

## Roteiro para Detecção de Objetos

### Instalação API TensorFlow

#### -----TensorFlow-GPU-----

A primeira coisa pra instalar é o TensorFlow, claro. Sem ele não tem detecção.

#### Step 1: Install NVIDIA CUDA 10.0

Remove previous cuda installation (if you installed cuda before):

```
sudo apt-get purge nvidia*  
sudo apt-get autoremove  
sudo apt-get autoclean  
sudo rm -rf /usr/local/cuda*
```

Add key and download:

```
sudo apt-key adv --fetch-  
keys http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86\_64/7fa2af80.pub  
echo  
"deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86\_64/ /" | sudo tee /etc/apt/sources.list.d/cuda.list
```

Install CUDA-10.0:

```
sudo apt-get update  
sudo apt-get -o Dpkg::Options::="--force-overwrite" install  
cuda-10-0 cuda-drivers
```

Reboot and type:

```
echo 'export PATH=/usr/local/cuda-10.0/bin${PATH:+:${PATH}}' >>  
~/.bashrc  
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-  
10.0/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}' >> ~/.bashrc  
source ~/.bashrc  
sudo ldconfig
```

For check if install was successful: after executing next command you need to see version of your nvidia-drivers and GPU: `nvidia-smi`

```
dmitriy@dmitriy-PC: ~
Файл Правка Вид Поиск Терминал Справка
dmitriy@dmitriy-PC:~$ nvidia-smi
Fri Mar 15 10:55:16 2019

+-----+
| NVIDIA-SMI 410.104      Driver Version: 410.104      CUDA Version: 10.0      |
+-----+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
|  0  GeForce GTX 106...    On          | 00000000:01:00.0 On |          N/A         |
| 31%   31C    P8           9W / 120W | 329MiB / 6075MiB |      0%      Default |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type    Process name                       Usage    |
|=====+=====+
|  0         1383    G      /usr/lib/xorg/Xorg                          147MiB |
|  0         1553    G      /usr/bin/gnome-shell                       135MiB |
|  0         2131    G      ...-token=93C69DF92624440FE9A5C4A499D294BA  44MiB   |
+-----+

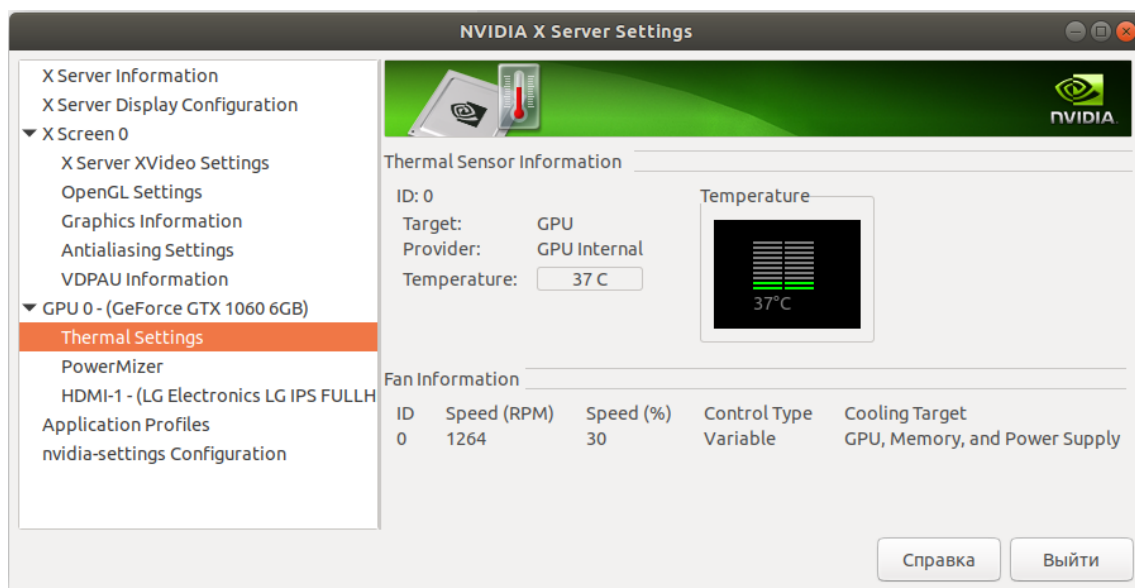
dmitriy@dmitriy-PC:~$
```

Output for `nvidia-smi` command

If you have low screen resolution, fix this with Xorg: `sudo nvidia-xconfig`

If this has not helped, check one of my previous installation (I have described in detail what should help if problems remain).

Also, don't forget to check `nvidia-settings` — here you can find out how much GPU is loaded (for example, if trained neuralnets using ML framework): `nvidia-settings`



This tab is the most useful, for my opinion

## Step 2: Install cuDNN 7.5.0

Go [here](#) and click Download CuDNN. Log in and accept the required agreement. Click the following: “Download cuDNN v7.5.0 (Feb 21, 2019), for CUDA 10.0” and then “cuDNN Library for Linux”.

### cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

[Download cuDNN v7.5.0 \[Feb 25, 2019\], for CUDA 10.1](#)

[Download cuDNN v7.5.0 \[Feb 21, 2019\], for CUDA 10.0](#)

#### Library for Windows, Mac, Linux, Ubuntu and RedHat/Centos(x86\_64 architecture)

[cuDNN Library for Windows 7](#)

[cuDNN Library for Windows 10](#)

[cuDNN Library for Linux](#)

[cuDNN Library for OSX](#)

[cuDNN Runtime Library for Ubuntu18.04 \[Deb\]](#)

[cuDNN Developer Library for Ubuntu18.04 \[Deb\]](#)

[cuDNN Code Samples and User Guide for Ubuntu18.04 \[Deb\]](#)

Download tgz from here

Then

install:

```
tar -xf cudnn-10.0-linux-x64-v7.5.0.56.tgz
sudo cp -R cuda/include/* /usr/local/cuda-10.0/include
sudo cp -R cuda/lib64/* /usr/local/cuda-10.0/lib64
```

## Step 3: Install Dependencies

Install libcupti:

```
sudo apt-get install libcupti-dev
echo 'export
LD_LIBRARY_PATH=/usr/local/cuda/extras/CUPTI/lib64:$LD_LIBRARY_PATH' >> ~/.bashrc
```

Python related:

```
sudo apt-get install python3-numpy python3-dev python3-pip
python3-wheel
```

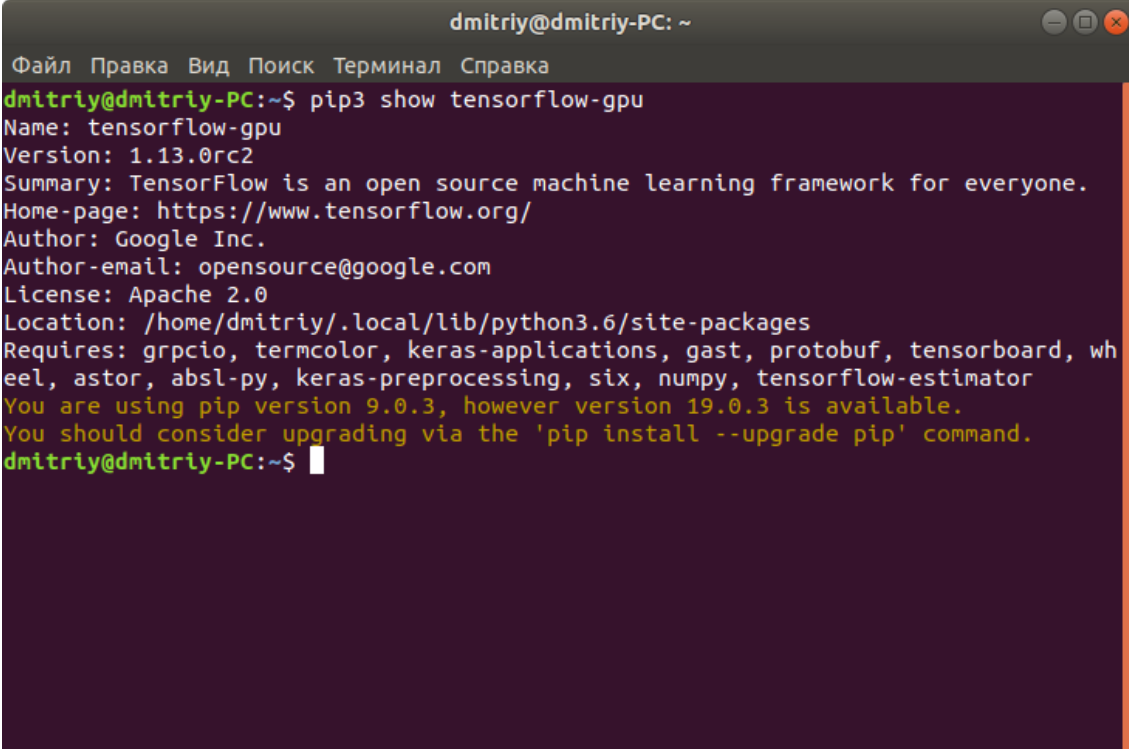
## Step 4: Install Tensorflow-GPU

Install Tensorflow-GPU 1.13 using pip:

```
sudo pip3 install tensorflow-gpu==1.13.1
```

Now you can check which tensorflow version you install:

```
pip3 show tensorflow-gpu
```

A terminal window titled 'dmitriy@dmitriy-PC: ~' with a menu bar in Russian (Файл, Правка, Вид, Поиск, Терминал, Справка). The command 'pip3 show tensorflow-gpu' has been executed, resulting in the following output: Name: tensorflow-gpu, Version: 1.13.0rc2, Summary: TensorFlow is an open source machine learning framework for everyone., Home-page: https://www.tensorflow.org/, Author: Google Inc., Author-email: opensource@google.com, License: Apache 2.0, Location: /home/dmitriy/.local/lib/python3.6/site-packages, Requires: grpcio, termcolor, keras-applications, gast, protobuf, tensorboard, wheel, astor, absl-py, keras-preprocessing, six, numpy, tensorflow-estimator. A yellow warning message states: 'You are using pip version 9.0.3, however version 19.0.3 is available. You should consider upgrading via the 'pip install --upgrade pip' command.' The prompt 'dmitriy@dmitriy-PC:~\$' is visible at the bottom.

```
dmitriy@dmitriy-PC:~$ pip3 show tensorflow-gpu
Name: tensorflow-gpu
Version: 1.13.0rc2
Summary: TensorFlow is an open source machine learning framework for everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: opensource@google.com
License: Apache 2.0
Location: /home/dmitriy/.local/lib/python3.6/site-packages
Requires: grpcio, termcolor, keras-applications, gast, protobuf, tensorboard, wh
eel, astor, absl-py, keras-preprocessing, six, numpy, tensorflow-estimator
You are using pip version 9.0.3, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
dmitriy@dmitriy-PC:~$
```

Yep! You are ready for using GPU!

FONTE: <https://medium.com/better-programming/install-tensorflow-1-13-on-ubuntu-18-04-with-gpu-support-239b36d29070>

É importante dizer que você só deve usar esse guia para instalação em máquina local, quando você não está preocupado com instalações de outras pessoas.

## **----TensorFlow SEM GPU ----**

Para instalar o TensorFlow em uma máquina sem GPU use o comando "sudo pip install tensorflow==1.13.1"

## TensorFlow Models

O próximo passo é instalar o TensorFlow models, repositório que contém tudo que precisamos para conseguir treinar os nossos modelos.

Primeiro você deve clonar o repositório models com o comando:

```
git clone https://github.com/tensorflow/models.git
```

Depois de clonado acesse o diretório models/research dentro do repositório clonado.

Estando nesse diretório você pode seguir o guia de instalação a seguir:

### Installation

#### Dependencies

Tensorflow Object Detection API depends on the following libraries:

- Protobuf 3.0.0
- Python-tk
- Pillow 1.0
- lxml
- tf Slim (which is included in the "tensorflow/models/research/" checkout)
- Jupyter notebook
- Matplotlib
- Tensorflow (>=1.9.0)
- Cython
- contextlib2
- Cocoapi

For detailed steps to install Tensorflow, follow the [Tensorflow installation instructions](#). A typical user can install Tensorflow using one of the following commands:

# For CPU

```
pip install tensorflow
```

# For GPU

```
pip install tensorflow-gpu
```

The remaining libraries can be installed on Ubuntu 16.04 using via apt-get:

```
sudo apt-get install protobuf-compiler python-pil python-lxml python-tk
```

```
pip install Cython
```

```
pip install contextlib2
```

```
pip install jupyter
```

```
pip install matplotlib
```

Alternatively, users can install dependencies using pip:

```
pip install Cython
pip install contextlib2
pip install pillow
pip install lxml
pip install jupyter
pip install matplotlib
```

**Note:** sometimes "sudo apt-get install protobuf-compiler" will install Protobuf 3+ versions for you and some users have issues when using 3.5. If that is your case, try the [manual](#) installation.

### Protobuf Compilation

The Tensorflow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be compiled. This should be done by running the following command from the tensorflow/models/research/ directory:

**# From tensorflow/models/research/**

```
protoc object_detection/protos/*.proto --python_out=.
```

**Note:** If you're getting errors while compiling, you might be using an incompatible protobuf compiler. If that's the case, use the following manual installation

### Manual protobuf-compiler installation and usage

#### **If you are on linux:**

Download and install the 3.0 release of protoc, then unzip the file.

**# From tensorflow/models/research/**

```
wget -O
protobuf.zip https://github.com/google/protobuf/releases/download/v3.0.0/protoc-3.0.0-linux-x86\_64.zip
```

```
unzip protobuf.zip
```

Run the compilation process again, but use the downloaded version of protoc

**# From tensorflow/models/research/**

```
./bin/protoc object_detection/protos/*.proto --python_out=.
```

Fonte: [https://git.byr.ac.cn/fdt/models/blob/master/research/object\\_detection/g3doc/installation.md](https://git.byr.ac.cn/fdt/models/blob/master/research/object_detection/g3doc/installation.md)

Quando terminar adicione a seguinte linha ao arquivo ".bashrc" contido na sua home:

```
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

Substitua 'pwd' pelo caminho absoluto até o diretório "research" dentro do repositório "models" que você clonou.

## Instalação Dodo\_Detector

### OBS: não utilizar a versão mais recente.

O próximo passo é instalar o dodo\_detector.

Instalar o dodo\_detector é muito simples, basta seguir o guia contido no github do incrível Douglas de Rizzo Meneghetti.

### Installation

Since this package is not on PyPI, you can install it via **pip** like this:

```
pip install git+https://github.com/douglasrizzo/dodo_detector.git
```

OpenCV is a hard dependency and is installed via the PyPI **opencv-python** package. If you already have OpenCV installed (*e.g.* from source), edit *setup.py* and remove the hard dependency before installing.

Fonte: [https://douglasrizzo.github.io/dodo\\_detector/index.html](https://douglasrizzo.github.io/dodo_detector/index.html) (O Dodô safadinho)

### -----dodo\_detector\_ros-----

Para usar a detecção de objetos com o ROS você deve clonar o pacote para a pasta src do seu "workspace" do ROS.

Clone o pacote com:

```
git clone https://github.com/douglasrizzo/dodo\_detector\_ros.git
```

Depois é só compilar usando catkin\_make.

Aqui você encontra mais informações sobre o pacote:

[https://github.com/douglasrizzo/dodo\\_detector\\_ros/blob/master/README.md](https://github.com/douglasrizzo/dodo_detector_ros/blob/master/README.md)

Alguns arquivos precisam ser configurados dentro do pacote, para escolher, por exemplo, qual modelo será usado para de detecção.

## Criação de um Dataset Sintético

Clonar o repositório:

```
->git clone https://github.com/debidatta/syndata-generation
```

Dentro da pasta clonada, clonar o seguinte pacote

```
->git clone https://github.com/yskmt/pb.git
```

Abrir a pasta clonada e copiar o arquivo pb.py e colocar na pasta syndata-generation

Instalar o pyamg

```
->pip install pyamg
```

instalar o opencv(caso já não tenha)

```
->pip install opencv-python
```

instalar PIL

```
->pip install Pillow
```

instalar o PyBlur

```
->pip install pyblur
```

executar através do terminal o seguinte código para verificar se tudo ocorreu corretamente:

```
->python dataset_generator.py --num 5 --dontocclude  
./demo_data_dir/objects_dir/ ./output_dir/
```



Baixar a pasta TensorFlow\_SynData do link:

<https://feiedu.sharepoint.com/sites/RoboFEI/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FRoboFEI%2FShared%20Documents%2FAttachments%2FHOME%2FComputacao%2FDetec%C3%A7%C3%A3o%20de%20objetos&parentid=%2F0155D8E5-4F41-4B4A-80D0-606060606060&originalPath=aHR0cHM6Ly9mZWlZHUuc2hhcmVwb2ludC5jb20vOjY6L3MvUm9ib0ZFSS9FbWl2a1ZheThSaEV0SE5DLUpJYThIVUJwcFVkcck1vakiQQ3ktbTk2MzdKYjdBP3J0aW1IPXJtY3JHJkprMkVn>

Para gerar as máscaras das imagens a serem inseridas no SynData

Generator dentro da pasta images no caminho

Tensorflow\_SynData/bgsubtractor/images

(imagens como .jpg)depois executar rmbg.py

->python rmbg.py

Dentro da pasta images poderemos encontrar duas outras pastas, a pasta com o nome do objeto, onde estão as imagens e as suas máscaras e a pasta annotations onde estão as anotações das imagens.

Depois copiar a pasta images e colar no caminho

*/Tensorflow\_SynData/synthetic/syndata-generation/demo\_data\_dir/objects\_dir/*

Agora o próximo passo é definir os objetos a serem usados, dentro da pasta */demo\_data\_dir/* existe um arquivo chamado *selected.txt*, nesse arquivo devemos colocar o nome das pastas dos objetos a serem usados no dataset sintético

Editar as configurações do arquivo *defaults.py* para satisfazer as necessidades do dataset.

Agora iremos criar o dataset a partir do comando abaixo:

**(OBS. o argumento --num é o número de imagens a serem criadas a partir de cada foto e para cada estilo de blending)**

->python dataset\_generator.py --num 5 --dontocclude  
./demo\_data\_dir/objects\_dir/ ./output\_dir/

---

Agora que temos o dataset criado, devemos criar as anotações tipo csv que serão usadas. Assim devemos seguir o caminho  
./Tensorflow\_SynData/synthetic/ e executar o script generate\_csv da seguinte maneira:

```
->python3 generate_csv.py xml ./syndata-  
generation/output_dir/annotations synthetic.csv ./syndata-  
generation/output_dir/images
```

---

Agora com arquivo synthetic.csv criado, copiamos ele e colamos na pasta que será criada no seguinte caminho:

/Tensorflow\_SynData/Scripts/detection\_util\_scripts/DataSet/ e executamos o script generate\_train\_eval.py com os seguintes argumentos:

```
->python generate_train_eval_modified.py './DataSet/synthetic.csv'  
-f 0.8 -o .
```

---

Pegar a pasta images no caminho:

/Tensorflow\_SynData/synthetic/syndata-  
generation/demo\_data\_dir/objects\_dir/

e o arquivo selected.txt(renomear para:Label\_Map.txt)

e colocar na pasta no caminho

/Tensorflow\_SynData/Scripts/detection\_util\_scripts/DataSet/

gerar o arquivo ptxt com o script:

```
->python3 generate_ptxt.py txt ./DataSet/Label_Map.txt  
./DataSet/Label_Map.ptxt
```

Com a pasta /DataSet/ pronta, iremos criar os arquivos tfrecord a partir dos comandos abaixo:

```
->python3 generate_tfrecord.py ./DataSet/synthetic_train.csv  
./DataSet/Label_Map.pbtxt ./DataSet/images ./DataSet/train.record
```

```
->python3 generate_tfrecord.py ./DataSet/synthetic_eval.csv  
./DataSet/Label_Map.pbtxt ./DataSet/images ./DataSet/eval.record
```

-----

## TREINAMENTO

Colocar a pasta DataSet com os arquivos recentemente gerados na HOME para ficar assim ./home/fernando/DataSet/

Os arquivos de modelos estarão no caminho /home/fernando/models/

Dentro da pasta do modelo encontraremos alguns arquivo:

- frozen\_interference\_graph.pb -> peso atual
- checkpoint um tipo de "save" do estado do treinamento, caso você pare e queira recomeçar
- pipeline.config é o arquivo que define as variáveis do treinamento do modelo

Agora iremos configurar o arquivo pipeline.config(nesse caso do ssd\_mobilenet\_v2\_coco) do modelo escolhido para treinar:

- linha 3: num\_classes: (número de classes)
- no fim do arquivo: verificar no espaços eval\_input\_reader e train\_input\_reader se os caminhos estão certos
- linha 176: shuffle:true
- linha 160: num\_steps: 200000

Para iniciar o treinamento devemos criar um arquivo com o seguinte script:

```
#!/bin/bash
```

```
PIPELINE_CONFIG_PATH=/home/robofeiathome/data/trained_models/larc2019/pipeline.config
```

```
MODEL_DIR=/home/robofeiathome/data/trained_models/larc2019/checkpoint
```

```
NUM_TRAIN_STEPS=200000
```

```
SAMPLE_1_OF_N_EVAL_EXAMPLES=10
```

```
python object_detection/model_main.py --  
pipeline_config_path=${PIPELINE_CONFIG_PATH} --  
model_dir=${MODEL_DIR} --num_train_steps=${NUM_TRAIN_STEPS} --  
sample_1_of_n_eval_examples=${SAMPLE_1_OF_N_EVAL_EXAMPLES} --  
alsologtostderr
```

Para visualizar o treinamento devemos criar um outro arquivo com o seguinte script:

```
#!/bin/bash
```

```
MODEL_DIR=/home/robofeiathome/data/trained_models/larc2019/checkpoint
```

```
tensorboard --logdir=${MODEL_DIR}
```

Para executar o treinamento devemos criar um arquivo de script com o seguinte conteúdo:

```
#!/bin/bash
```

```
INPUT_TYPE=image_tensor
```

```
PIPELINE_CONFIG_PATH=/home/robofeiathome/data/trained_models/larc2019/pipeline.config
```

```
TRAINED_CKPT_PREFIX=/home/robofeiathome/data/trained_models/larc2019/checkpoint/model.ckpt-52802
```

```
EXPORT_DIR=/home/robofeiathome/data/trained_models/larc2019/exported
```

```
python object_detection/export_inference_graph.py --  
input_type=${INPUT_TYPE} --  
pipeline_config_path=${PIPELINE_CONFIG_PATH} --  
trained_checkpoint_prefix=${TRAINED_CKPT_PREFIX} --  
output_directory=${EXPORT_DIR}
```

**\*\*TRAINED\_CKPT\_PREFIX** - PEGAR O ÍNDICE DO CHECKPOINT DO CAMINHO MODEL\_DIR ONDE O MODELO ESTÁ SENDO TREINADO. Ex: 52802

Para exportar o modelo criar um arquivo de script com o seguinte conteúdo:

```
#!/bin/bash
```

```
INPUT_TYPE=image_tensor
```

```
PIPELINE_CONFIG_PATH=/home/robofeiathome/data/trained_models/larc2019/pipeline.config
```

```
TRAINED_CKPT_PREFIX=/home/robofeiathome/data/trained_models/larc2019/checkpoint/model.ckpt-52802
```

```
EXPORT_DIR=/home/robofeiathome/data/trained_models/larc2019/exported
```

```
python object_detection/export_inference_graph.py --
```

```
input_type=${INPUT_TYPE} --
```

```
pipeline_config_path=${PIPELINE_CONFIG_PATH} --
```

```
trained_checkpoint_prefix=${TRAINED_CKPT_PREFIX} --
```

```
output_directory=${EXPORT_DIR}
```

**\*\*TRAINED\_CKPT\_PREFIX** - PEGAR O ÍNDICE DO CHECKPOINT DO CAMINHO MODEL\_DIR ONDE O MODELO ESTÁ SENDO TREINADO. Ex: 52802

-----

Com o treinamento terminado devemos ir até o caminho  
/home/fernando/dodo\_detector-0.6.1/dodo\_detector e colar os arquivos  
gerados na pasta exported aqui.

Criar uma pasta chama images e colocar as imagens que você queira  
detectar, criar uma pasta chamada savedImages e executar o seguinte  
comando

->python3 DetecionOnBatch.py