Build your own CV (as in Computer Vision) with Keras and Tensorflow

machinalis
Machine Learning Solutions Delivery

Who Am I?

machinalis

Systems Engineer

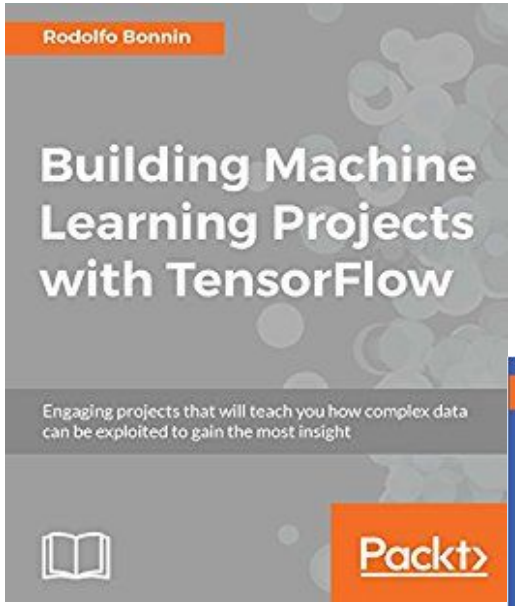
Ph.D Student, UTN

Working on Deep Learning since 2008

Ex ML @ Mercadolibre

CV Specialist @ Machinalis

Author of “Building Machine Learning Projects with Tensorflow” and “Machine Learning for Developers”



About us

Machinalis

Talk summary

machinalis

Introduction
Image Classification
Image Detection
Image Segmentation
The Future

Thanks:CS213 Stanford, EdjeElectronics, matterport

machinalis

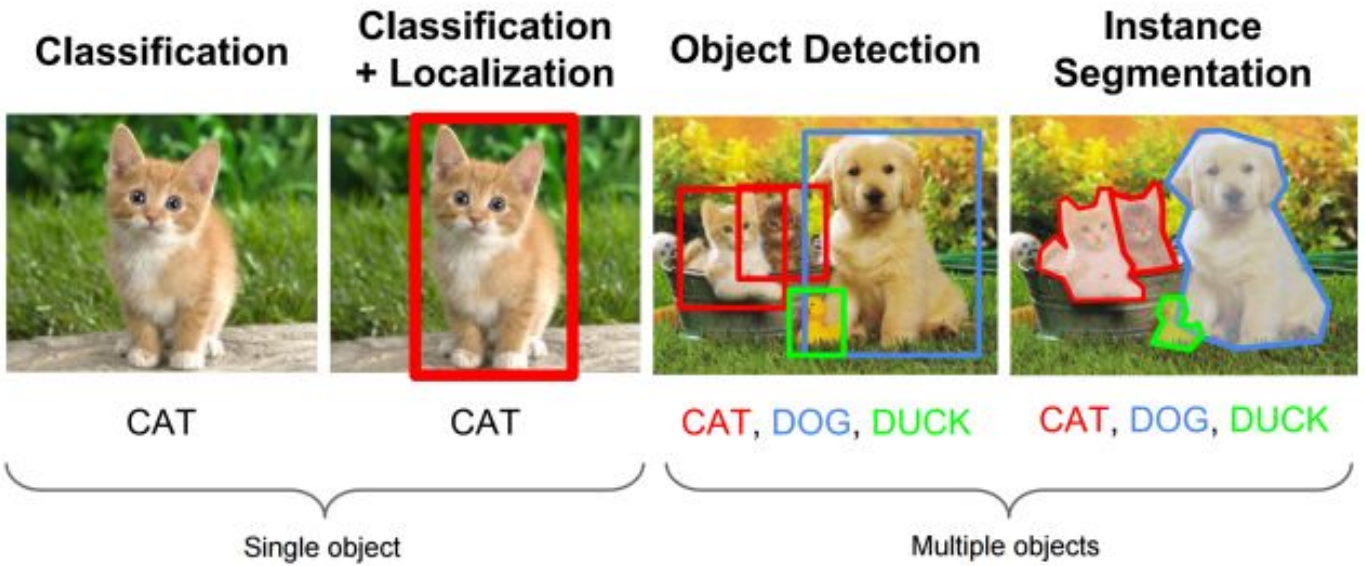
Introduction

Evolution of architectures for Computer Vision

machinalis

1940's	
1950's	**Perceptrons**
1960's	**First Winter (Minsky & Papert)**
1970's	**MLP's**
1980's	**First RNN, first Convolutional layers**
1990's	**Second winter, SVMs taking place**
2000's	
2010's	**Deep Architectures**

Types of tasks for Computer Vision



- **Image classification** is the prediction of the presence/absence of an instance of class in a test image.
- **Object detection** is the prediction of the bounding box and label of each object from the twenty target classes in a test image.
- **Semantic instance segmentation** is the assignment of one of the several labels to every pixel in a test image.

machinalis

Keras: An easy way to build neural models

Keras

machinalis

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow or CNTK.

- Allows for easy and fast prototyping
- Supports both convolutional networks and recurrent networks, as well as combinations of the two) and others via complementary projects.
- Runs seamlessly on CPU and GPU.

Model compilation

machinalis

A basic Keras model needs

- An **optimizer**. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class.
- A **loss function**. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as categorical_crossentropy or mse), or it can be an objective function.
- A list of **metrics**.

```
model.compile(loss='mean_squared_error',
              optimizer='sgd',
              metrics=[metrics.mae, metrics.categorical_accuracy])
```

Keras Model Training

machinalis

Keras models are trained on Numpy arrays of input data and labels.

For training a model, you will typically use the ***fit*** function.

Main Layer Types

machinalis

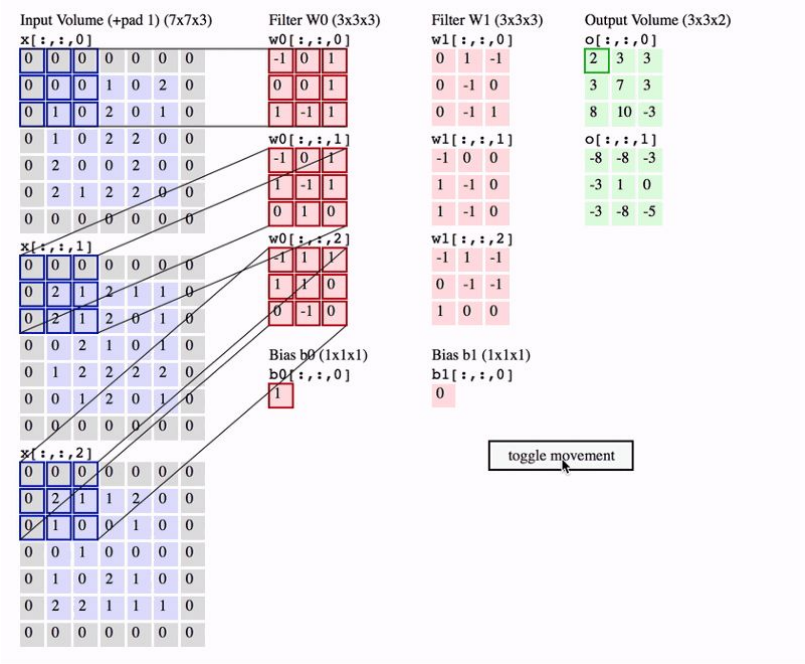
Dense

Convolutional: `Conv1D, Conv2D ,Conv3D`

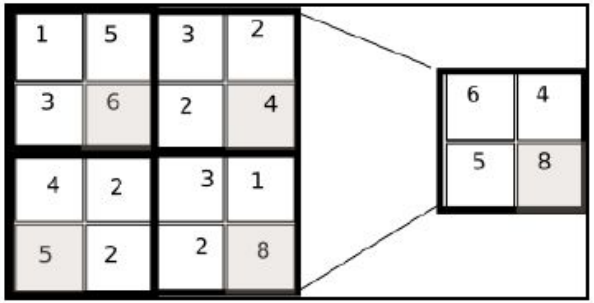
`Pooling:`

Recurrent: `RNN, GRU, LSTM`

Main special layers types in Computer Vision



Convolutional



Pooling

https://hackernoon.com/dl05-convolutional-neural-networks-1d3bb7fff586
(Original from CS231)

Losses

machinalis

```
mean_squared_error
mean_squared_error(y_true, y_pred)
mean_absolute_error(y_true, y_pred)
mean_absolute_percentage_error(y_true, y_pred)
mean_squared_logarithmic_error(y_true, y_pred)
categorical_crossentropy(y_true, y_pred)
```

Metrics

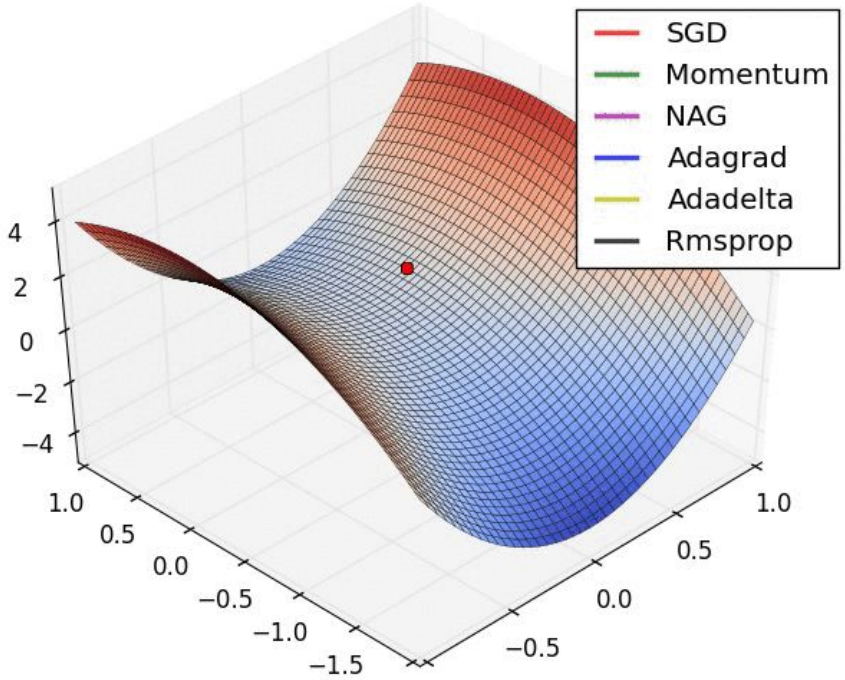
machinalis

- `accuracy`
- `binary_accuracy`
- `binary_accuracy(y_true, y_pred)`
- `categorical_accuracy(y_true, y_pred)`
- `top_k_categorical_accuracy(y_true, y_pred, k=5)`

Optimizers

machinalis

```
SGD()

RMSprop()

Adagrad()

Adadelta()

Adam()
```



Source: Alec Radford

Activations

machinalis

```
softmax
relu
tanh
sigmoid
hard_sigmoid
linear
```

Example:

```
from keras.layers import Activation, Dense
model.add(Dense(64))
model.add(Activation('tanh'))
```

machinalis

Image classification:

History of neural models

machinalis

Training (Images + labels)

Model (in Training)

Per class probability	Label
0.3	1.0
0.2	0
0.1	0
0.02	0
0.03	0
0.8	0
0.03	0

Airplane (Ground Truth)

Backprop (Normally)

-

Error

Inference (labels from images)

Trained Model

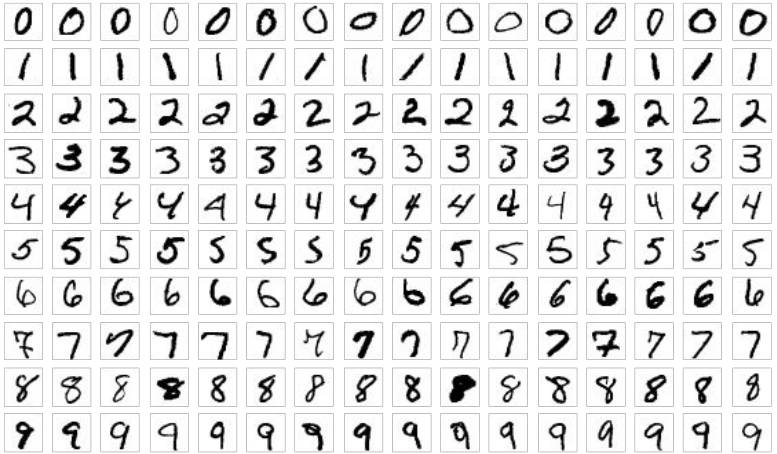
0.8
0.1
0.01
0.02
0.03
0.01
0.03

Per class probability

Airplane (most likely)

From Numbers to clothes

machinalis



First notebook: 1_History_of_neural_models.ipynb

machinalis

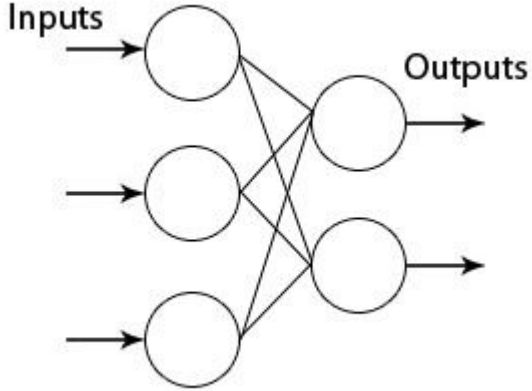
Open in Colaboratory:

https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/1_History_of_neural_models.ipynb

Fashion Mnist: History of neural models

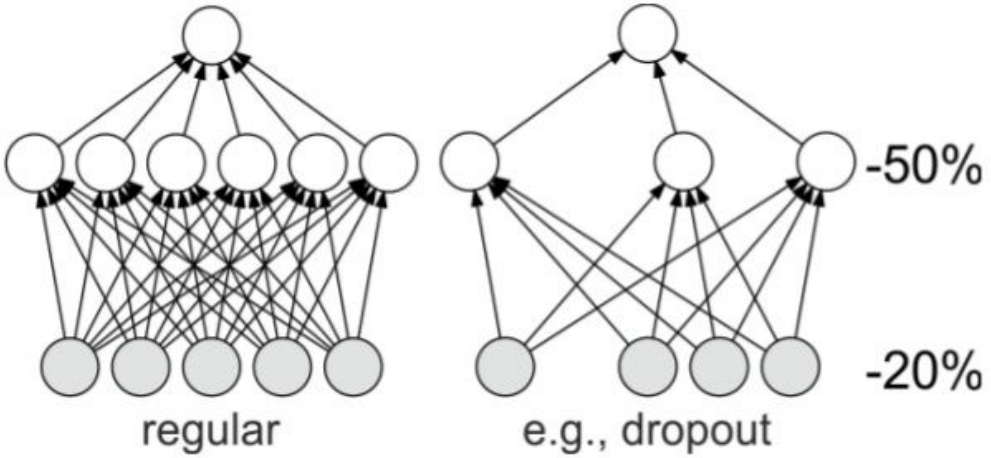
```
from __future__ import print_function

import keras
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten
from keras.optimizers import RMSprop
from keras.optimizers import SGD
import matplotlib.pyplot as plt
%matplotlib inline
batch_size = 128
num_classes = 10
epochs = 20

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

print (y_train[1000])
fig= plt.figure()
plt.imshow(x_train[1000], cmap='gray')

print (y_train[4000])
fig= plt.figure()
```

From Numbers to clothes: Single layer Perceptron



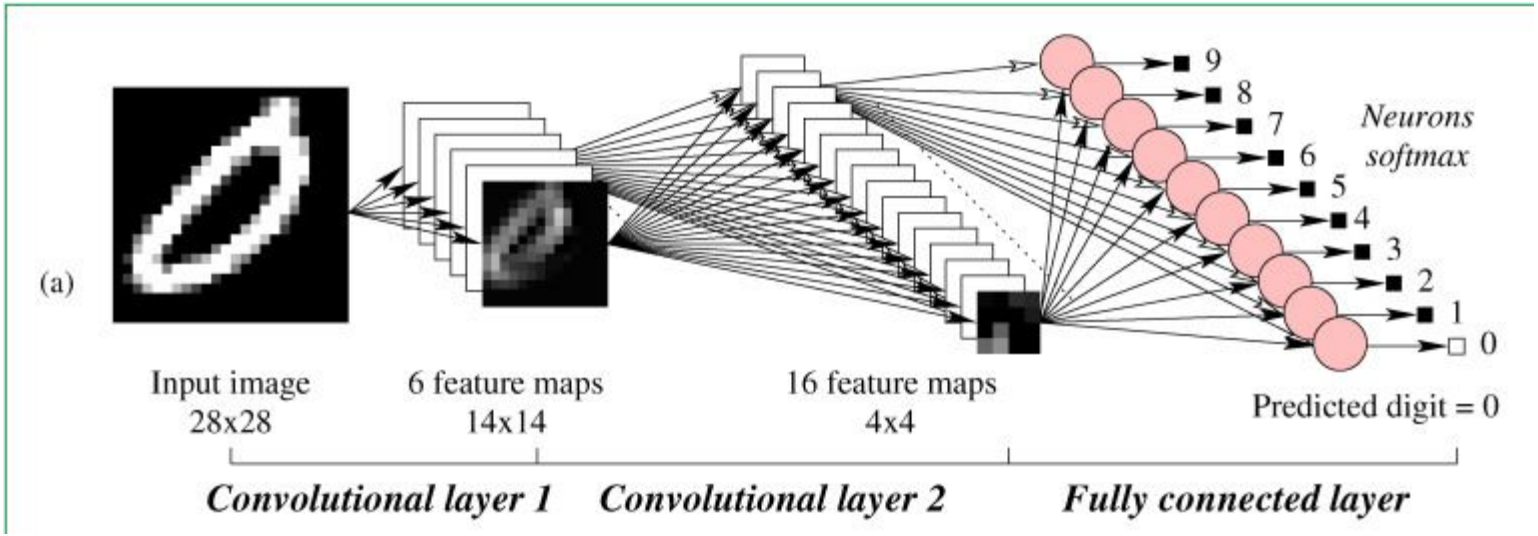
From Numbers to clothes

machinalis



From Numbers to clothes: Convolutional layer

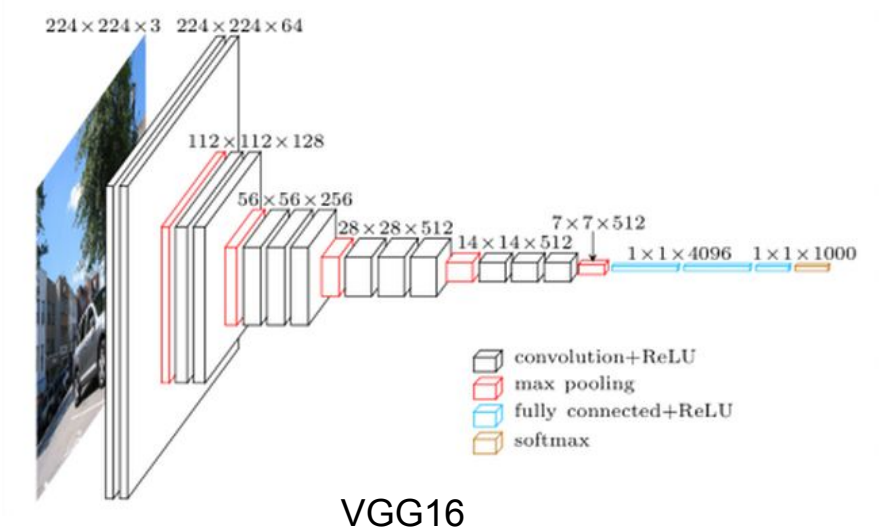
machinalis



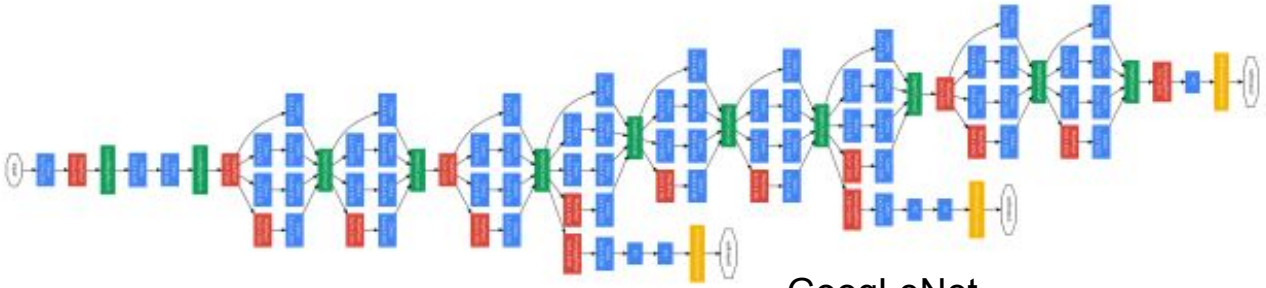
machinalis

Deep Learning and Transfer Learning Classification

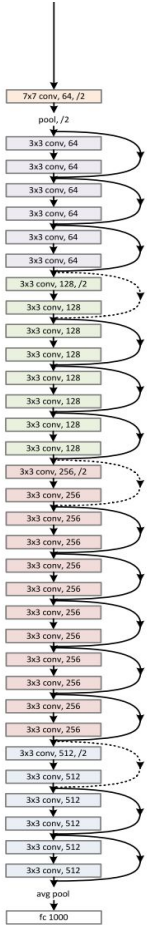
Main Deep learning models



VGG16



GoogLeNet



ResNet

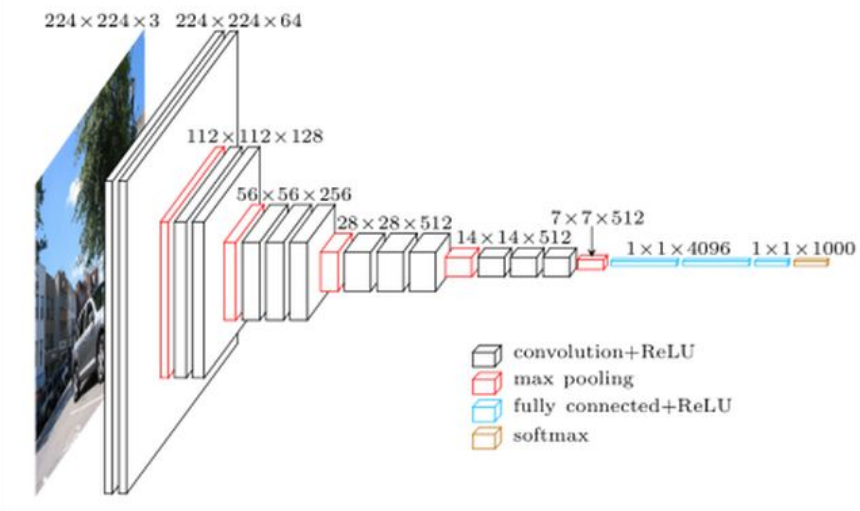
First things: Imagenet

machinalis

- S: (n) Eskimo dog, **husky** (breed of heavy-coated Arctic sled dog)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
        - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
          - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
            - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
              - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
                - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
                  - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                    - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
                      - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                        - S: (n) living thing, animate thing (a living (or once living) entity)
                          - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"*; *"the team is a unit"*
                            - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
                              - S: (n) physical entity (an entity that has physical existence)
                                - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Demo time! Exploring CNN layers on a Deep Network with Quiver

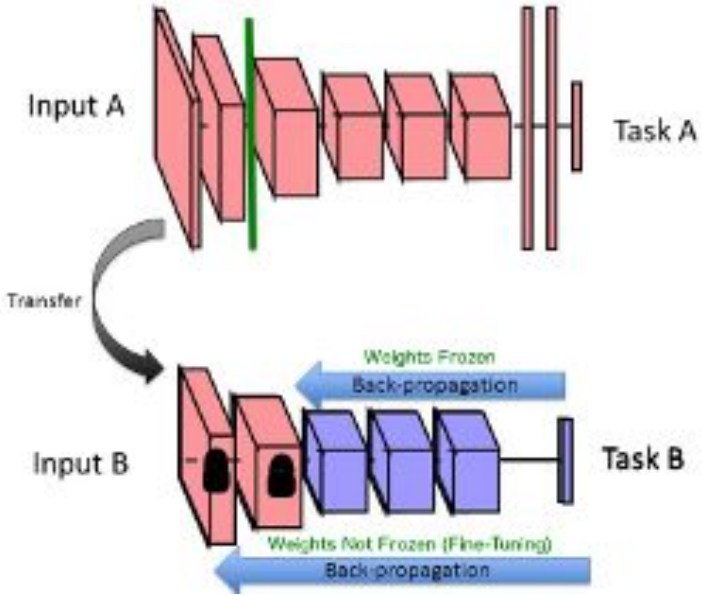
machinalis



Applying Transfer Learning to build a classifier

Open in Colaboratory:

https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/3_Transfer%20Learning%20For%20classification.ipynb



machinalis

Tensorflow &

Tensorflow Object Detection API

Tensorflow Object detection API

machinalis

machinalis

Object localization & Detection

The PASCAL Visual Object Classes (2005/2012)

machinalis

- A publicly available dataset of images and annotations:
  - 2.5 million labeled instances in 328k images, taken from consumer photographs
  - 20 Object types

- An annual competition and workshop, held until 2012

.

The PASCAL Visual Object Classes (2005/2012)

machinalis

- Example annotation:

```
<annotation>
	<folder>GeneratedData_Train</folder>
	<filename>000001.png</filename>
	<path>/my/path/GeneratedData_Train/000001.png</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>224</width>
		<height>224</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>21</name>
		<pose>Frontal</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<occluded>0</occluded>
		<bndbox>
			<xmin>82</xmin>
			<xmax>172</xmax>
			<ymin>88</ymin>
			<ymax>146</ymax>
		</bndbox>
	</object>
</annotation>
```

Types of detectors

machinalis

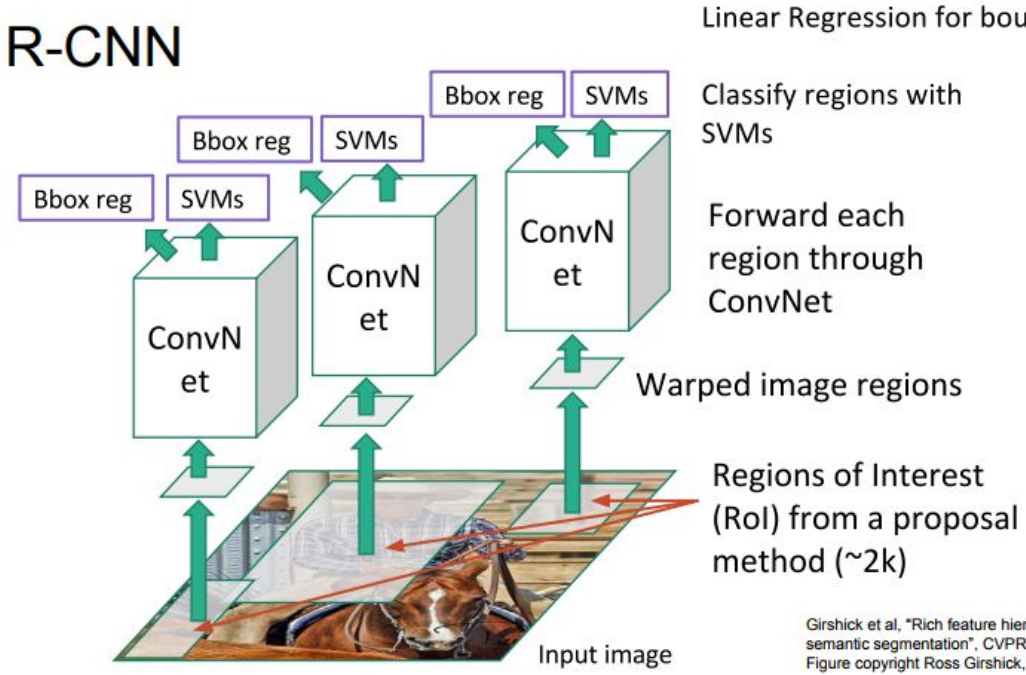
Main types of deep neural Detection Architectures

With proposals:
R-CNN, Faster-CNN

Without Proposals:
YOLO, SSD

R-CNN variants

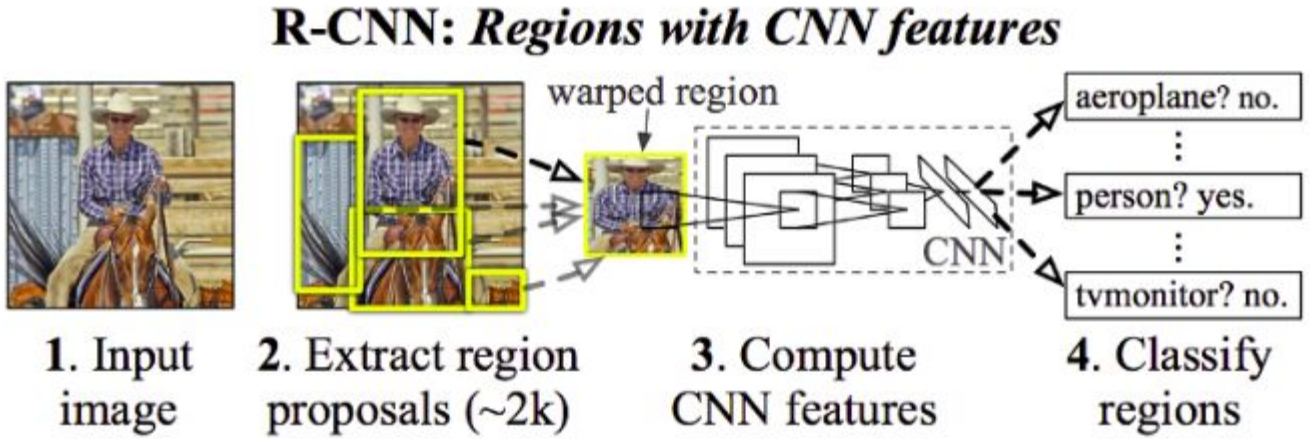
machinalis



R-CNN creates bounding boxes, or region proposals, using a process called Selective Search which you can read about here. At a high level, Selective Search looks at the image through windows of different sizes, and for each size tries to group together adjacent pixels by texture, color, or intensity to identify objects

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Thanks CS231



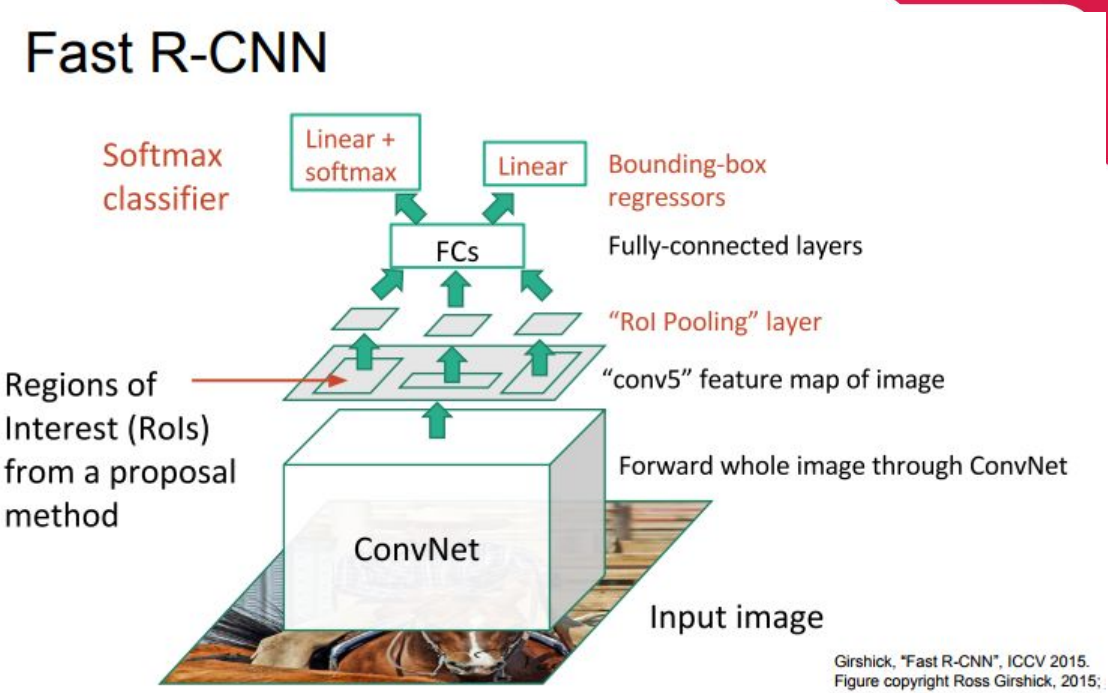
After creating a set of region proposals, R-CNN passes the image through a modified version of AlexNet to determine whether or not it is a valid region. Source: https://arxiv.org/abs/1311.2524.

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

R-CNN variants Fst R-CNN

R-CNN works really well, but is slow for a few simple reasons:

- It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image (that's around 2000 forward passes per image!).
-
- It has to train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train.



- RoIPool shares the forward pass of a CNN for an image across its subregions.
- One pass of the original image as opposed to ~2000
- Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor), **Fast R-CNN instead used a single network to compute all three.**

machinalis

Only one CNN needs to be trained and we get region proposals almost for free

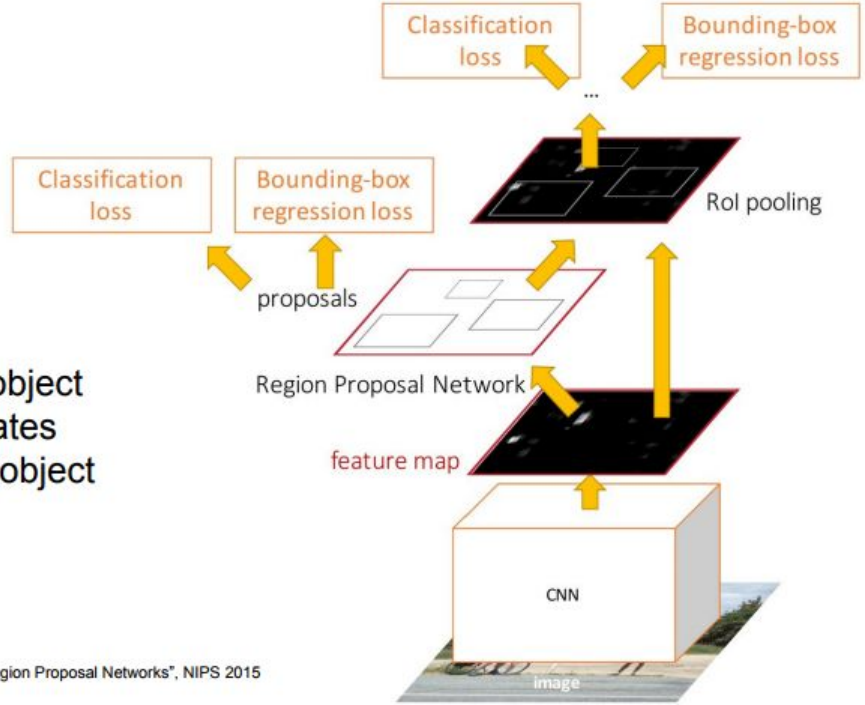
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

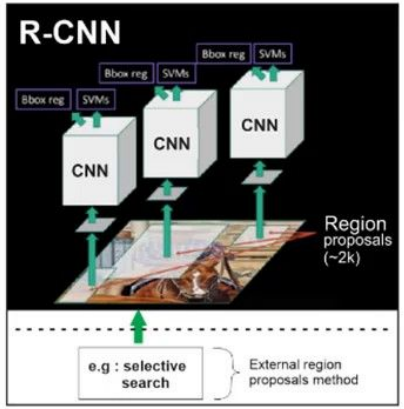
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

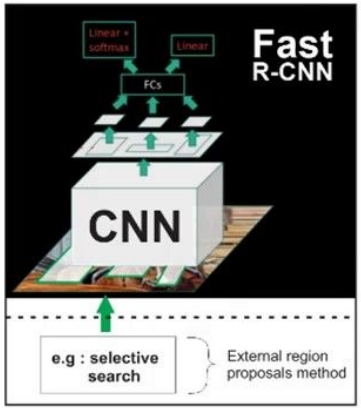


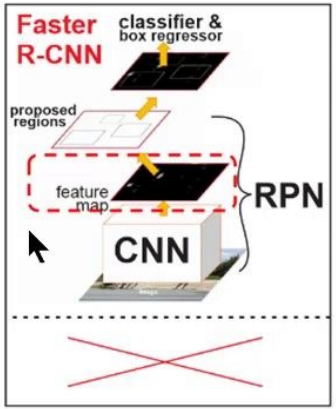
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

R-CNN variants

machinalis







	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy

By Ardian Umam

Open in Colaboratory:

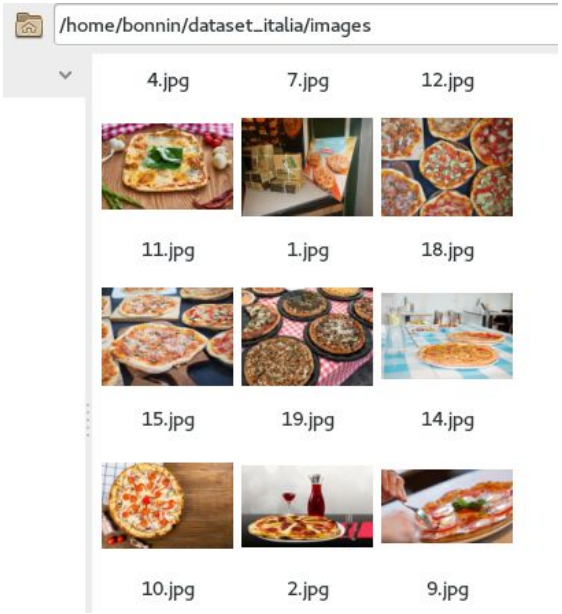
https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/4A_Object_detection.ipynb

machinalis

Training your own

Pizza detector with Faster-RCNN

Training setup



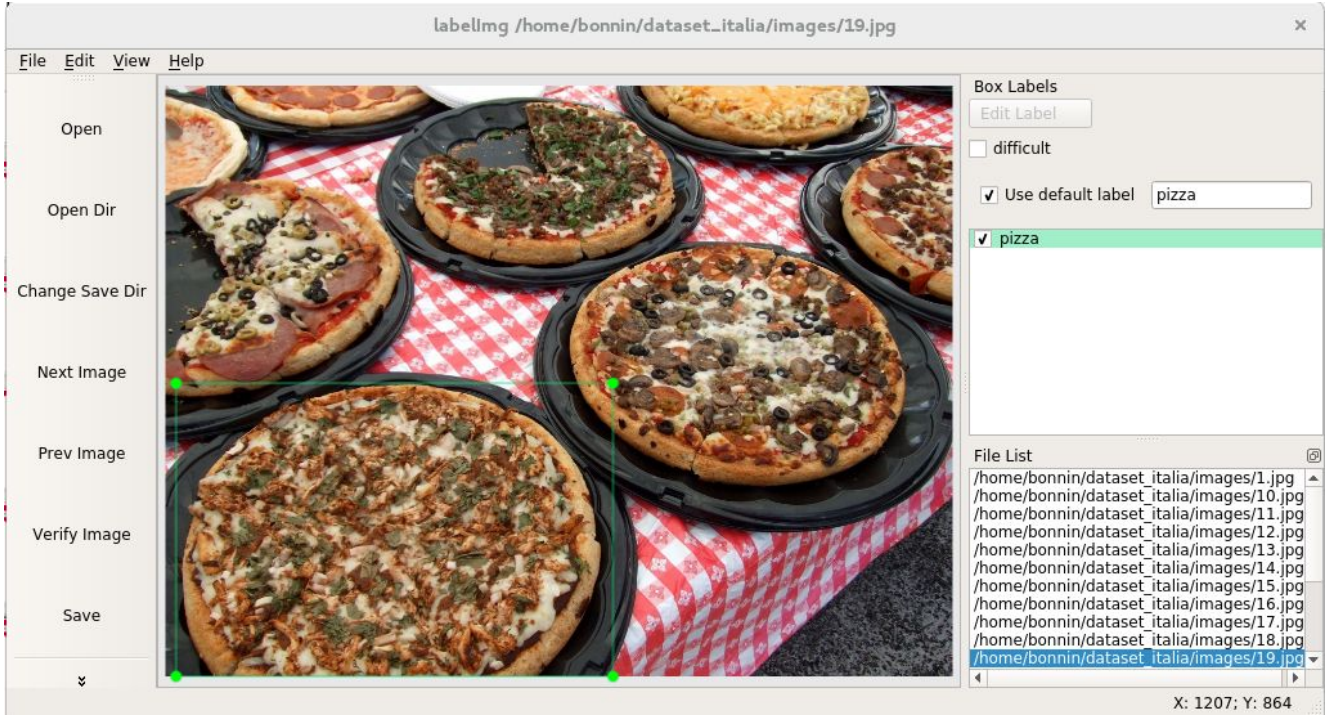
Training:
object_detection.pbtxt

```
item {
  id: 1
  name:
'pizza'
}
```

Data:
Empty(before training)

Building the dataset: Labeling the images

```
sudo apt-get install pyqt5-dev-tools
sudo pip3 install lxml
make qt5py3
python3 labelImg.py
```



Model Training and final object snapshot saving

```
sudo apt-get install -y python-pip
sudo apt-get install -y protobuf-compiler python-pil python-lxml python-tk unzip
sudo pip install Cython
sudo pip install jupyter
sudo pip install matplotlib
sudo pip install pandas
sudo pip install tensorflow

git clone https://github.com/tensorflow/models.git
cd models
git checkout 079d67d9a0b3407e8d074a200780f3835413ef99
cd research
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
protoc object_detection/protos/*.proto --python_out=.
python setup.py build
python setup.py install
cd object_detection
wget https://s3.amazonaws.com/italia18/dataset_italia_final.zip
unzip dataset_italia_final.zip

wget http://download.tensorflow.org/models/object_detection/faster_rcnn_inception_v2_coco_2018_01_28.tar.gz
tar -zxvf faster_rcnn_inception_v2_coco_2018_01_28.tar.gz

python xml_to_csv.py

python generate_tfrecord.py --csv_input=images/train_labels.csv --image_dir=images/train --output_path=train.record
python generate_tfrecord.py --csv_input=images/test_labels.csv --image_dir=images/test --output_path=test.record

python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.config
```

Demo Time: Inference of our trained model on test images

machinalis

https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/Object_detection_trained_model.ipynb

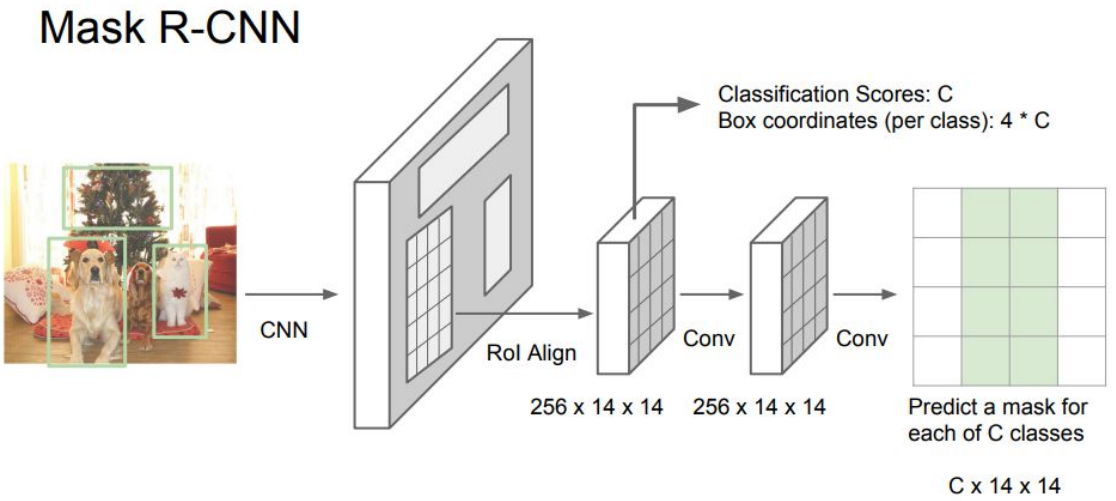
machinalis

Image Segmentation

Mask-RCNN

Adds a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object.
It's a Fully Convolutional Network on top of a CNN based feature map.

Inputs: CNN Feature Map.
Outputs: Matrix with 1s on all locations where the pixel belongs to the object and 0s elsewhere (this is known as a binary mask).



It consists of complex everyday scenes containing common objects in their natural context.
Objects are labeled using per-instance segmentations to aid in precise object localization.
Its canonical dataset has :

- **330K images (>200K labeled)**
- **1.5 million object instances**
- **80 object categories**
- **91 stuff categories**
- **5 captions per image**
- **250,000 people with keypoints**

The most common tasks applied to this dataset are:

- **Object segmentation ✓**
- **Recognition in context**
- **Superpixel stuff segmentation**

MSCOCO general format

machinalis

Coco json sample format:

```
{
"info" : info, "images" : [image], "annotations" : [annotation],
"licenses" : [license],
}

info{
"year" : int, "version" : str, "description" : str, "contributor" : str,
"url" : str, "date_created" : datetime,
}

image{
"id" : int, "width" : int, "height" : int, "file_name" : str, "license"
: int, "flickr_url" : str, "coco_url" : str, "date_captured" : datetime,
}

license{
"id" : int, "name" : str, "url" : str,
}
```

MSCOCO Annotation format

machinalis

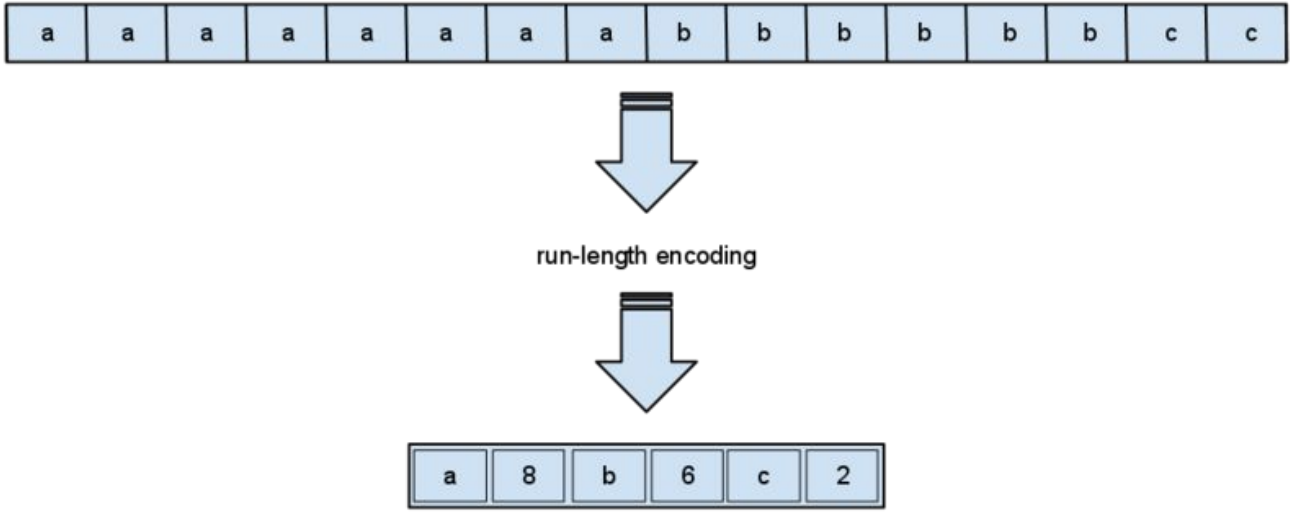
Coco json annotation sample:

```
annotation{
"id" : int, "image_id" : int, "category_id" : int, "segmentation" : RLE
or [polygon], "area" : float, "bbox" : [x,y,width,height], "iscrowd" : 0
or 1,
}

categories[{
"id" : int, "name" : str, "supercategory" : str,
}]
```

MSCOCO Annotation format

machinalis

RLE



Demo time: Evaluating a pre-trained COCO model

machinalis

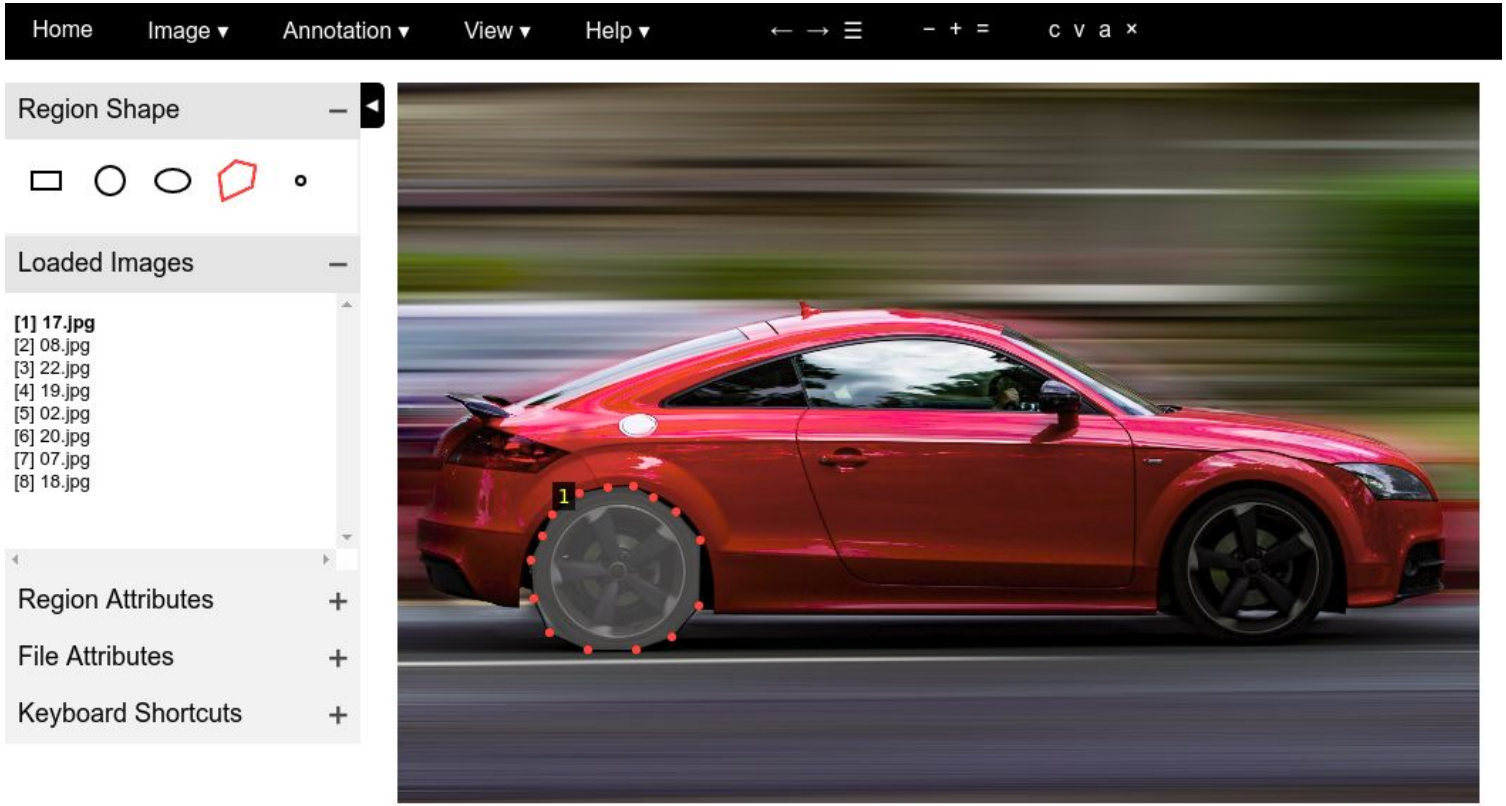
https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/5A_Object Segmentation.ipynb

machinalis

Training a new Segmentation model.

Mask-RCNN: Training a custom model: Tire segmenter

machinalis



Mask-RCNN: Training the model

machinalis

https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/5B_Object_segmentation_training.ipynb

Segmenting test images

machinalis

https://colab.research.google.com/github/rnditdev/PyData_London_2018_Computer_Vision/blob/master/5B_Object_segmentation_training.ipynb

machinalis

The future

Questions?