Overview of Deep Learning-based segmentation architectures

Petr Shulzhenko

shulzhenko.pa@phystech.edu
@petr_shul

girafe ai

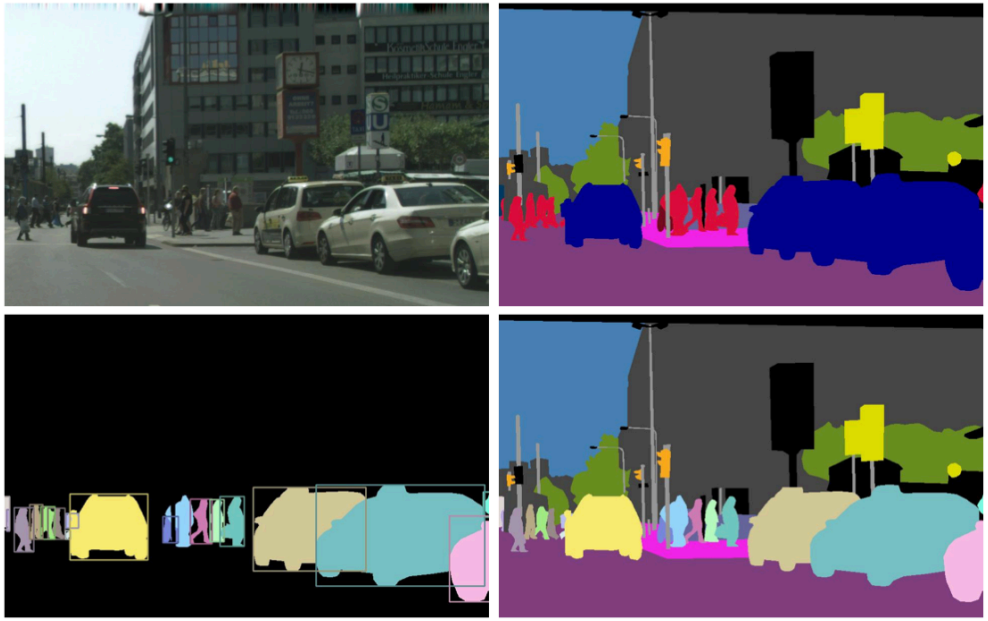
Presentation outline

- Segmentation problem definition
- Key metrics
- Overview of architecture classes
  - Naive → Fully Convolutional → U-Net
  - Object Detection-based
  - Modern approaches
- Related topics
  - CPU inference, annotation acceleration, class imbalance and synthetic data

Segmentation problem definition

Main types:

- Semantic
- Instance
- Panoptic



Examples of semantic, segmentation, and panoptical segmentation maps from Cityscapes dataset [1]

Segmentation problem definition

Image (+ classes)

Same-size image

with highlighted pixels

Segmentation problem definition

Image + Annotation mask:

- Polygon

  image.txt

  [<class-index> <x1> <y1> ... <xn> <yn>]

  [….]

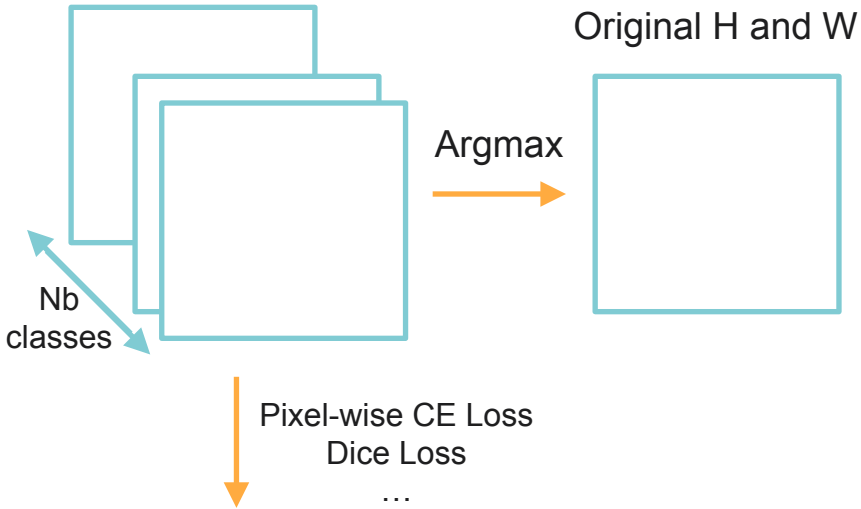
- Mask

  image in which the pixel

  values = class codes

Segmentation mask

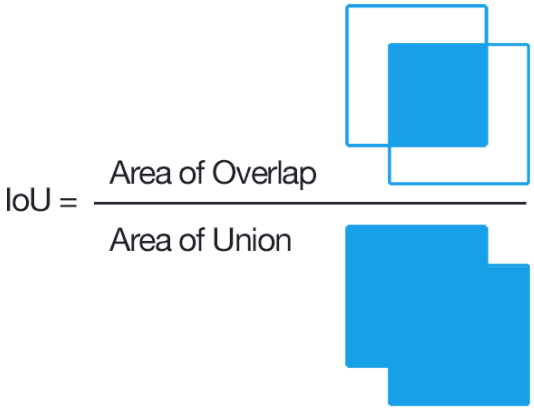
For example:



Key metrics

1.IoU - Intersection over Union

- takes FP and FN into account, but since they are in sum, this metric cannot measure their significance
- measures only the number of correctly labeled pixels, without taking into account how accurate the segmentation boundaries are



2. mIoU and FwIoU

- IoU averaged by classes / weighted by frequencies

$$IoU = \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k} n_{ij} + n_{ji} + n_{jj}}, i \neq j$$

girafe ai

[Image source]

Key metrics

3. AP - Average Precision

- Finding AP is done by finding the area under PRC via averaging the precision values obtained for certain selected values of recall.
- The simultaneous consideration of both shape and area under the PRC in a single number is a significant advantage of AP.
- Note in different datasets and different versions of datasets: interpolation method, mean by classes, threshold variation etc

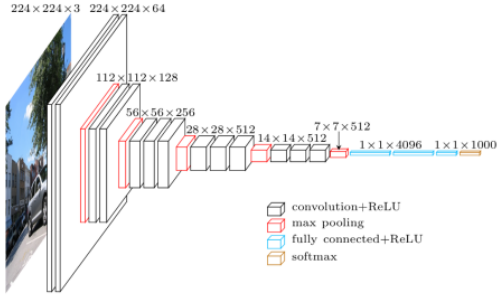
$$AP = \int_0^1 p(r)\, dr$$

$$AP = \frac{1}{n} \sum_{r=[r_1 \ldots r_n]} AP_r$$

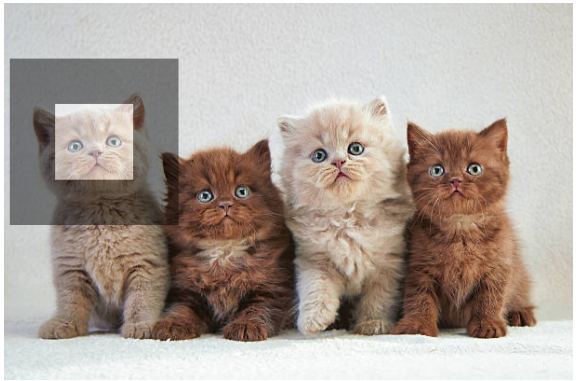
Overview of architectures

1.Naive approaches

Direct prediction via FC



Sliding window



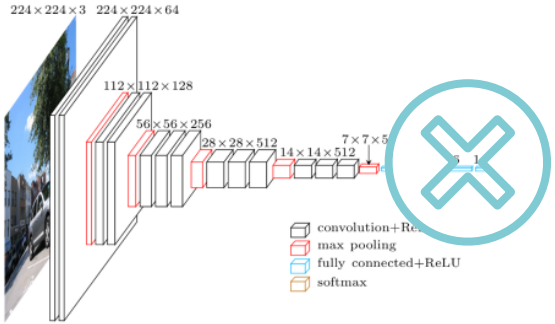
Extremely computationally inefficient, overfitting, poor performance

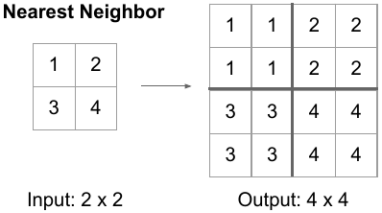
girafe ai

Image source

Overview of architectures

2. Fully Convolutional





Transposed convolution

$$O = (I - 1) * s + k - 2 * p$$

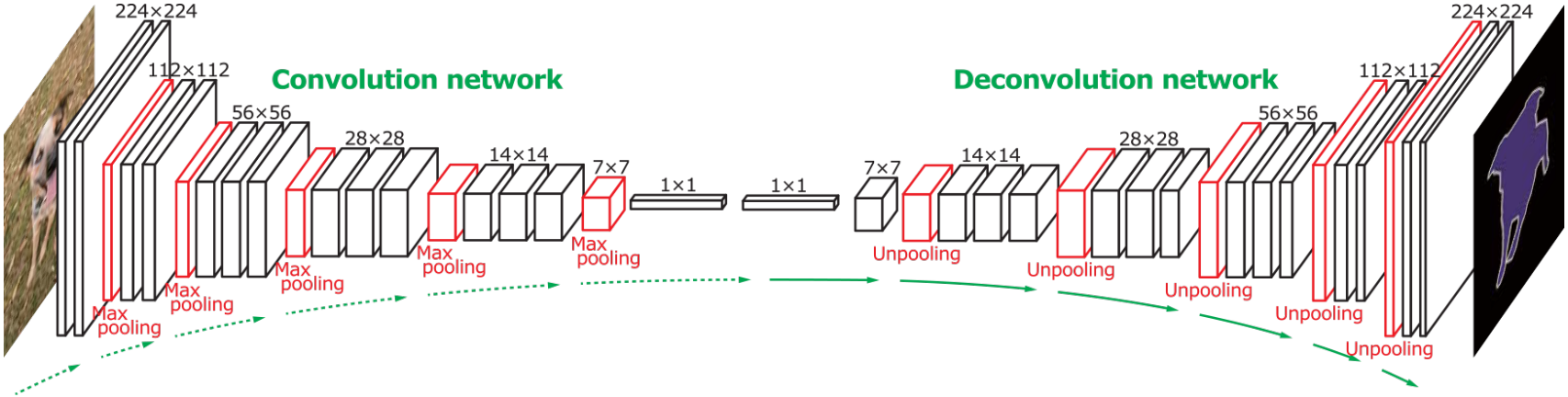
- Downsampling degrades spatial information
- Upsampling adds noise
- It is necessary to apply postprocessing methods (CRF..)

girafe ai

Image source

Overview of architectures

3. Encoder-Decoder architecture



- Spatial info loss - Dilated convolutions
- Scale dependence - Multi-Scale DilConv, Average Pooling, Spatial-Pyramid pooling

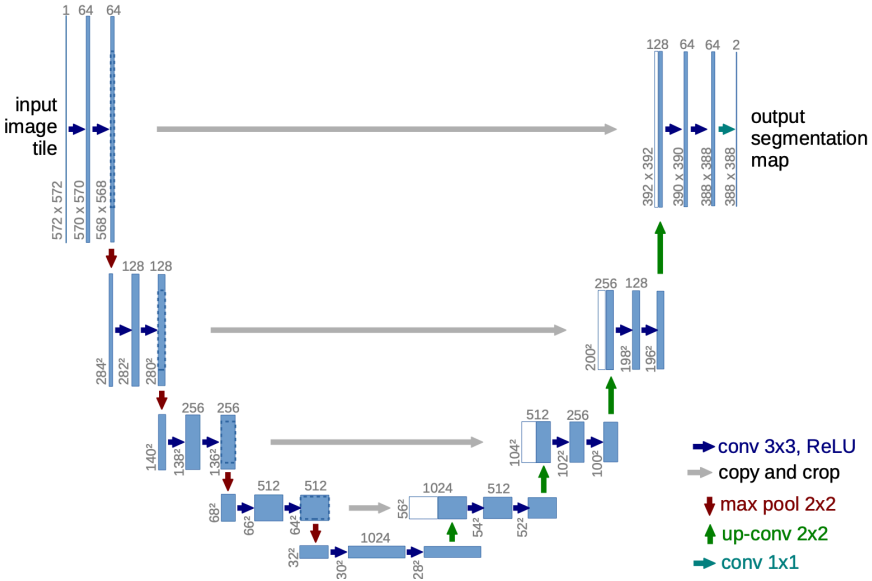
girafe ai

Image source

Overview of architectures

4. U-Net

- 72.7% mIoU on PASCAL2012 [2]
- +10% compared to FC nets
- Average speed



girafe ai

Image source

Overview of architectures

5. Object Detection-based

Mask R-CNN

- 37.1% mIoU on COCO dataset [3]
- 35.8% mAP on COCO dataset [3]
- 8 fps on GPU

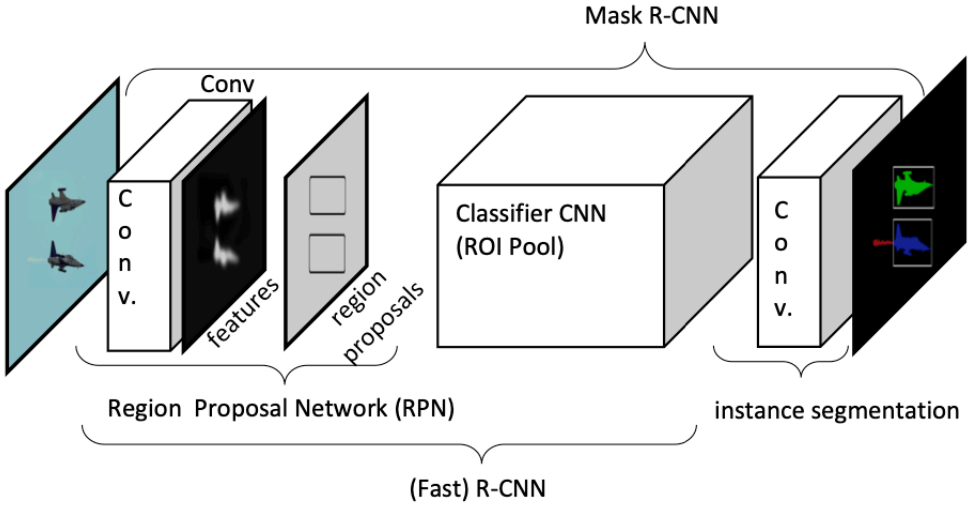


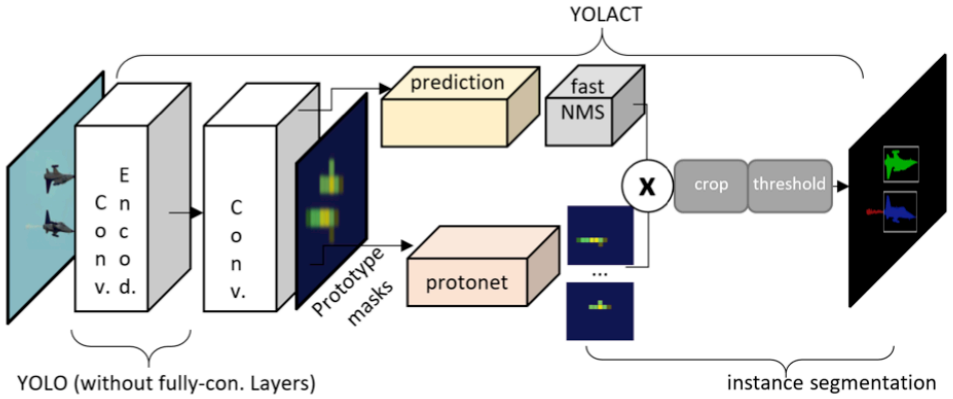
Image source

Overview of architectures

5.1. Object Detection-based

YOLACT

- 31.2% mAP on COCO dataset [3]
- ~ 30 fps



girafe ai

Image source

Overview of architectures

5.2. MultiNet

Single encoder for multiple tasks

- MultiNet
- Valeo JOSN - YOLOv2 / FCN
- YOEO - YOLOv4-tiny / U-Net

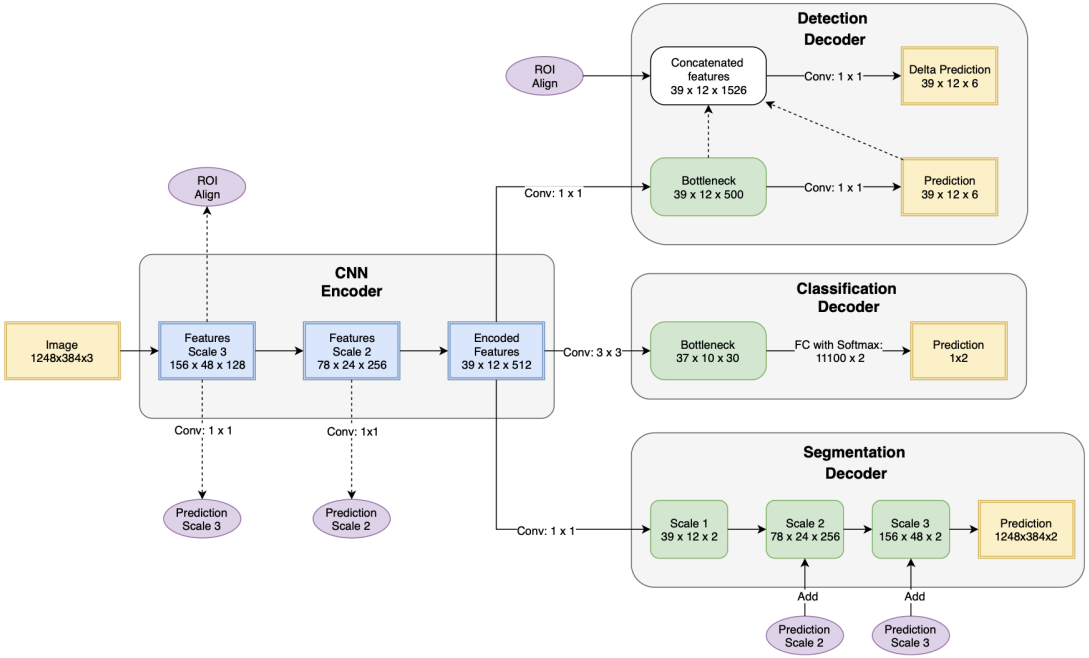
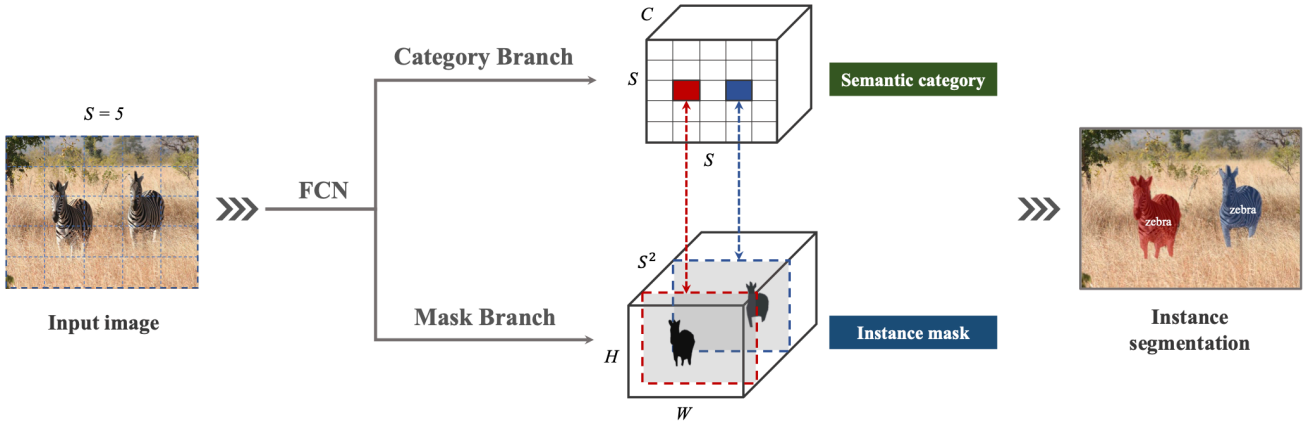


Image source

Overview of architectures

6. Modern approaches - SOLO



Instance segmentation - two simultaneous tasks:

- predicting a semantic category
- creating an instance mask

- 37.8% mAP on COCO [4]
- ~ 31.3 fps

girafe ai

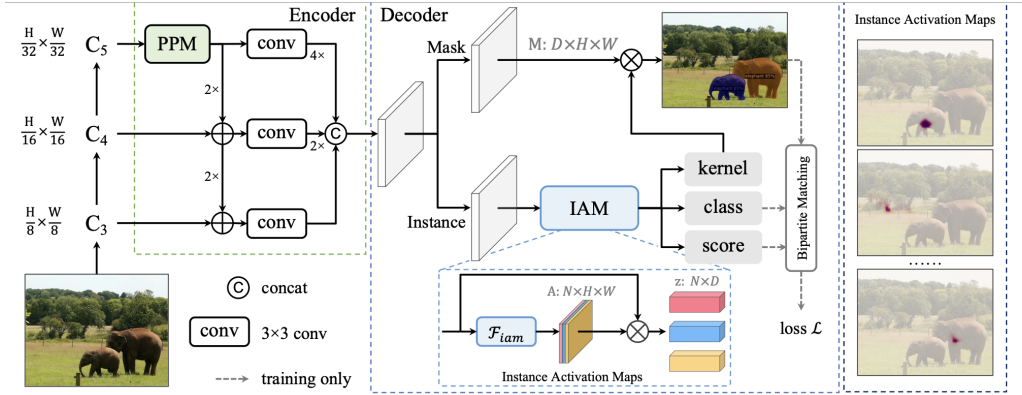
Image source

Overview of architectures

6. Modern approaches - SparseInst

Instance activation maps
instead of regions or centers

- 37.9% mAP on COCO [5]
- ~ 40 fps



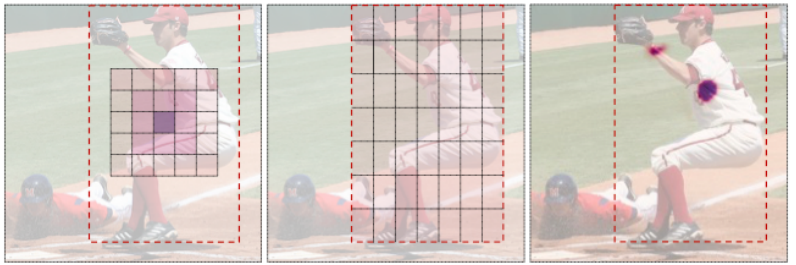


Image source

Overview of architectures

6. Modern approaches - BiSeNet / v2

BiSeNet v2
- 76.2% mIoU on Cityscapes [6]
- ~156 fps (GeForce GTX 1080 Ti)

BiSeNet
- 31.3 mIoU [7]
- ~105 fps (Titan XP)

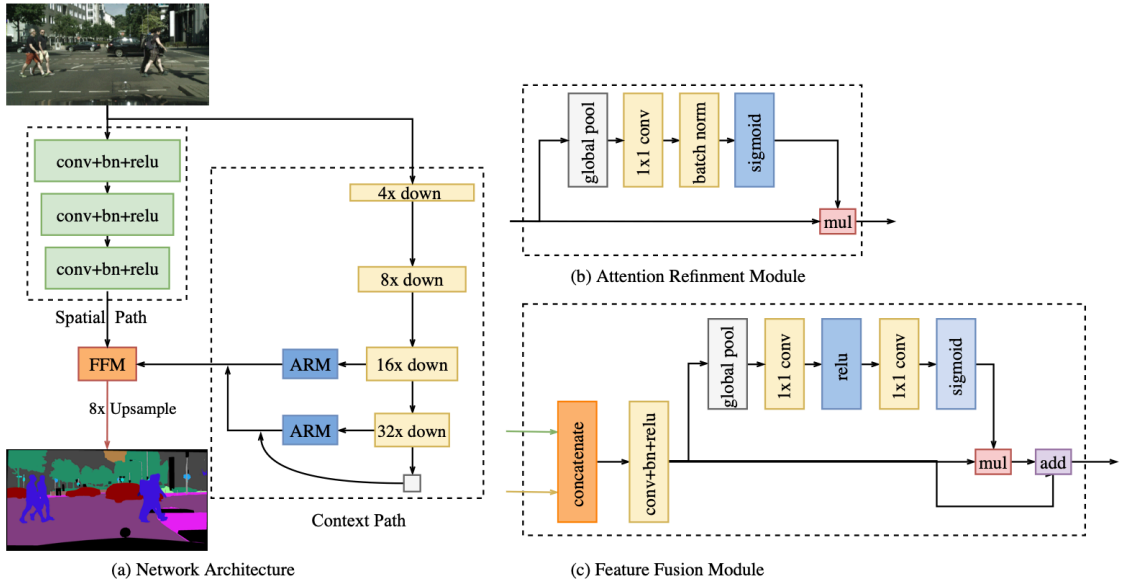


Image source

Overview of architectures

6. Modern approaches - Dual Attention Network

- 39.7% mIoU on COCO-stuff
- 80.4% mIoU on PASCAL VOC 2012
- 81.5% mIoU on Cityscapes

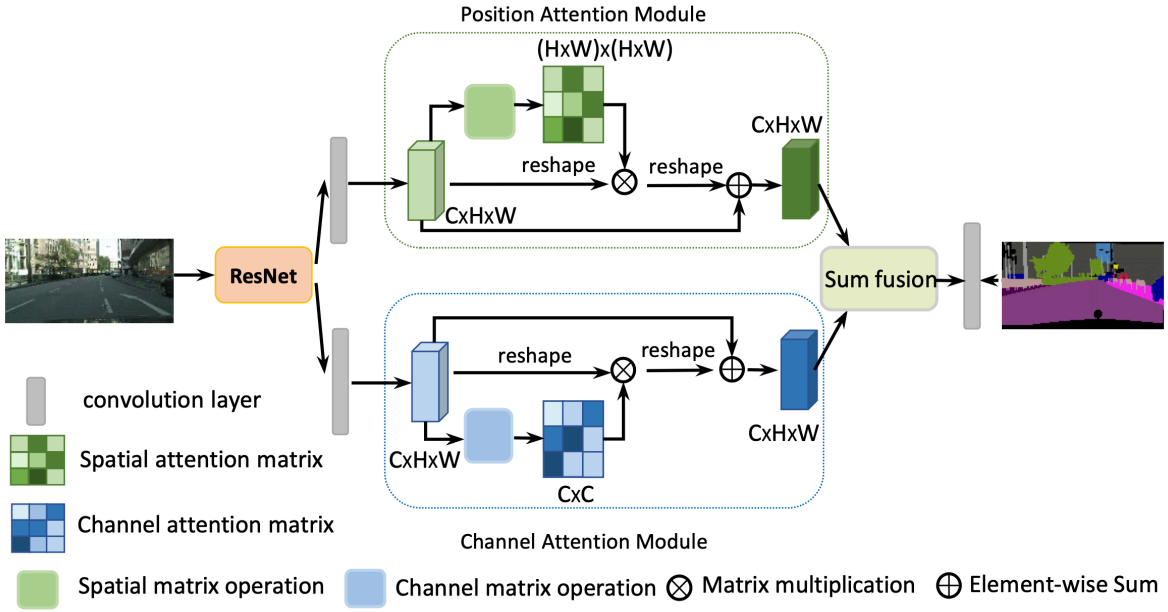


Image source

Overview of architectures

6. Modern approaches
ViT-based

TeViT ~ 68.9 fps [8]

1	**InternImage-H**	59.6%	InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions		2022	
2	**ViT-Adapter-L** (Mask2Former, BEiT pretrain)	54.2%	Vision Transformer Adapter for Dense Predictions		2022	
3	**EVA**	53.4%	EVA: Exploring the Limits of Masked Visual Representation Learning at Scale		2022	
4	**RSSeg-ViT-L** (BEiT pretrain)	52.6%	Representation Separation for Semantic Segmentation with Vision Transformers		2022	
5	**RSSeg-ViT-L**	52.0%	Representation Separation for Semantic Segmentation with Vision Transformers		2022	
6	**ViT-Adapter-L** (UperNet, BEiT pretrain)	51.4%	Vision Transformer Adapter for Dense Predictions		2022	
7	**SegViT** (ours)	50.3%	SegViT: Semantic Segmentation with Plain Vision Transformers		2022	
8	**SenFormer** (Swin-L)	50.1%	Efficient Self-Ensemble for Semantic Segmentation		2021	
9	**CAA** (Efficientnet-B7)	45.4%	Channelized Axial Attention for Semantic Segmentation -- Considering Channel Relation within Spatial Attention for Semantic Segmentation		2021	EfficientNet
10	**HRNetV2 + OCR + RMI** (PaddleClas pretrained)	45.2%	Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation		2019	
11	**DRAN** (ResNet-101)	41.2%	Scene Segmentation with Dual Relation-aware Attention Network		2019	ResNet

CPU inference

Why

- Requirements and constraints of the task
  In RoboCup Soccer - only CPU, limited RAM, real-time speed requirement
- Costs and budget optimization

How

1. Architecture choice
   - MultiNets
   - Object Detection-based with light one stage detectors
     YOLO-LITE, YOLOv4-tiny, YOLOv8 etc
   - Fast modern approaches
2. Model choice - size vs quality trade-off

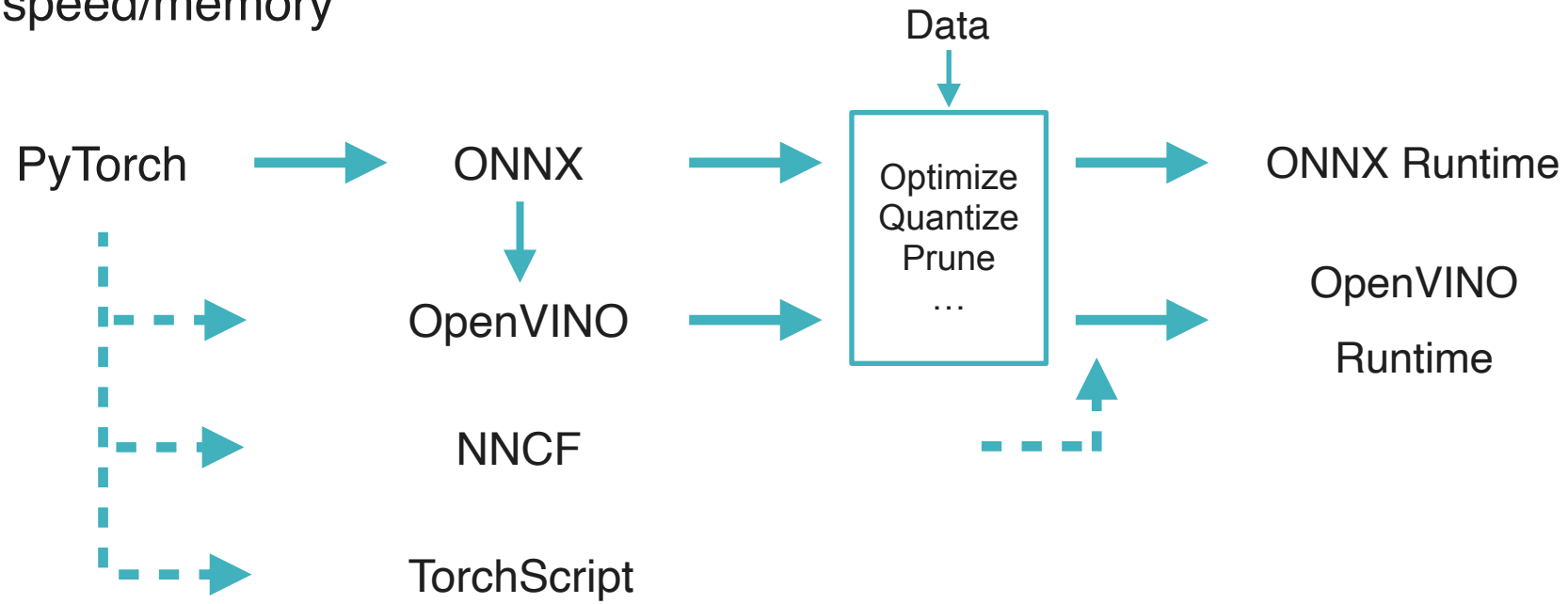
CPU inference

3. Reformat and optimize with a compromise between quality and speed/memory

- Depend on the target hardware
- Optimizations :
    A. During the training (e.g.Quantization-Aware Training)
    B. Post-training

CPU inference

3. Reformat and optimize with a compromise between quality and speed/memory

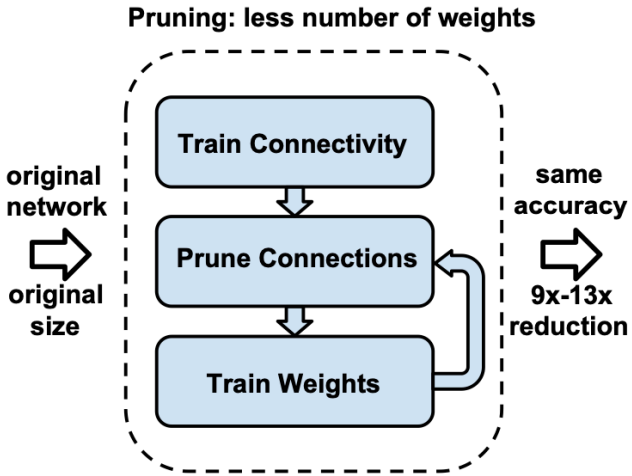


CPU inference

Pruning - ‘Like pruning decision trees’

- Criteria
- Connections
- Channels
- Layers

- Iterative or one-shot



girafe ai

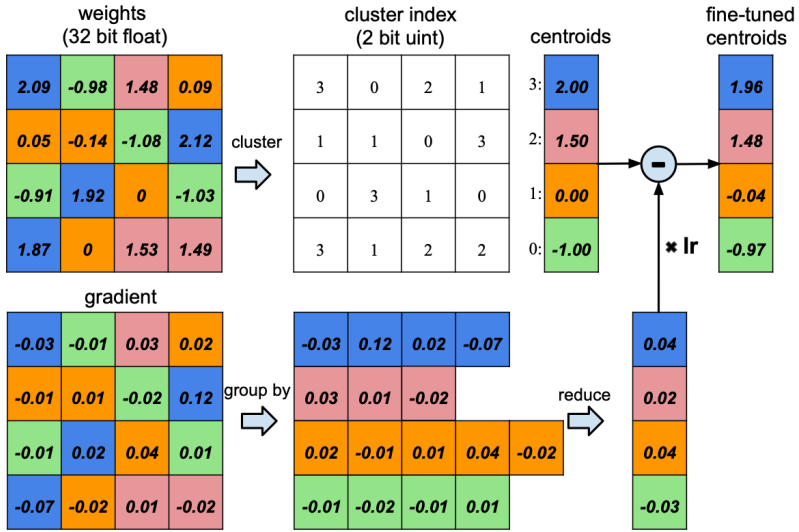
Image source

CPU inference

Quantization

- post-training or during training
- simple fp32 ⟶ fp16/int16/int8/etc
- Trained scalar quantization + weights sharing (KMeans,…)
- Details matter

  For example: changing the way the quantization range is calculated and using an asymmetric activation quantization method



girafe ai

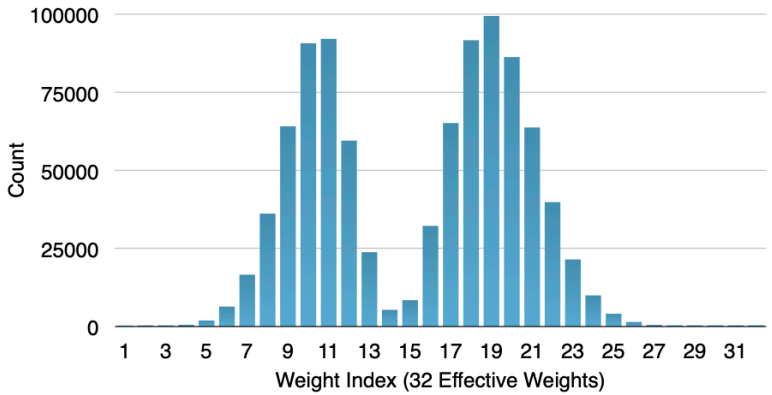
Image source

CPU inference

3. Reformat and optimize with a compromise between quality and speed/memory

Huffman Coding

- since the distribution of quantized weights is biased



4. Alternative - combination of DL and classic CV

girafe ai

Image source

CPU inference

Model	GPU Inference	CPU Basic	CPU Optimized
YOEO (416x416)	137 fps (2080Ti)	0.5 fps (M1)	6.25 fps
YOLOv8 n (640)	800 fps (A100)	2.4 fps (M1)	~ 11 fps
SparseInst (640)	45 fps (2080Ti)	1.6 fps (M1)	7.4 fps
BiSeNet v1 (2048x1024)	78 fps (fp16, Tesla T4)	0.44 fps (x20 on 416 input) (M1)	(> 4 fps)
SegFormer (2048x1024)	48 fps (V100)	2.3 fps (M1)	-

Annotation acceleration idea

Interactive segmentation / Weakly Supervised segmentation from extreme points

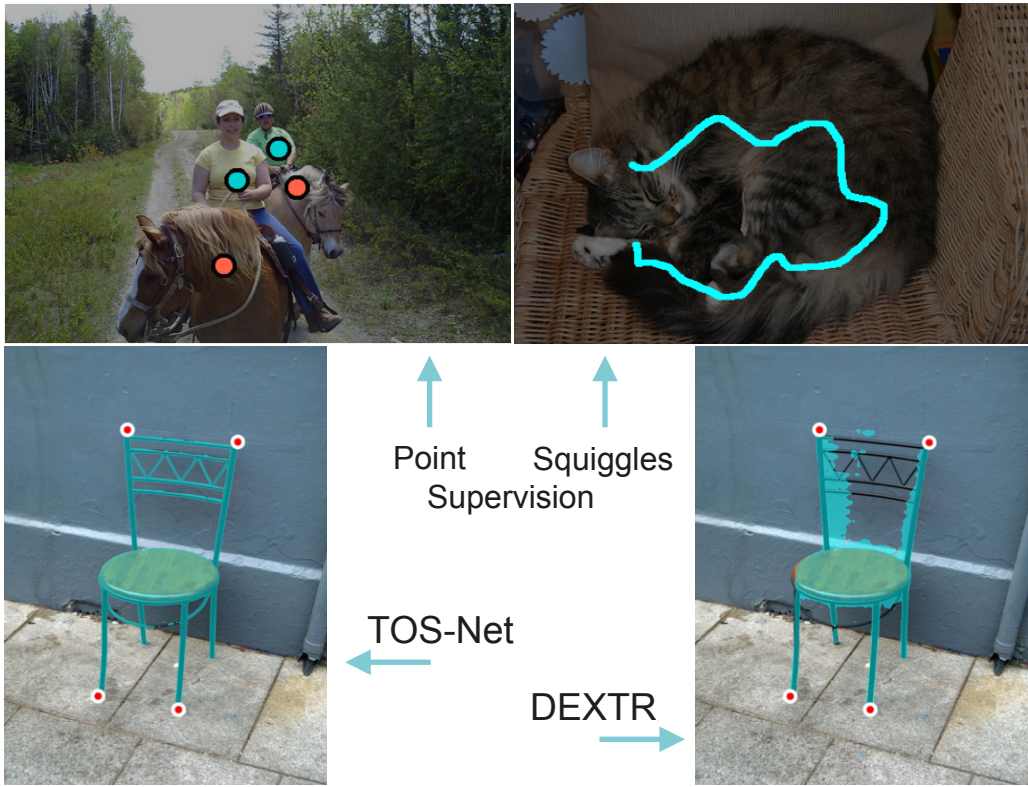
In particular:

DEXTR - Deep Extreme Cut: From Extreme Points to Object Segmentation

Deep Interactive Thin Object Selection

Extreme points labeling:

- x10 faster than polygons [9]
- x2 faster than SAM [10]

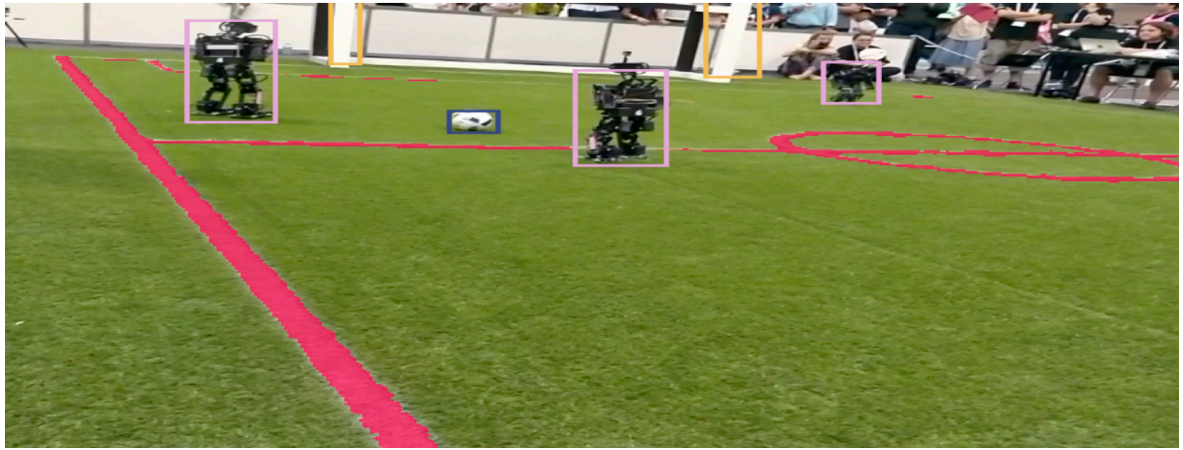


girafe ai

Image source

Class imbalance

In practical applications it's often necessary to segment small or/and thin objects



Segmentation of the marking lines of a footbal pitch

Class imbalance

Loss functions

- Pixel-wise CE Loss
  - standard approach
  - works well with a more or less uniform class distribution
- Weighted CE Loss
- Dice loss [11]: 1 - Dice Coef $= 1 - 2\frac{|PG|}{|P| + |G|}$
  - Dice Coef is a mesure of overlap of the predicted mask and the groundtruth
  - only considers the foreground (segmentation) class
- Tversky Index [12] - generalization of Dice Coef and IoU - $TI = \frac{|PG|}{|PG| + \alpha|P \backslash G| + \beta|G \backslash P|}$
  - gives more room for adjustements (e.g. penalize FN)
  - Dice if $\alpha = \beta = 0.5$, F-score if $\alpha + \beta = 1$

Class imbalance

Loss functions

- Focal Loss [13]

  Unlike OHEM, Focal Loss doesn't completely discard 'easy' examples, but assigns them a lower weight

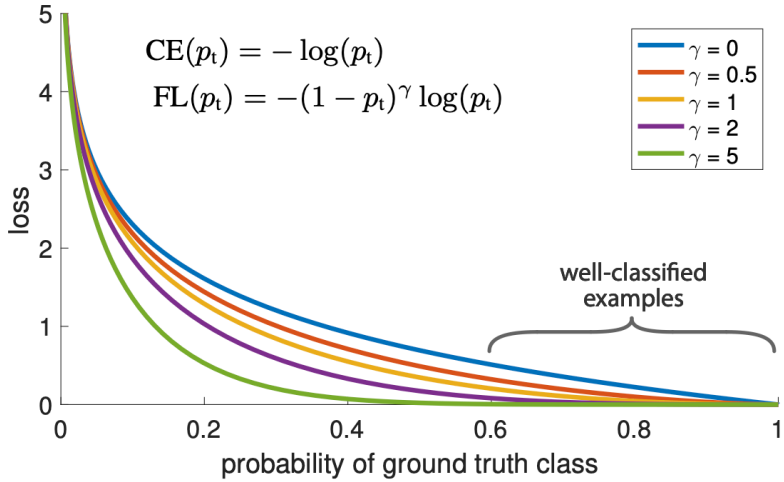


Image source

Class imbalance

Loss functions

- Morphological CE Loss [14]

  a weighted combination of CE and CE for small objects

$$MorCE = \alpha * CE + \beta * CE_{small}(k)$$

$$CE_{small}(k) = CE(WTH(y_{true}, k), WTH(y_{pred}, k))$$

  WTH - White-Top-Hat - morphological transformation, defined as the difference between the input image and its opening by a structuring element. This operation is capable of extracting small objects/details

Class imbalance and boundaries

Loss functions

- Boundary loss for highly unbalanced segmentation [15]

  Changes loss computation from unbalanced integrals over regions to metric on the space of contours

Class imbalance and boundaries

Loss functions

- Active boundary loss (ABL) [16]

  Every iteration guides a predicted boundary pixel in a direction pointing to the closest GT boundary pixel.

$$L = CE + IoU + \omega_\alpha ABL \qquad ABL = \frac{1}{N_b}\sum_{i}^{N_b} \Lambda(M_i) CE(D_i^p, D_j^q) \text{ , } \qquad \Lambda(x) = \frac{min(x,\theta)}{\theta}$$

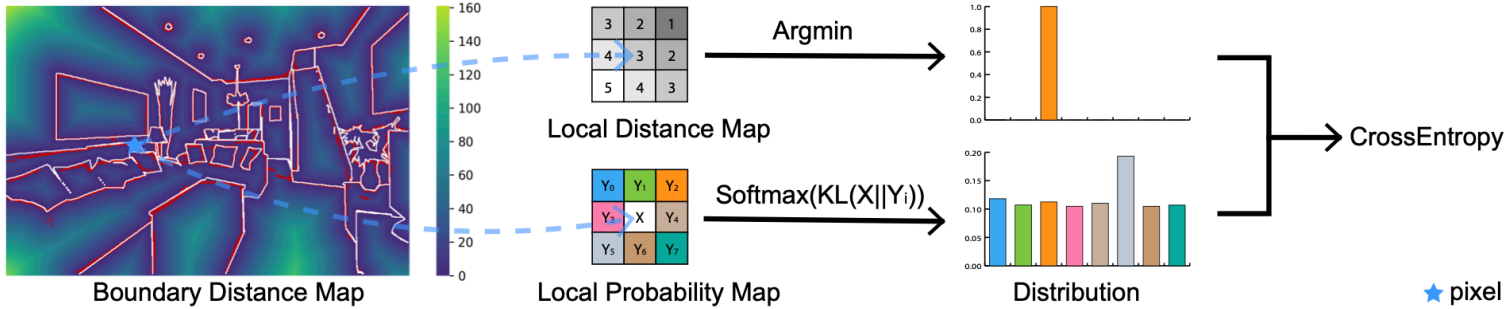


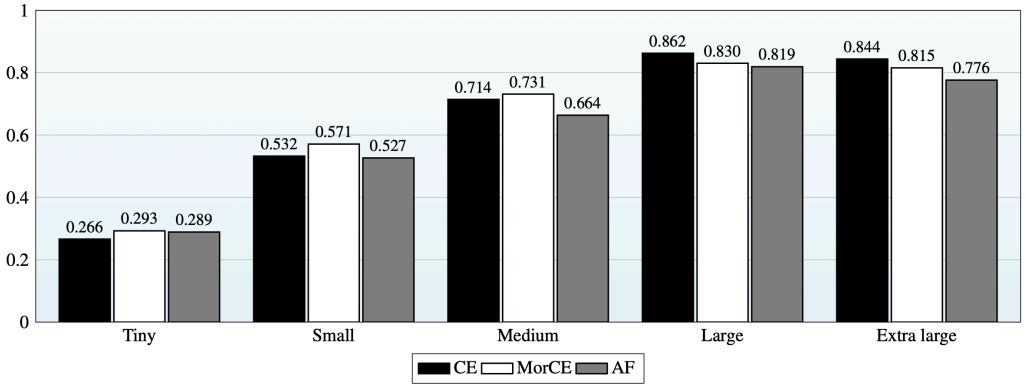
Image source

Class imbalance and boundaries

Experimental results:

- Focal Loss - IoU worse than pixel-wise CE (~ - 7%)
- Morphological Loss and ABL - slightly better than CE (< +0.5%)
- The set of samples with bad metrics changed insignificantly

Due to limited computational resources, combinations of parameters from the corresponding papers were used, which may not be optimal for this application



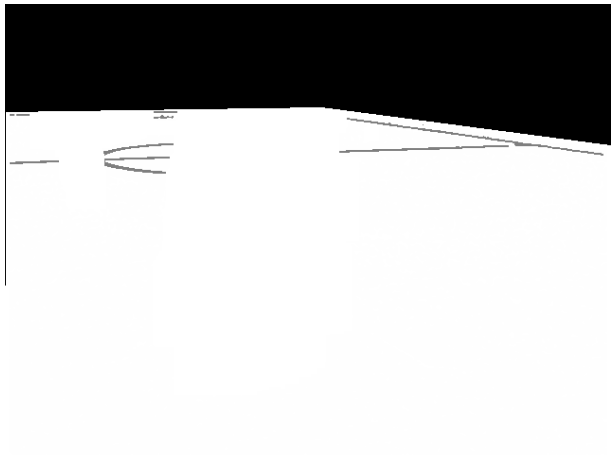
girafe ai

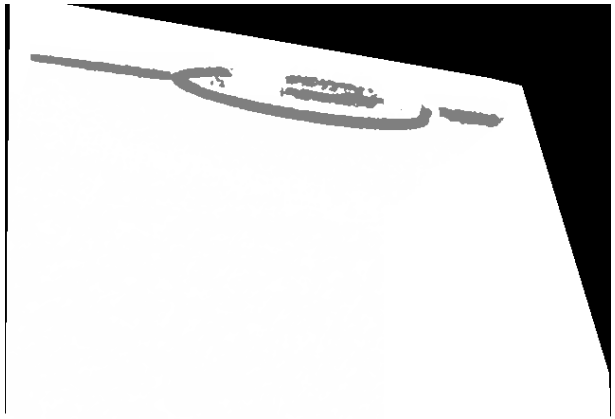
Image source

Class imbalance and boundaries

The limitation of the metric is likely related to the respective quality of the annotations

- enhance annotations
- use simulation / synthetic data





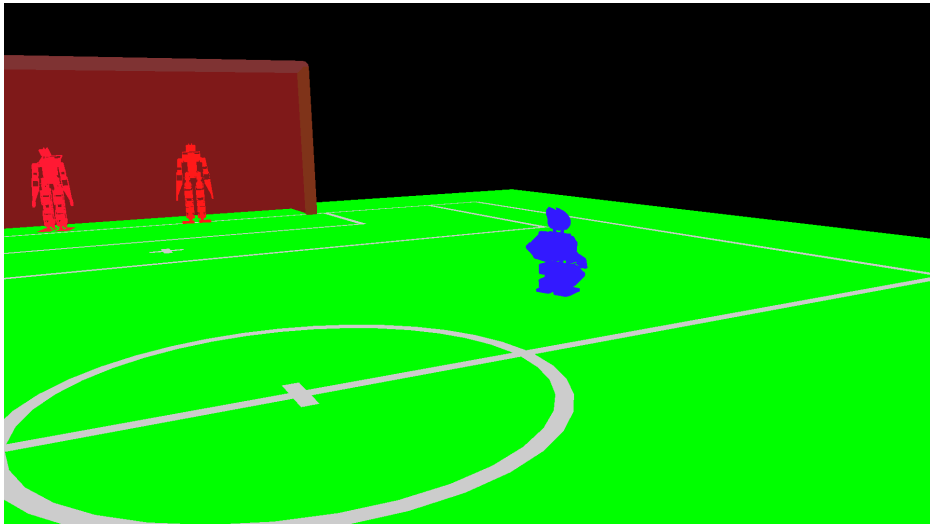
Class imbalance and boundaries

Simulation data - Starkit case

Adding 40 % (of the size of real imgs dataset):

> + 2 % IoU

> + 5 % mAP (for object detection task)



girafe ai

Class imbalance and boundaries

Synthetic data - Production line case

Challenges:

- Thin and small objects
- High precision (especially on the boudaries) and stability requirements
- Expensive quality annotations
- Low data variance

Tested approaches:

- Simplify «domain adaptation» using more analogous datasets in pre-training
- Create synthetic dataset

Class imbalance and boundaries

Synthetic data - Production line case

Analogously to ThinObject-5K dataset, 4000 images with target objects were created

- Free quality annotations
- Great flexibility and object awareness compared to Mosaic augmentation for example
- Data variety

girafe ai

Image source

Class imbalance and boundaries

Synthetic data - Production line case

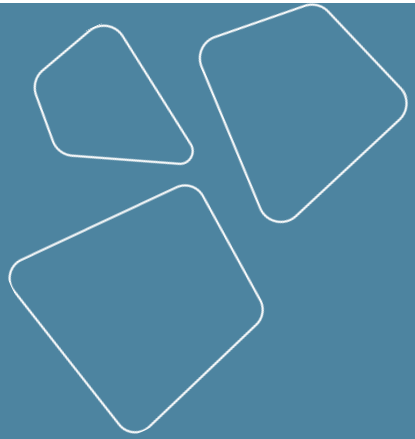
Used:

- mix of high-low resolution backgrounds depending on the resolution of objects
- random: rotations, changes in size, positions
- Color backgrounds, «grayscale», backgrounds from real dataset

Results:

- Classification task: +36% accuracy without real dataset, +4% after fine-tuning compared to models using real dataset with augmentations (Rotations, Flips, Affines..)
- Segmentation: +3% mIoU with less border blur
- Background is not significant; bg from real dataset had similar to augmentations results

“Your time is limited, so don’t waste it living ~~someone else’s life~~ without high-quality segmentation maps.” – «Steve Jobs»



girafe ai