\*\*\* Caso não esteja instalando no environment principal do python, sempre que abrir o terminal, ativar o environment a ser utilizado \*\*\*

1) Instalar as bibliotecas necessárias:

Tensorflow

OpenCV

PIL

Numpy

Cython

\* Caso não tenha git instalado, instalar via conda (conda install git)

--------------------------------------------------------------------------------------------------------------

2) Instalar CUDA e CUDNN

--------------------------------------------------------------------------------------------------------------

**\*\*\* Necessário bastante cuidado nos passos 3, 4 e 5 \*\*\***

--------------------------------------------------------------------------------------------------------------

3) Instalar a API de object detection do tensorflow

3.1) Baixar a API {{{[https://github.com/tensorflow/models}}](https://github.com/tensorflow/models%7D%7D)} (Remover chaves)

3.2 Protobuf

3.2.1) Baixar e instalar Protobuf (conda install protobuf)

3.2.2) Incluir a variável de ambiente necessária\*\*\*

\*\*\* Abrir as variáveis de ambiente do windows, em [Variáveis do sistema > Path], adicionar 'caminho\_protobuf'/bin onde 'caminho\_protobuf' é o local onde foi instalado. Isto varia com o ambiente do python no qual foi executado o comando conda install.

3.2.3) Abri terminal (Conda ou CMD) [Reiniciar caso já esteja aberto]

3.2.4) Navegar (cd) até a pasta na qual foi extraída a API > .../models/research

3.2.5) Executar no terminal: {{{for /f %i in ('dir /b object\_detection\protos\\*.proto') do protoc object\_detection\protos\%i --python\_out=.}}}

--------------------------------------------------------------------------------------------------------------

4) Instalar e compilar a Python COCO API

4.1) Baixar e instalar o Microsoft Visual C++ 14.0 (ou superior)

4.2) Reiniciar o computador (Aqui precisei reiniciar duas vezes para dar certo)

4.3) Baixar a COCO API usando: pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI

--------------------------------------------------------------------------------------------------------------

5) Compilar a API de object detection

5.1) Abri terminal (Conda ou CMD) [Reiniciar caso já esteja aberto]

5.2) Navegar (cd) até a pasta na qual foi extraída a API > .../models/research

5.3) Executar no terminal: {{{cp object\_detection/packages/tf2/setup.py .}}}

5.4) Executar no terminal: {{{python -m pip install --use-feature=2020-resolver .}}}

--------------------------------------------------------------------------------------------------------------

6) Baixar o modelo pré-treinado desejado (<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md>)

\*\* No trabalho foi utilizado o SSD MobileNet V2 FPNLite 320x320

-------------------------------------------------------------------------------------------------------------------

7) Criar o label map p/ o transfer learning segundo o modelo (Ver arquivo labelmap.pbtxt)

-------------------------------------------------------------------------------------------------------------------

8) Editar e Executar o script 'organiza\_dataset.py' (Corrigir os caminhos p/ sua máquina)

-------------------------------------------------------------------------------------------------------------------

9) Modificar o arquivo de configurações (pipeline.config)

\*\* Alterar com calma e bastante atenção.

- Quantidade de classes p/ transferlearning;

- Caminhos dos arquivos de treino e teste (após execução do script do passo 8)

- Caminho do label map

- Batch Size

- Caminhos de Checkpoints do treinamento

- Tipo do modelo (alterar de classification p/ detection)

- Demais parâmetros que achar conveniente (hiperparâmetros, camadas, etc...)

-------------------------------------------------------------------------------------------------------------------

\*Nesse ponto reiniciei o computador p/ garantir

10) Treinamento do modelo (c/ transfer-learning)

10.1) Abrir o terminal e ativar o environment do projeto

10.2) Executar no terminal: {{{'''caminho\_api'''/research/object\_detection/model\_main\_tf2.py --pipeline\_config\_path='''caminho\_arquivo\_config'''/pipeline.config --model\_dir='''caminho\_salvar\_modelo'''/Modelo\_Treinado}}}

\*\*\* Substituir:

'''caminho\_api''' pelo caminho onde foi baixada e compilada a API de object detection

'''caminho\_arquivo\_config''' pelo caminho onde ficou salvo o arquivo de config baixado junto do modelo (após alterações no passo 8)

'''caminho\_salvar\_modelo''' pelo caminho onde deseja que seja salvo o modelo treinado

10.3) Aguardar (No meu pc com GPU descente demorou +- 8 horas cada treinamento)

-------------------------------------------------------------------------------------------------------------------

11) Salvar o modelo treinado em graph tensorflow lite

11.1) Executar no treminal: {{{python '''caminho\_api''' /research/object\_detection/export\_tflite\_graph\_tf2.py --pipeline\_config\_path '''caminho\_arquivo\_config''' --trained\_checkpoint\_dir '''caminho\_salvar\_modelo''' --output\_directory '''caminho\_modelo\_graph'''}}}

\*\*\* '''caminho\_api'''; '''caminho\_arquivo\_config''' ; '''caminho\_salvar\_modelo''' devem ser exatamente os mesmos utilizados no passo 10

\*\*\* '''caminho\_modelo\_graph''' deve ser subsituido pelo caminho onde deseja que seja salvo o modelo final

-------------------------------------------------------------------------------------------------------------------

12) Converter o modelo p/ tensorflow lite

12.1) Se deseja converter sem quantizar, editar e executar o script 'converte\_sem\_qtd.py'

12.2) Se deseja converter com quantização. editar e executar o script 'converte\_qtd.py'

12.2.1) Se desejar compilar p/ TPU, executar o script oficial disponível em: <https://colab.research.google.com/github/google-coral/tutorials/blob/master/compile_for_edgetpu.ipynb>

\*\*\* Antes de executar os scritps em 12.1 e 12.2, é necessário corrigir os caminhos p/ a sua máquina.

-------------------------------------------------------------------------------------------------------------------