* O diretório classify contém imagens classificadas pela rede em categorias de acordo com a marca do produto. As imagens em train foram usadas para treinar a rede, e as imagens em test somente para testar. O diretório clean contém as imagens originais sem classificação.
* O diretório count contém imagens dos produtos vistos de cima que precisam ser contados pela rede.

Instruções para utilizar o framework de redes neurais Darknet, com a rede **YOLOv3-tiny**:

1. **Baixar a instalar o framework**

Instruções originais: <https://pjreddie.com/darknet/yolo/>

Clonar o repositório <https://github.com/pjreddie/darknet>:

*git clone* [*https://github.com/pjreddie/darknet*](https://github.com/pjreddie/darknet)

*cd darknet*

*make*

Para habilitar o uso de GPUs é necessário trocar a linha GPU=0 para GPU=1 no Makefile antes de compilar.

Fazer o download do arquivo de pesos:

*wget https://pjreddie.com/media/files/yolov3-tiny.weights*

Testar (o resultado será salvo em *predictions.jpg*):

*./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg*

1. **Preparar o dataset**

Instruções originais: <https://medium.com/@manivannan_data/yolo-annotation-tool-new-18c7847a2186>

Antes de começar a treinar a rede, é preciso anotar as imagens com a região e a classe associada a cada região. Por exemplo, com três regiões e duas classes:



Clonar o repositório: <https://github.com/ManivannanMurugavel/Yolo-Annotation-Tool-New->

Modificar o arquivo *classes.txt* com as classes desejadas (uma classe por linha).

Colar a pasta com as imagens que precisam ser anotadas no diretório */Images* e rodar *main.py*. A ferramenta permite anotar manualmente a região e a classe de cada objeto em cada imagem.

Vídeo: <https://www.youtube.com/watch?v=WiIDe2jNuc0>

Para cada imagem anotada será gerado um arquivo *.txt* no diretório correspondente em */Labels* com a informação de regiões e classes. Esse arquivo será necessário para o treinamento.

No fim devemos ter uma pasta dentro de */Labels* com a anotação de todas as imagens, um *.txt* por imagem. Colar os *.txt* na mesma pasta onde estão as imagens.

1. **Treinar**

Instruções originais: <https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2>

Antes de treinar é preciso preparar arquivos *.data*, *.names* e *.cfg*. Exemplos de cada um dos arquivos na pasta exemplos do drive.

Utilizar o ***.cfg*** já disponível em *cfg/yolov3-tiny.cfg.* Ele deve ser modificado de acordo com o número de classes que serão treinadas:

* Linha 127: filters = (número de classes + 5) \* 3
* Linha 135: classes = número de classes
* Linha 171: filters = (número de classes + 5) \* 3
* Linha 177: classes = número de classes

Preparar dois arquivos, ***test.txt*** e ***train.txt*** com o path para cada uma das imagens anotadas no passo 2. Por exemplo, se as imagens estão em /home/exemplos, fazer um arquivo com uma imagem em cada linha:

*/home/exemplos/imagem1.jpg*

*/home/exemplos/imagem2.jpg*

*/home/exemplos/imagem3.jpg*

Os arquivos *.txt* de anotações gerados no passo anterior devem estar na mesma pasta. As imagens no arquivo *train.txt* serão usadas para treinar a rede, e as imagens no arquivo *test.txt* serão usadas para testar a rede.

O arquivo ***.names*** deve conter o nome das classes (o mesmo que classes.txt do passo anterior), uma em cada linha.

O arquivo ***.data*** deve ser preenchido conforme:

*classes=2*

*train=PATH/train.txt*

*valid=PATH/test.txt*

*names=PATH/exemplo.names*

*backup=backup*

Classes é o número de classes a serem treinadas, train e valid devem indicar o diretório dos arquivos train.txt e test.txt anteriores, names deve indicar o diretório do arquivo .names, e backup indica o diretório onde os pesos intermediários serão salvos durante o treinamento.

Para treinar:

*./darknet detector train cfg/exemplo.data cfg/yolov3-tiny.cfg yolov3-tiny.weights -clear -1*

Um arquivo de pesos será salvo a cada 100 iterações na pasta backup. Depois de 900 iterações, esse arquivo será salvo a cada 10.000 iterações. O arquivo .backup contém os pesos mais atualizados.