read the conv chapter before this, it's essential

we are going to use U-Nets → Up Convolution.

this is what we're trying to do.

hundreds of papers on image segmentation.

this is also called Semantic Segmentation.

never seen

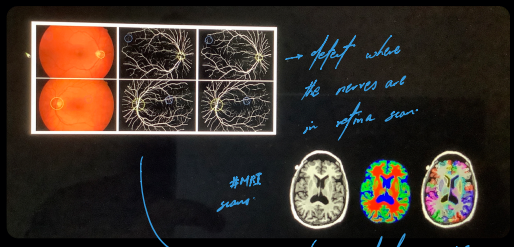
not a new problem. it's 3 decades old ← meaningful segmentation.

→ can say you've cancer

→ can show where it thinks the cancer is.

background blur
self driving car
medical → mark the tumor/problems → CT scans, etc.
colorization

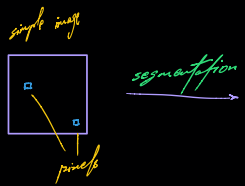
something like this.



→ defect where the nerves are in retina scan.

↳ Biomedical imaging

↳ fast application of UNet.



classify each pixel into some color coding scheme

general dataset that are are general purpose datasets.

COCO dataset → ImageNet Equivalent for segmentation.

- 1.5 million marked/labeled objects
- 330k images
- 80 classes

example

Domain specific Dataset

cityscape → for traffic & self driving cars.

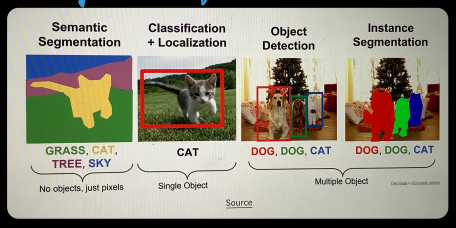
kaggle Medical Segmentation

segmentation vs object detection.

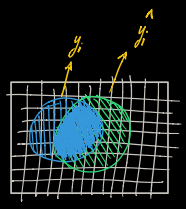
→ shows but more useful

→ faster & easier coz less precise

→ very very fast



segmentation is just object detection but pixel level, instead of bounding box.



Evaluation Metric

① Average precision & recall for each class/color → pixel level.

② Jaccard Similarity = $\frac{R_1 \cap \hat{R}_1}{R_1 \cup \hat{R}_1}$

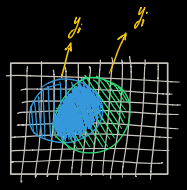
R stands for region

Ideal value = 1 → ∩ & ∪ is same.

worst case = ∩ is 0 → prediction too far.

Intersection Over Union

IOU



Classical Approaches

Clustering using K-means, etc → easy to mess up with shifted images

Edge Detection → traditional CV methods.

Graph Theory → they were state of the art, but no more. RIP.

Deep Learning Approaches

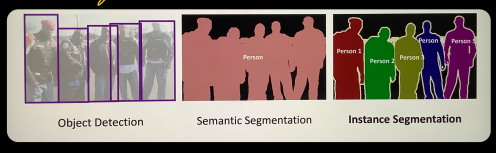
CHOICE FOR THIS SESSION # UNet → made for biomedical but works everywhere

Fully Convolutional Networks (FCN)

Mask R-CNN

SegNet → for mobile devices.

just keep this in mind




input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

```python
def unet(pretrained_weights = None,input_size = (256,256,1)):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
```

This is fucking simpler than I expected.

they haven't cropped the skip connections because the dimensions link up.

dropout was not there but this code has it.

encoder ends here

→ makes axis 3 go from 1024 to 512

#Data Augmentation is necessary

→ shift, rotate, shear, zoom, etc (problem specific) → some applications won't allow this.

→ grayscale adjustments

Mask Region CNN

↳ does bounding box & then highlights objects pixel wise.

#resources

Marshall Lambá's blog

Google AI's blog on this

Divam's repo on GitHub for popular implementations.

This is just the architecture, we can change the I/O resolution as long as the numbers add up.

follow top researchers & conferences

↳ that's how we filter the noise → finding good papers → one good paper a week

or Arxiv Sanity