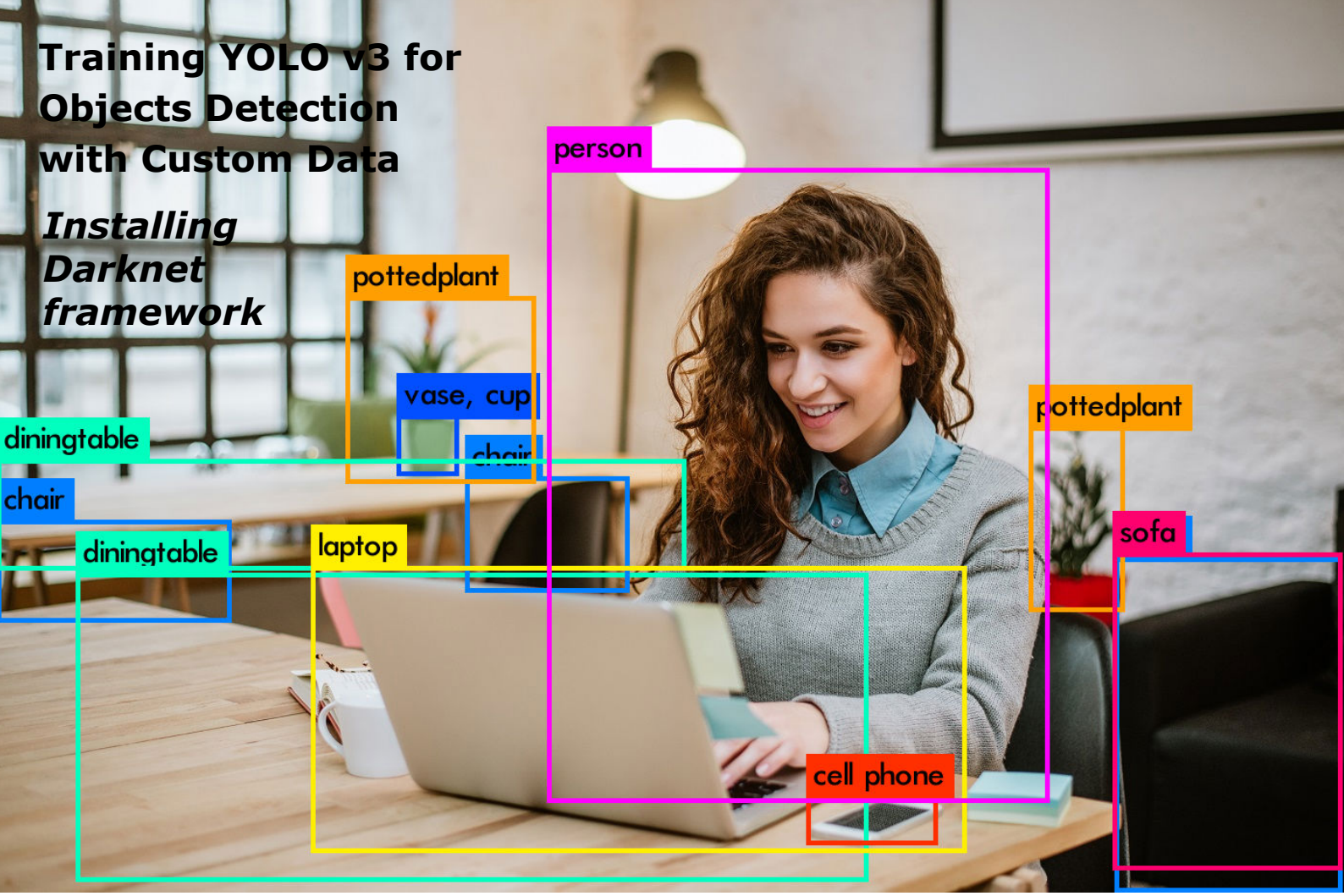
Training YOLO v3 for Objects Detection with Custom Data

Installing Darknet framework



Installing Darknet framework

Let's consider some examples on how to install *Darknet framework*, namely, the most popular fork *AlexeyAB*, that also supports *Windows* and has huge community to find answers from. If you encounter any issues with installation, let's discuss them in *Question & Answer* board. Together with course-mates we will find solution. Try also to find answer by using links at the end of this *pdf* in *Useful Links* section. Somebody else might already encountered the same issue and solution might be described there.

General Algorithm

In most of the cases the general approach is as following or quite similar:

- Install prerequisites (*OpenCV, CUDA, cuDNN, etc.*)
- Clone repository
- Adjust options in *Makefile*
- Compile it

Linux

Instructions on how to install *Darknet* on *Linux*.

Base CPU system on Linux

Installing Basic option of *Darknet framework* to use it with *CPU* on Linux consists of the steps described below.

- **Clone repository.** The first step to implement is *cloning repository*. Activate your *Python v3* environment, navigate to the desired directory and run following command in *Terminal*:
```
git clone https://github.com/AlexeyAB/darknet.git
```
If you don't have *git* been installed, run following command in *Terminal*:
```
conda install git
```

- **Compile Darknet framework.** Navigate to the root directory of *Darknet framework* by using following command in *Terminal*:
```
cd darknet
```
Compile *Darknet framework* by using following command in *Terminal*:
```
make
```

- **Verify successful installation** by using following command in *Terminal*:
```
./darknet
```
As a respond, following line should appear:
```
usage: ./darknet <function>
```

Base CPU system on Linux + OpenCV

Installing Basic option of *Darknet framework* to use it with *CPU* on Linux along with OpenCV consists of the steps described below. OpenCV has to be installed beforehand (Section-1 of the course).

- **Clone repository.** The first step to implement is *cloning repository*. Activate your *Python v3* environment, navigate to the desired directory and run following command in *Terminal*:
```
git clone https://github.com/AlexeyAB/darknet.git
```
If you don't have *git* been installed, run following command in *Terminal*:
```
conda install git
```

- **Change flag in Makefile.** Navigate to the root directory of *Darknet framework* and open file *Makefile* in any editor like *notepad* or any other. Change flag in the following line from *0* to *1*:
```
OPENCV=1
```

- **Compile Darknet framework.** Navigate to the root directory of *Darknet* framework by using following command in *Terminal*:
```
cd darknet
```
Compile *Darknet framework* by using following command in *Terminal*:
```
make
```

- **Verify successful installation** by using following command in *Terminal*:
  `./darknet`

  As a respond, following line should appear:
  `usage: ./darknet <function>`

GPU system on Linux
Installing option of *Darknet framework* to use it with *GPU* on Linux.

- **CMake installation.** Install *CMake* for modern *CUDA* support by following instructions described on official resource here.

- **CUDA 10.0 installation**
  - **Pre-installation actions.** Implement following instructions described on official *nvidia guide* here. By doing these steps, you will:
    - Verify the system has a *CUDA-capable GPU*
    - Verify the system is running a supported version of *Linux*
    - Verify the system has *gcc* installed
    - Verify the system has the correct kernel headers and development packages installed
  - **Download** the *NVIDIA CUDA Toolkit version 10.0* from archive here
  - **Install *CUDA Toolkit version 10.0*** by following one of the instructions:
    - *Package Manager Installation* here
    - *Runfile Installation* here. For instance, for *runfile* option, sequence of commands to run in *Terminal* looks like following or quite similar:
      ```
      sudo apt-get update
      sudo apt-get upgrade
      sudo sh cuda_10.0.130_410.48_linux.run
      ```
  - **Post-installation actions.** Implement actions that have to be taken after the installation before the *CUDA Toolkit 10.0* can be used. They are described on official *nvidia guide* here.
  - Check out *Frequently Asked Questions* described on official *nvidia guide* here.

- **cuDNN 7.4.1 installation** – *GPU-accelerated library* of for deep neural networks.
  - **Download archive** of *cuDNN version 7.4.1* for *CUDA 10.0 version* (might need to try different versions like 7.3, 7.4, etc.) from official resource here.
  - **Run installation** by following instructions described on official resource here.

- **OpenCV installation.** Till this point we have to have OpenCV been installed. Anyway, check options on how to install it from Section-1 of the course.

- **Clone repository.** Navigate to the desired directory and run following command in *Terminal*:
  ```
  git clone https://github.com/AlexeyAB/darknet.git
  ```

  If you don't have *git* been installed, run following command in *Terminal*:
  ```
  conda install git
  ```

- **Change flags in Makefile.** Navigate to the root directory of *Darknet framework* and open file *Makefile* in any editor like *notepad* or any other. Change following flags from *0* to *1*:
  ```
  GPU=1
  CUDNN=1
  OPENCV=1
  ```

- **Compile Darknet framework.** Navigate to the root directory of *Darknet* framework by using following command in *Terminal*:
  ```
  cd darknet
  ```

  Compile *Darknet framework* by using following command in *Terminal*:
  ```
  make
  ```

- **Verify successful installation** by using following command in *Terminal*:
  ```
  ./darknet
  ```

  As a respond, following line should appear:
  ```
  usage: ./darknet <function>
  ```

Windows

Instructions on how to install *Darknet* on *Windows*.

Base CPU system on Windows + OpenCV

Installing Basic option of *Darknet framework* to use it with *CPU* on Windows along with OpenCV consists of the steps described below.

- **OpenCV 3.3.0 installation.** In order to use *Darknet framework* on Windows it is needed to install *OpenCV version 3.3.0*.
  - **Download installation file** from official resource here and extract it to the folder *C:\opencv_3.0*

- **Create system variable** with name *OpenCV_DIR* and path to where extracted folders *include* and *x64* are located. To do so:
  - in the *File Explorer* right click on *This PC* and open *Advanced system settings*; choose *Environment Variables*
  - click *New* in *System variables*
  - create new variable with name *OpenCV_DIR*

    and path *C:\opencv_3.0\opencv\build*
  - click *Ok*

- **Visual Studio installation.** In order to build *Darknet framework* on Windows it is needed to install *Visual Studio*.
  - **Download installation file** of free, community version *Visual Studio* from official resource here, double-click on it and follow prompts

- **Clone repository.** Navigate to the desired directory and run following command in *Anaconda Prompt*:

  ```
  git clone https://github.com/AlexeyAB/darknet.git
  ```

  If you don't have *git* been installed, run following command in *Anaconda Prompt*:

  ```
  conda install git
  ```

- **Compile Darknet framework.** By using installed *Visual Studio* open following file in the directory *darknet\build\darknet*:

  ```
  darknet_no_gpu.sln
  ```

  Then, in *Visual Studio* drop-down menus set following:

  ```
  x64
  ```

  and

  ```
  Release
  ```

  Finally, choose *Build* and *Build darknet_no_gpu*

  Find executable file *darknet.exe* created in the directory *\build\darknet\x64*

- **Verify successful installation.** Navigate to the directory with executable file *\build\darknet\x64* and type following command in *command line or Anaconda Prompt*:

  ```
  darknet.exe
  ```

GPU system on Windows

Installing option of *Darknet framework* to use it with *GPU* on Windows.

- **Clone repository.** Navigate to the desired directory and run following command in *Anaconda Prompt*:

  ```
  git clone https://github.com/AlexeyAB/darknet.git
  ```

  If you don't have *git* been installed, run following command in *Anaconda Prompt*:

  ```
  conda install git
  ```

- **OpenCV 3.3.0 installation.** In order to use *Darknet framework* on Windows it is needed to install *OpenCV version 3.3.0*.
  - **Download installation file** from official resource here and extract it to the folder *C:\opencv_3.0*
  - **Create system variable** with name *OpenCV_DIR* and path to where extracted folders *include* and *x64* are located. To do so:
    - in the *File Explorer* right click on *This PC* and open *Advanced system settings*; choose *Environment Variables*
    - click *New* in *System variables*
    - create new variable with name *OpenCV_DIR*

      and path *C:\opencv_3.0\opencv\build*
    - click *Ok*
  - **Find following files** in the folder *C:\opencv_3.0\opencv\build\x64\vc14\bin:*

    ```
    opencv_world330.dll
    ```

    ```
    opencv_ffmpeg330_64.dll
    ```
  - **Copy paste** these two files in the directory *darknet\build\darknet\x64* where executable file *darknet.exe* will appear after compilation

- **Visual Studio installation.** In order to build *Darknet framework* on Windows it is needed to install *Visual Studio*. Pay attention that *Visual Studio* has to be installed before *CUDA.*
  - **Download installation file** of free, community version *Visual Studio* from official resource here, double-click on it and follow prompts.

- **CUDA 10.0 installation**
  - **Download** *NVIDIA CUDA Toolkit version 10.0* from archive here
  - **Install *CUDA Toolkit version 10.0*** by executing the *CUDA* installer and following the on-screen prompts. Find also installation instructions on official *nvidia guide* here.

- **cuDNN 7.4.1 installation** – *GPU-accelerated library* of for deep neural networks.
  - **Download archive** of *cuDNN version 7.4.1* for *CUDA 10.0 version* from official resource here.
  - **Install cuDNN** by following instructions on official *nvidia guide* here.
  - **Create system variable** with name *CUDNN* and path to where *cuDNN* was installed. To do so:
    - in the *File Explorer* right click on *This PC* and open *Advanced system settings*; choose *Environment Variables*
    - click *New* in *System variables*
    - create new variable with name *CUDNN*

      and path where *cuDNN* was installed, like following or quite similar: *C:\Program Files\NVIDIA GPU Computing Toolkit*
    - click *Ok*
  - **Find following file:**
    `cudnn64_7.dll`
  - **Copy paste** this file in the directory *darknet\build\darknet\x64* where executable file *darknet.exe* will appear after compilation

- **Compile Darknet framework.** By using installed *Visual Studio* open following file in the directory *darknet\build\darknet*:
  `darknet.sln`

  Then, in *Visual Studio* drop-down menus set following:
  `x64`
  and
  `Release`

  Finally, choose *Build* and *Build darknet*

  Find executable file *darknet.exe* created in the directory *\build\darknet\x64*

- **Verify successful installation.** Navigate to the directory with executable file *\build\darknet\x64* and type following command in *command line or Anaconda Prompt*:
  `darknet.exe`

MacOS

Instructions on how to install *Darknet* on *MacOS*.

Base CPU system on MacOS + OpenCV

Installing Basic option of *Darknet framework* to use it with *CPU* on MacOS along with OpenCV consists of the steps described below.

- **CMake installation.** Install *CMake* for by following instructions described on official resource here.

- **OpenCV installation.** One of the simplest way to install *OpenCV* on Mac is with Homebrew. If you don't have Homebrew installed, use following command in *Terminal* (zoom in and copy-paste):

  ```
  /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
  ```

  Then, install OpenCV by following command in *Terminal*:

  ```
  brew install opencv
  ```

- **Clone repository.** The first step to implement is *cloning repository*. Activate your *Python v3* environment, navigate to the desired directory and run following command in *Terminal*:

  ```
  git clone https://github.com/AlexeyAB/darknet.git
  ```

  If you don't have *git* been installed, run following command in *Terminal*:

  ```
  conda install git
  ```

- **Change flag in Makefile.** Navigate to the root directory of *Darknet framework* and open file *Makefile* in any editor like *notepad* or any other. Change flag in the following line from *0* to *1*:

  ```
  OPENCV=1
  ```

  Also, change in *Makefile* line 139:

  ```
  $(CPP) -std=c++11 $(COMMON) $(CFLAGS) -o $@ src/yolo_console_dll.cpp $(LDFLAGS) -L ./ -l:$(LIBNAMESO)
  ```

  this `-l:$(LIBNAMESO)`
  to this `-l$(EXEC)`

- **Compile Darknet framework.** Navigate to the root directory of *Darknet* framework by using following command in *Terminal*:

  ```
  cd darknet
  ```

  Compile *Darknet framework* by using following command in *Terminal*:

  ```
  ./build.sh
  ```

- **Verify successful installation** by using following command in *Terminal*:
  ```
  ./darknet
  ```

  As a respond, following line should appear:
  ```
  usage: ./darknet <function>
  ```

GPU system on MacOS
Installing option of *Darknet framework* to use it with *GPU* on MacOS.

- **CMake installation.** Install *CMake* for by following instructions described on official resource here.

- **OpenCV installation.** One of the simplest way to install *OpenCV* on Mac is with Homebrew. If you don't have Homebrew installed, use following command in *Terminal* (zoom in and copy-paste):
  ```
  /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
  ```

  Then, install OpenCV by following command in *Terminal*:
  ```
  brew install opencv
  ```

- **CUDA 10.0 installation**
  - **Download** *NVIDIA CUDA Toolkit version 10.0* from archive here
  - **Install *CUDA Toolkit version 10.0*** by following installation instructions on official *nvidia guide* here.

- **cuDNN 7.4.1 installation** – *GPU-accelerated library* of for deep neural networks.
  - **Download archive** of *cuDNN version 7.4.1* for *CUDA 10.0 version* from official resource here.
  - **Install cuDNN** by following instructions on official *nvidia guide* here.

- **Clone repository.** The first step to implement is *cloning repository*. Activate your *Python v3* environment, navigate to the desired directory and run following command in *Terminal*:
  ```
  git clone https://github.com/AlexeyAB/darknet.git
  ```

  If you don't have *git* been installed, run following command in *Terminal*:
  ```
  conda install git
  ```

- **Change flag in Makefile.** Navigate to the root directory of *Darknet framework* and open file *Makefile* in any editor like *notepad* or any other. Change flag in the following line from *0* to *1*:
  ```
  GPU=1
  CUDNN=1
  OPENCV=1
  ```

Also, change in *Makefile* line 139:

```
$(CPP) -std=c++11 $(COMMON) $(CFLAGS) -o $@ src/yolo_console_dll.cpp $(LDFLAGS) -L ./ -l:$(LIBNAMESO)
```

this `-l:$(LIBNAMESO)`

to this `-l$(EXEC)`

- **Compile Darknet framework.** Navigate to the root directory of *Darknet* framework by using following command in *Terminal*:

  `cd darknet`

  Compile *Darknet framework* by using following command in *Terminal*:

  `./build.sh`

- **Verify successful installation** by using following command in *Terminal*:

  `./darknet`

  As a respond, following line should appear:

  `usage: ./darknet <function>`

Useful Links

Check out these links with official resources for installing and using *Darknet framework* as well as the links to *possible issues*:

[1] Fork of AlexeyAB – the most popular fork of *Darknet framework* **chosen for this course**, that has improvements on performance, answers on the popular issues

[2] Issues - try to find answer to encountered issue from more than 2000 community posts on how to install and use *Darknet framework*

[3] Issue #500 – describes which version of *OpenCV* is needed for installing *Darknet framework* on Windows

[4] Issue #2983 – describes needed changes in *Makefile* needed for installing *Darknet framework* on MacOS

[5] CUDA Toolkit – installation guide with step-by-step instructions on how to install and verify *CUDA* on Linux, Windows and MacOS

[6] NVIDIA CUDA Deep Neural Network library (cuDNN) – installation guide with step-by-step instructions on how to install and check cuDNN on Linux, Windows and MacOS