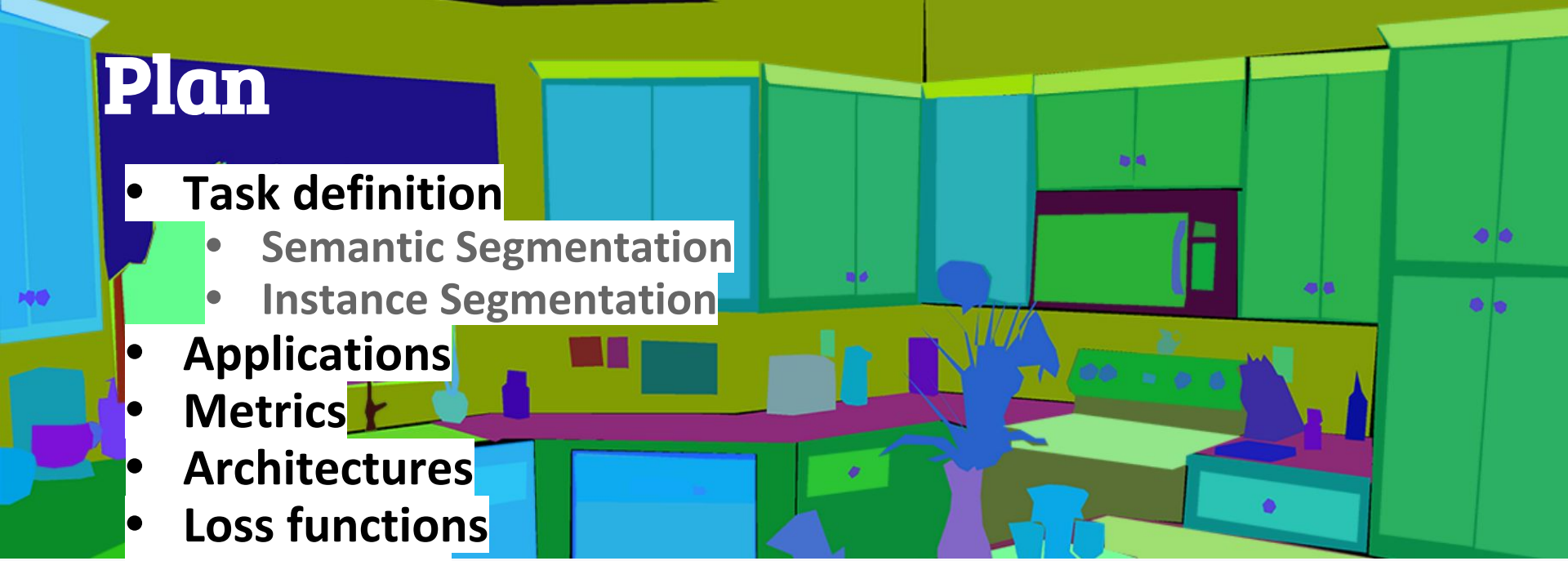
OTUS
ОНЛАЙН ОБРАЗОВАНИЕ

Онлайн-образование

Image Segmentation

Ческидова Евгения

Deep Learning engineer
Wolf3d
t.me/fogside

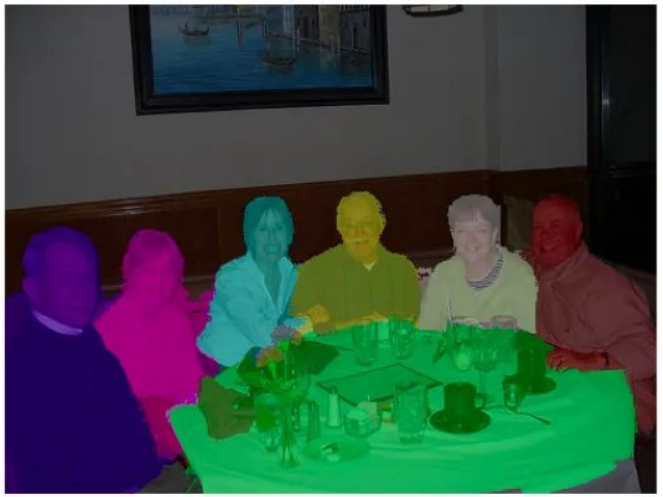


Plan

- **Task definition**
  - Semantic Segmentation
  - Instance Segmentation
- **Applications**
- **Metrics**
- **Architectures**
- **Loss functions**
- **Frameworks**
- Datasets
- Tutorials

Semantic and instance segmentation

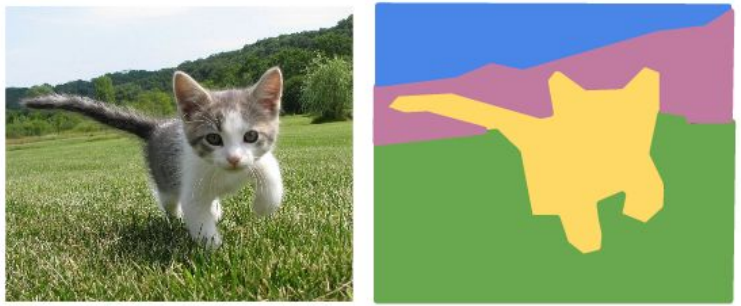
Semantic Segmentation



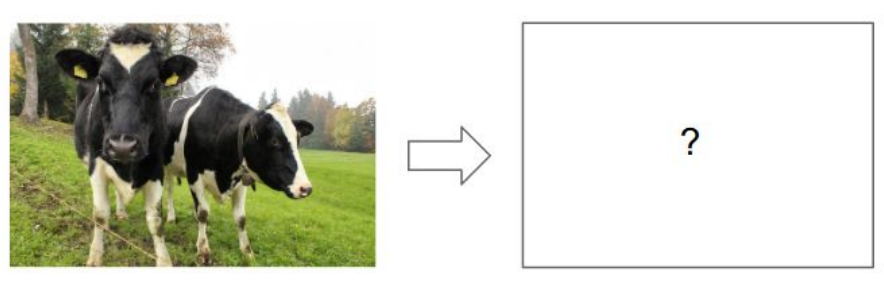
Instance Segmentation

- In **semantic segmentation**, all objects of the same type are marked using one class label while in **instance segmentation** similar objects get their own separate labels.

Semantic Segmentation: The Problem



Paired **training data**: for each training image, each pixel is labeled with a semantic category



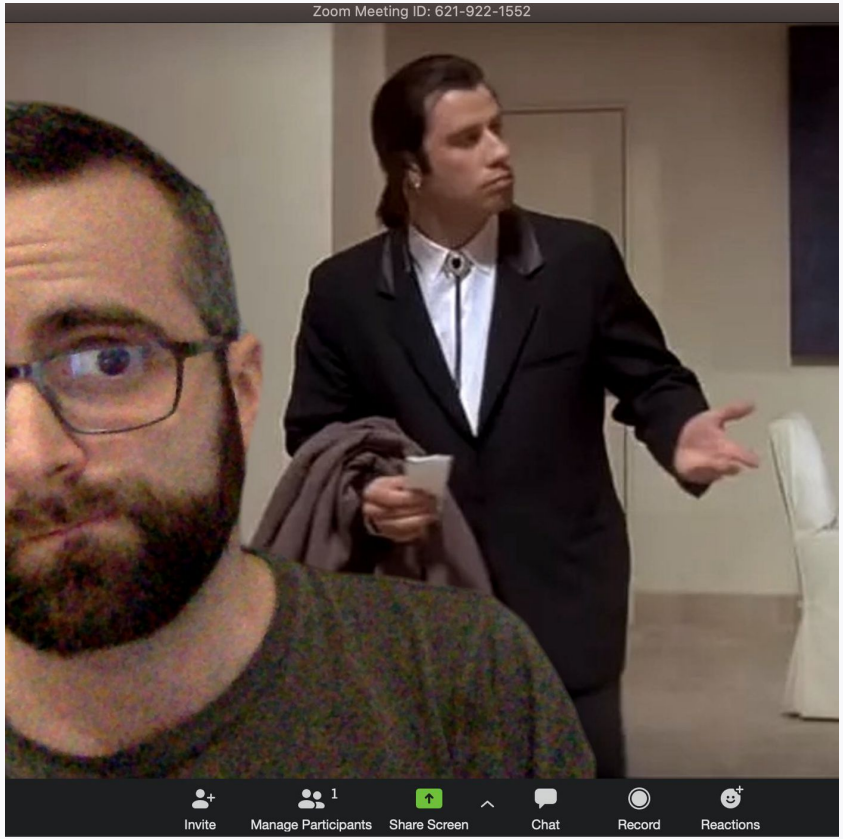
At **test time**, classify each pixel of a new image.


OTUS
ОНЛАЙН ОБРАЗОВАНИЕ
Applications

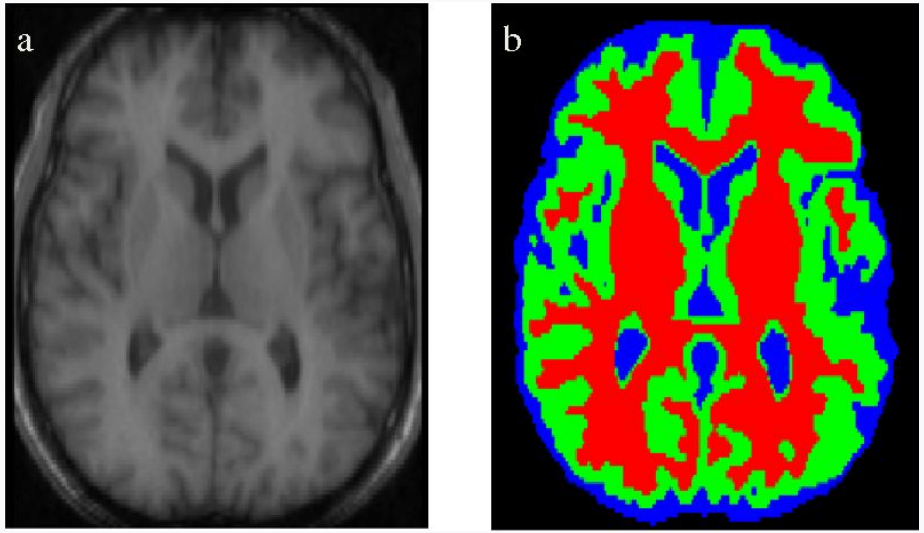
Applications: Autonomous Driving

Semantic segmentation for autonomous vehicles

Background replacement



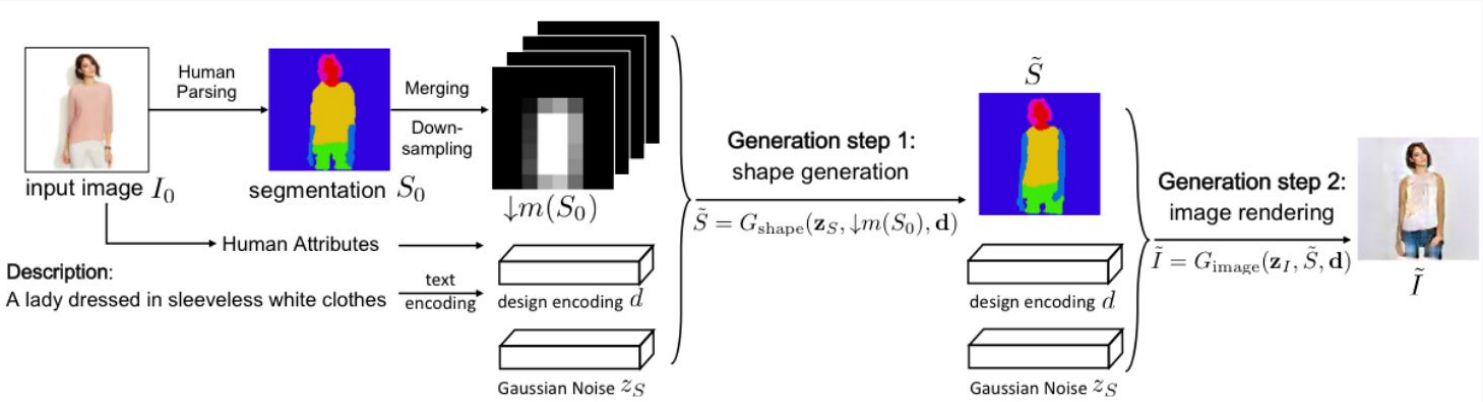
Applications: Medical Image Segmentation



Segmentation of medical scans. (Source)

- 2D segmentation (U-Net)
- 3D segmentation (V-Net)

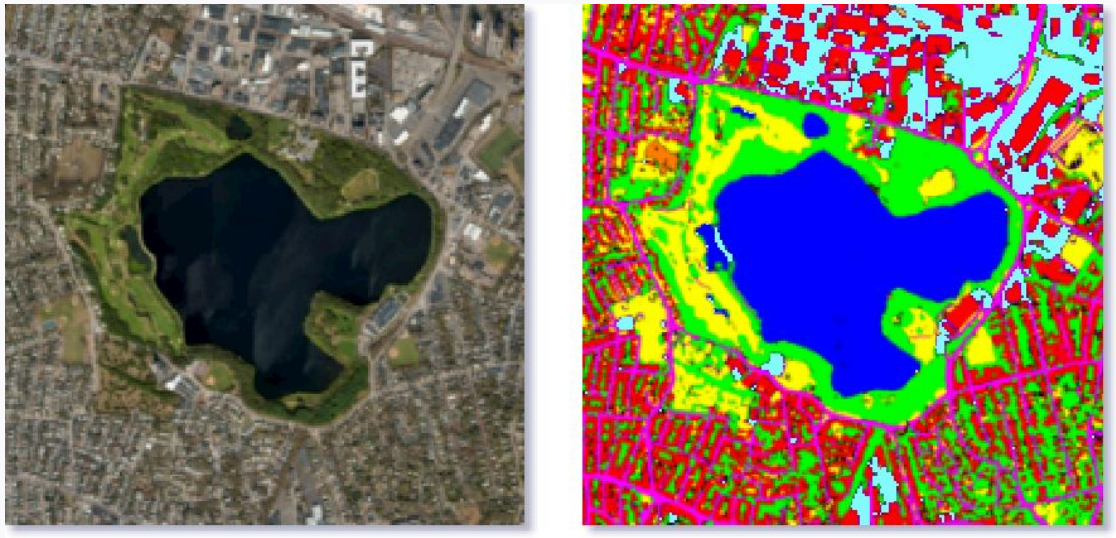
Applications: Fashion Industry



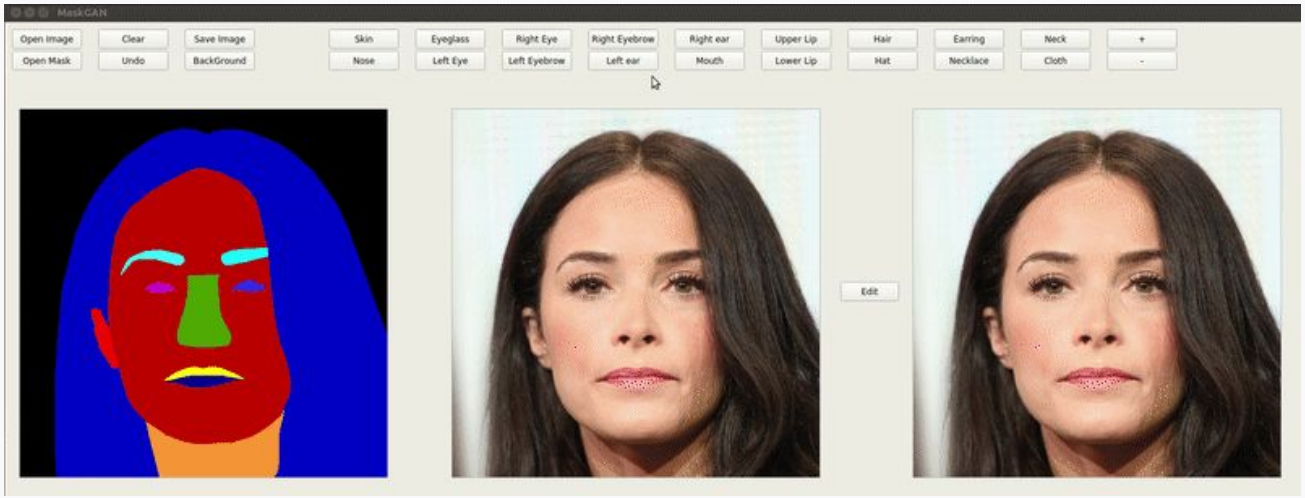
Semantic segmentation used as an intermediate step to redress a human based on text input.

- Be Your Own Prada: Fashion Synthesis with Structural Coherence (2017, Shizhan Zhu et al)
- https://github.com/switchablenorms/DeepFashion2

Applications: Satellite Image Processing



Applications: Image editing

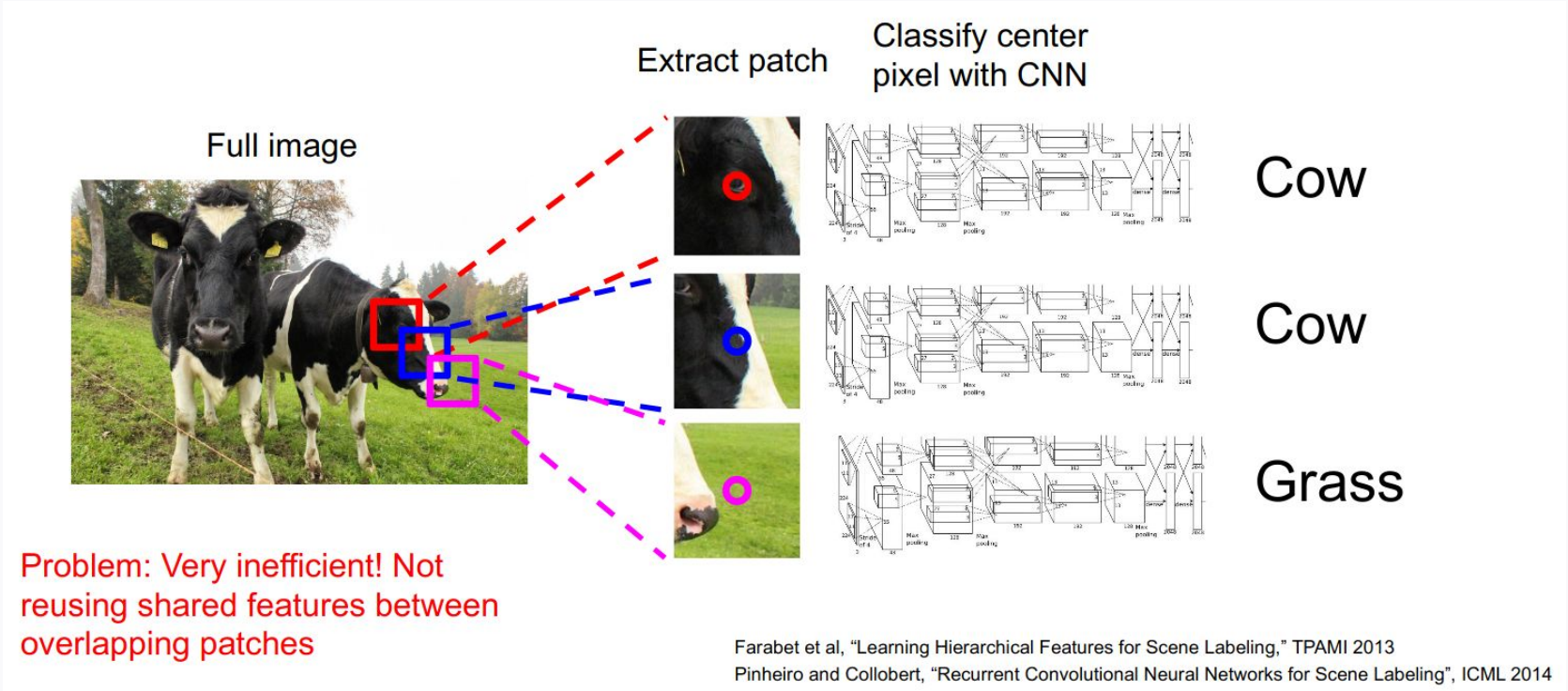


Segmentation used as an intermediate step
for GAN face generation

https://github.com/switchablenorms/CelebAMask-HQ

Architectures

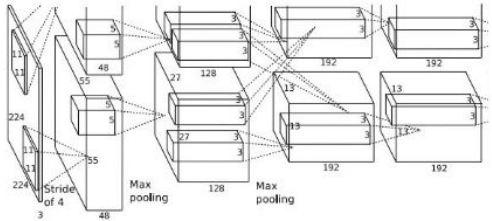
Semantic Segmentation Idea: Sliding Window



source: cs231 2020, lection 12

Semantic Segmentation Idea: Convolution

Full image

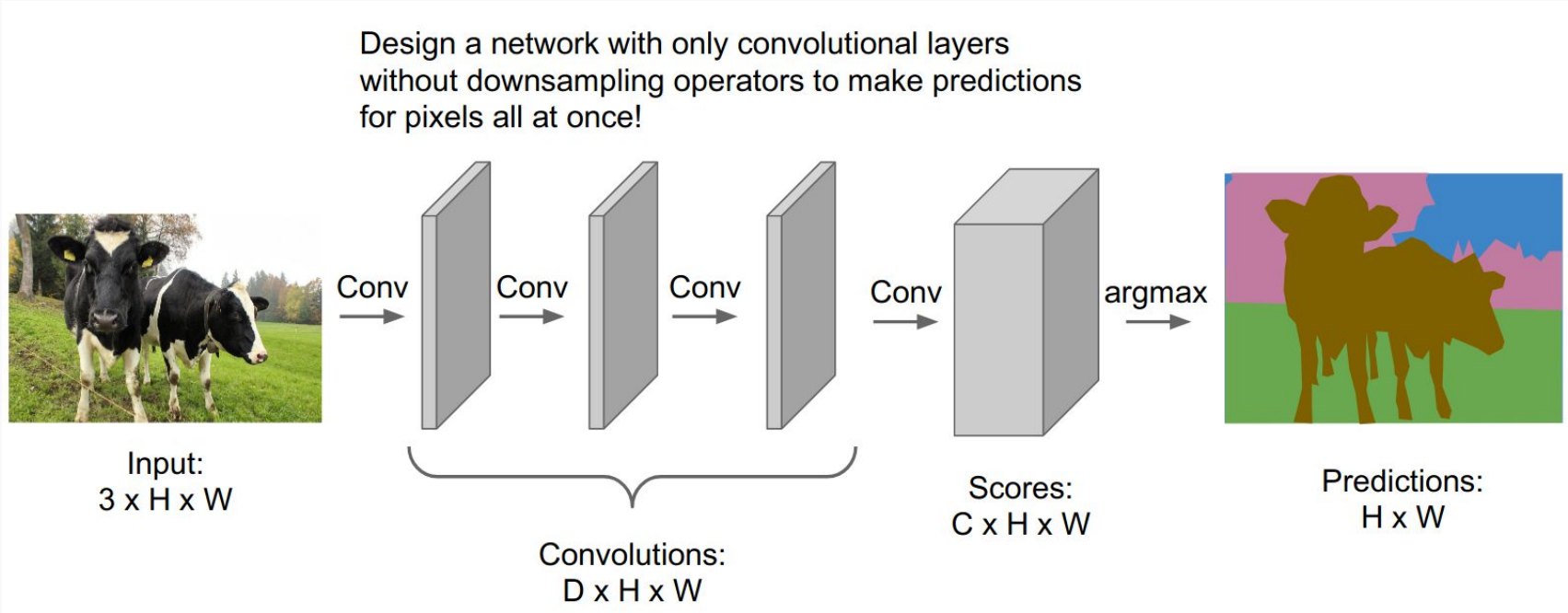


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

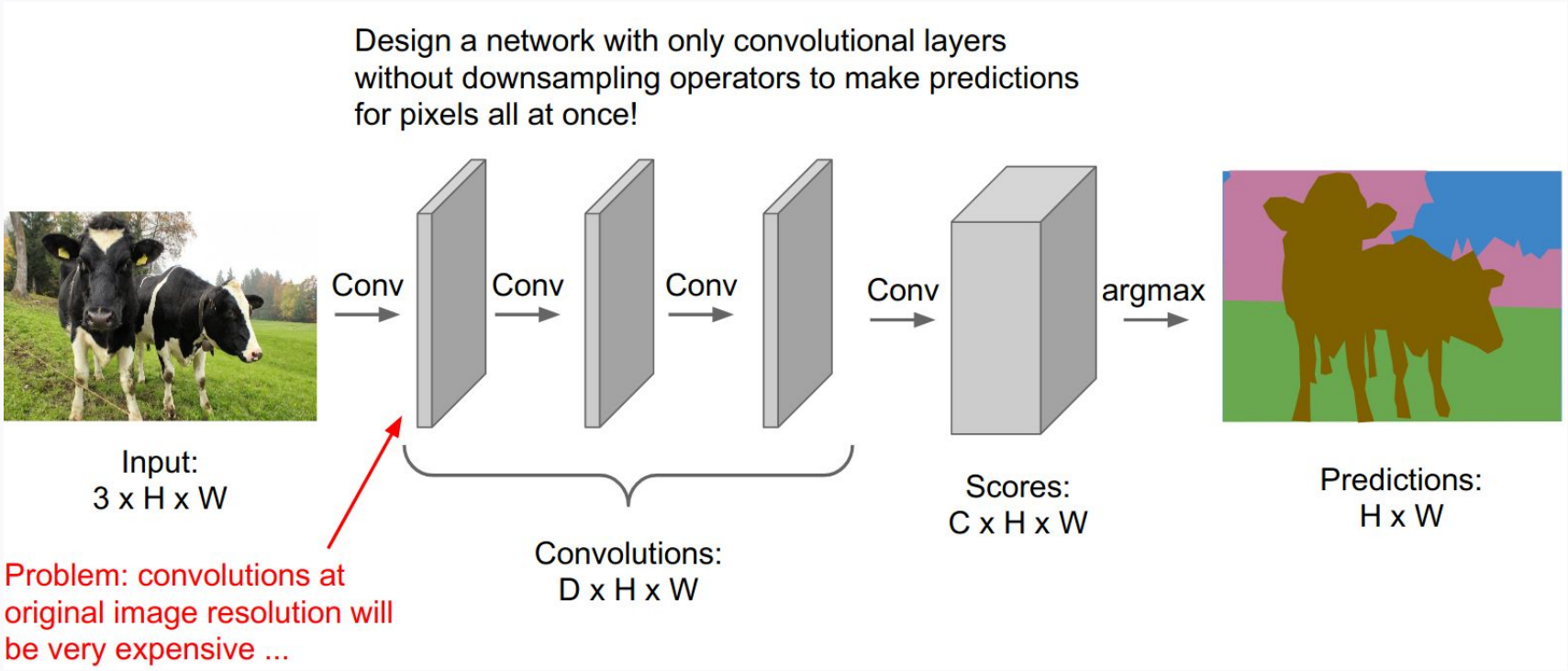
source: cs231 2020, lection 12

Semantic Segmentation Idea: Fully Convolutional



source: cs231 2020, lection 12

Semantic Segmentation Idea: Fully Convolutional



source: cs231 2020, lection 12

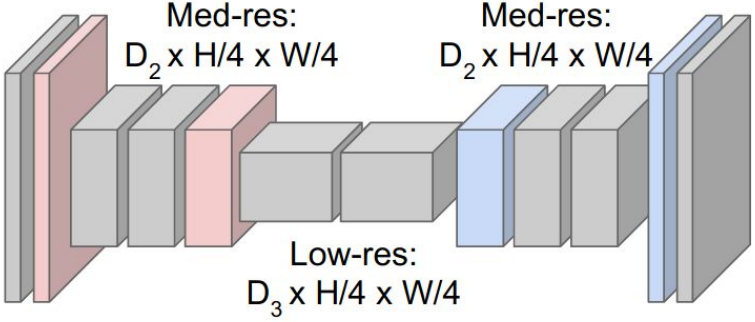
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

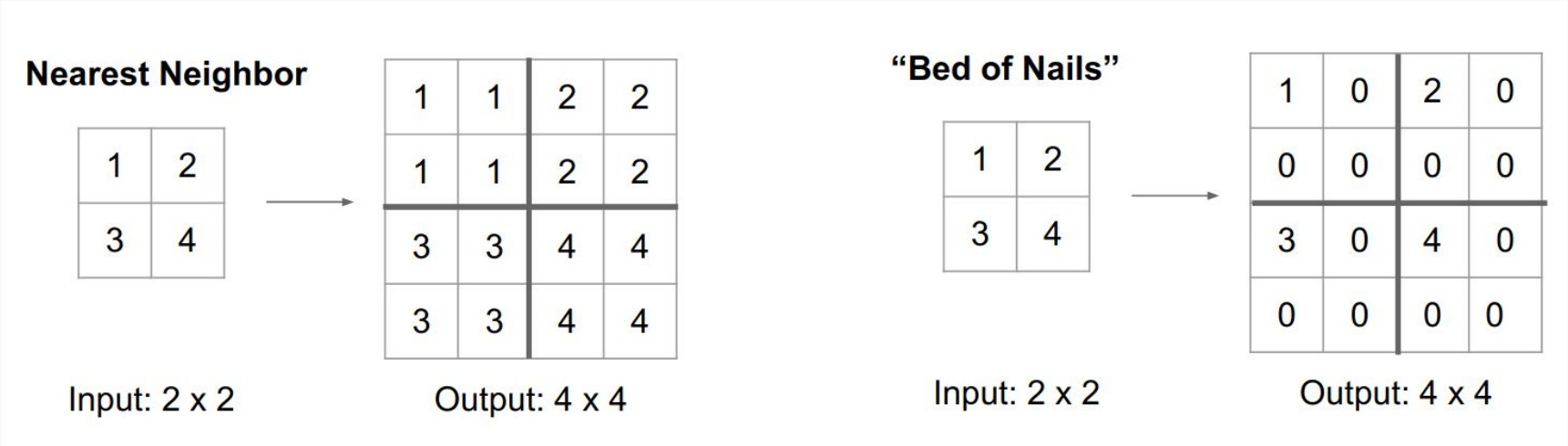
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

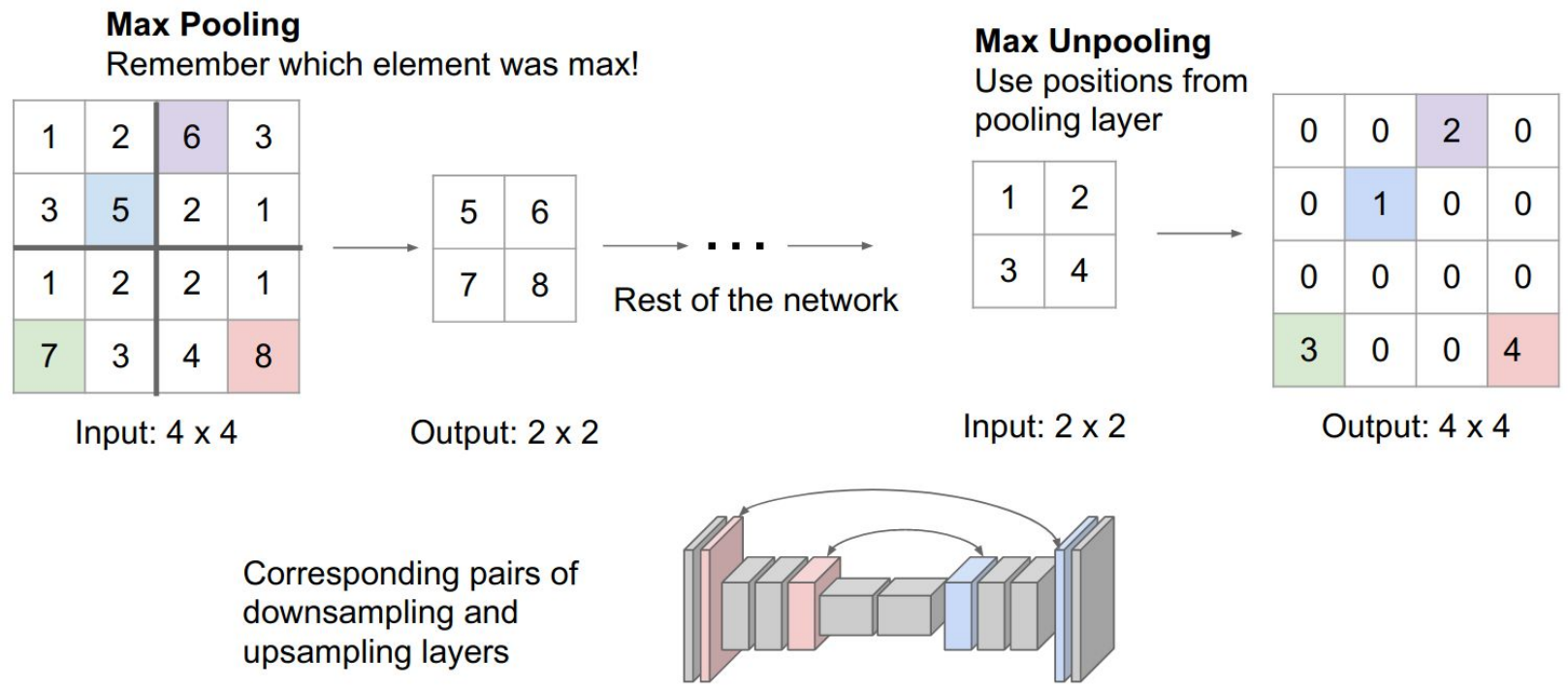
source: cs231 2020, lection 12

In-Network upsampling: “Unpooling”



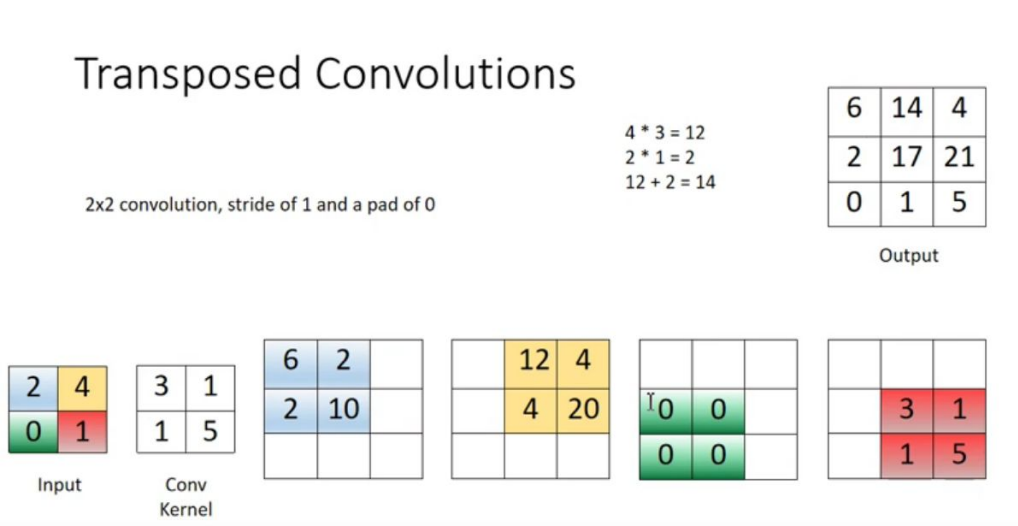
- Keras layer: https://keras.io/api/layers/reshaping_layers/up_sampling2d/
- source: cs231 2020, lection 12

In-Network upsampling: “Max Unpooling”



source: cs231 2020, [lection 12]

Learnable Upsampling: Transposed Convolutions



Deconvolution and Checkerboard Artifacts
Use upsampling + convolution instead of deconvolution to avoid it

https://distill.pub/2016/deconv-checkerboard/

U-net

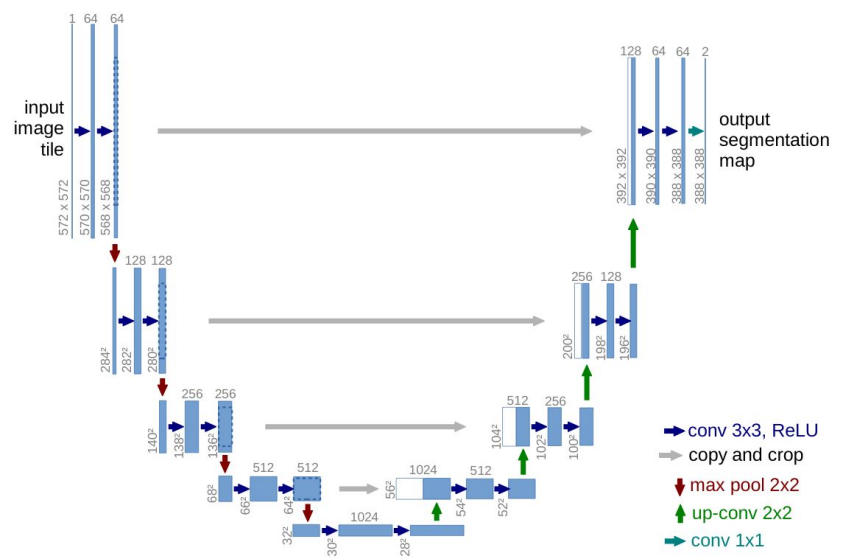


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

- Skip connections allows gradients to flow better and provides information from multiple scales of the image size.
- Example: https://github.com/petrosgk/Kaggle-Carvana-Image-Masking-Challenge/blob/master/model/u_net.py

U-Net: Convolutional Networks for Biomedical Image Segmentation (Ronneberger et al, 2015)

U-net: pretrained weights usage

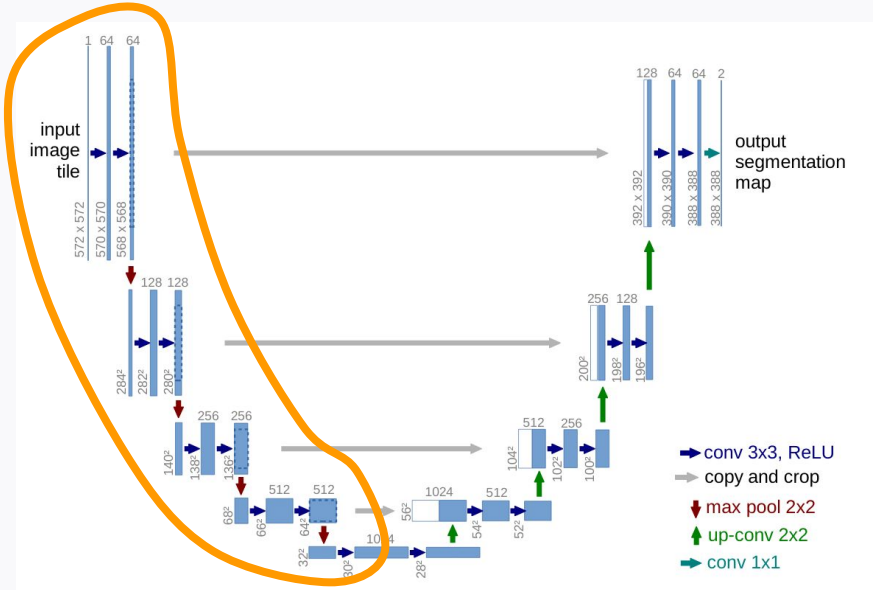
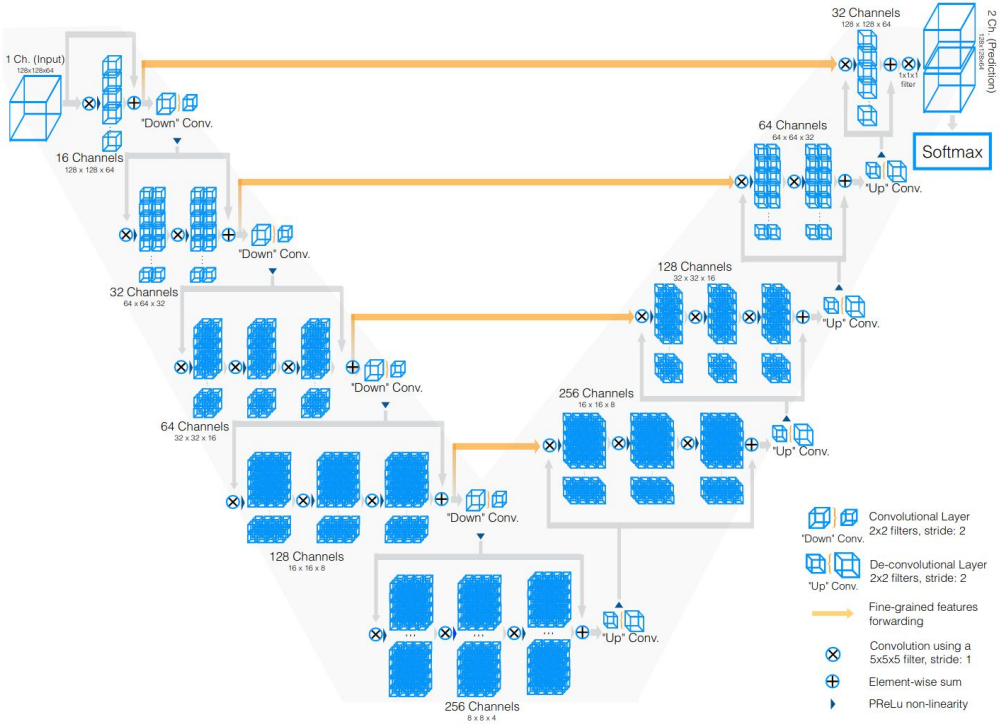


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

- Weights from pretrained backbone (like ResNet18) could be used to initialize encoder part of U-net

U-Net: Convolutional Networks for Biomedical Image Segmentation (Ronneberger et al, 2015)

V-net



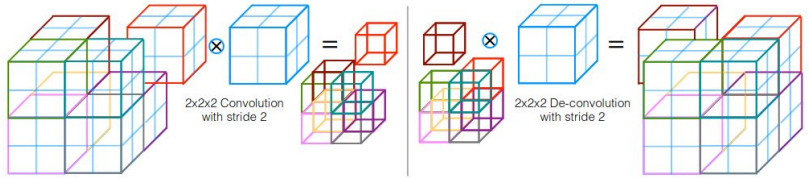


Fig. 3. Convolutions with appropriate stride can be used to reduce the size of the data. Conversely, de-convolutions increase the data size by projecting each input voxel to a bigger region through the kernel.

- 3D-convolutions

V-Net: Fully Convolutional Neural Networks for volumetric Medical Image Segmentation

Tiramisu Model

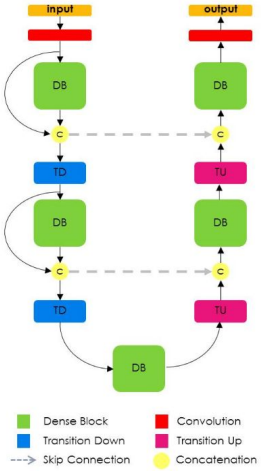


Figure 1. Diagram of our architecture for semantic segmentation. Our architecture is built from dense blocks. The diagram is composed of a downsampling path with 2 Transitions Down (TD) and an upsampling path with 2 Transitions Up (TU). A circle represents concatenation and arrows represent connectivity patterns in the network. Gray horizontal arrows represent skip connections, the feature maps from the downsampling path are concatenated with the corresponding feature maps in the upsampling path. Note that the connectivity pattern in the upsampling and the downsampling paths are different. In the downsampling path, the input to a dense block is concatenated with its output, leading to a linear growth of the number of feature maps, whereas in the upsampling path, it is not.

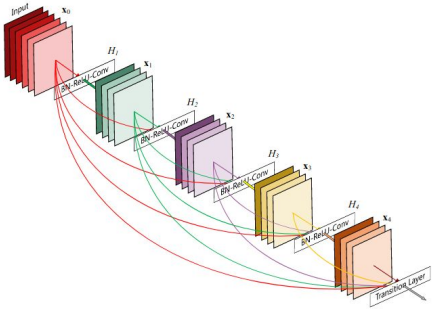


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

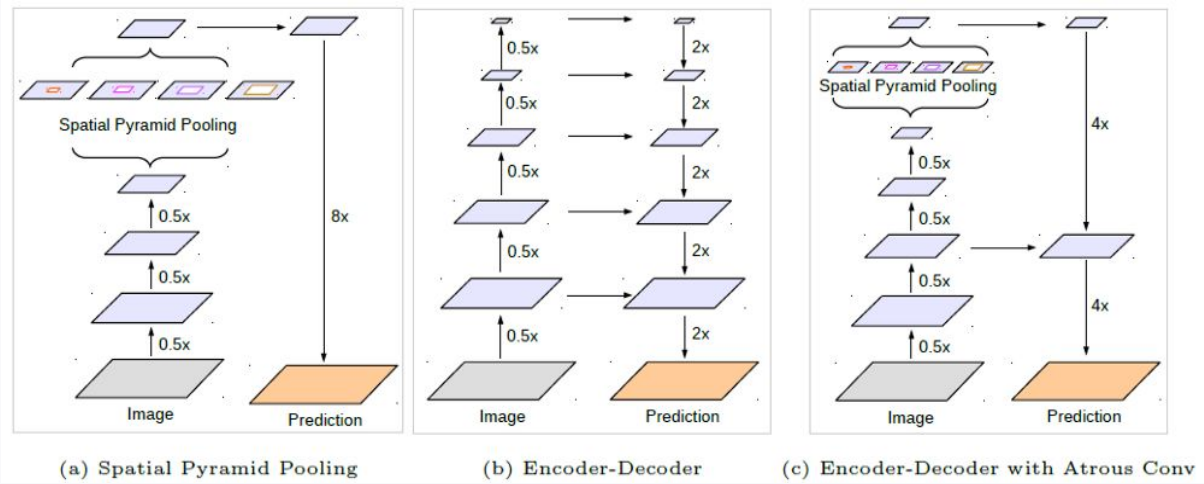
Dense block (source)

- Uses dense blocks
- The resultant network is extremely parameter efficient and can better access features from older layers
- Due to the nature of the concatenation operations in several ML frameworks, it is not very memory efficient

The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation (Jegou et al, 2017)

DeepLab by Google

- DeepLabv1 (2015 ICLR), DeepLabv2 (2018 TPAMI), and DeepLabv3 (arXiv)

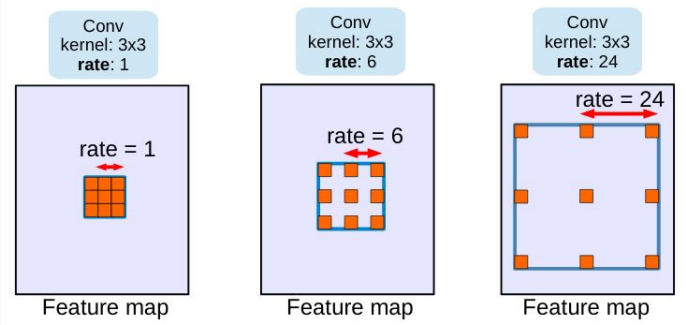


(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

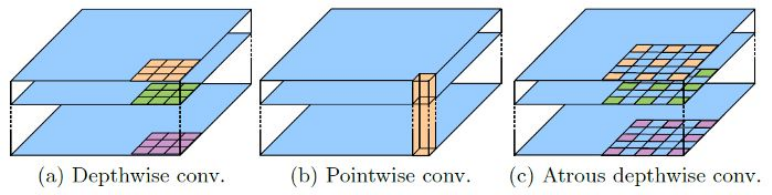
(a): With Atrous Spatial Pyramid Pooling (ASPP), able to encode multi-scale contextual information.
(b): With Encoder-Decoder Architecture, the location/spatial information is recovered.
(c): DeepLabv3+ makes use of (a) and (b) and Atrous Separable Convolution

Deeplab repo: https://github.com/tensorflow/models/tree/master/research/deeplab

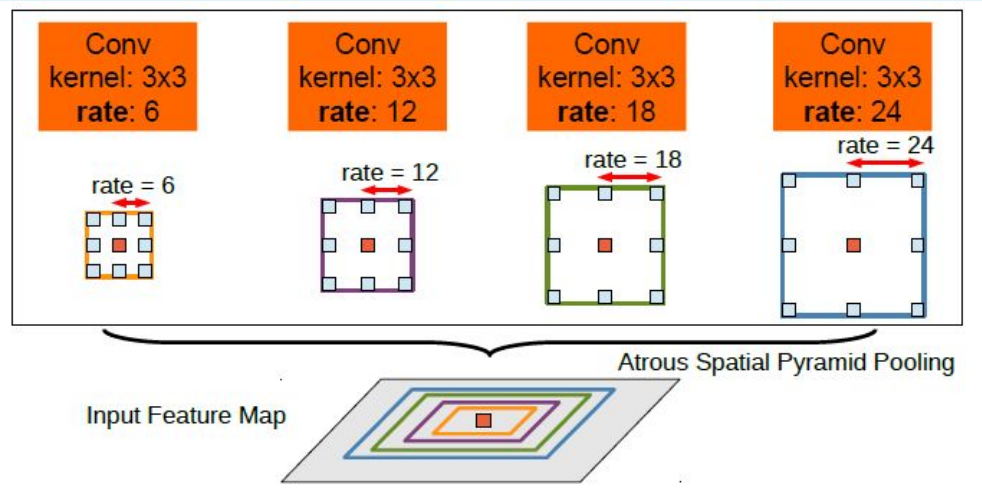
DeepLabV3+: Atrous (dilated) Convolutions



Atrous convolutions



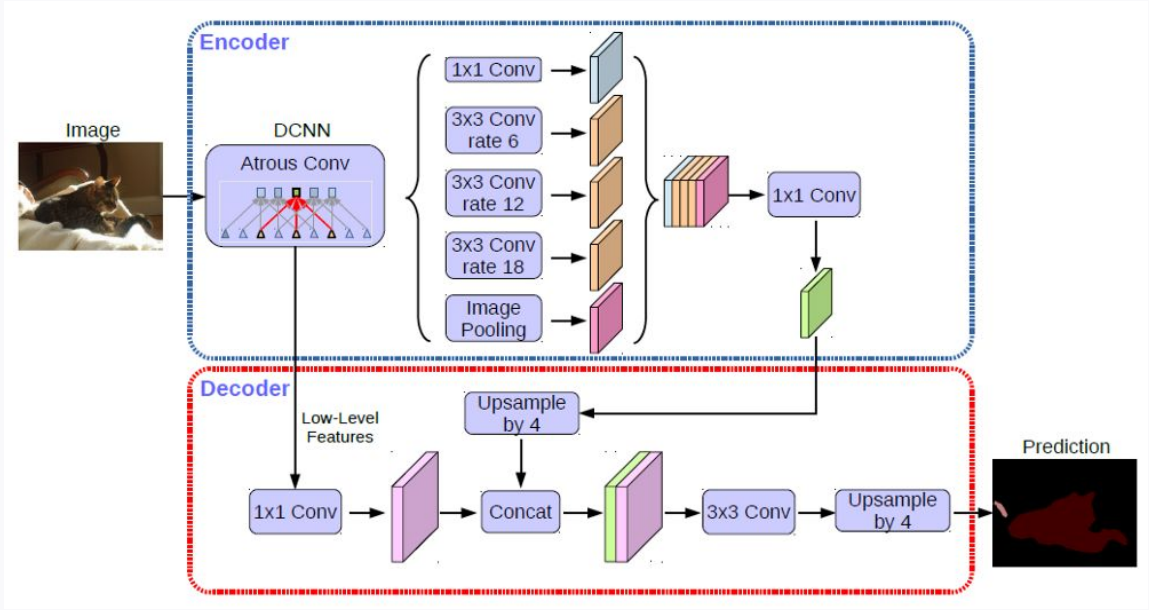
Depthwise Separable Convolution Using Atrous Convolution



Atrous Spatial Pyramid Pooling (ASPP)

(rate - 1) -- это расстояние между 2-мя пикселями, на которые применяется фильтр.
Иллюстрация не точная по соотношению площадей и указанного rate.
Более подробно смотреть в [этой заметке].

DeepLabV3+: architecture



Encoder-Decoder architecture

- DeepLabv3 as Encoder
- Atrous Separable Convolution instead of Atrous Convolution
- Modified Aligned Xception as Backbone

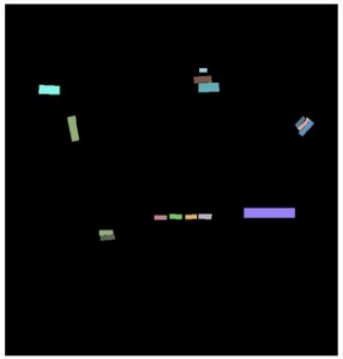
More information in [the article]

Metrics

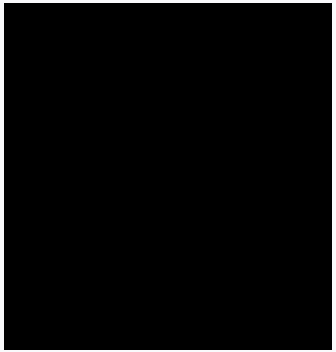
Pixel Accuracy

Percent of pixels in your image that are classified correctly

Input image



ground truth mask

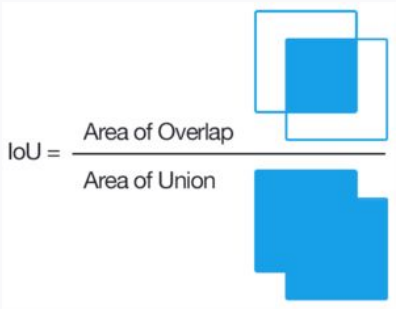


your prediction that gives 95% accuracy

Class imbalance issue

Intersection-Over-Union (IoU, Jaccard Index)

Jaccard Similarity Index is the most intuitive ratio between the intersection and union



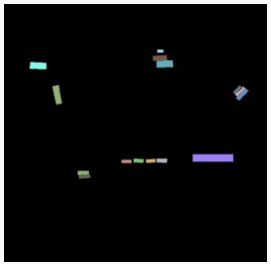
- ranges from 0–1 (0–100%)
- mean IoU for multi-class segmentation

How to calculate?

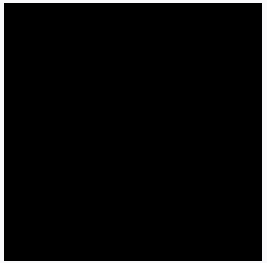
Ships
Area of Overlap = 0
Area of Union = (5+0)-0 =5
Area of Overlap/Area of Union = 0%

Background
Area of Overlap = 95,
Area of Union = (95+100)–95 = 100
Area of Overlap/Area of Union =95%

Mean IoU = (Ships + Background)/2 =
(0%+95%)/2 = 47.5%



ground truth



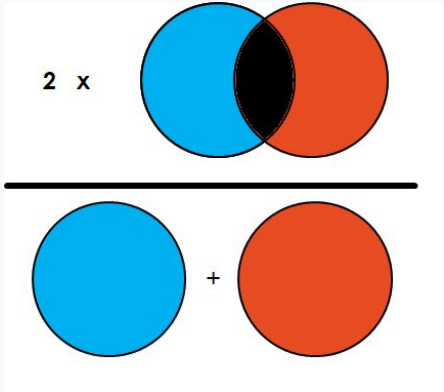
prediction

$$IoU = \frac{TP}{(TP + FP + FN)}$$

https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Dice Similarity Coefficient

2 * the Area of Overlap divided by the total number of pixels in both images



How to calculate?

Let total Number of Pixels for both images combined = 200

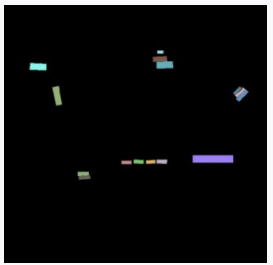
Ships: Area of Overlap = 0

*(2 * Area of Overlap)/(total pixels combined) = 0/(5+0) = 0*

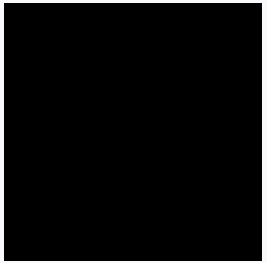
Background: Area of Overlap = 95

*(2 * Area of Overlap)/(total pixels combined) = 95*2/(100+95) ≃ 0.97*

***Dice** = (Ships + Background)/2 = (0%+97%)/2 =* **48.5%**



ground truth



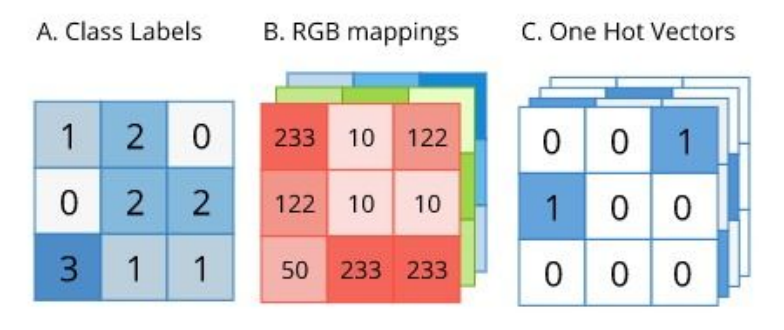
prediction

$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Loss functions

Pixel-wise Softmax with Cross Entropy



- Labels in a one-hot form
- Softmax must be applied pixel-wise on the predicted output before applying cross entropy, as each pixel can belong to any one of our target classes

Focal Loss

Good choice in cases with extreme class imbalance

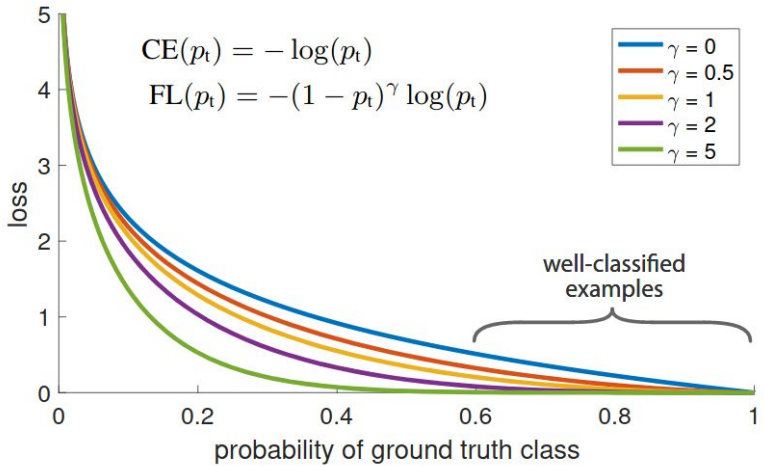


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

Focal Loss for Dense Object Detection [RetinaNet] (2018, Lin et al, FAIR)

Dice Loss

Good choice in cases with extreme class imbalance

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

Dice coefficient between **p** (predicted) and **g** (ground truth) *binary* volumes ranging between 0 and 1

$$\frac{\partial D}{\partial p_j} = 2\left[\frac{g_j\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right) - 2p_j\left(\sum_i^N p_i g_i\right)}{\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right)^2}\right]$$

Derivative of dice coefficient computed with respect to the j-th voxel of the prediction

V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation (2016, Milletari et al)

Image Segmentation Datasets

- **Common Objects in COntext—Coco Dataset**

COCO is a large-scale object detection, segmentation, and captioning dataset. The dataset contains 91 classes. It has 250,000 people with key points. Its download size is 37.57 GiB. It contains 80 object categories. It is available under the Apache 2.0 License and can be downloaded from here.

- **PASCAL Visual Object Classes (PASCAL VOC)**

PASCAL has 9963 images with 20 different classes. The training/validation set is a 2GB tar file. The dataset can be downloaded from the official website.

- **The Cityscapes Dataset**

This dataset contains images of city scenes. It can be used to evaluate the performance of vision algorithms in urban scenarios. The dataset can be downloaded from here.

- **The Cambridge-driving Labeled Video Database—CamVid**

This is a motion-based segmentation and recognition dataset. It contains 32 semantic classes. This link contains further explanations and download links to the dataset.

- The Berkeley Segmentation Dataset and Benchmark
- ADE20K
- DeepFashion2

Frameworks

- For instance segmentation detectron2
- For semantic segmentation:
  - DeepLab
  - Catalyst.Segmentation (Unet)
  - SMP

Tutorials

- Detectron2 -- instance segmentation with pretrained mask-RCNN and custom baloon dataset

- DeepLab on ADE20k
- DeepLab train on your own dataset
  - Background segmentation example -- collecting data, making labels, preparing tfdata and training
  - But it's possible to solve background segmentation just with inference of pretrained model like this tutorial shows.

- Advanced segmentation tutorial with Catalyst *[Highly recommended]*

Pretrained models from SMP

- PyTorch (you can use them in Catalyst as well):
  - https://github.com/qubvel/segmentation_models.pytorch
- Keras/Tensorflow
  - https://github.com/qubvel/segmentation_models

Tools for annotations

- LabelMe: One of the most known tools. The UI was a bit too slow, though, especially when zooming in on large images.
- RectLabel: Simple and easy to work with. Mac only.
- LabelBox: Pretty good for larger labeling projects and has options for different types of labeling tasks.
- VGG Image Annotator (VIA): Fast, light, and really well designed. This is the one I ended up using.
- COCO UI: The tool used to annotate the COCO dataset.