

Detectron object detection

By Facebook



DETECTRON

- Detectron is Facebook AI Research's (FAIR) software system
- Implements state-of-the-art object detection algorithms, including Mask R-CNN.
- It is written in Python and powered by the Caffe2 deep learning framework.



GOAL OF DETECTRON

The goal of Detectron is to provide a

- high-quality,
- high-performance
- codebase for object detection research.

It is designed to be flexible in order to support rapid implementation & evaluation of novel research.

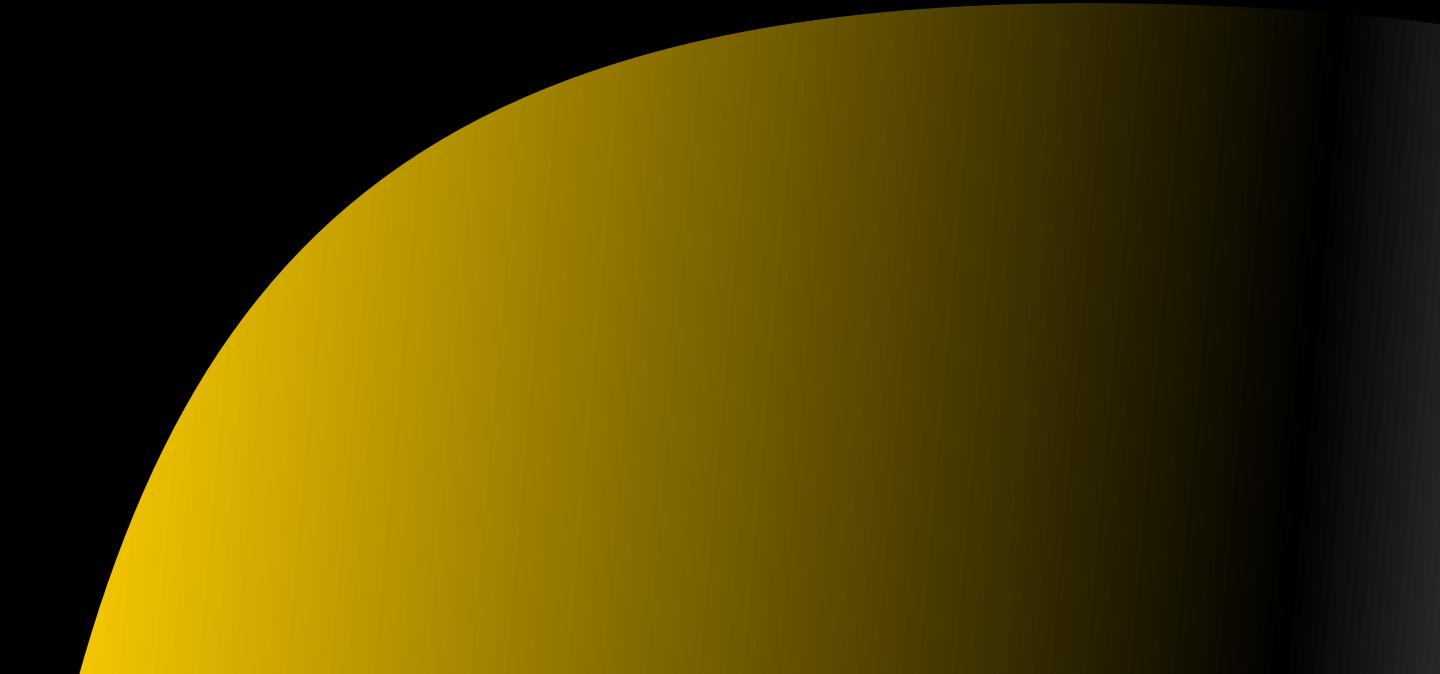


WHY DETECTRON

- High quality
- Industry standard codebase for object detection research
 - Incredibly accurate
 - Object related algorithms are embedded in Detectron
 - Leverages the Caffe2 deep learning framework underneath.
 - In a popular language, Python

"Detectron model is meant to advance object detection by offering speedy training and addressing the issues companies face when making the step from research to production "

– Mark Roosevelt 



DETECTRON includes
implementations of the
following object detection
ALGORITHMS:

- Mask R-CNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN



R-FCN

- A type of region-based object detector
-
- **Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.**
- The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.
- **Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.**
- **Mask R-CNN outperforms all existing, single-model entries on every task.**
- **Mask R-CNN is easy to generalize to other tasks.**



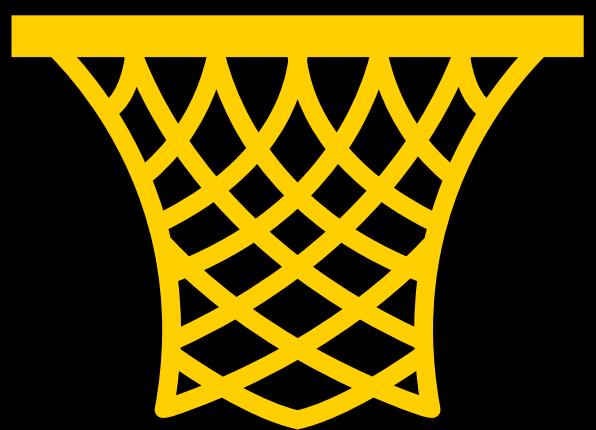
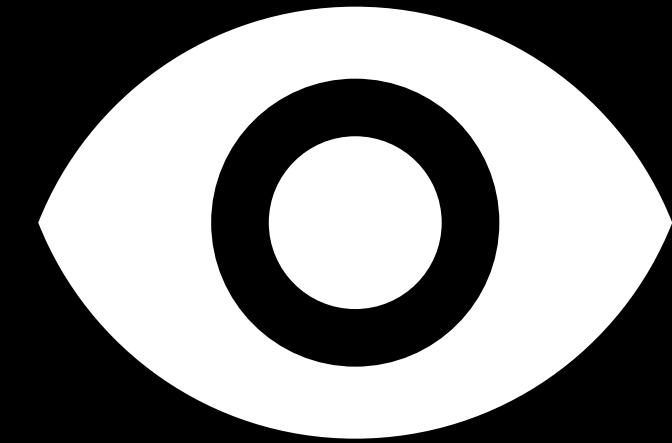
MASK R-CNN

- Conceptually simple & flexible framework for object instance segmentation.
- Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.
- The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.
- Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.
- Mask R-CNN outperforms all existing, single-model entries on every task.
- Mask R-CNN is easy to generalize to other tasks.



RETINANET

- One of the best, one-stage object detection models.
- Work well with dense and small scale objects.
- Formed by making two improvements over existing single stage object detection models -
 - a. Feature Pyramid Networks (FPN) .
 - b. Focal Loss.
- Using larger scales allows RetinaNet to surpass the accuracy of all two-stage approaches, while still being faster.
- Except YOLOv2 (which targets on extremely high frame rate), RetinaNet outperforms SSD, DSSD, R-FCN and FPN.
- Built on top of ResNet and is responsible for computing convolutional feature maps of an entire.



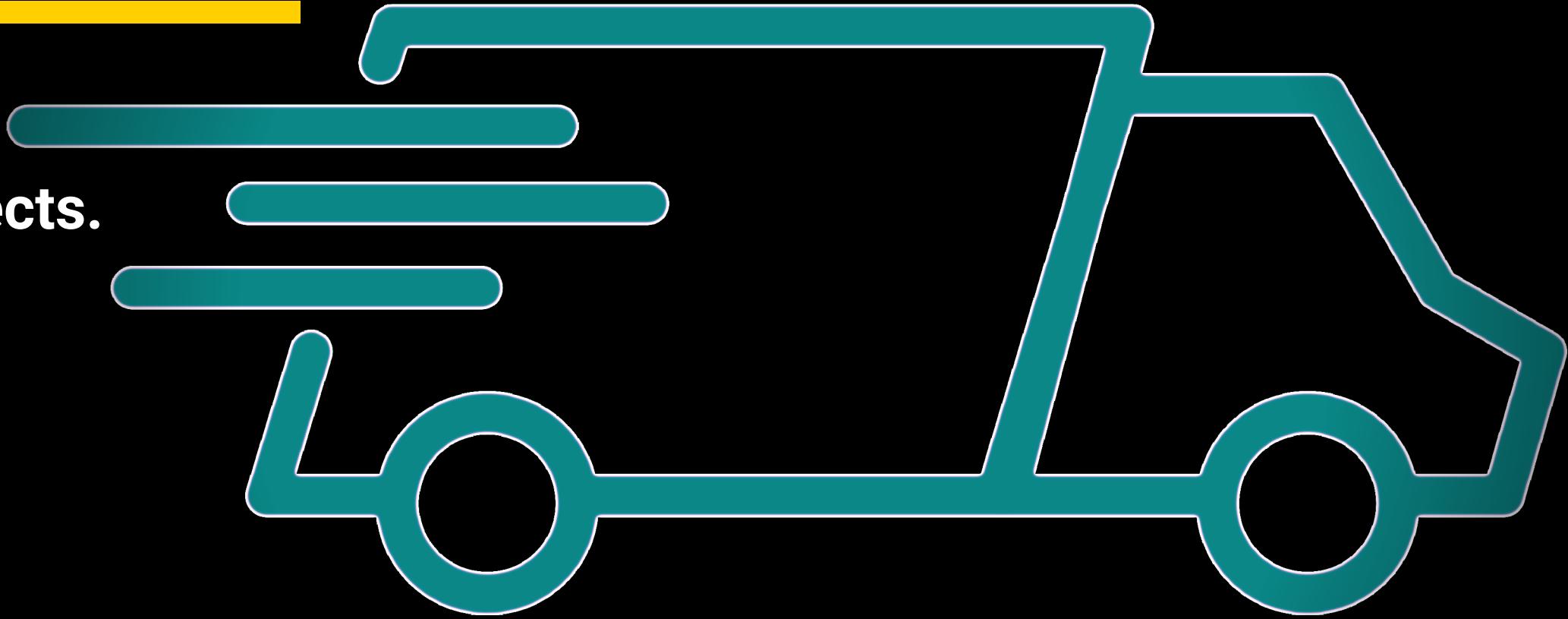
FAST R-CNN

- Instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map.
- From the convolutional feature map, we identify the region of proposals and warp them into squares .
- By using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer.
- From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.
- Uses selective search algorithm on the feature map to identify the region proposals is used to predict the region proposals.



FASTER R-CNN

- Work well with dense & small scale objects.
- Uses convolution neural networks like
 - a. YOLO (You Look Only Once)
 - b. SSD (Single Shot Detector)
- You don't have to feed 2000 region proposals to the convolutional neural network every time.
- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

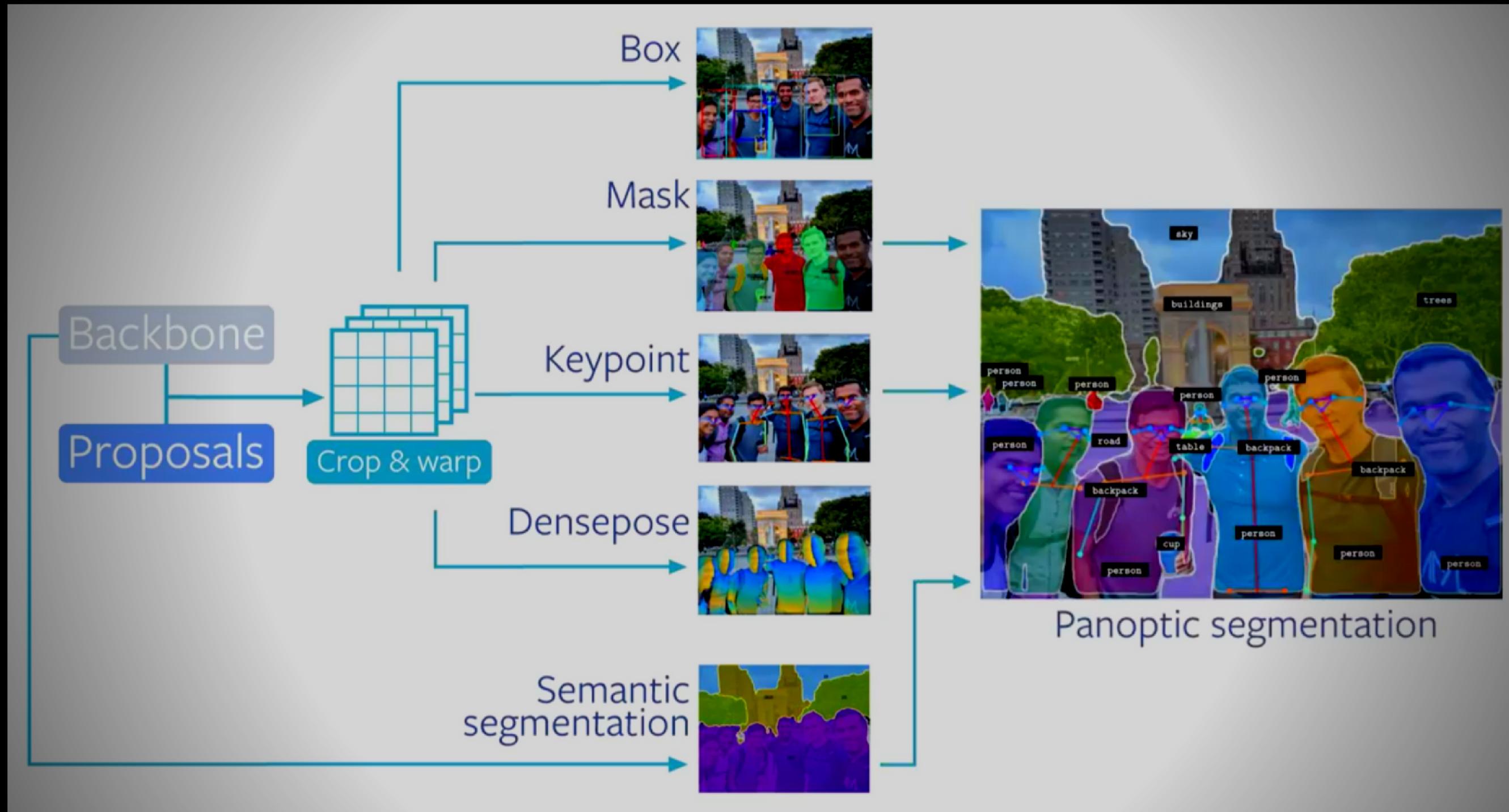


USING THE FOLLOWING BACKBONE NETWORK ARCHITECTURES:

- ResNeXt{50,101,152}
- **ResNet{50,101,152}**
- Feature Pyramid Networks
(with ResNet / ResNeXt)
- **VGG16**



FRAMEWORK OF DETECTRON



**WANT TO USE
DETECTRON?**

Lets make your dream come true !!!

- **STEP -1 -OPEN your python IDE & use the below command -**

!pip install 'git+https://github.com/facebookresearch/detectron2.git'

```
!pip install 'git+https://github.com/facebookresearch/detectron2.git'
# (add --user if you don't have permission)

Collecting git+https://github.com/facebookresearch/detectron2.git
  Cloning https://github.com/facebookresearch/detectron2.git to /tmp/pip-req-build-brrbyhcg
    Running command git clone -q https://github.com/facebookresearch/detectron2.git /tmp/pip-req-build-brrbyhcg
Requirement already satisfied: Pillow>7.1 in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (7.1.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (3.2.2)
Requirement already satisfied: pycocotools>2.0.2 in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (2.0.2)
Requirement already satisfied: termcolor>1.1 in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (1.1.0)
Collecting yacs>0.1.6
  Downloading https://files.pythonhosted.org/packages/28/4f/f9e4ad472a867878ce3bb7efb16654c5d63672b80dc8e6e953a67018433/yacs-0.1.8-py3-none-any.whl
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (0.8.9)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (1.3.0)
Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (4.41.1)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (2.5.0)
Collecting fvcore>0.1.6,>=0.1.5
  Downloading https://files.pythonhosted.org/packages/e6/69/805702ba4c2ae87ab887665619f8478a18ad8aa92efdf7c28e8761c72a6b/fvcore-0.1.5.post20210624.tar.gz (49kB)
    51KB 4.3MB/s
Collecting iopath>0.1.9,>=0.1.7
  Downloading https://files.pythonhosted.org/packages/21/d9/22104caed16fa41382792fed959f4a9b88b2f905e7a82e4483180a2ec2a/iopath-0.1.8-py3-none-any.whl
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (0.16.0)
Requirement already satisfied: pydot in /usr/local/lib/python3.7/dist-packages (from detectron2==0.4.1) (1.3.0)
Collecting omegaconf>2.1.0rc1
  Downloading https://files.pythonhosted.org/packages/f9/96/1966b49bf6ca64bfadfa7bcc9a8d73c5d83b4be769321fcc5d617abeb8c/omegaconf-2.1.0-py3-none-any.whl (74kB)
    81KB 6.9MB/s
    2m 16s completed at 12:35 AM
```

WARNING: The following packages were previously imported in this runtime:
[pydevd_plugins]
You must restart the runtime in order to use newly installed versions.

RESTART RUNTIME

**YOU WILL SEE YOUR IDE GOING CRAZY !!!
BUT DON'T GET SCARED !!
KEEP CALM & TRUST THE DETECTRON**

PS: Just installing detectron2

Restart your runtime, and run it, you will get the below message

Successfully built detectron2



- **STEP -2 -USE the below command & import all the libraries required -**

```
# install dependencies:  
!pip install pyyaml==5.1 pycocotools>=2.0.1  
import torch, torchvision  
print(torch.__version__, torch.cuda.is_available())  
!gcc -v  
# opencv is pre-installed on colab  
  
# install detectron2: (colab has CUDA 10.1 + torch 1.6):  
# See https://detectron2.readthedocs.io/tutorials/install.html for instructions  
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html
```

Libraries Required

- **CUDA**
- **PyTorch**
- **Torchvision**
- **detectron2**
- **Numpy**
- **JSON**
- **OpenCV**
- **random**

```
# Some basic setup:  
# Setup detectron2 logger  
import detectron2  
from detectron2.utils.logger import setup_logger  
setup_logger()  
  
# import some common libraries  
  
import numpy as np  
import os, json, cv2, random  
from google.colab.patches import cv2_imshow  
  
# import some common detectron2 utilities  
  
from detectron2 import model_zoo  
from detectron2.engine import DefaultPredictor  
from detectron2.config import get_cfg  
from detectron2.utils.visualizer import Visualizer  
from detectron2.data import MetadataCatalog, DatasetCatalog
```

• **STEP -3 -Detectcting & Labelling of Image -**

```
#getting Image from COCO dataset
!wget http://images.cocodataset.org/val2017/000000439715.jpg -q -O input.jpg
im = cv2.imread("/content/image.jpg")
cv2_imshow(im)

#Creating a detectron2 config and a detectron2 `DefaultPredictor` to run inference on this image.
cfg = get_cfg()
# add project-specific config (e.g., TensorMask) here if you're not running a model in detectron2's core library
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5 # set threshold for this model
# Find a model from detectron2's model zoo. You can use the https://dl.fbaipublicfiles... url as well
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
outputs = predictor(im)

# look at the outputs. See https://detectron2.readthedocs.io/tutorials/models.html#model-output-format for specification
print(outputs["instances"].pred_classes)
print(outputs["instances"].pred_boxes)

#Last and final step is to visualize our processed image.
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

REFERENCE LINKS FOR YOU

Facebook's AI team Releases Detectron – A Platform for Object Detection Research

www.analyticsvidhya.com/blog/2018/01/facebook-launched-detectron-platform-object-detection-research/

Image Labelling Using Facebook's Detectron

<https://towardsdatascience.com/image-labelling-using-facebooks-detectron-4931e30c4d0c>

THANK YOU

- **STEP - 4 -Manipulating the Image**

CARTOONING AN IMAGE



OPEN CV

Free Icons

Use these elements in your
Canva presentation.

