Deep Learning
Lecture 9

Object Detection
Object Segmentation

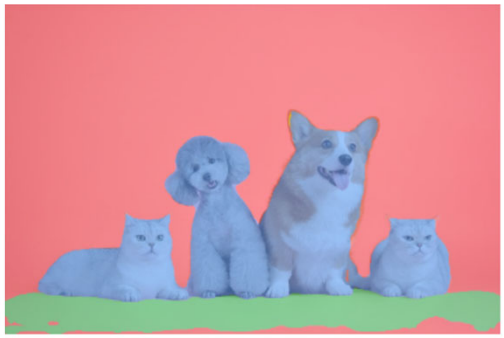
Prof. Dr. Rainer Herrler
Phone: 09721/ 940-8710
Email: rainer.herrler@fhws.de

Pictures from Wikipedia / Pixabay
Some Pictures generated with Stable Diffusion

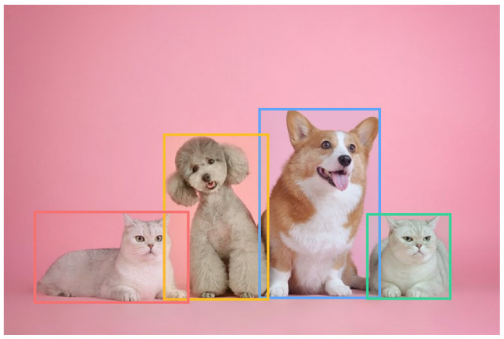
Object Detection

- One Task with two subtasks
  - Classification
  - Object Localization
- Localization is typically done by a “bounding box”
- More sophisticated approach is called Image Segmentation

Definition of terms



Semantic Segmentation: Pixelwise assignment to classes



Object Detection: Multiple Instances can be found in one Image, Location is given by a bounding box

Instance Segmentation: Multiple Instances can be found and are marked by a segmentation map

thws

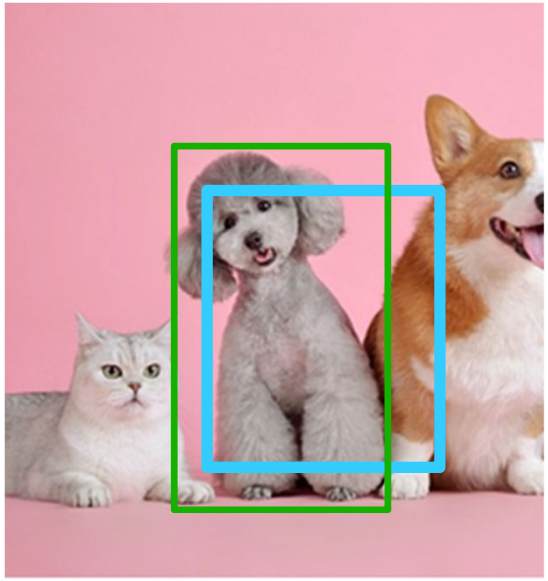
Excursion: Huggingface Hub

- Huggingface Hub is a website that hosts code repositories, datasets and models.
- The website provides a good overview on ML tasks and different models.
- Many models can be tested directly at the huggingface
- Examples:
  - Object detection https://huggingface.co/tasks/object-detection
  - Segmentation: https://huggingface.co/tasks/image-segmentation

What is needed for Object Detection

- Suitable Labeled data
  - Classes and Regions have to be specified by the user
  - Tool Support
- Label Representation
  - Bounding box Representation (Corners or Center and length/width
- Suitable Loss Function
  - A way to describe how close the prediction is to the data label
  - Loss needs to deal with multiple classes
  - Loss needs to deal with correct and position
- Network Structure/Algorithm

Different Losses for object detection



- Classification Loss
  - CrossEntropy Loss

- Bounding Box regression loss:
  - Measures the deviation of all coordinates
  - MSE or Huber Loss

$$huberloss(x,y) = \begin{cases} 0.5(x-y)^2, if\ |x-y| < 1 \\ |x-y| - 0.5, otherwise \end{cases}$$

- Intersection over Union Loss:
  - Measures the overlap between the ground truth bounding box and the predicted

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Huber Loss

- Besides MSE its also a common Loss for regression

$$huberloss(x,y) = \begin{cases} 0.5(x-y)^2, if\ |x-y| < 1 \\ |x-y| - 0.5, otherwise \end{cases}$$

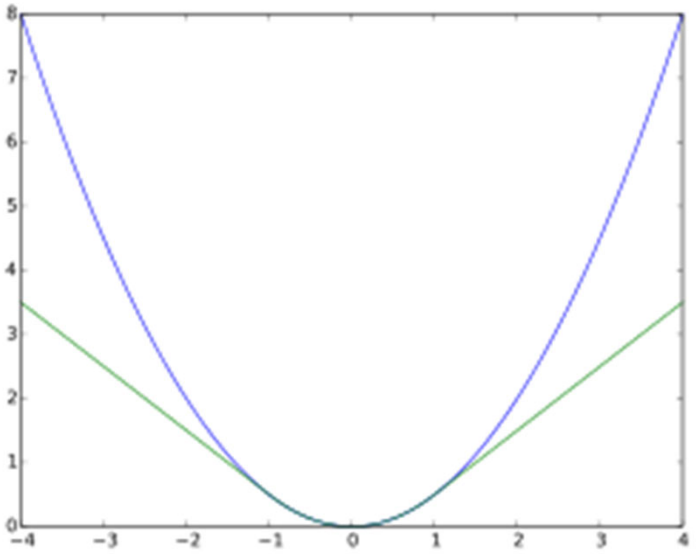
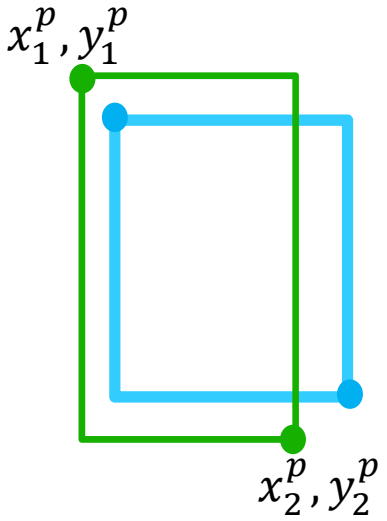


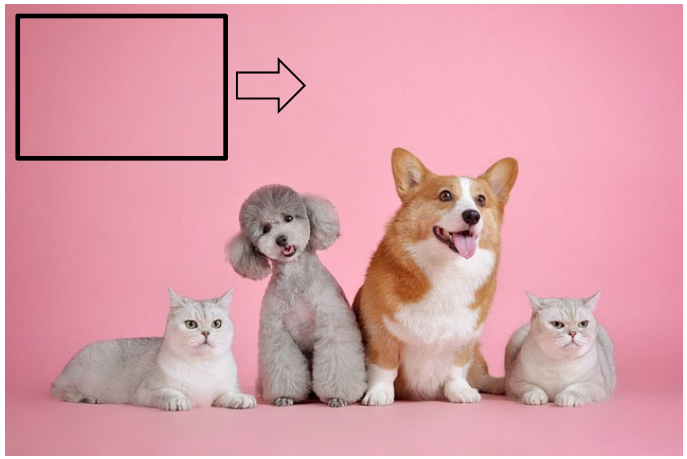
Image: Wikipedia

- It has less emphasis in strong outliers
- Also Called Smooth L1 Loss



- Regression variables:
  - Predicted rectangle: $x_1^p, y_1^p, x_2^p, y_2^p$
  - True label: $x_1^l, y_1^l, x_2^l, y_2^l$

Algorithm Approaches



- Sliding window
  - Shift an area of a certain size across the image
  - Apply a CNN on those areas
- Region based approaches (starting 2014)
  - Two Stages
    - Find an interesting region in the image first
    - then classify with an CNN
  - RCNN- Simple Region detection – Then CNN
  - FastRCNN – CNN First - Region Detection In Feature Maps,
  - FasterRCNN - Region proposal Network
- Single Shot Detectors (starting 2016)
  - Joint detection and classification
  - YOLO (You only look once)
  - Single Shot Multibox Detector (SSD)

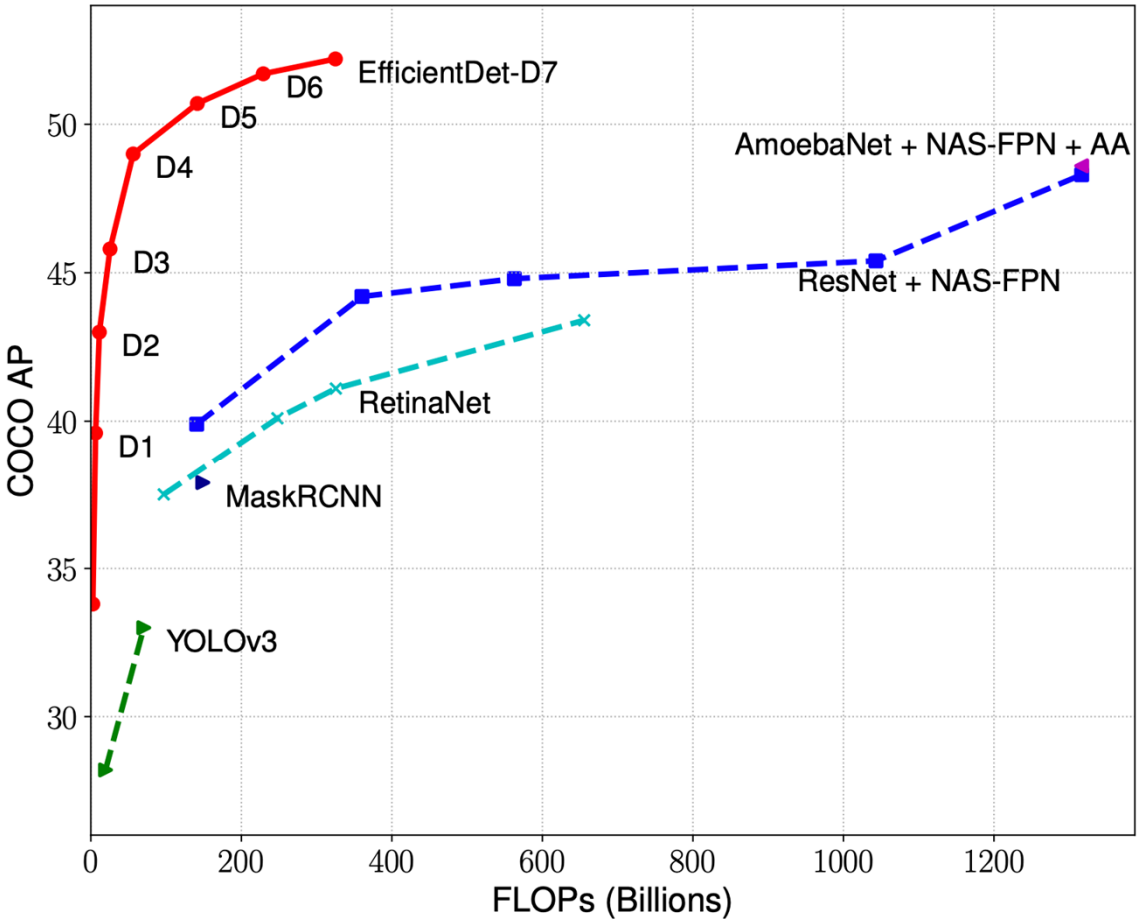
Age of the One-Stage Object Detection Algorithms

- YOLO (2016)
- SSD (2016)
- RetinaNet (2017)
- YOLOv3 (2018)
- EfficientDet (2020) Google Brain / best Model in Tensorflow
- YOLOv4 (2020)
- YOLOR (2021)
- YOLOv7 (2022)
- YOLOv8 (2022) Ultralytics /
- YOLO-NAS (2023) Deci-AI / Just for Pytorch Currently

Further readings:

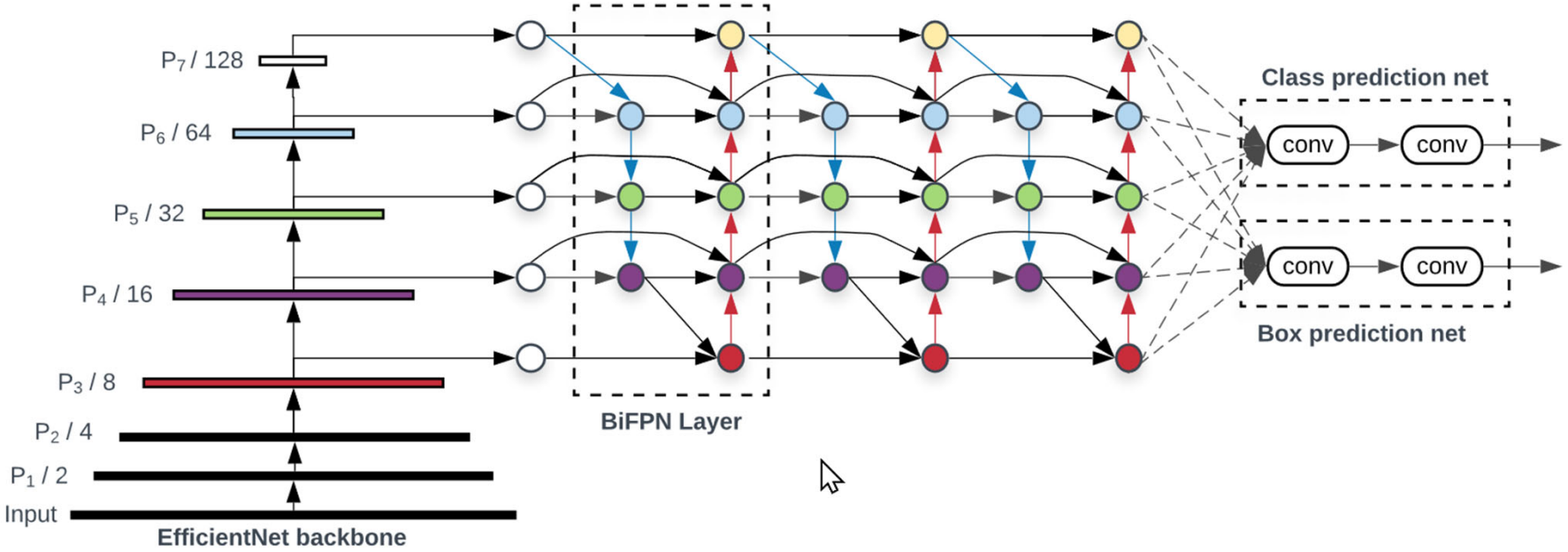
https://viso.ai/deep-learning/object-detection/

Performance Comparison SOTA 2020



https://arxiv.org/pdf/1911.09070.pdf

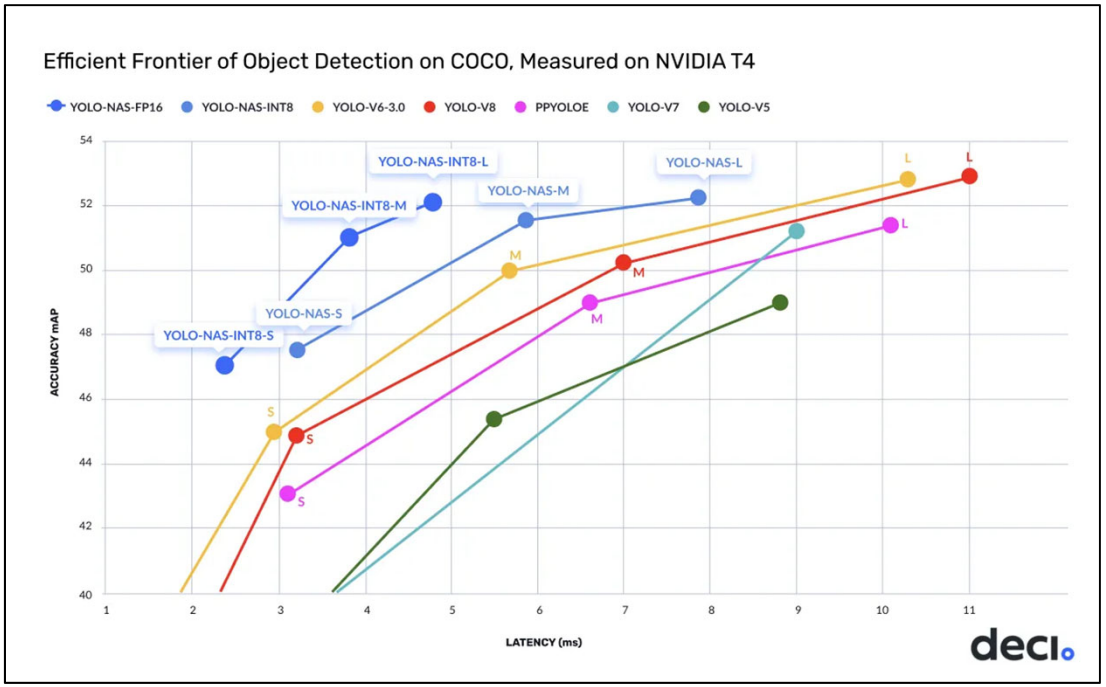
Network Architecture EfficientDet



EfficientNet Backboan outputting Features at different scales

Bidirectional Feature Pyramid Network for Feature Fusions

Networks to derive classification and box prediction

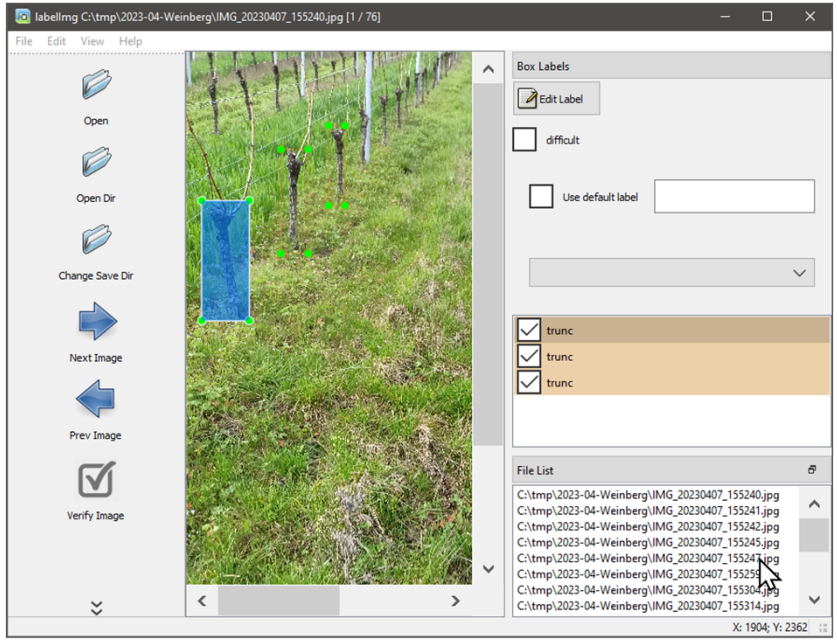


Source: Deci-AI. License: Apache License 2.0

Newest Model is YOLO NAS even outperforming all previous models!
Faster but just available as a Pytorch based implementation

Tools for Labeling your Data

```
conda install -c conda-forge labelimg
labelImg
```



- labelImg
  - Opensource
  - Easy to install
  - Create an Environment with python==3.9
- Roboflow Annotate
  - Commercial service with free plan
  - With the free plan your data is published
  - Many formats supported
- CVAT (intel/openCV)
  - Computer Vision Annotation Tool
  - Open Source / Web Based (can be used a s hosted Service or self-hosted)
- Many Others:
  - E.g . Open Labeling, Labelbox, VoTT (Microsoft)…
  - Comparison: https://squareshade.com/de/2021/01/09/best-open-source-image-annotation-tool-in-2021/

Data formats for Labeling your Data

- PASCAL VOC
  - Created for the Pascal VOC Challange
  - One XML per image
- YOLO
  - Different Versions
  - yaml (central description)
  - One txt file per image (classes + coordinates)
- CreateML
  - Apple 2018
- TFRecord
  - Supported by Tensorflow Object Detection API)
  - Binary format but there is a csv based preformat as well
- More about Formats:
  - https://roboflow.com/formats

```
<annotation>
  <folder></folder>
  <filename>000001.jpg</filename>
  <path>000001.jpg</path>
      <source> <database>Vineyard</database>
      </source>
      <size>
         <width>600</width>
         <height>400</height>
         <depth>3</depth>
      </size>
      <segmented>0</segmented>
      <object>
         <name>Trunk</name>
         <pose>Unspecified</pose>
         <truncated>0</truncated>
         <difficult>0</difficult>
         <occluded>0</occluded>
            <bndbox>
               <xmin>179</xmin>
               <xmax>231</xmax>
               <ymin>85</ymin>
               <ymax>144</ymax>
            </bndbox>
      </object>
</annotation>
```

Pacal VOC Example

```
1  path: ../datasets/coco128  # dataset root dir
2  train: images/train2017  # train images
3  val: images/train2017  # val images
4  test:  # test images (optional)
5
6  # Classes
7  names:
8    0: person
9    1: bicycle
10   2: car
11   3: motorcycle
```

Yolo Yaml Example

Comparing the two networks

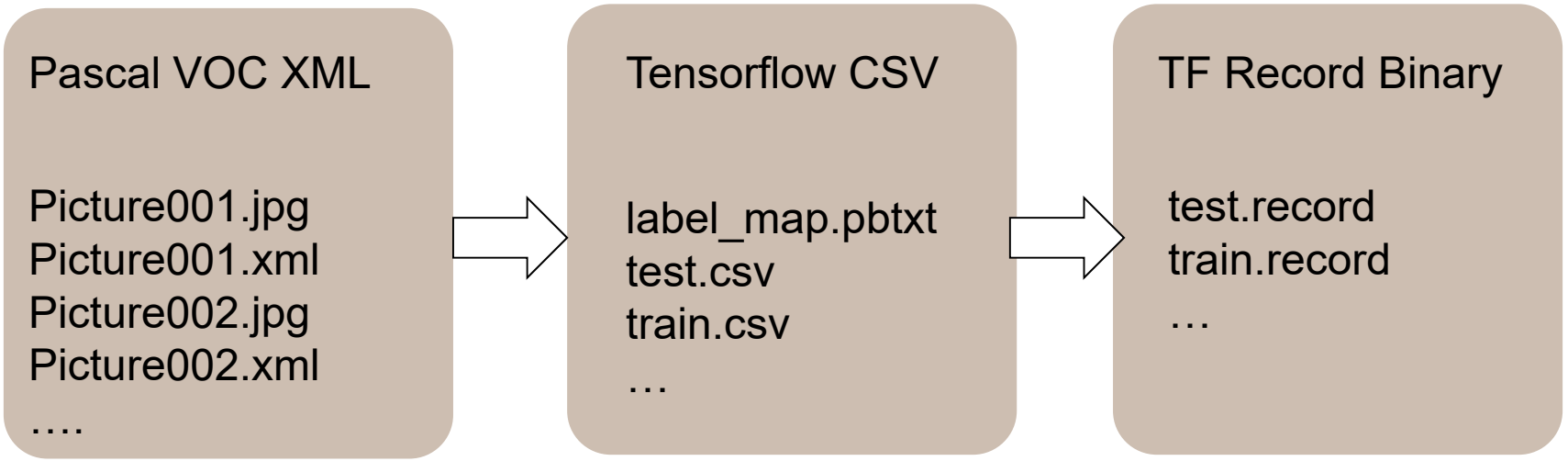
- EfficientDet
  - Efficient and fast Model with high accuracy
  - Fits best into the Tensorflow Ecosystem
  - Available in Tensorflow model zoo.
  - Algorithms can be swaped out easily

- Yolo v8 or YOLO NAS
  - YOLO is short for “You Only look once”
  - NAS stands for “Neural Architecture Search”
  - Many new versions on the last years improving performance
  - SOTA Model, outperforms previous models
  - https://github.com/roboflow/notebooks/blob/main/notebooks/train-yolo-nas-on-custom-dataset.ipynb

Scenario with Tensorflow API

- Data Preparation
  - Get Data or Label Data yourself
  - Convert to TFRecord (e.g. from Pascal VOC XML to Tensorflow csv)
- Install Modell
  - Clone the tensorflow git repository & Install TensorFlow Object Detection API
  - Download Pretrained Model from Model Zoo (e.g MobileNet or EfficientNet)
- Edit Settings in Pipeline Config file
  - Specify where to find the data
  - How many target classes
  - Initial Model Checkpoint and where to save intermediate results
  - …
- Start Training with a given script `model_main_tf2.py`
- Load last Checkpoint and Export Inference Model

Data preparation for the Tensorflow API




```
<object>
    <name>Scissors</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <occluded>0</occluded>
    <bndbox>
        <xmin>138</xmin>
        <xmax>532</xmax>
        <ymin>322</ymin>
        <ymax>480</ymax>
    </bndbox>
</object>
```

filename	width	height	class	xmin	ymin	xmax	ymax
20220216_221550_jpg	640	640	Scissors	138	322	532	480
20220216_221819_jpg	640	640	Rock	265	283	438	436
20220216_221856_jpg	640	640	Rock	239	316	451	465
20220216_222153_jpg	640	640	Paper	122	369	457	493
8ddb9b104a99c82.jpg	640	640	Scissors	1	31	115	293

https://colab.research.google.com/drive/1soqqi93MSBzv13GKLaLh5o-OwzlSALkW?usp=sharing

Installing and finetuning a TF model

- Install Modell
  - Clone the tensorflow git repository & Install TensorFlow Object Detection API
  - Download Pretrained Model from Model Zoo (e.g MobileNet or EfficientNet)
- Edit Settings in Pipeline Config file
  - Specify where to find the data
  - How many target classes
  - Initial Model Checkpoint and where to save intermediate results
  - …
- Start Training with a given script `model_main_tf2.py`
- Load last Checkpoint and Export Inference Model

`Original Post:`

- https://medium.com/analytics-vidhya/training-a-model-for-custom-object-detection-tf-2-x-on-google-colab-4507f2cc6b80

Adapted Version:

- https://colab.research.google.com/drive/1uHHdTLlb9-L33p1GWriNTLrpkT8TPT7m?usp=sharing

Using the most recent model YOLO NAS

- Installation of libs
  - supergradient, roboflow, supervision
- Instanciate model with supergradient
  - models.get(architecture, numclasses, weights|checkpoint)
- Test inference
  - model.predict(picture,confidence)
- Retrive Data via roboflow library
- Finetuning
  - trainer.train(…)
- Test & Evaluate

`Original Post:`

- https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolo-nas-on-custom-dataset.ipynb

`Adapted Version:`

- https://colab.research.google.com/drive/17PgoWHRJFUAf1Sq3HZ9610irx92MBf5T?usp=sharing

Zero Shot Object Detection and Segmentation

- Zero Shot Object Classification
  - OpenAI Clip
- Zero Shot Object Detection
  - OWL-ViT
  - Grounding Dino
- Zero shot segmentation
  - Segment Anything
- Combination
  - Grounded Segment Anything
  - https://huggingface.co/spaces/yizhangliu/Grounded-Segment-Anything