Semantic Segmentation and Instance Segmentation

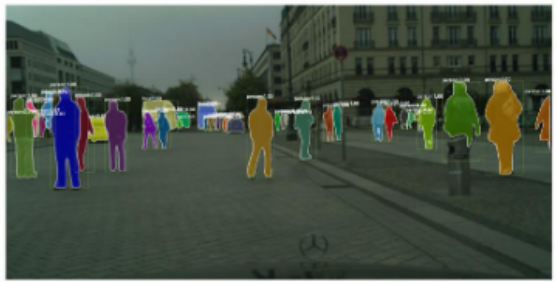
Mask R-CNN

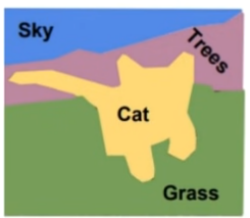
By Kaiming He, Georgia Gkioxari, Piotr Dollar and Ross Girshick

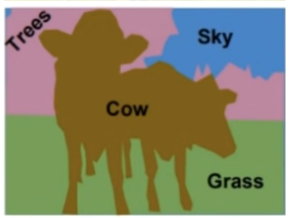
Presented By Aditya Sanghi

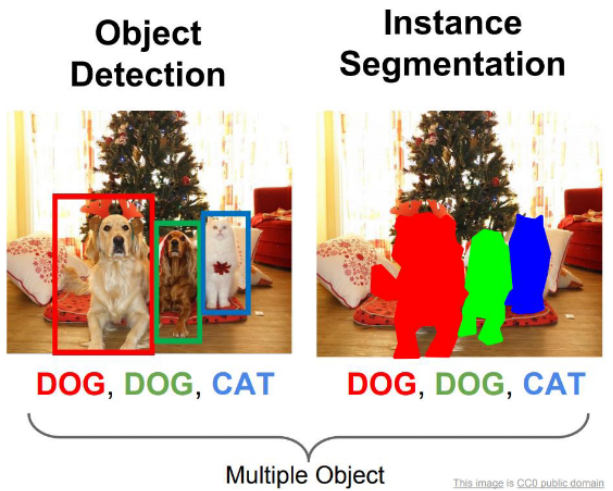
https://www.cs.toronto.edu/~fidler/teaching/2018/slides/CSC2548/MaskRCNN.pdf

Semantic segmentation --- > instance segmentation

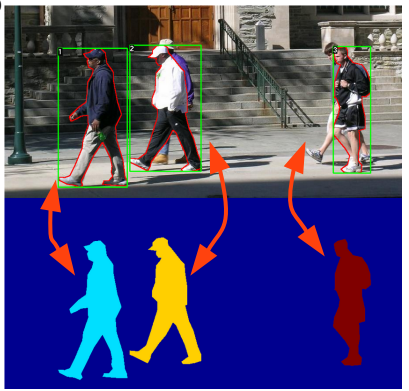








https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/torchvision_finetuning_instance_segmentation.ipynb



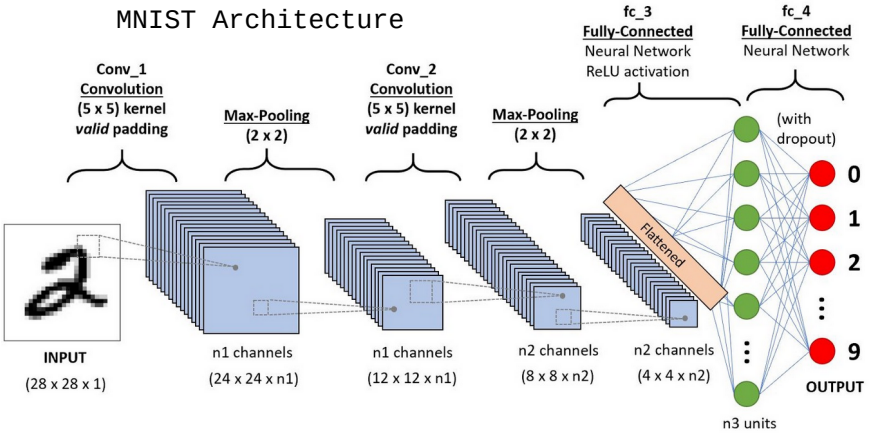
segmentation masks

Each image has a corresponding segmentation mask, where each color correspond to a different instance.

Reference: Detecting Pedestrians using PyTorch – A Helpful Guide

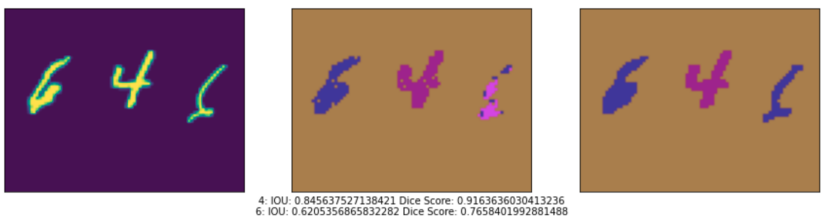
https://www.data-blogger.com/pedestrian-detection-using-pytorch/

From MNIST to Semantic Segmentation of Handwritten Digits



sinarazi / Image-Segmentation-of-Handwritten-Digits https://github.com/sinarazi/Image-Segmentation-of-Handwritten-Digits

Semantic segmentation: to predict/classify the hand written digits with segmentation mask on each pixel basis



MNIST extended: a dataset for semantic segmentation

https://awaywithideas.com/mnist-extended-a-dataset-for-semantic-segmentation-and-object-detection/

Input Image

Semantic Segmentation Result

Four Step Design Guidelines

General Guide (4 Steps)

1. Replace FC layers with convolutional layers.
2. Convert the last layer output to the original resolution.
3. Do softmax-cross entropy between the pixelwise predictions and segmentaion ground truth.
4. Backprop and SGD

Entropy:

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log \mathrm{P}(x_i) \quad (1)$$

Cross Entropy: Measure of difference between 2 probabilities

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x) \quad (2)$$

Define loss function (Cross entropy)

$$D(\hat{Y}, Y) = -\frac{1}{N} \cdot \sum Y_i \cdot \log(\hat{Y}_i) \quad (3)$$

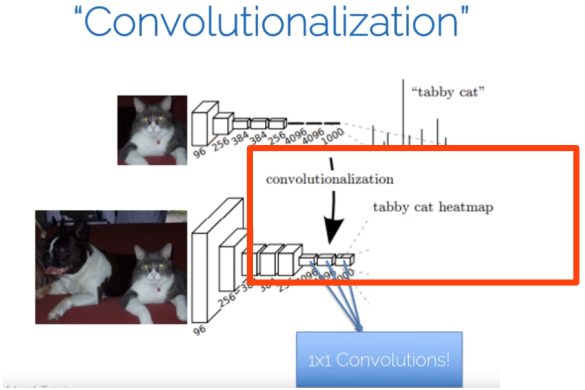
Stochastic Gradient Descent (SGD)

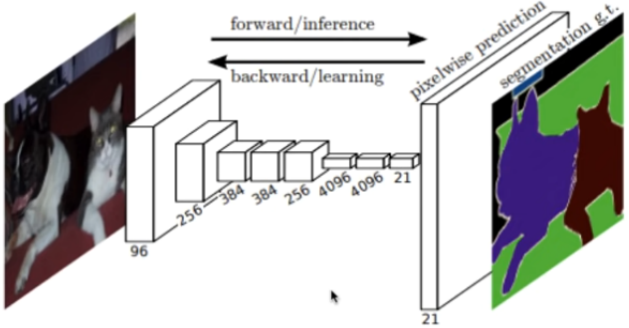
$$Q(w) = \sum_{i=1}^{n} Q_i(w) \quad (4) \text{ Objective function}$$

$$w := w - \eta \nabla Q(w) = w - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w) \quad (5) \text{ SGD}$$

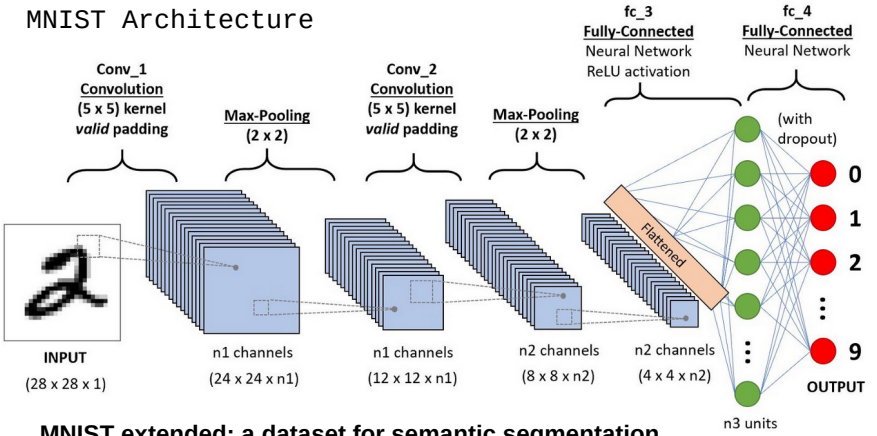
Convolutionalization: Change FC Layer to Convolutional Layer

General Guide: change FC (fully connected) layer to convolutional layer by 1x1 convolution kernel





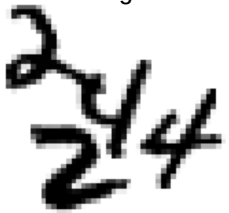
MNIST Architecture



MNIST extended: a dataset for semantic segmentation

https://awaywithideas.com/mnist-extended-a-dataset-for-semantic-segmentation-and-object-detection/

Input Image



Semantic Segmentation Result

Semantic segmentation: to predict/classify the hand written digits with segmentation mask on each pixel basis

SGD Example

Example [edit]

Let's suppose we want to fit a straight line $\hat{y} = w_1 + w_2 x$ to a training set with observations $(x_1, x_2, \ldots, x_n)$ and corresponding estimated responses $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ using least squares. The objective function to be minimized is:

$$Q(w) = \sum_{i=1}^{n} Q_i(w) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} (w_1 + w_2 x_i - y_i)^2.$$

The last line in the above pseudocode for this specific problem will become:

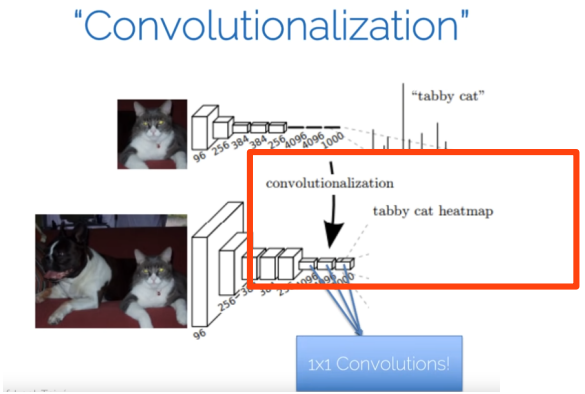
$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial}{\partial w_1}(w_1 + w_2 x_i - y_i)^2 \\ \frac{\partial}{\partial w_2}(w_1 + w_2 x_i - y_i)^2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} 2(w_1 + w_2 x_i - y_i) \\ 2x_i(w_1 + w_2 x_i - y_i) \end{bmatrix}.$$

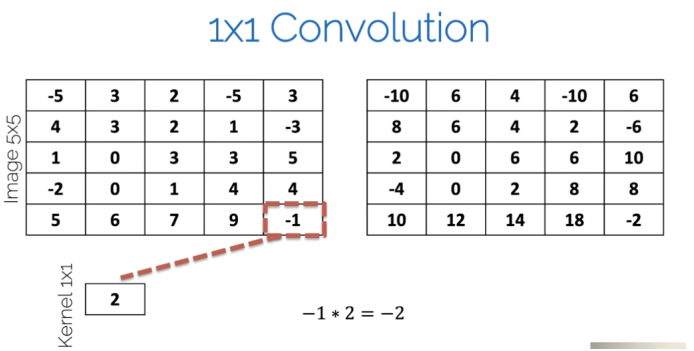
Note that in each iteration (also called update), the gradient is only evaluated at a single point x_i instead of at the set of all samples.

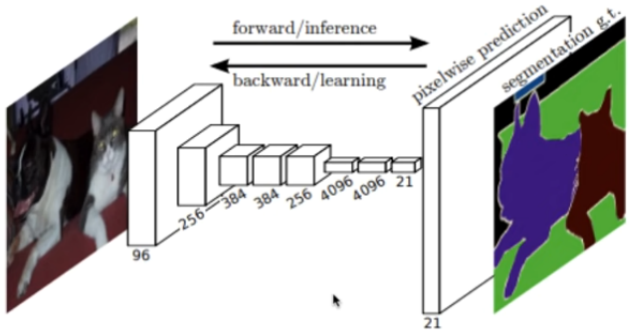
The key difference compared to standard (Batch) Gradient Descent is that only one piece of data from the dataset is used to calculate the step, and the piece of data is picked randomly at each step.

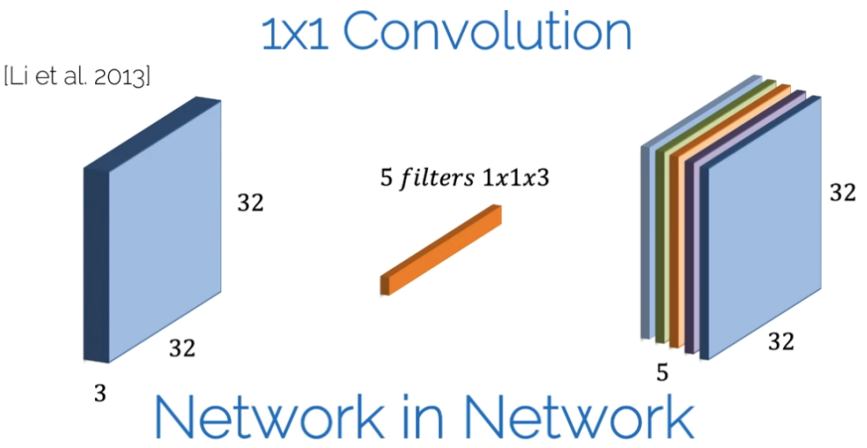
Convolutionalization: Change FC Layer to Convolutional Layer

General Guide: change FC (fully connected) layer to convolutional layer by 1x1 convolution kernel

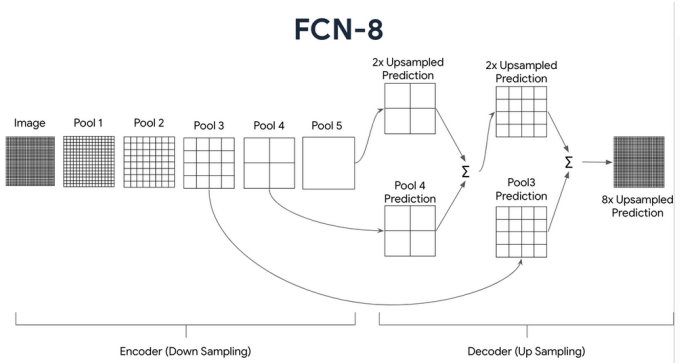


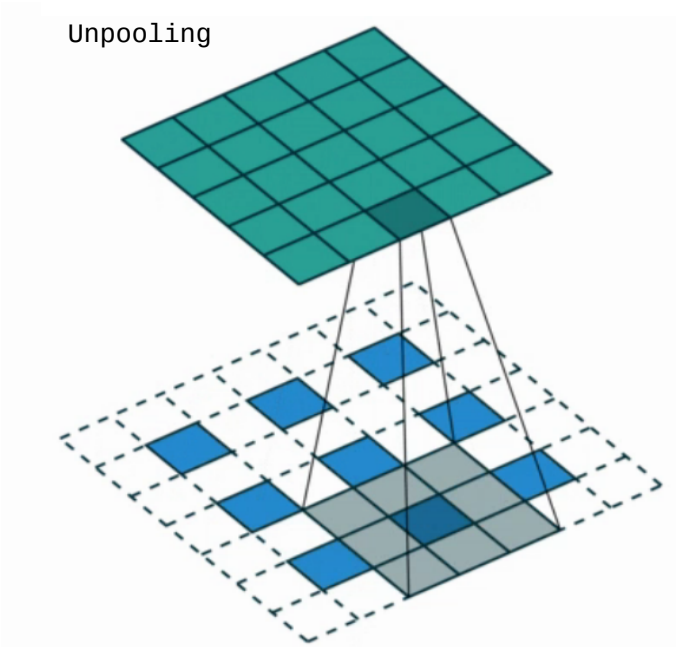




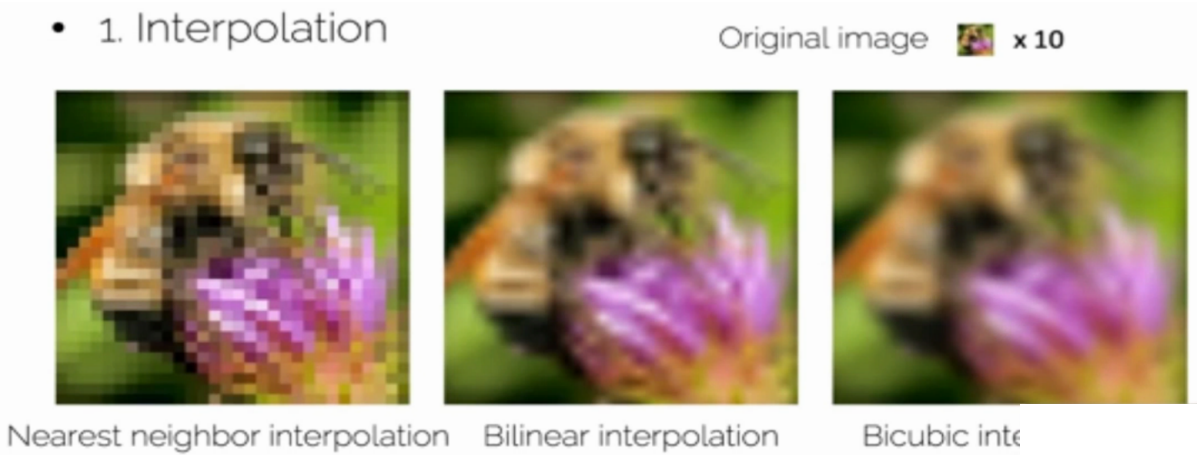


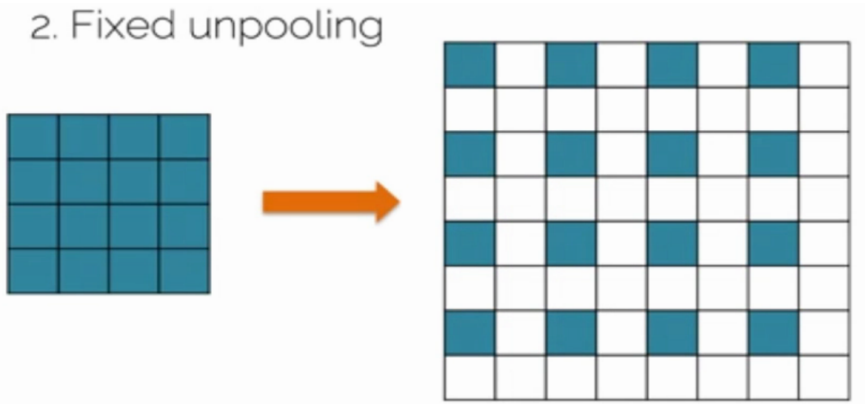
Up Sampling





- 1. Interpolation





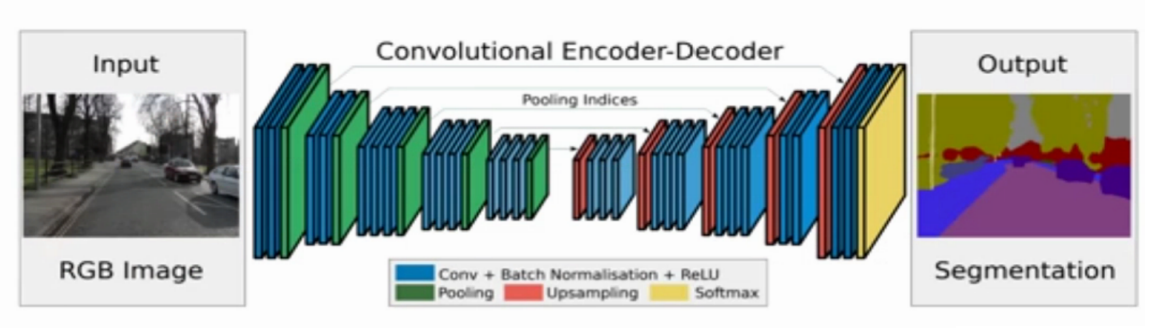
RCNN Architecture Examples

https://www.youtube.com/watch?v=XMSjOatyH0k

Mask R-CNN is a state of the art model for instance segmentation, developed on top of Faster R-CNN. Faster R-CNN is a region-based convolutional neural networks [2], that returns bounding boxes for each object and its class label with a confidence score.

1. Using Mask R-CNN, which is based on top of Faster R-CNN. Faster R-CNN is a model that predicts both bounding boxes and class scores for potential objects in the image.
2. Mask R-CNN adds an extra branch into Faster R-CNN, which also predicts segmentation masks for each instance.

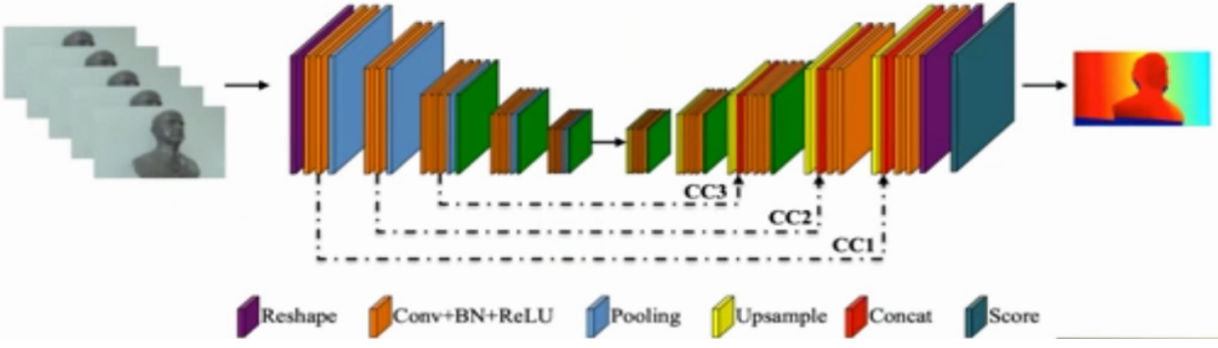
Architecture: Deep Depth from Focus.



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segm

Skip Connections

- Concatenation connections



Mask R-CNN for Semantic Segmentation

https://github.com/buseyaren/Installation-MaskRCNN

Step 6: Running the setup.py file. $python3 setup.py installpip

Step 7. install git+ https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI

Step 8. $python3 balloon.py train --dataset=/path/to/dataset --model=coco

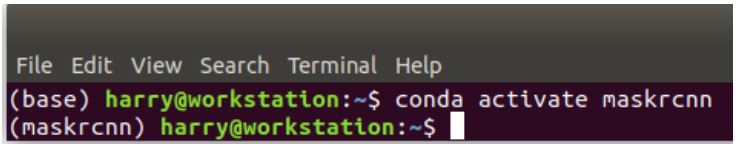
```
Check tensorFlow in Python before down loading:

import tensorflow as tf
print(tf.__version__)
```

https://www.youtube.com/watch?v=tcu4pr948n0

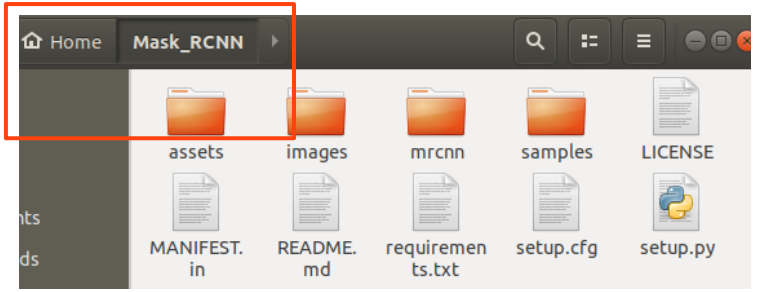
Software architecture: https://blog.paperspace.com/mask-r-cnn-in-tensorflow-2-0/

Step 1. use conda to create a virtual environment $conda create -n maskrcnn python=3.6.12

Step 2. Activate the environment: $conda activate maskrcnn



Step 3. $git clone https://github.com/matterport/Mask_RCNN.git



Step 4. python3 setup.py install

Step 4. $pip3 install -r requirements.txt (add the following 2 lines to enable gpu: (add this line) tensorflow-gpu==1.15.0 (then add this line) keras==2.2.5

Step 5. Download the pre-trained weights from https://github.com/matterport/Mask_RCNN/releases. Download the file mask_rcnn_balloon.h5 from Mask_RCNN_2.1 file. These 2 models should be placed in the samples folder.

download: mask_rcnn_coco.h5

Pedestrian Detection

https://data-flair.training/blogs/pedestrian-detection-python-opencv/

https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/torchvision_finetuning_instance_segmentation.ipynb

TorchVision Instance Segmentation

Step 1. start colab, install pycocotools-2.0 COCO metric for intersection over union.

```
%%shell

pip install cython
# Install pycocotools, the version by default in
# has a bug fixed in https://github.com/cocodata
pip install -U 'git+https://github.com/cocodatas
```

Cursor over it to run the code above on colab

dataset for Penn-Fudan, download and extract the data
https://www.cis.upenn.edu/~jshi/ped_html/PennFudanPed.zip

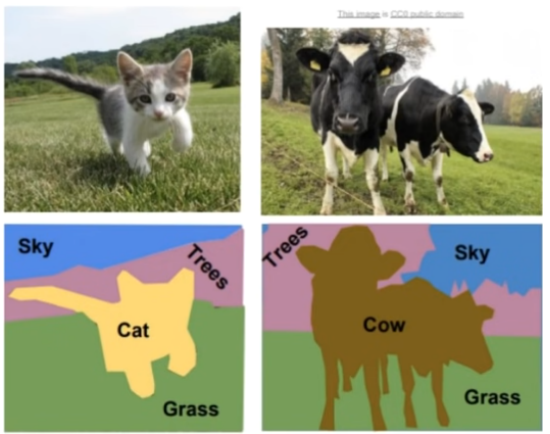
The data is structured as follows

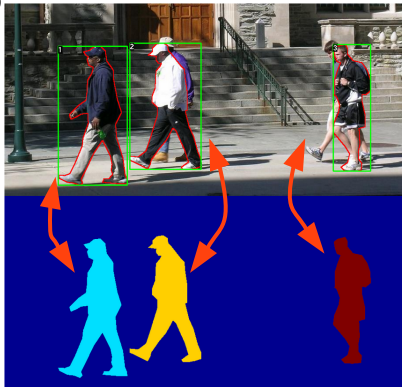
```
PennFudanPed/
  PedMasks/
    FudanPed00001_mask.png
    FudanPed00002_mask.png
    FudanPed00003_mask.png
    FudanPed00004_mask.png
    ...
  PNGImages/
    FudanPed00001.png
    FudanPed00002.png
    FudanPed00003.png
```

1. Using Mask R-CNN, which is based on top of Faster R-CNN. Faster R-CNN is a model that predicts both bounding boxes and class scores for potential objects in the image.

2. Mask R-CNN adds an extra branch into Faster R-CNN, which also predicts segmentation masks for each instance.

Semantic segmentation --- > instance segmentation





segmentation masks

Each image has a corresponding segmentation mask, where each color correspond to a different instance.

Detecting Pedestrians using PyTorch – A Helpful Guide
https://www.data-blogger.com/pedestrian-detection-using-pytorch/