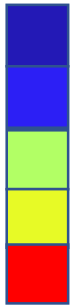
Deep Learning for Semantic Image Segmentation -----SegNet

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

The Task of Semantic Image Segmentation



person
grass
trees
motorbike
road

The Task of Semantic Image Segmentation

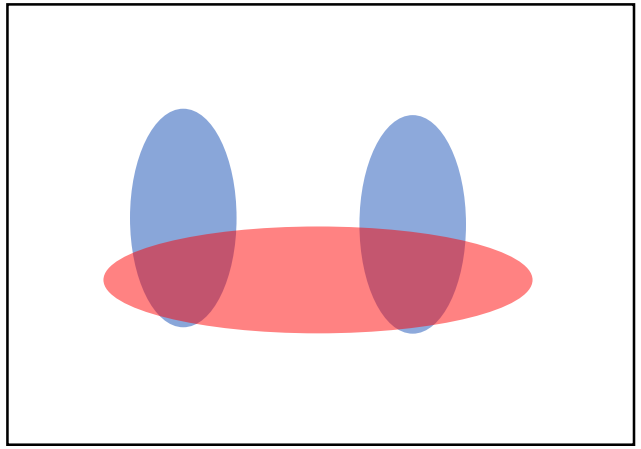
- Pixel-level classification

(Badrinarayanan, Kendall, & Cipolla, 2015)

(Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2014)

Evaluation metric

- Pixel classification!
- Accuracy?
  - Heavily unbalanced
  - Common classes are over-emphasized
- *Intersection over Union*
  - Average across classes and images
- Per-class accuracy
  - Compute accuracy for every class and then average



Things *vs* Stuff

THINGS

- Person, cat, horse, etc
- Constrained shape
- Individual instances with separate identity
- May need to look at objects

STUFF

- Road, grass, sky etc
- Amorphous, no shape
- No notion of instances
- Can be done at pixel level
- “texture”

Challenges in data collection

- Precise localization is hard to annotate
- Annotating every pixel leads to heavy tails
- Common solution: annotate few classes (often things), mark rest as “Other”
- Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)

Pre-convnet semantic segmentation

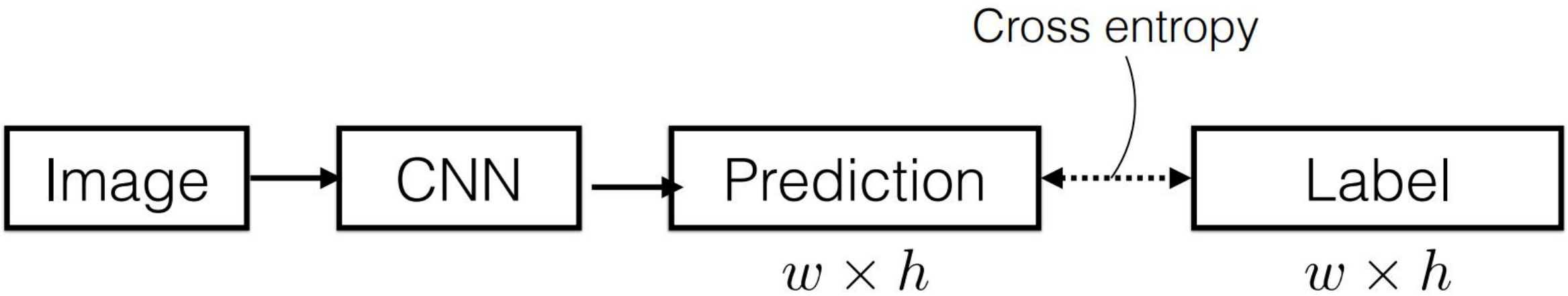
- Things
  - Do object detection, then segment out detected objects
- Stuff
  - ”Texture classification”
  - Compute histograms of filter responses
  - Classify local image patches

Previously…

- Hand-engineered features, various classifiers
- Deep Convolutional Neural Nets
  - Success at other *high-level* vision tasks (abstract representations)
- DCNN hurdles for low-level tasks:
  - Signal down-sampling ---> reduced signal resolution
  - Spatial invariance ---> limits spatial accuracy
  - In general – hard to train

Typical formulation

- Tackling this problem with CNN, usually it's formulated as:

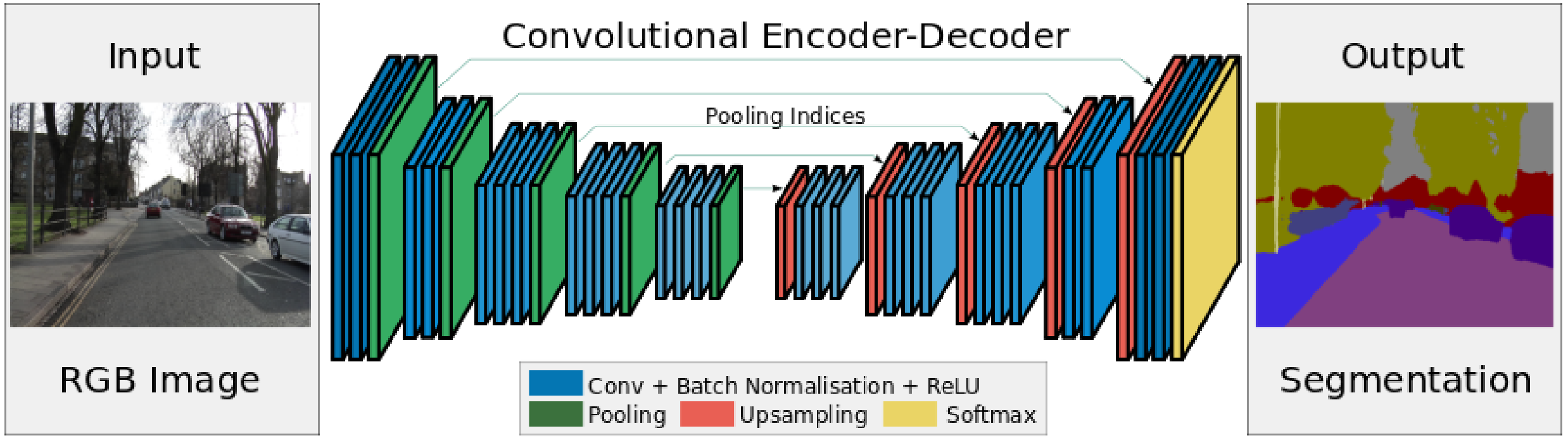


- The loss is calculated for **each pixel independently**
- It leads to the problem: "How can the model consider the context to make a single prediction for a pixel?"

Common problems

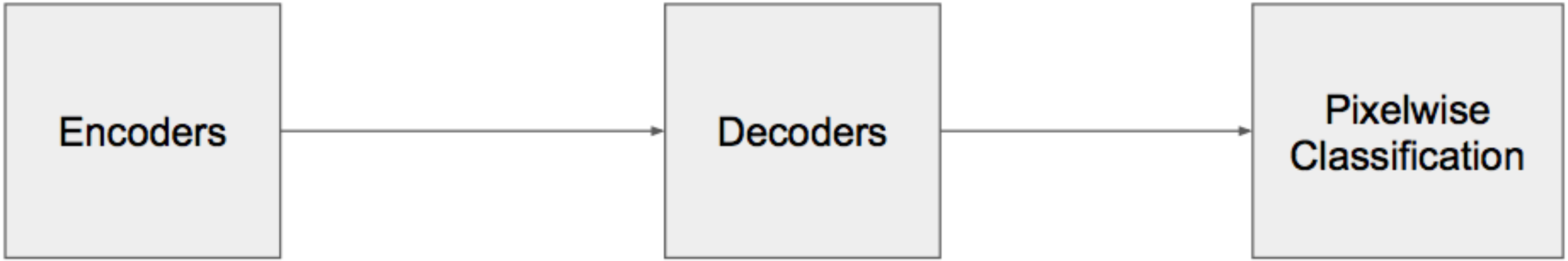
- How to leverage context information
- How to use law-level features in upper layers to make detailed predictions
- How to create dense prediction

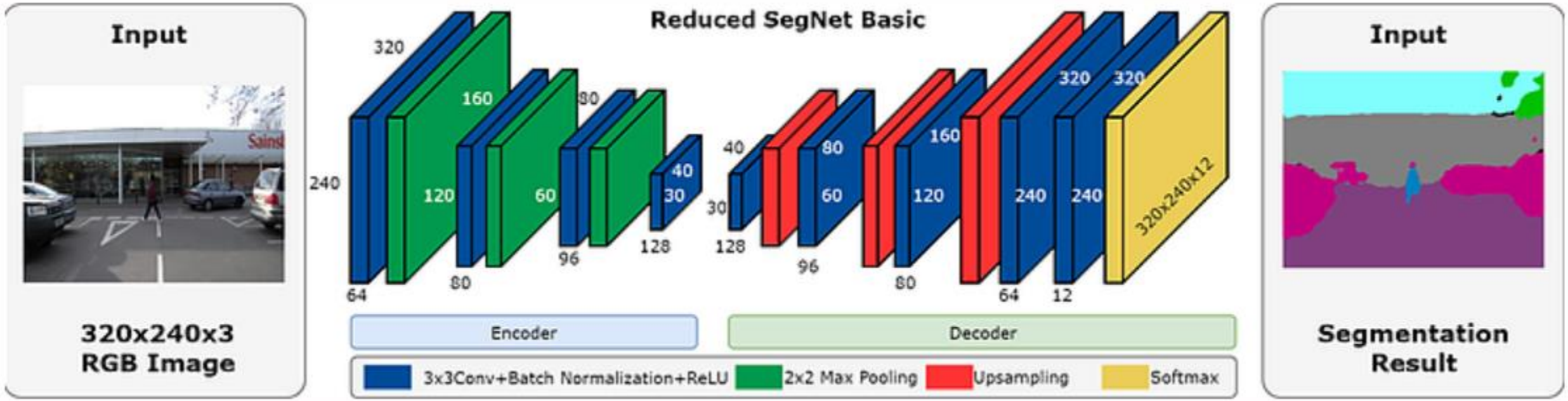
Network Architecture for SegNet

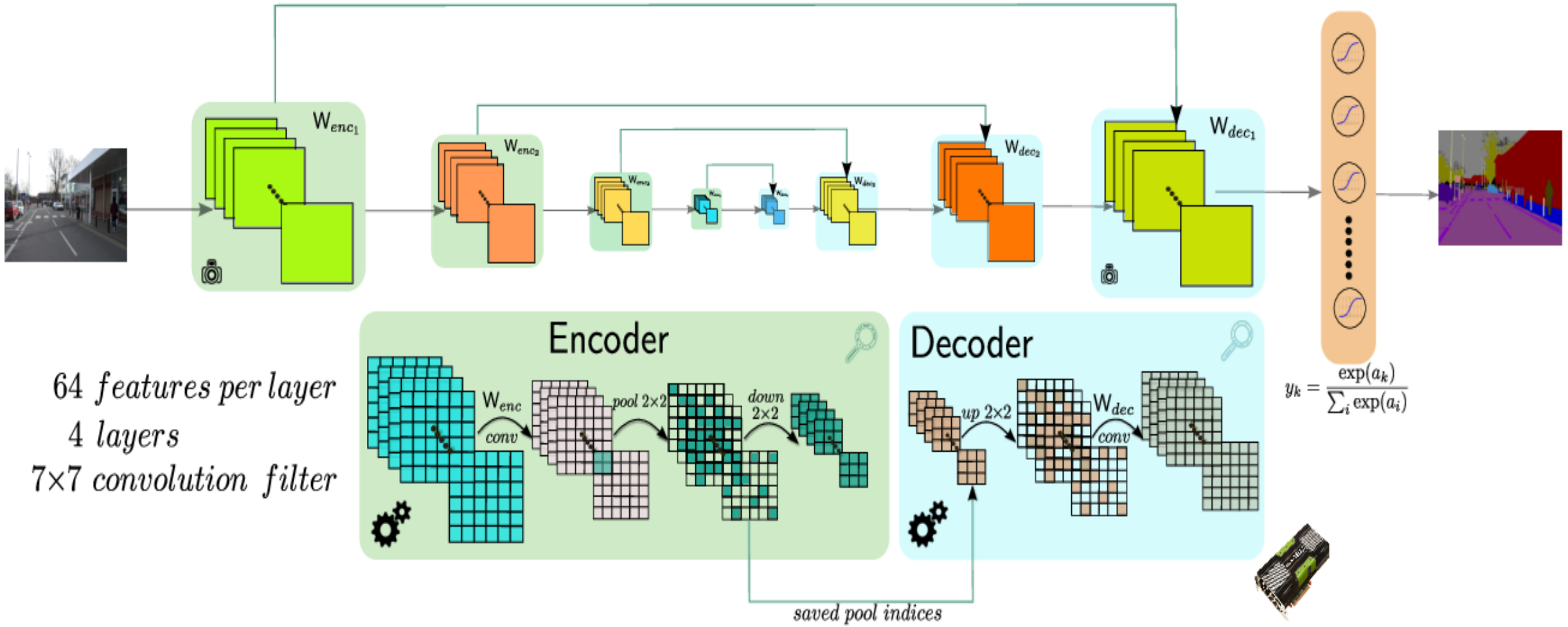


1. **The architecture consists of an encoder-decoder followed by a pixel-wise classifier.**
2. **Each consists of one or more convolutional layers with batch normalization and a ReLU non-linearity, followed by non-overlapping max-pooling and sub-sampling.**
3. **Each decoder consists of one or more de-convolutional layers with upsamling.**
4. **One key ingredient of the SegNet is the use of max-pooling indices in the decoders to perform upsampling of low resolution feature maps.**
5. **The entire architecture can be trained end-to-end using stochastic gradient descent.**

Encoder-Decoder pairs are used to create feature maps for classifications of different resolutions.



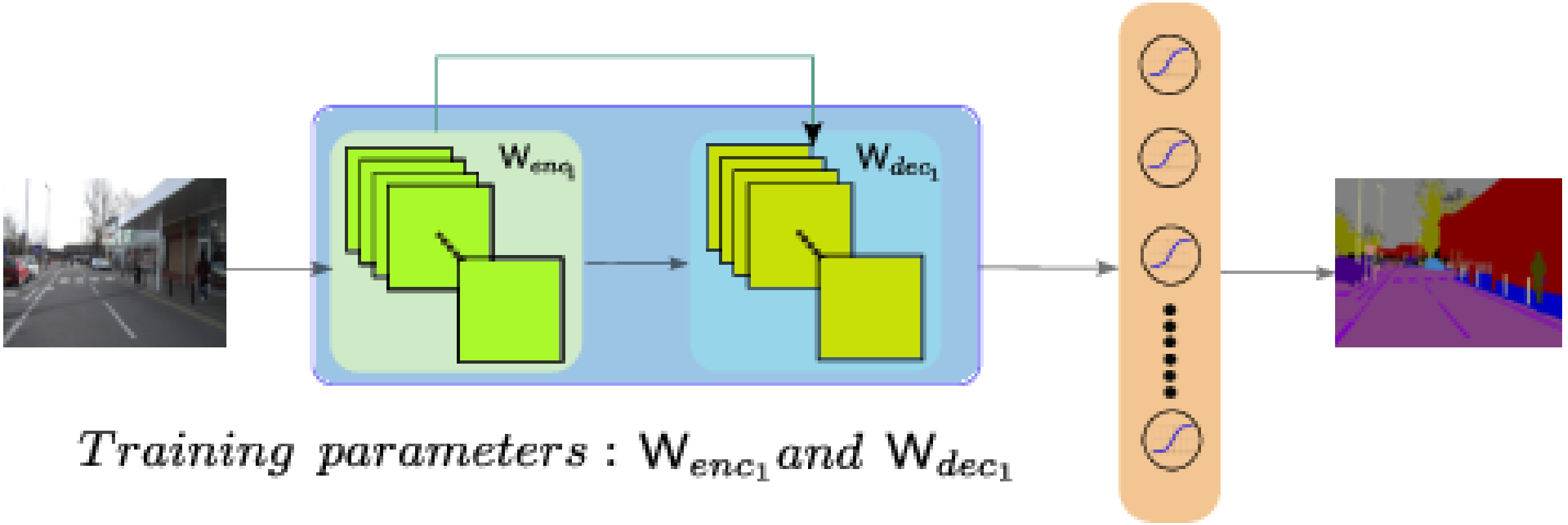




64 *features per layer*

4 *layers*

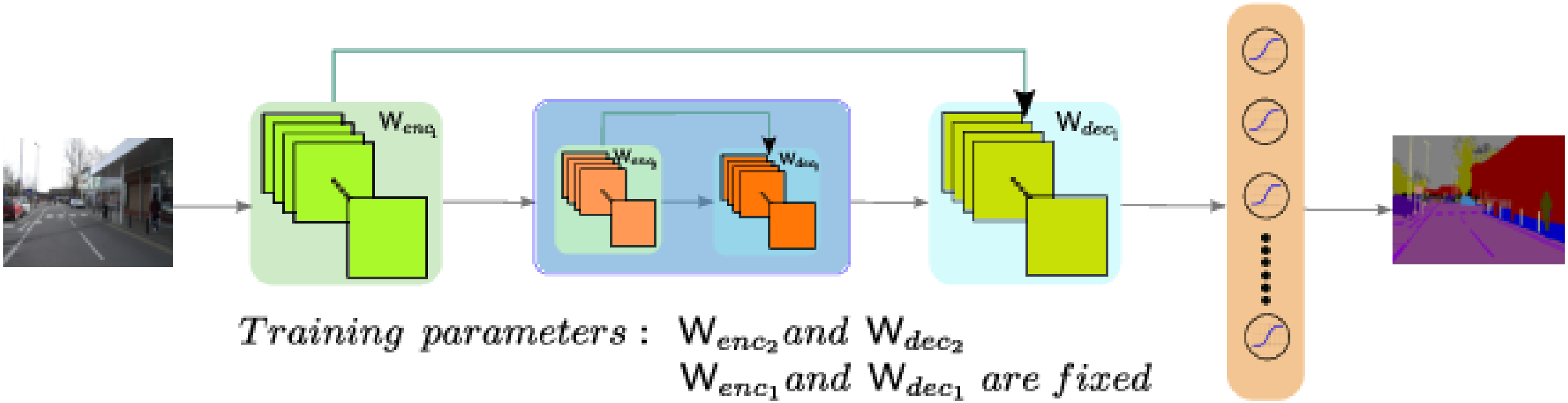
7×7 *convolution filter*



$Training\ parameters : W_{enc_1} and\ W_{dec_1}$

(a)

$$y_k = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$$



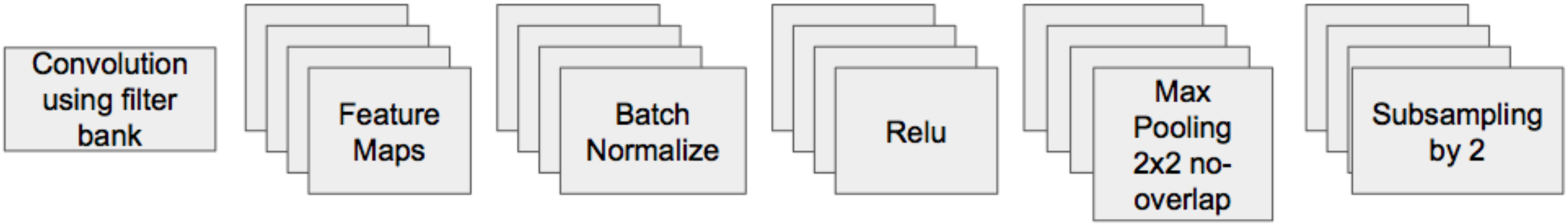
$Training\ parameters:\ W_{enc_2}\ and\ W_{dec_2}$
$W_{enc_1}\ and\ W_{dec_1}\ are\ fixed$

(b)

$$y_k = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$$

Encoder

1. 13 VGG16 Conv layers
2. Not fully connected, this reduces parameters from 134M to 14.7M
3. Good initial weights are available hence these layers are made non trainable



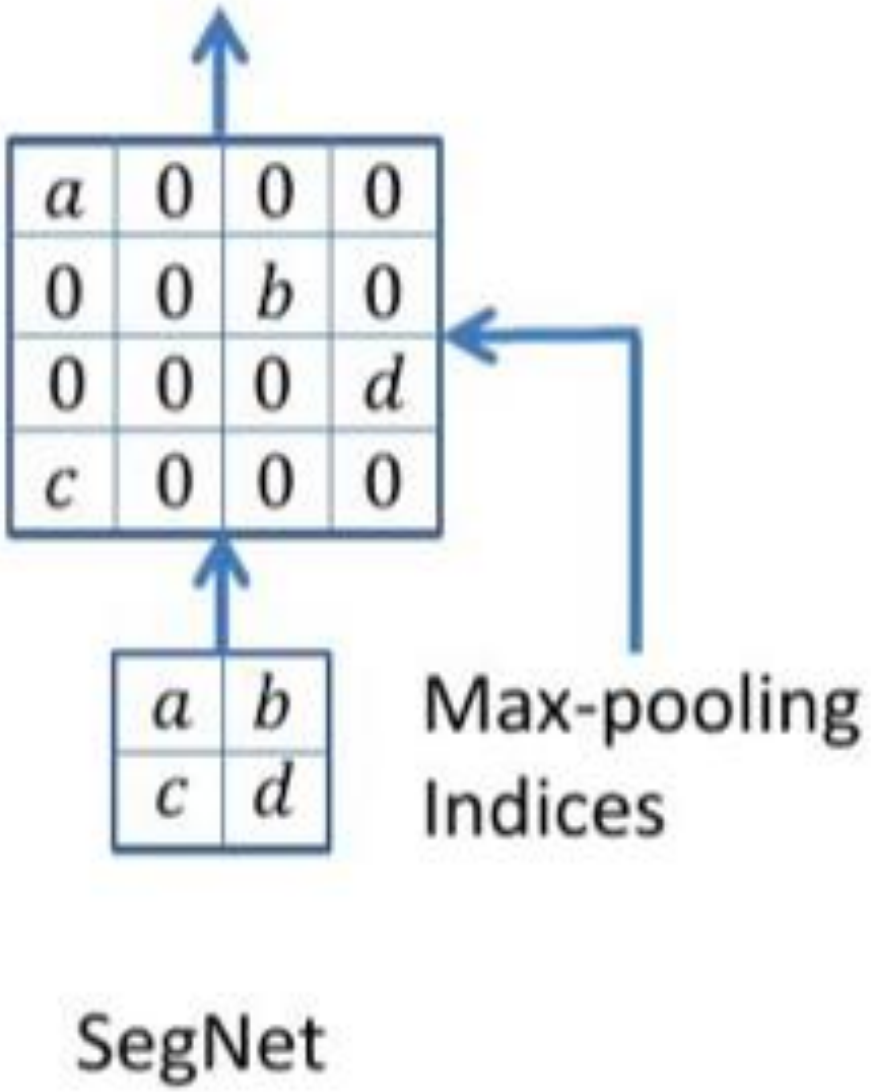
Advantages

1. **Improved boundary delineation**
2. **Less number of parameters**

1. To reduce the feature map size, each encoder contains a **subsampling stage**, max-pooling is used to achieve translation invariance over small spatial shifts in the image, combine that with subsampling and it leads to each pixel governing a ***larger input image context*** (spatial window).

2. To guide up-sampling at decoder, SegNet stores only the ***max-pooling indices*** i.e. the locations of maximum feature value in each pooling window is memorized for each encoder map, which are used to *capture and store* boundary information in the encoder feature maps before sub-sampling.

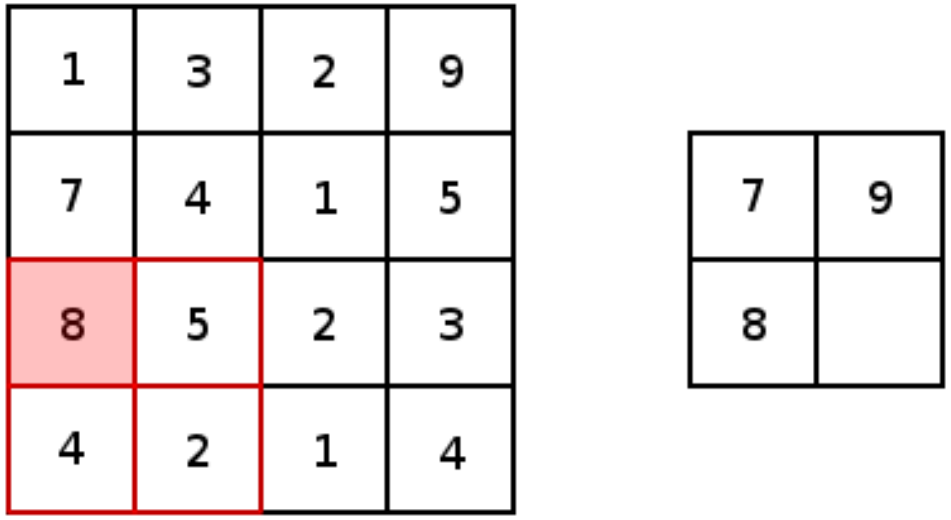
3. Only 2 bits are needed for each window of 2x2, slight loss of precision, but *tradeoff*.



This form of up-sampling can be incorporated in any encoder-decoder architecture

SegNet stores only the ***max-pooling indices*** i.e. the locations of maximum feature value in each pooling window is memorized for each encoder map, which are used to *capture and store* boundary information in the encoder feature maps before sub-sampling.

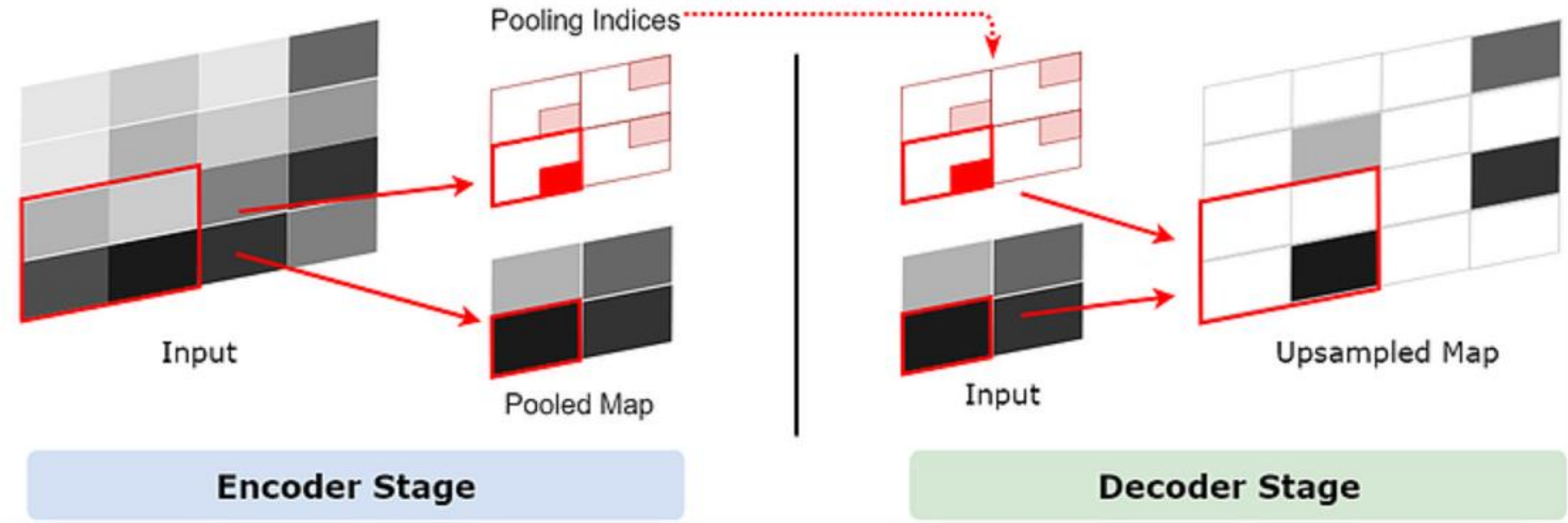
Only 2 bits are needed for each window of 2x2, slight loss of precision, but *tradeoff*.



In SegNet, a non-overlapped 2x2 max pooling is adopted as illustrated below. Only the maximum value inside this 2x2 region is kept and propagated to the next layer.

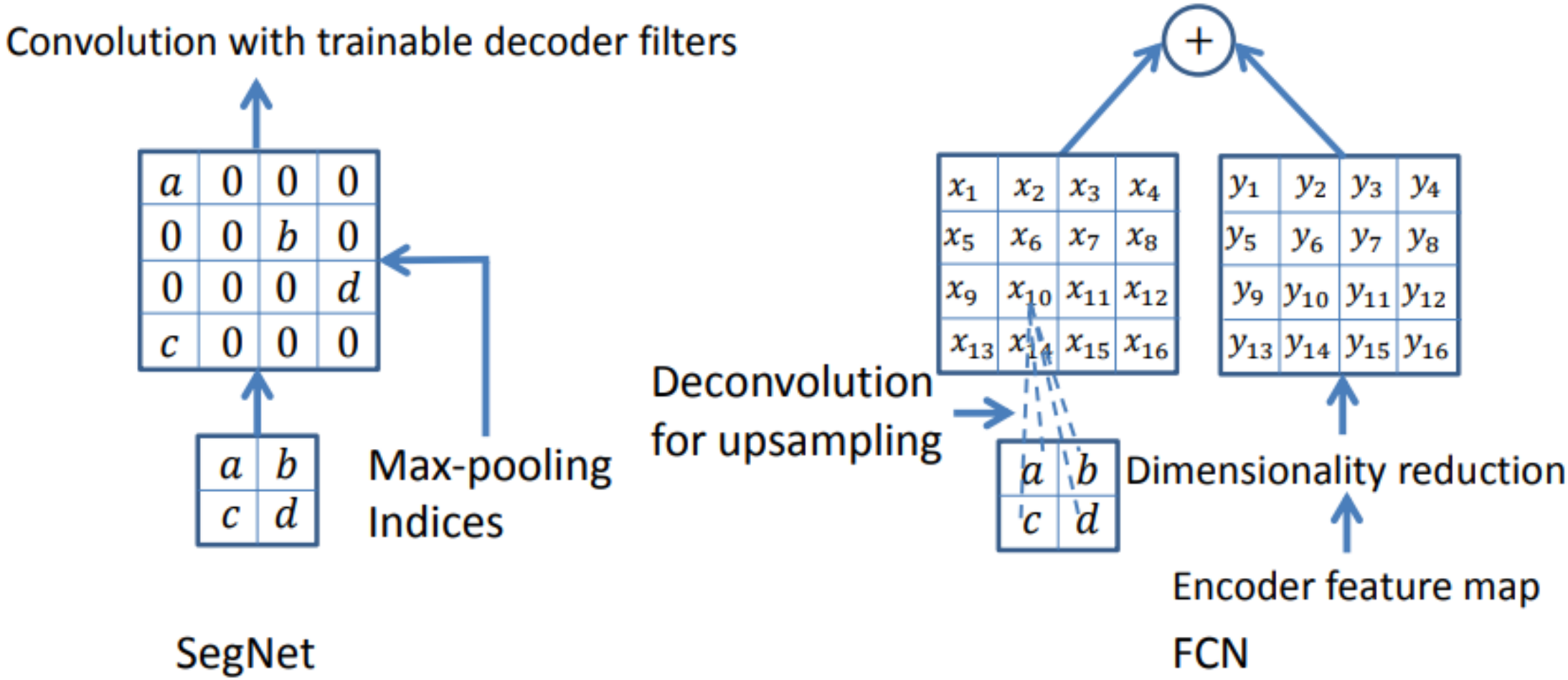
Upsampling, is just the opposite way of pooling. However, there is one uncertain fact that during Upsampling, the size of 1x1 becomes 2x2, one of which will be taken by the original 1x1 feature and others will be empty. But which location should be assigned to the original 1x1 feature? We can assign it randomly or in a fixed way, but this might add errors to the next layer. The deeper it goes, the more it affects the layer that follows so it is important to keep the 1x1 feature in its proper place.

In SegNet, this is implemented through a so-called **Pooling Indices**. On the Encoder stage, the index of the maximum feature in each 2x2 area is saved for Decoder to use. Now that we have the Pooling Indices and given the fact that the whole network is symmetric, during Upsampling on the Decoder stage, the 1x1 feature goes to the exact location where the corresponding Pooling Index shows, as illustrated in the graph below.



Pooling

Up-Pooling


Convolution with trainable decoder filters
a 0 0 0
0 0 b 0
0 0 0 d
c 0 0 0
a b
c d
Max-pooling
Indices
SegNet
+
x1 x2 x3 x4
x5 x6 x7 x8
x9 x10 x11 x12
x13 x14 x15 x16
y1 y2 y3 y4
y5 y6 y7 y8
y9 y10 y11 y12
y13 y14 y15 y16
Deconvolution
for upsampling
a b
c d
Dimensionality reduction
Encoder feature map
FCN

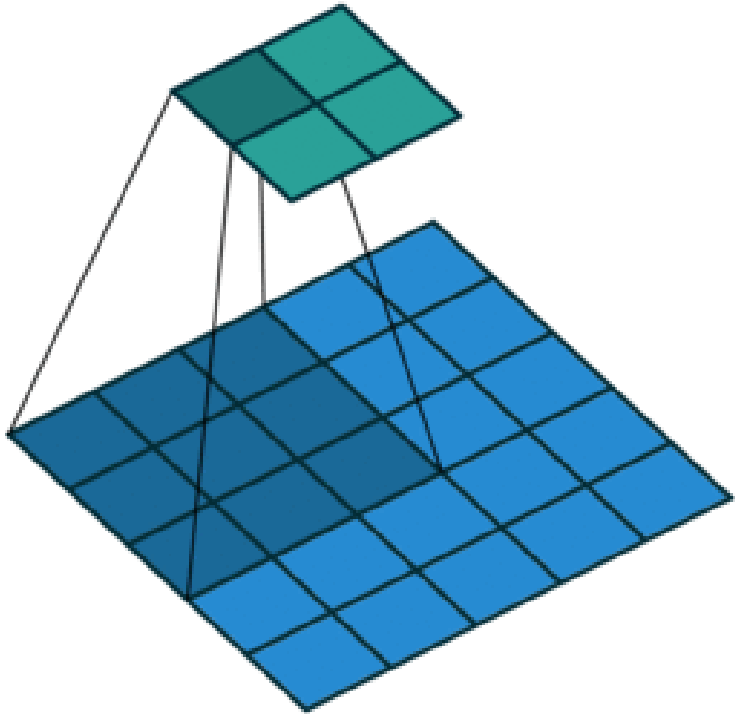
Convolution Layer in Encoder and Decoder

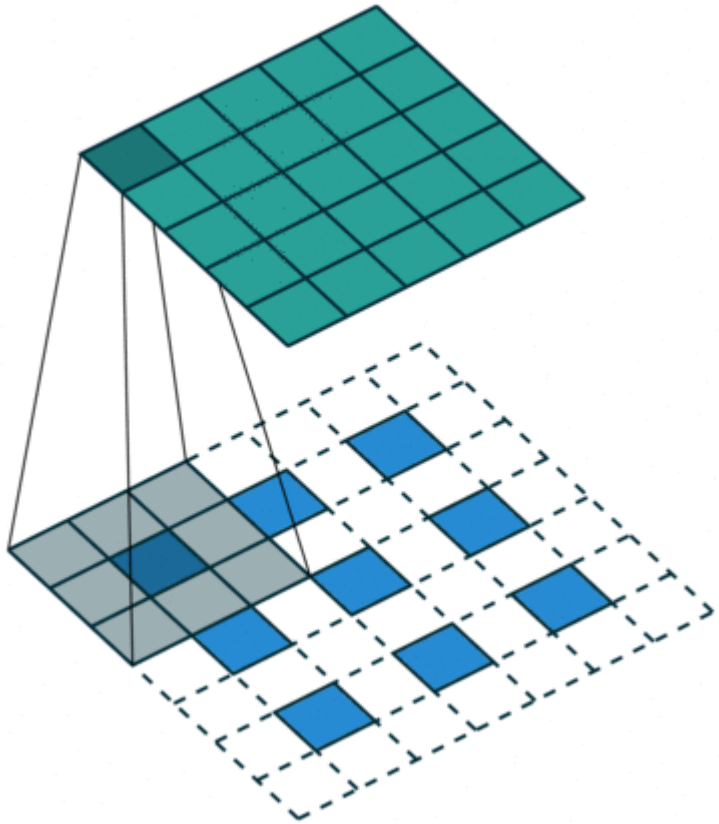
The Convolution Layers exist in both Encoder Stage and Decode Stage. They work mechanically the same but have slightly different meanings.

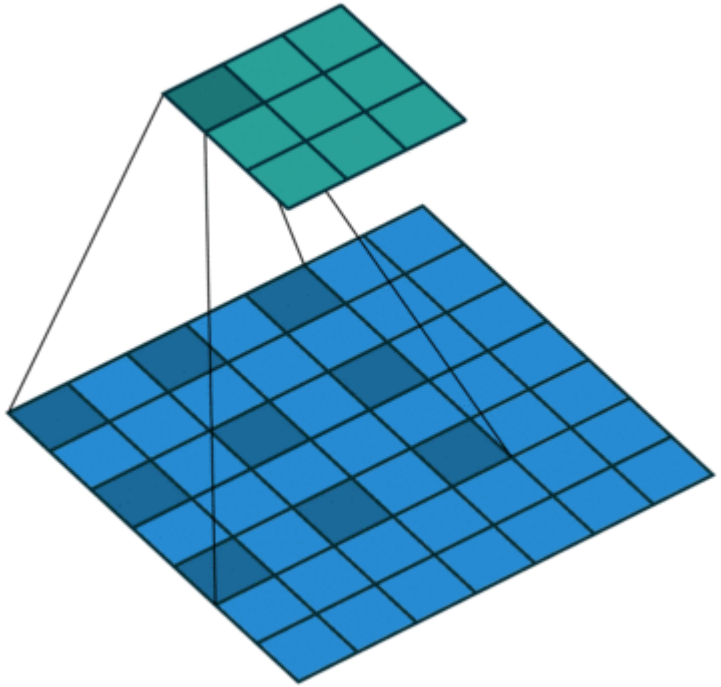
In Encoder Stage, the Convolution Layer extracts local features and pass them to Pooling Layers, which subsamples the input feature map by keeping the max value in the 2x2 area. You can think that the Convolution+Pooling in Encoder Stage learns to "extract" features.

On the contrary, in the Decoder Stage, the input feature map is first up-sampled before going to Convolution Layers. In a 2x2 upsampled area, only one value is propagated from previous layer and others are empty. Later the empty spots will be filled up by Convolution Layer. So here, the Upsampling+Convolution learns to "add" features to the feature map in order to make it smother.

In a word, the computation behind these Convolution Layers are totally the same, but they somehow results in a impression that they behave differently. Some people name them after "Convolution" in the Encoder Stage and "Deconvolution" or "Transposed Convolution" in the Decoder Stage, but in fact the computation of these two make no difference. Here are two animations of "Convolution" (left) and so-called "Deconvolution"(right).





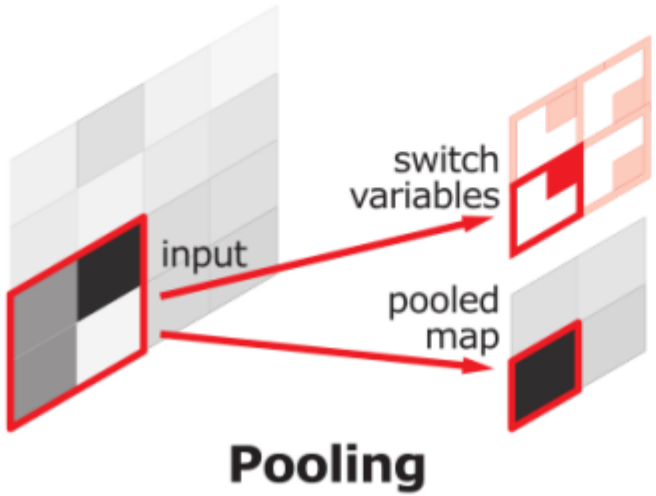


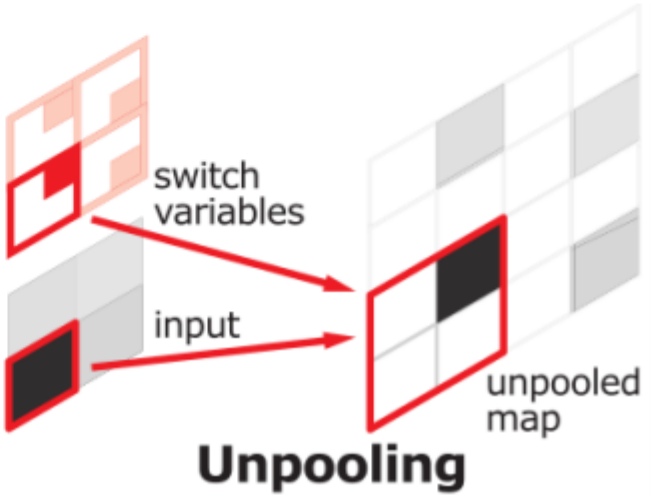
- Encoder Network: extract image features using deep convolutional network
  - Each layer: bank of trainable convolutional filters, followed by ReLUs and max-pooling to downsample image features
- Decoder Network: upsamples feature map back to image resolution with final output having same number of channels as there are pixel classes
  - Where the methods differ most dramatically
  - Network mirrors encoder network
- Pixel-wise softmax over final feature map and cross-entropy loss function for training using SGD.

Decoder

1. For each of the 13 encoders, there is a corresponding decoder which up-samples the feature map using memorized *max-pooling indices*
2. Sparse feature maps of higher resolutions produced
3. Sparse maps are fed through a *trainable filter bank* to produce dense feature maps
4. The last decoder is connected to a *softmax classifier* which classifies each pixel

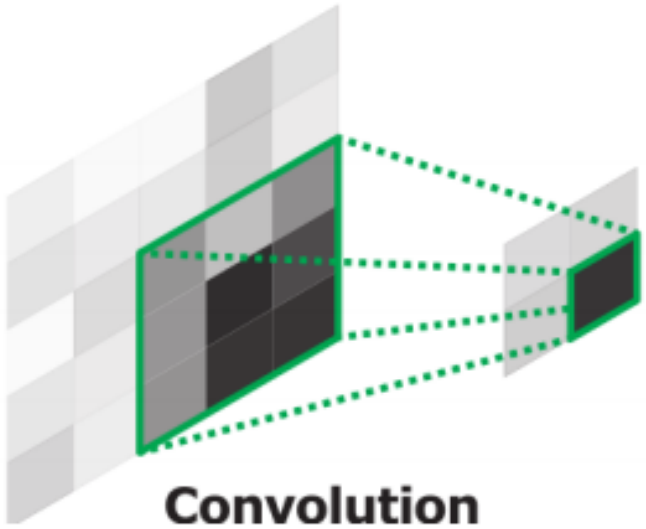
- Upsampling is needed to return feature map to higher resolution for pixel classification
- Pooling destroys spatial information, which is useful for precise localization
- To reconstruct (partially): store max-pooling indices from encoder and place each activation back to its original pooled location
- Pad zeros to other locations



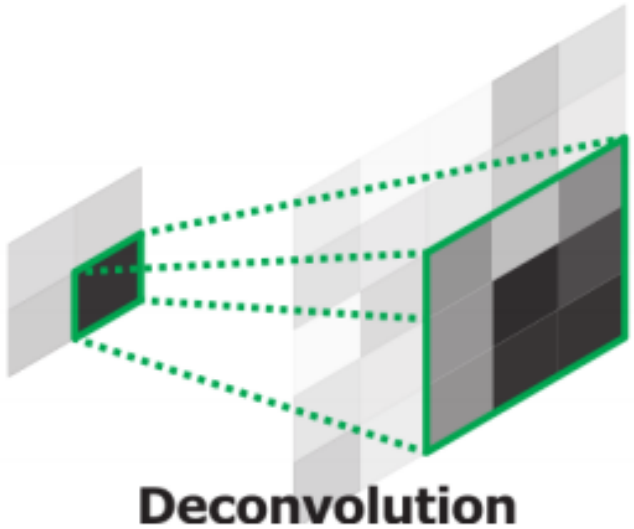


Slide credit to Gregory P. Spell

- Upsampling provides sparse feature maps
- Use trainable (de)convolution filters to densify maps



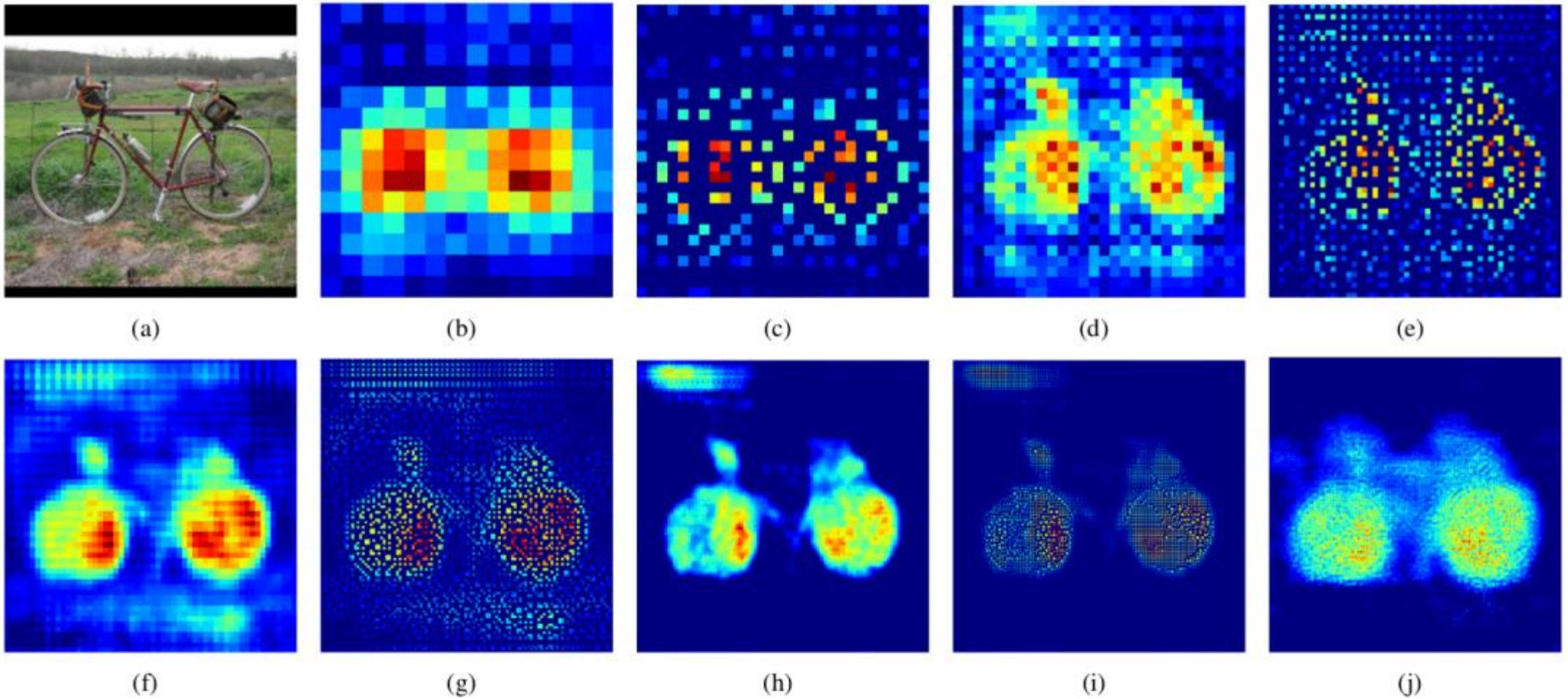
Convolution



Deconvolution

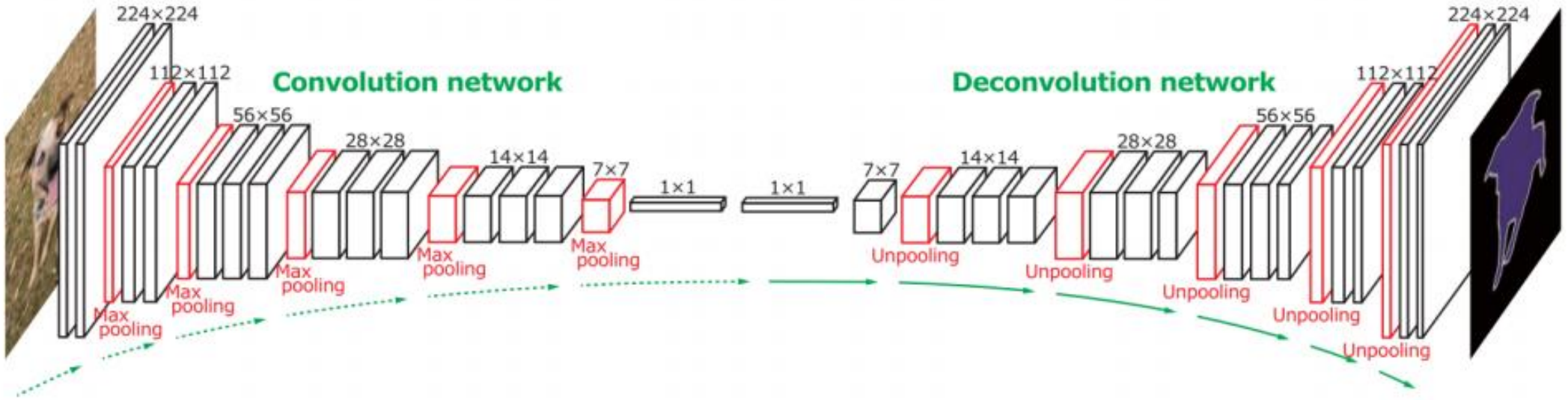
Slide credit to Gregory P. Spell

- Unpooling captures example-specific structures
- Deconvolution captures class-specific shapes
- Hierarchical structure reconstructs shape details from coarse to fine



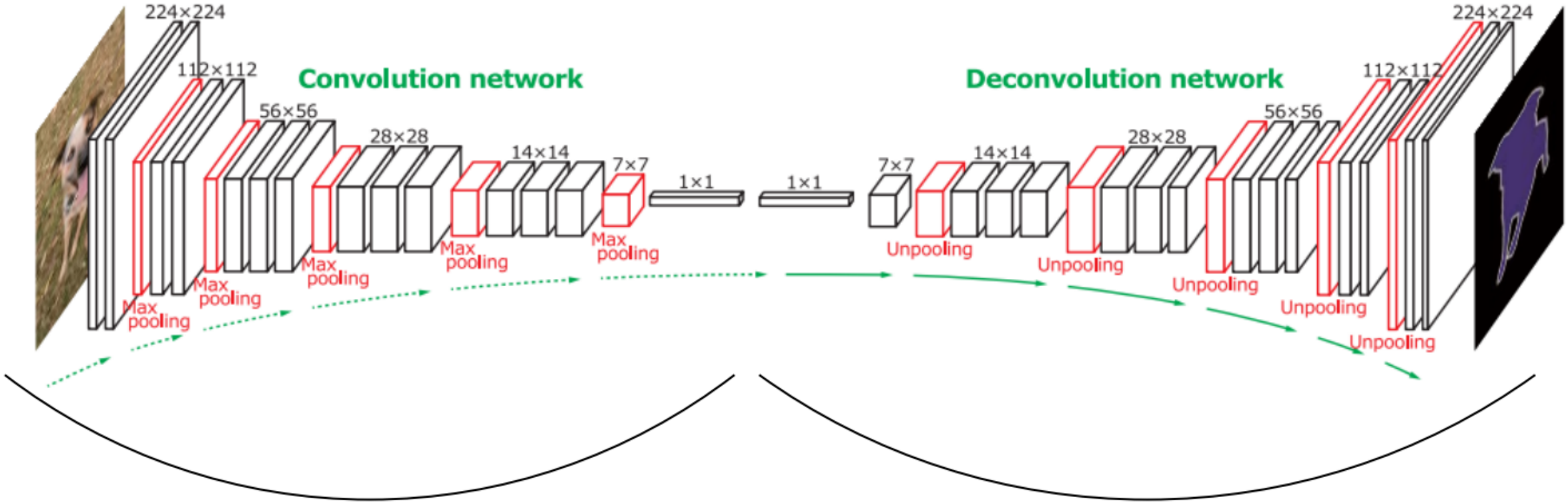
Slide credit to Gregory P. Spell

- Instance-wise segmentation: use edge-box[1] algorithm to generate object proposals from which to predict pixel classes. Aggregate all proposal outputs for an image via pixel-wise max (or average)
- Two-stage training: train on easy examples (cropped bounding boxes centered on a single object) first and then more difficult examples (proposals from edge-box)

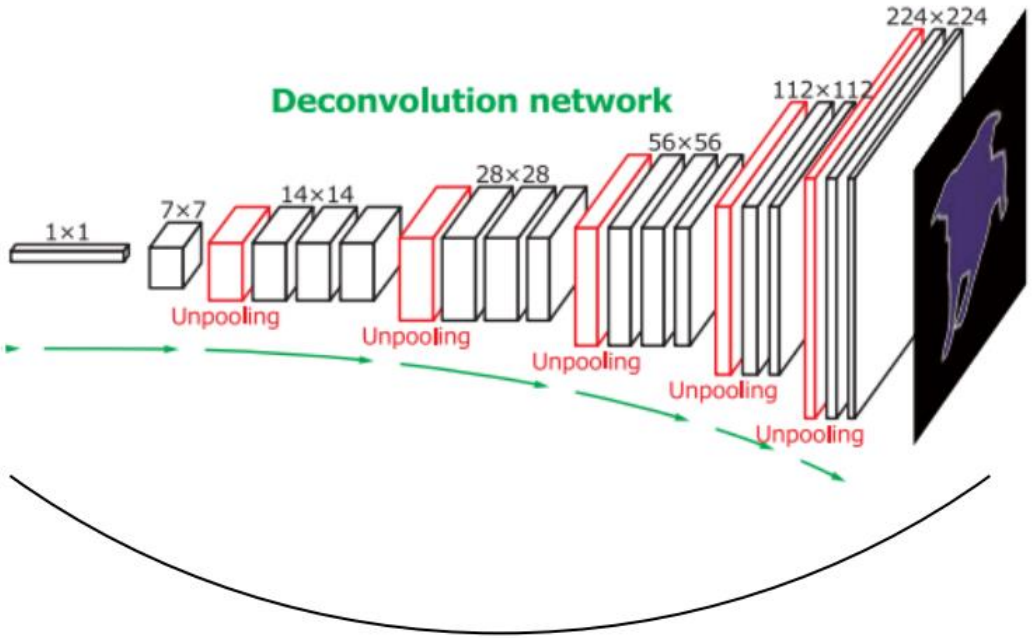


Slide credit to Gregory P. Spell

Deconvolution Network

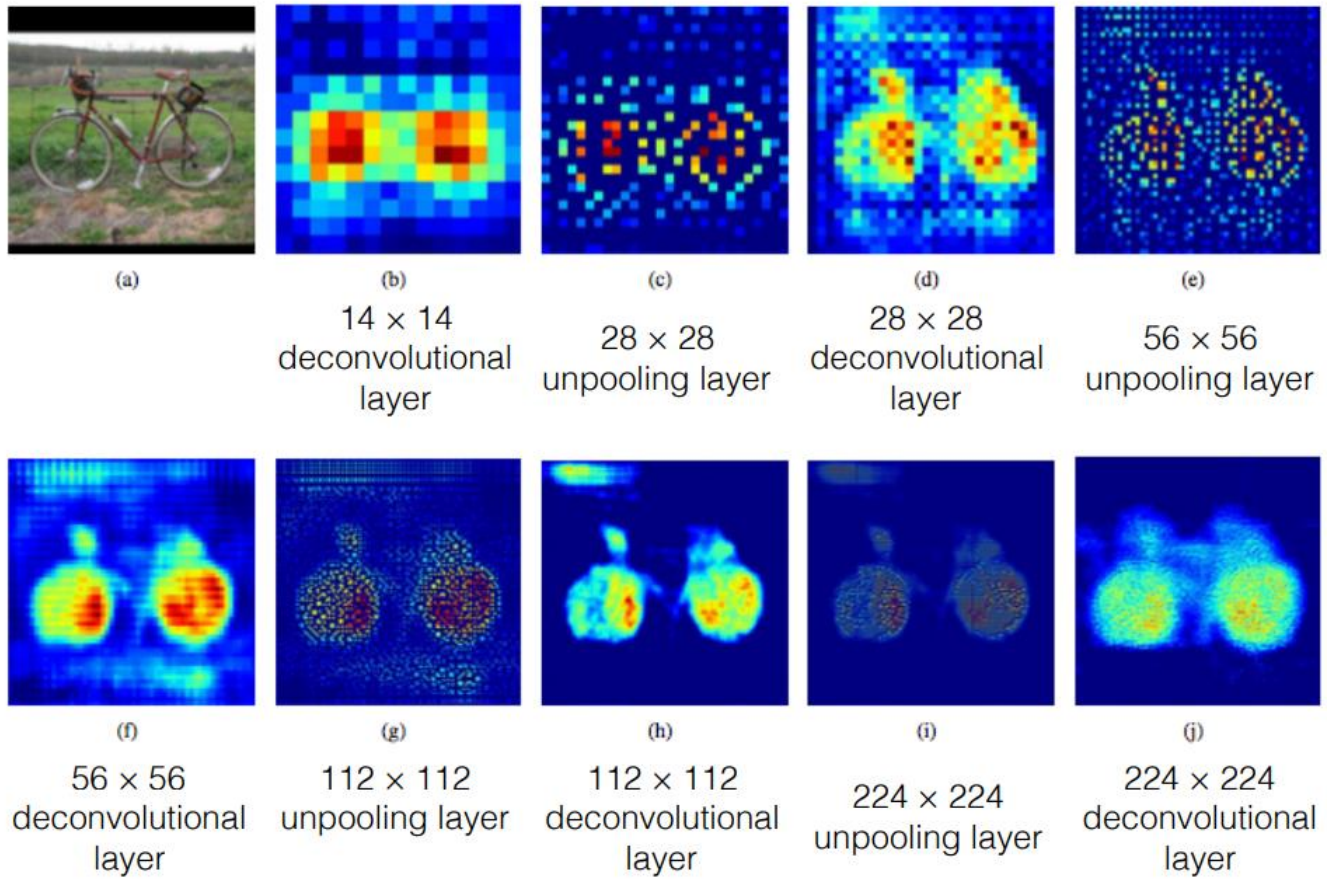


Deconvolution Network



Shape generator

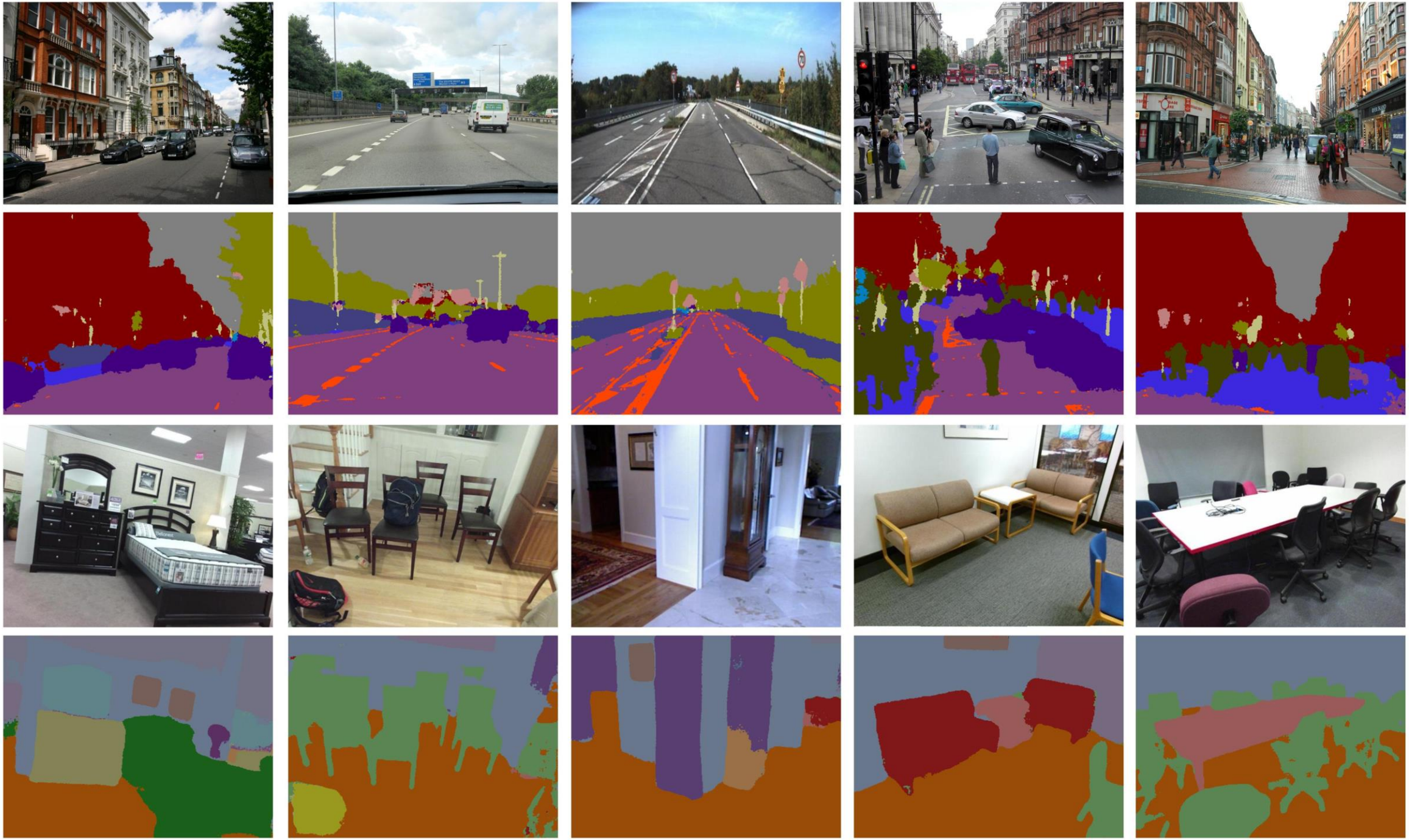
Activations of each layer



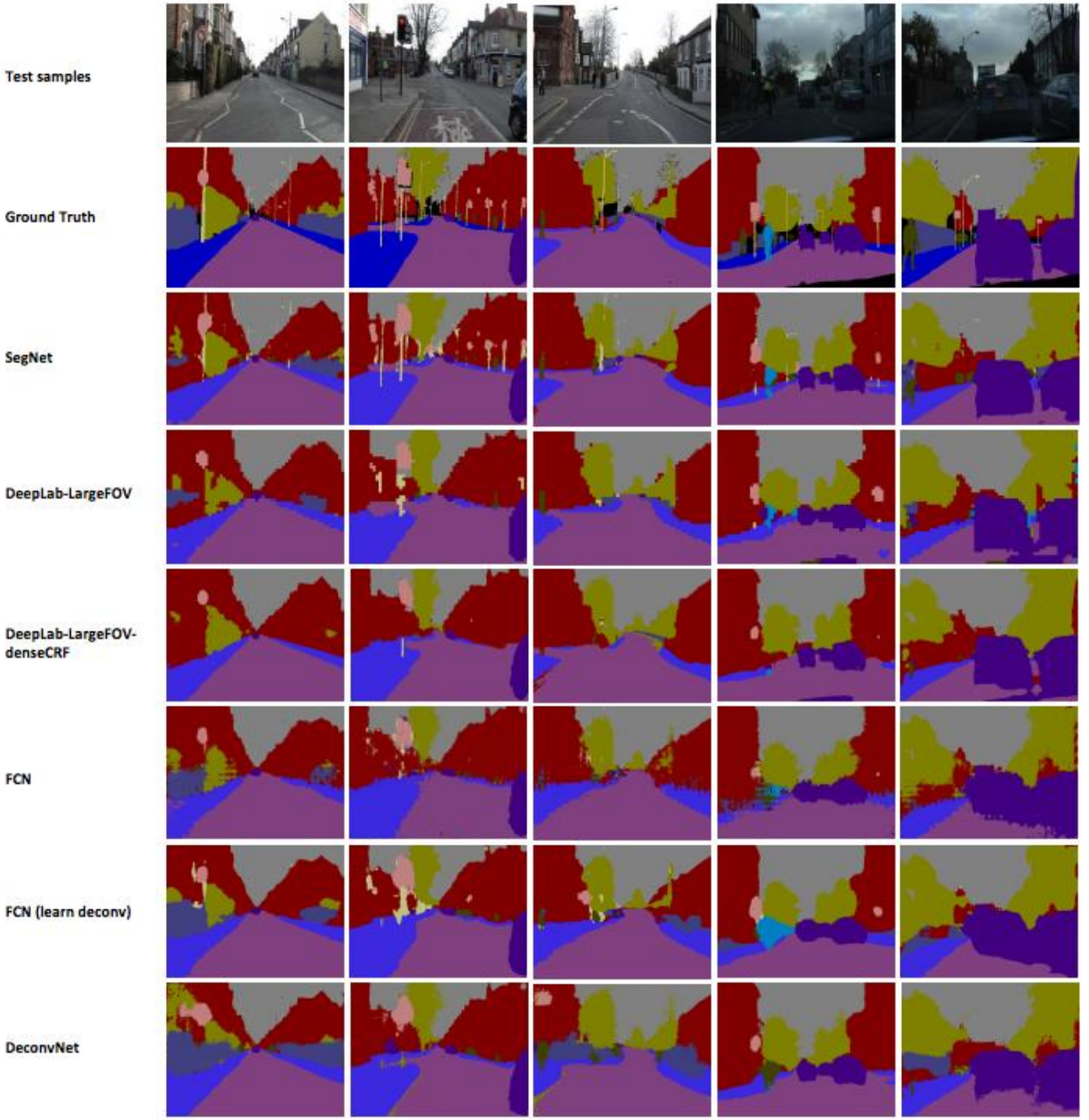
Advantages of SegNet:

1. Uses a novel technique to up-sample encoder output which involves **storing the max-pooling indices** used in pooling layer. This gives reasonably good performance and is space efficient

2. VGG16 with only forward connections and non trainable layers is used as encoder. This leads to very less parameters.



Comparison Results




STORE
STORE