Deep Learning & Engineering Applications

6. Image Segmentation

JIDONG J. YANG

COLLEGE OF ENGINEERING

UNIVERSITY OF GEORGIA

Semantic Segmentation

Semantic segmentation recognizes and understands what are in images in pixel level.

The labeling and prediction of semantic regions are in pixel level.

It will be convenient if the spatial dimensions of the input and output are the same.

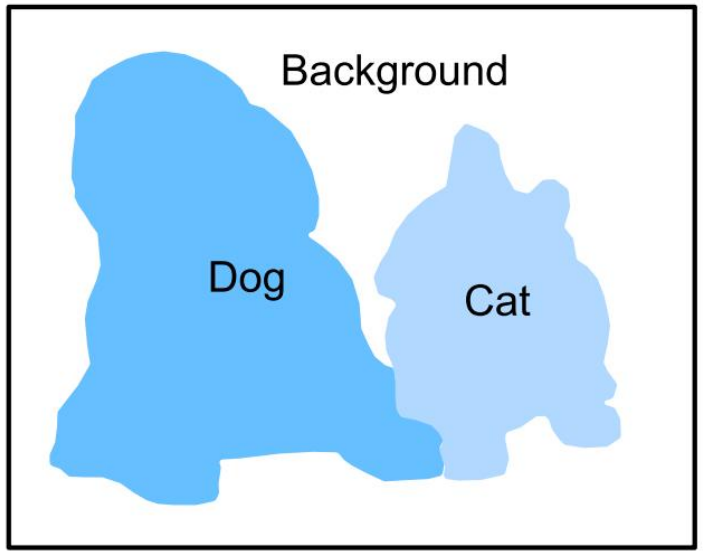


Image Segmentation

Semantic Segmentation

- Divide an image into regions belonging to different semantic classes

Instance Segmentation

- Recognize the pixel-level regions of each object instance in an image, i.e., distinguishing not only semantics, but also different object instances.

Panoptic Segmentation

- Combine semantic and instance segmentations

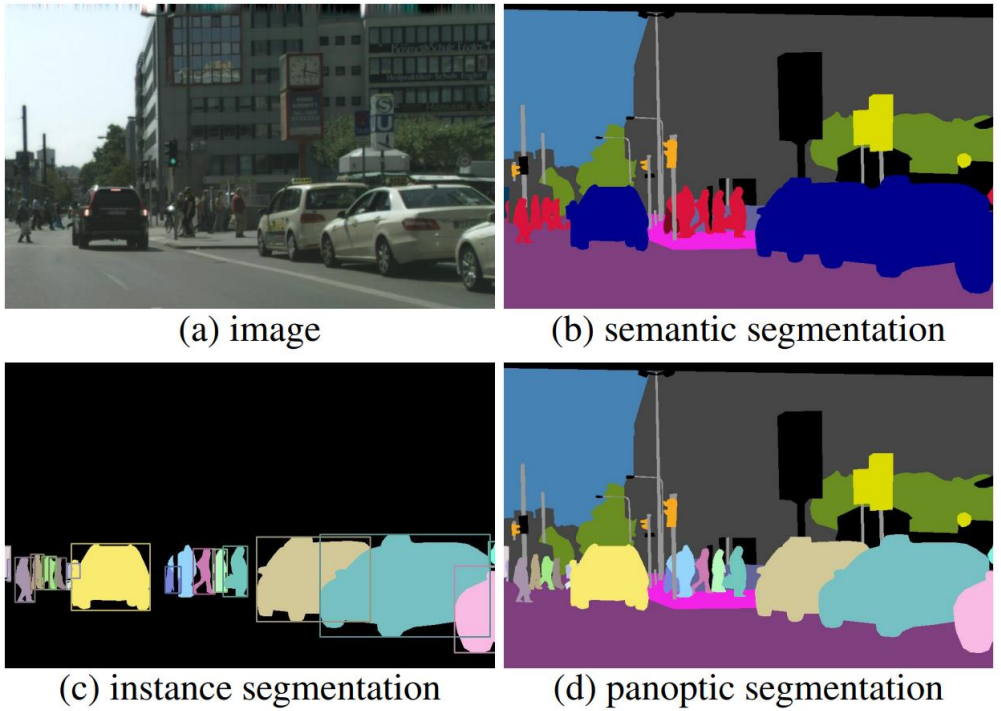


Figure 1: For a given (a) image, we show *ground truth* for: (b) semantic segmentation (per-pixel class labels), (c) instance segmentation (per-object mask and class label), and (d) the proposed *panoptic segmentation* task (per-pixel class+instance labels). The PS task: (1) encompasses both stuff and thing classes, (2) uses a simple but general format, and (3) introduces a uniform evaluation metric for all classes. Panoptic segmentation generalizes both semantic and instance segmentation and we expect the unified task will present novel challenges and enable innovative new methods.

[Kirillov et al., 2018] https://arxiv.org/pdf/1801.00868.pdf

Transposed Convolution

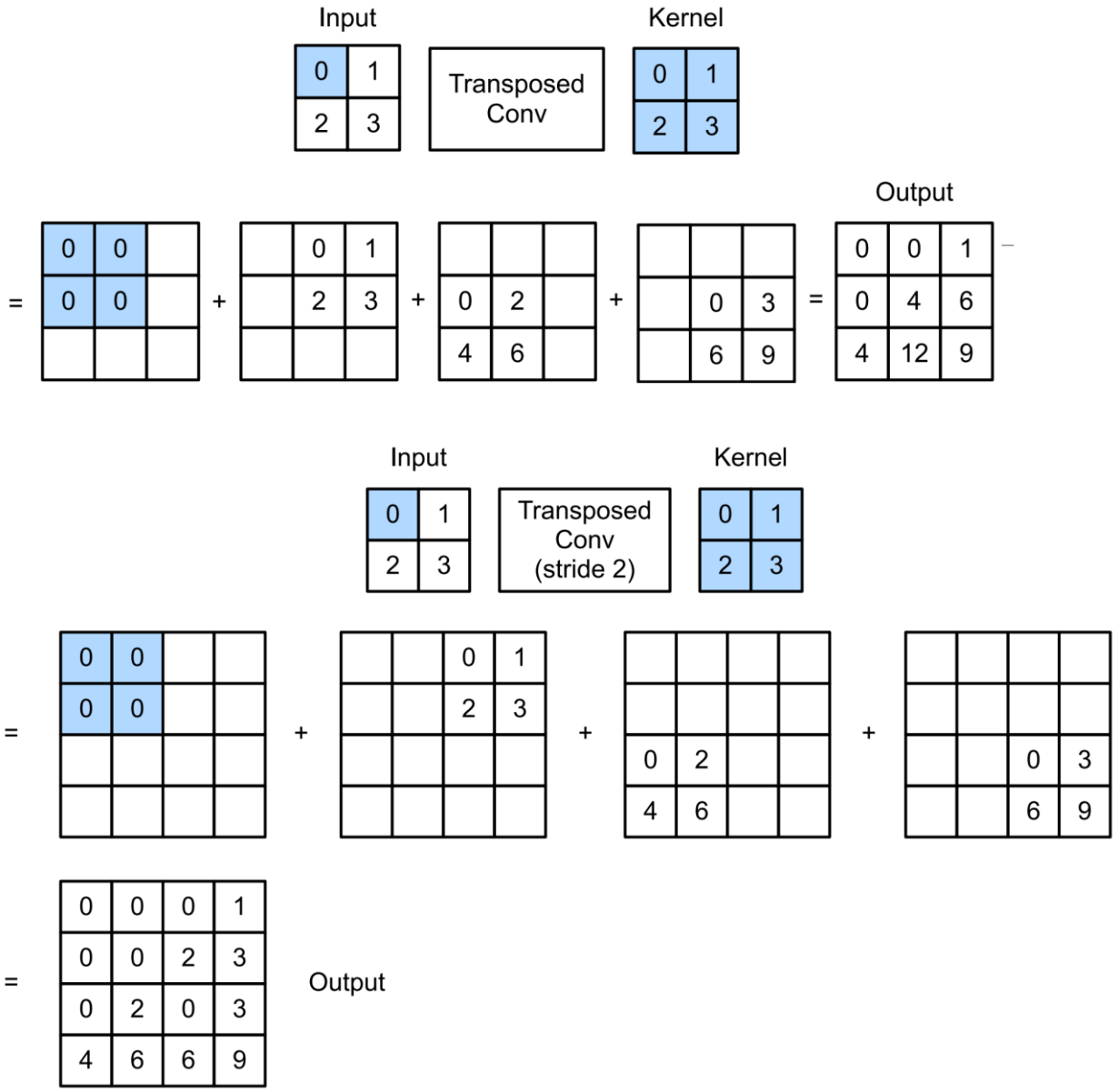
Upsampling

Padding = p, remove the first and last p rows and columns from the output.

Stride: specified for the output.

Named after the matrix transposition

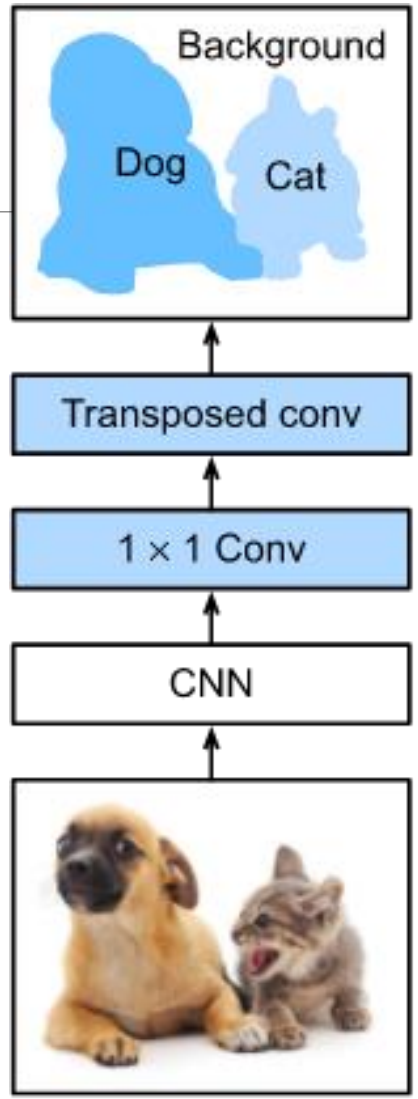
Can be implemented as matrix multiplication



Fully Convolutional Networks (FCN)

Basic design of FCN

1. Use a CNN to extract image features.
2. Transform the number of channels into the number of classes via a 1×1 convolution.
3. Transform the height and width of the feature maps to those of the input image via transposed convolution.



[Long et al., 2015] https://arxiv.org/abs/1411.4038

Fully Convolutional Networks (FCN) for Semantic Segmentation

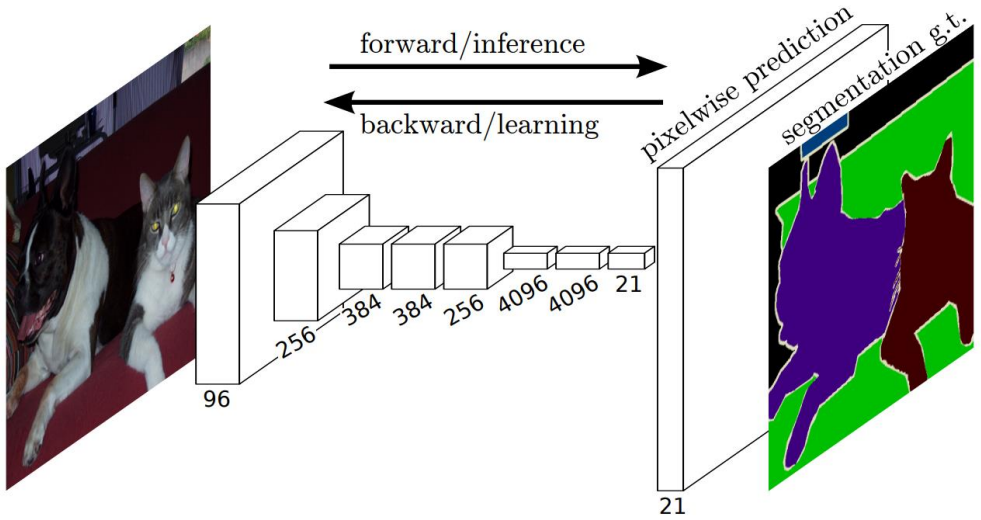


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

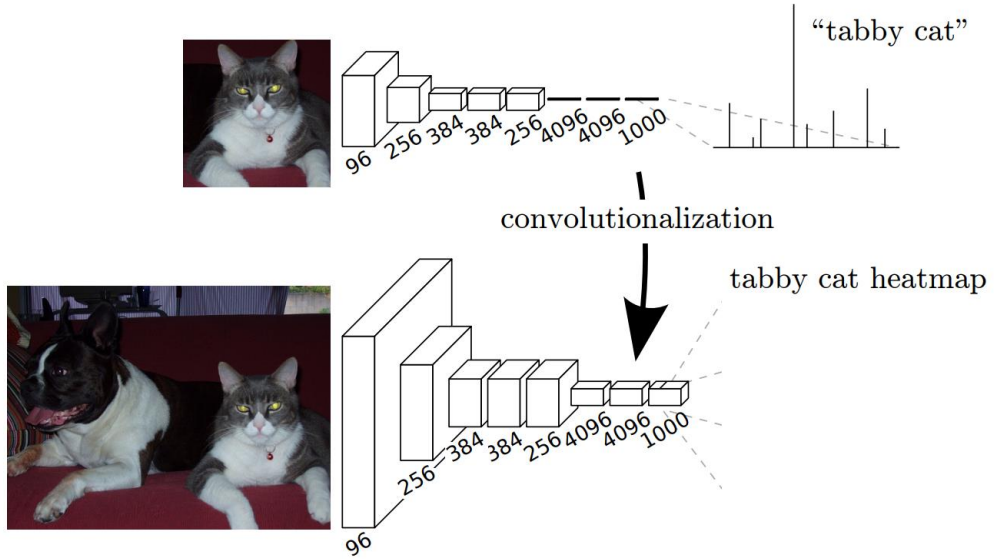


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

[Long et al., 2015] https://arxiv.org/abs/1411.4038

FCN

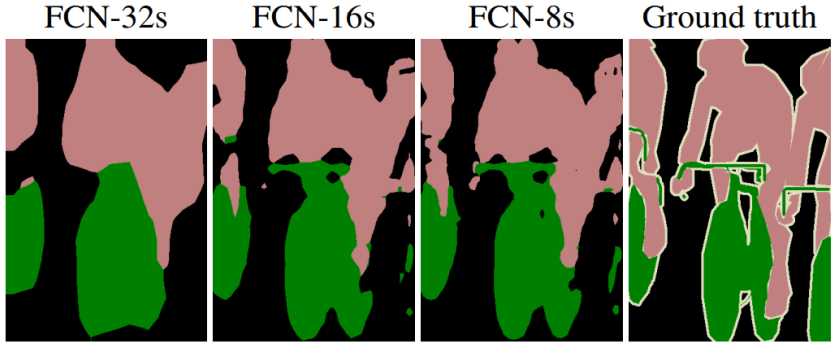


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

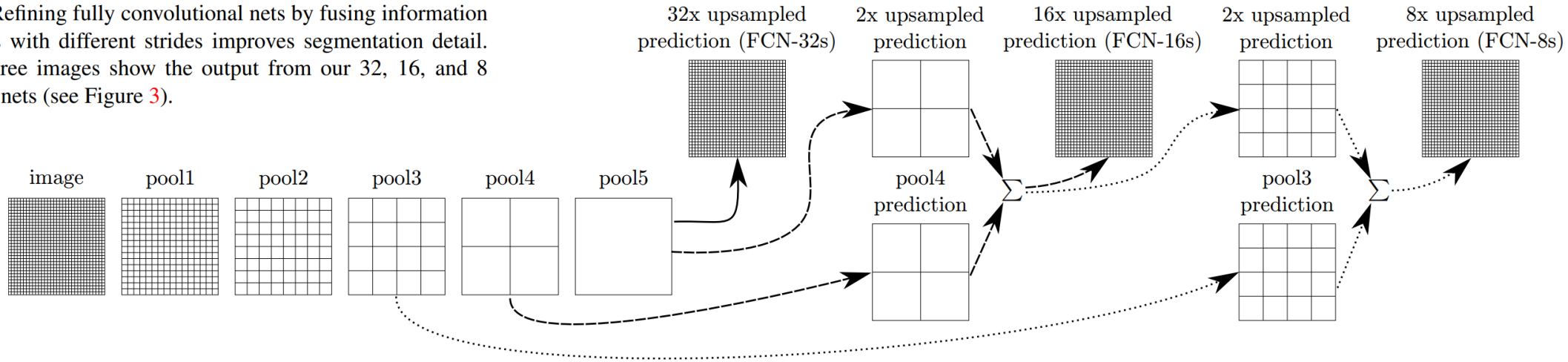


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

[Long et al., 2015] https://arxiv.org/abs/1411.4038

U-net

Proposed for the medical imaging community

Built upon “fully convolutional network”. The architecture includes

- a contracting path to **capture context**
- a symmetric expanding path that **enables precise localization**.

Transfer the entire feature map (**at the cost of more memory**) to the corresponding decoders and concatenates them to upsampled (via deconvolution) decoder feature maps.

Upsampling allows the network to propagate context information to higher resolution layers.

Trained end-to-end from a few images (e.g., random elastic deformations of the training samples)

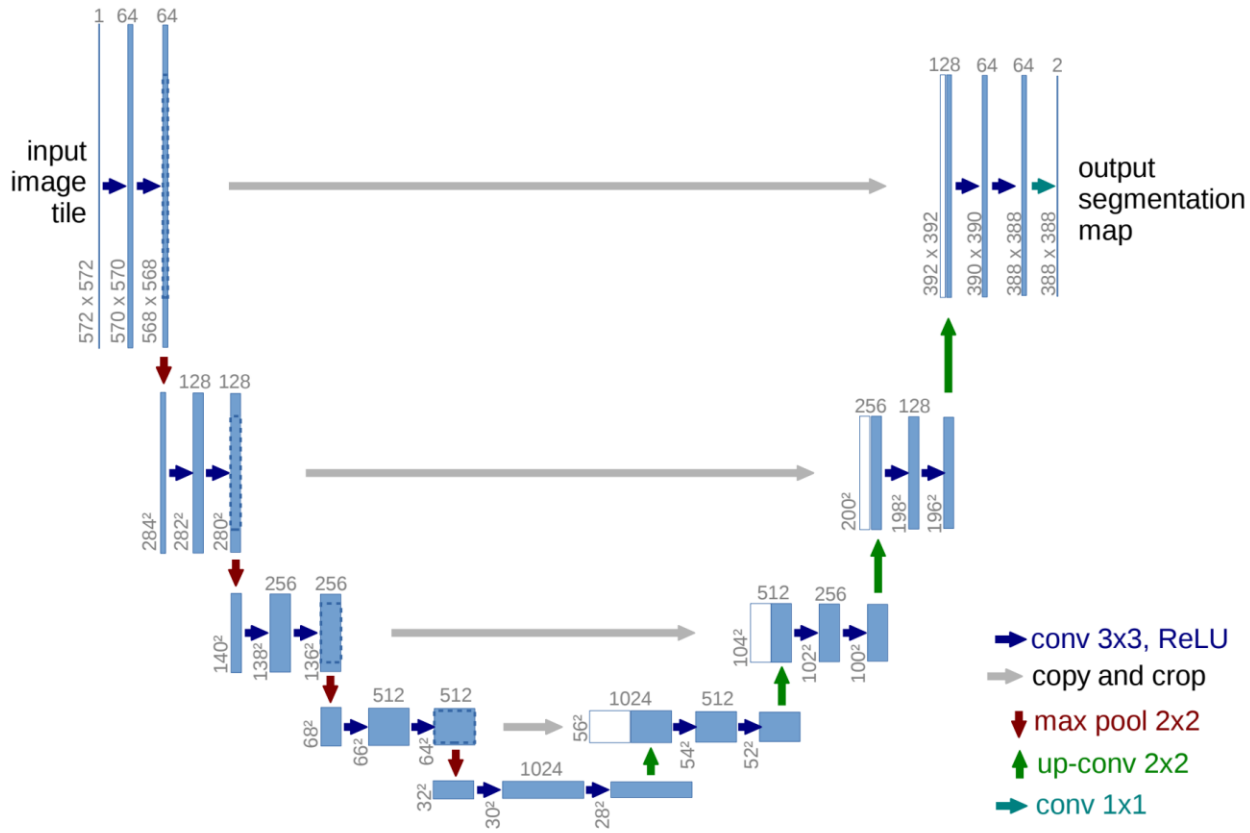


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows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

[Ronneberger et al., 2015] https://arxiv.org/pdf/1505.04597.pdf

SegNet

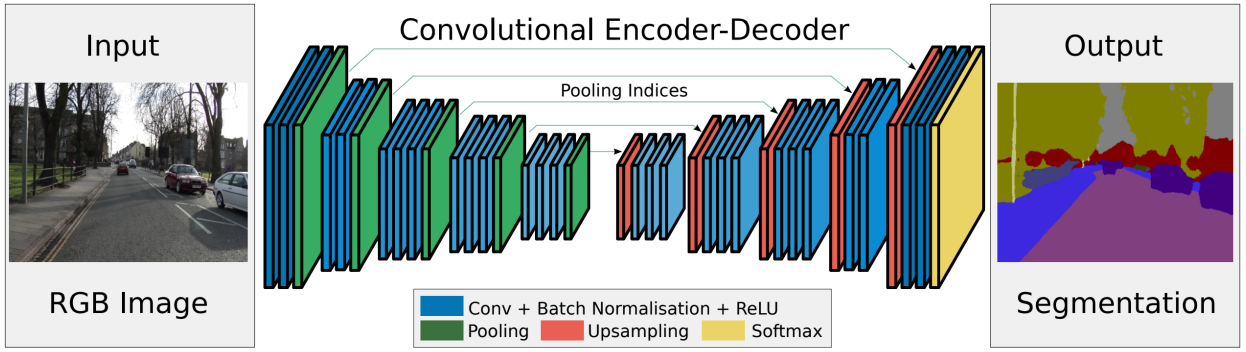


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Semantic pixel-wise segmentation

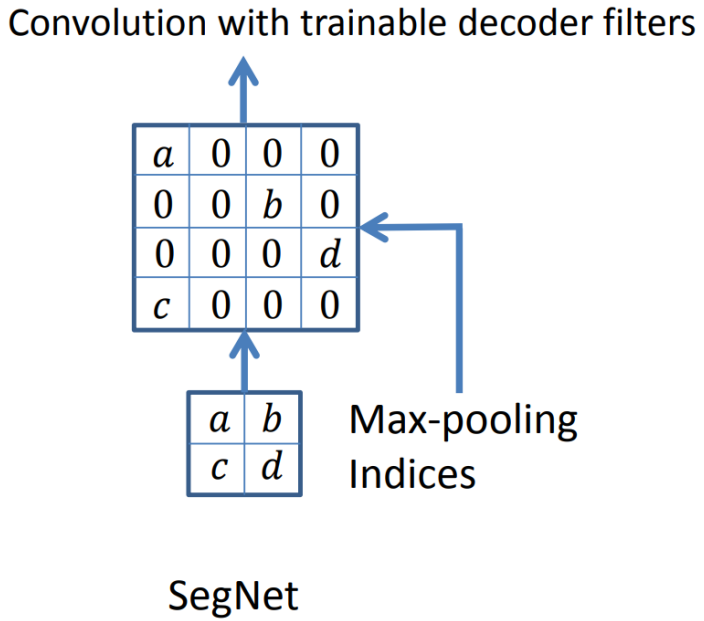
The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network.

The role of the decoder network is to map the low-resolution encoder feature maps to full input resolution feature maps for pixel-wise classification.

It involves storing only the max-pooling indices (memory efficient). Specifically, the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. **This eliminates the need for learning to upsample.**

The upsampled maps are sparse and are then convolved with trainable filters to produce dense feature maps.

Finally, a K-class softmax classifier is used to predict the class for each pixel.



[Badrinarayanan et al., 2016] https://arxiv.org/abs/1511.00561

SegNet

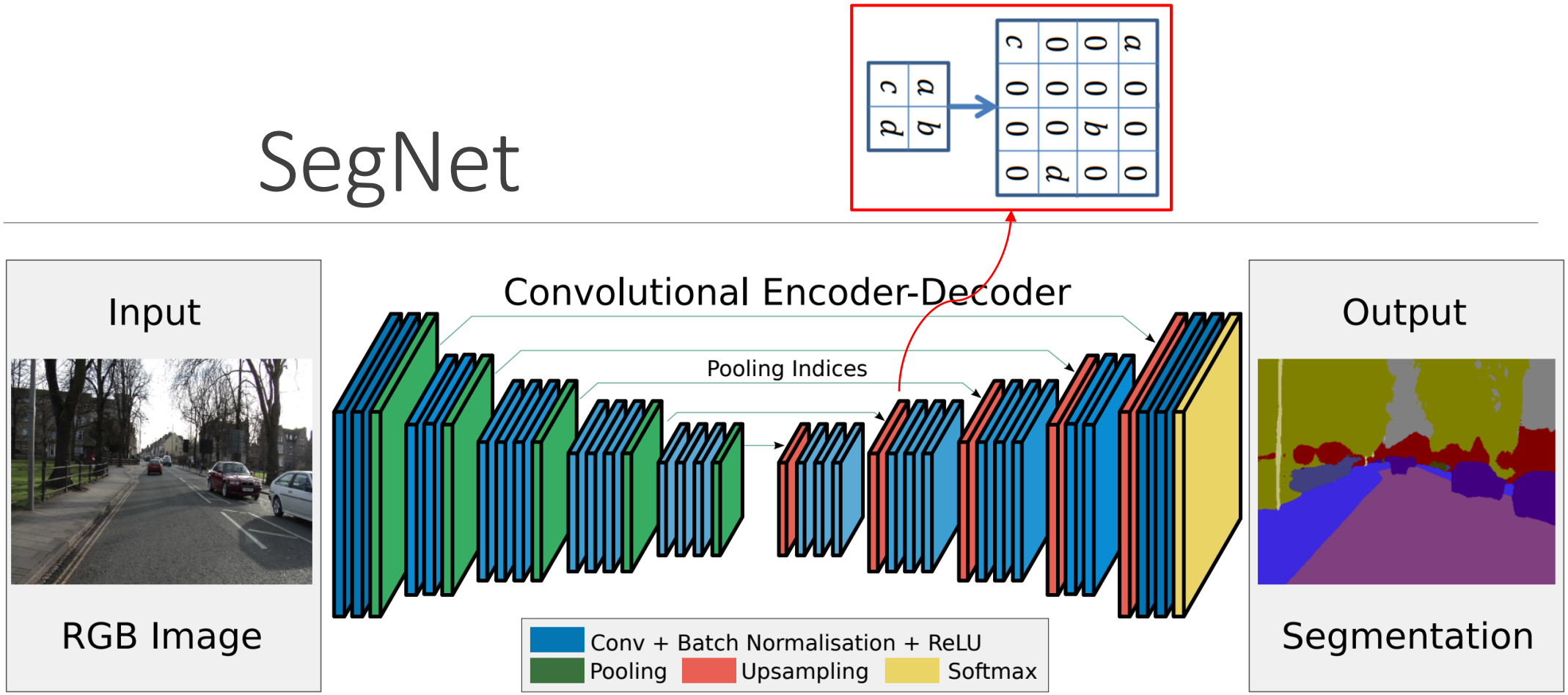


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Mask R-CNN

Mask R-CNN adopts the two-stage procedure as Faster R-CNN, with an identical first stage (which is RPN).

In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI.

Use multitask loss on each sampled RoI.

- $L = L_{cls} + L_{box} + L_{mask}$

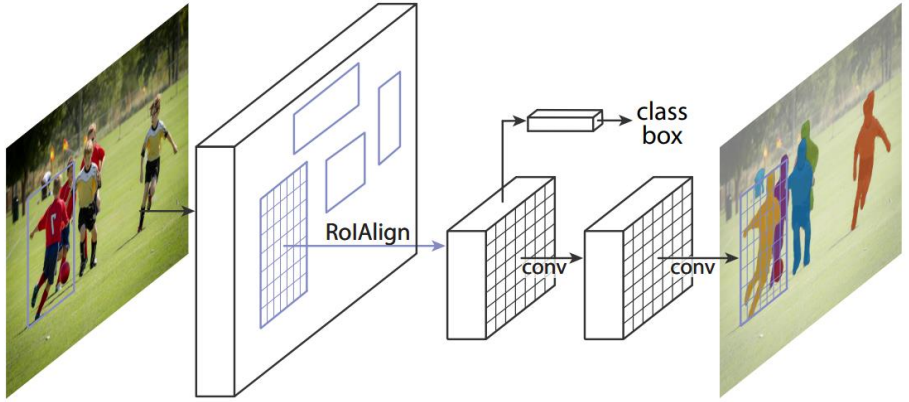


Figure 1. The **Mask R-CNN** framework for instance segmentation.

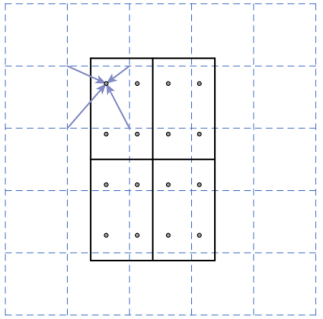
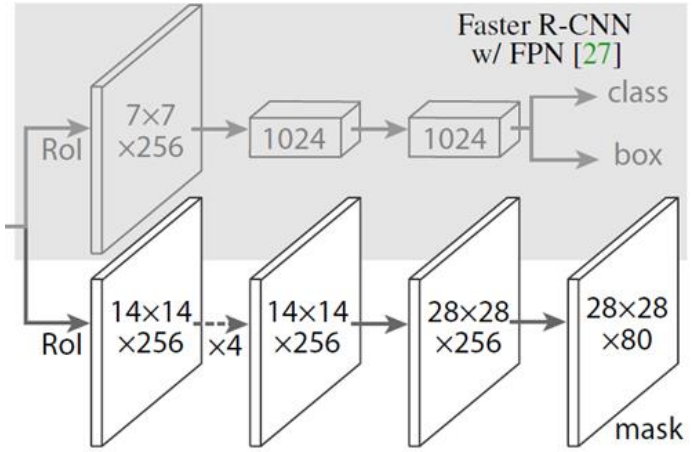


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.



[He et al., 2017] https://arxiv.org/pdf/1703.06870.pdf