COMPUTER VISION LAB.,
School of Artificial Intelligence,
Inha University

Fully Convolutional Networks, CVPR 2015

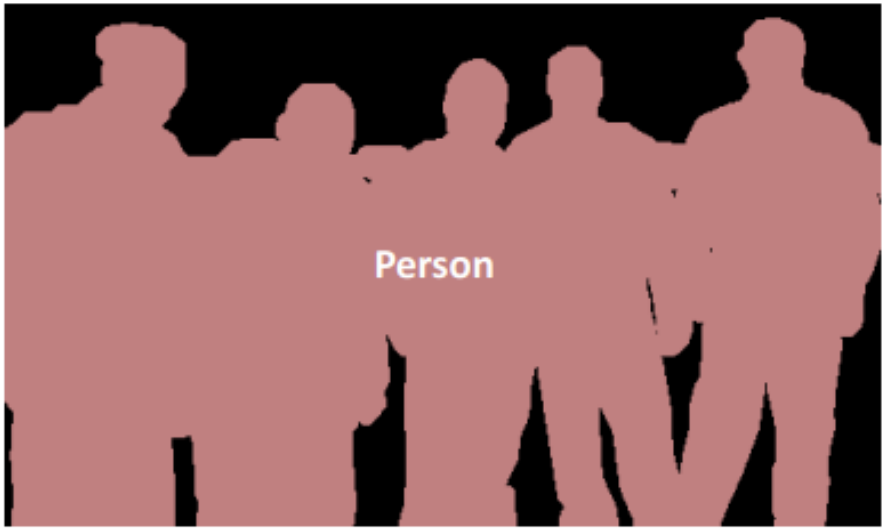
COMPUTER VISION LAB

22211226 JHNam

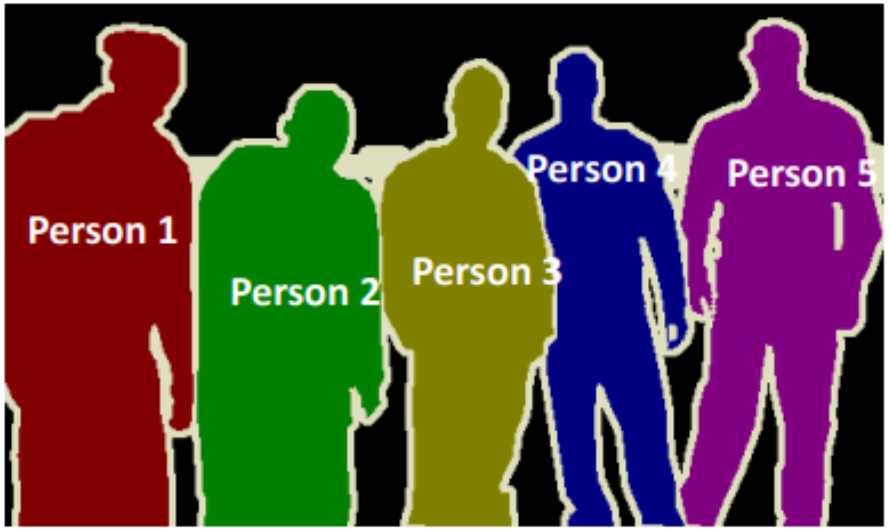
CONTENT

- Introduction
- Fully Convolutional Networks
- Segmentation Architecture
- Results

- **Semantic Segmentation**
  - Make a prediction at every each pixel.

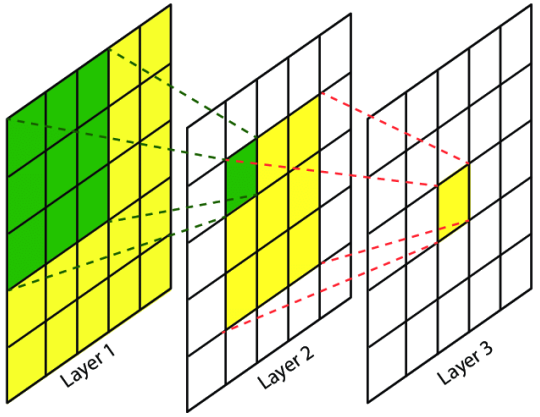


Semantic Segmentation



Instance Segmentation

Fully Convolutional Networks



- **Basic Concept**
  - Shape of each conv net : $h \times w \times d$
    - h, w : spatial resolution
    - d : channel dimension
  - *Receptive Fields* : Locations in higher layers correspond to the locations in the image they are path-connected to.
  - *Translation Invariant* : Max Pooling Layer
    - $\mathbf{x}_{i,j}$: data vector at (i, j) in particular layer

$$\mathbf{y}_{i,j} = f_{ks}(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \le \delta i, \delta j \le k})$$

      - k : kernel size
      - s : stride
      - f_{ks} : layer type

$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s', ss'}$$

  - Fully Convolutional Network : A net with only layers of this form.
    - FCN1 : operates on an input of any size
    - FCN2 : produces an output of corresponding spatial dimensions.

- **Basic Concept**
  - Loss function $l(\mathbf{x};\theta) = \sum_{ij} l'(\mathbf{x}_{ij};\theta)$
    - SGD with $l(\mathbf{x};\theta)$ on whole image = SGD with $l'(\mathbf{x}_{ij};\theta)$ taking all of the final layer receptive fields

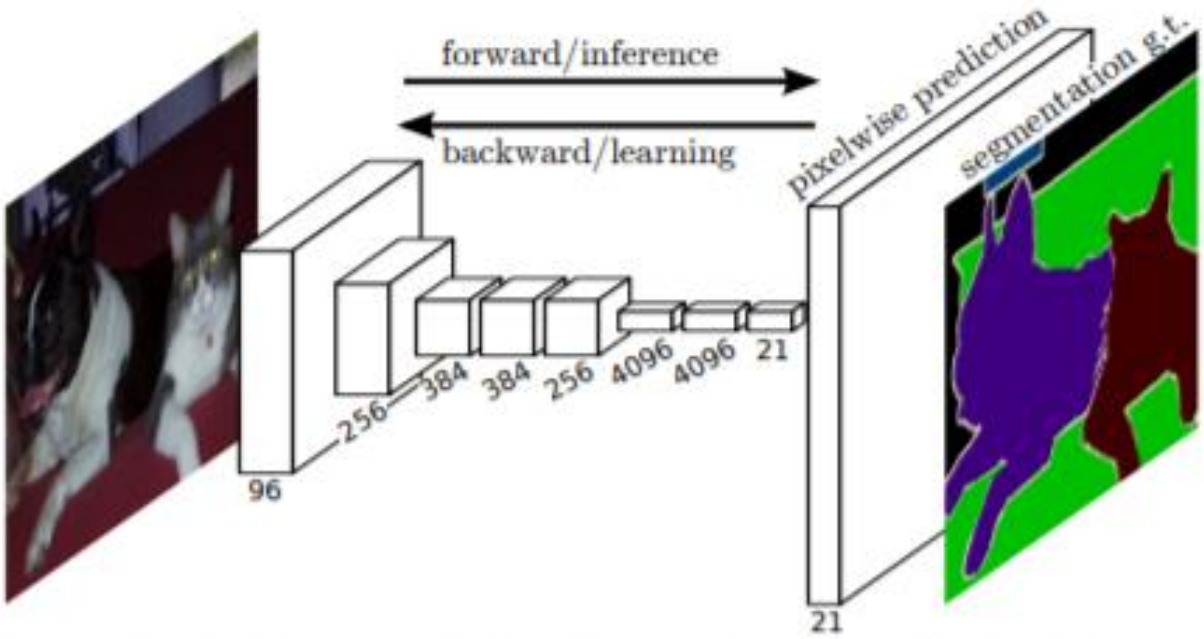


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

- **Adapting Classifiers for Dense Prediction**
  - Original Segmentation Network : Fixed-size input & Produce Non-spatial output
  - FCL → Convolution with kernels that cover their entire input regions
  - Convolutionalization

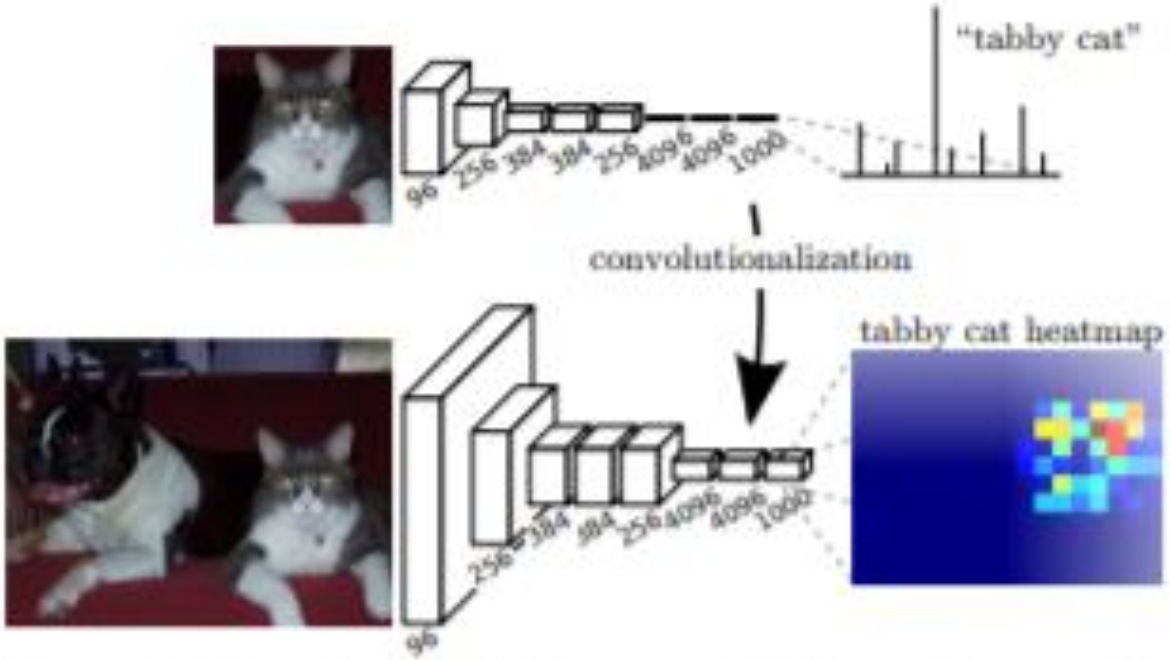
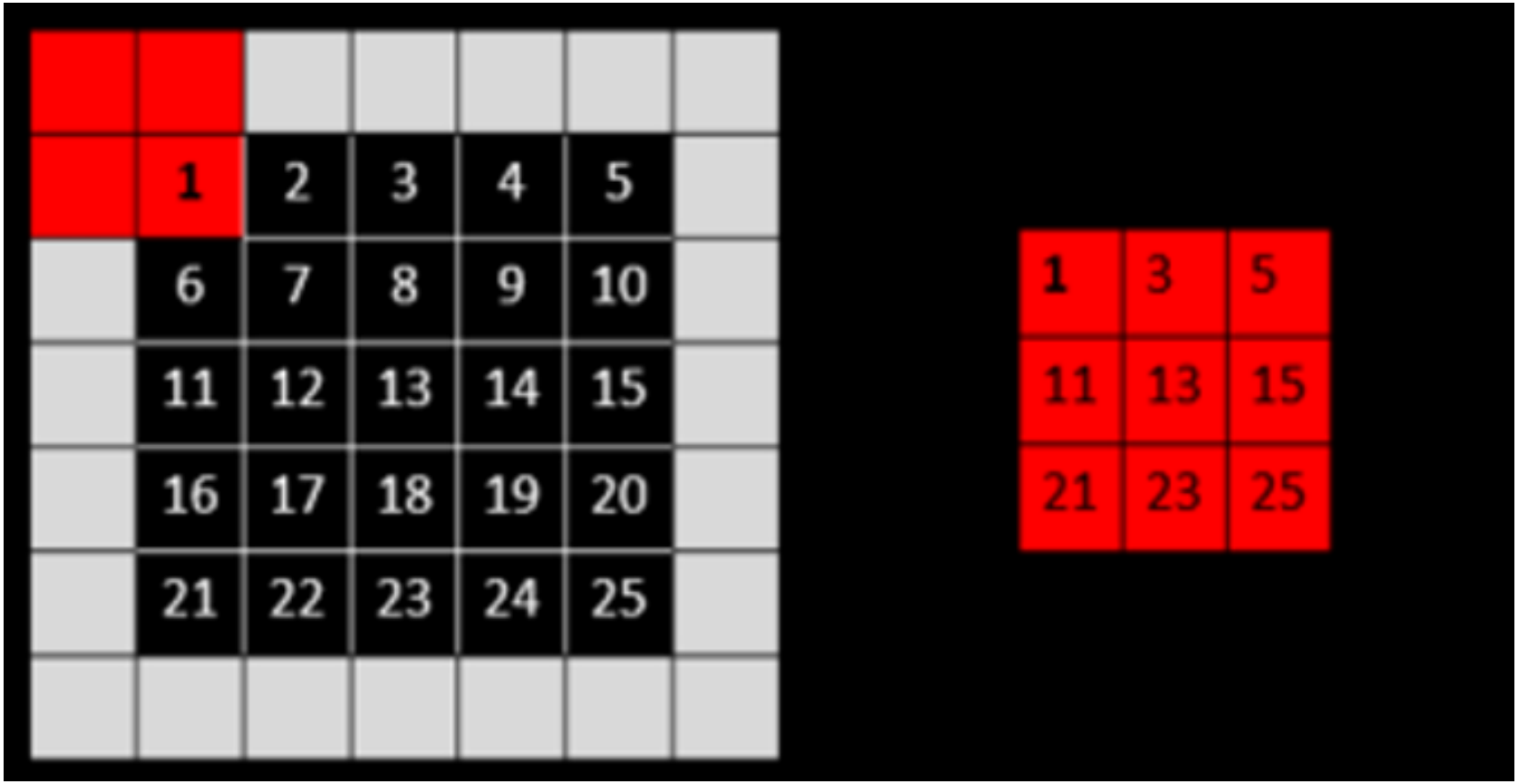


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

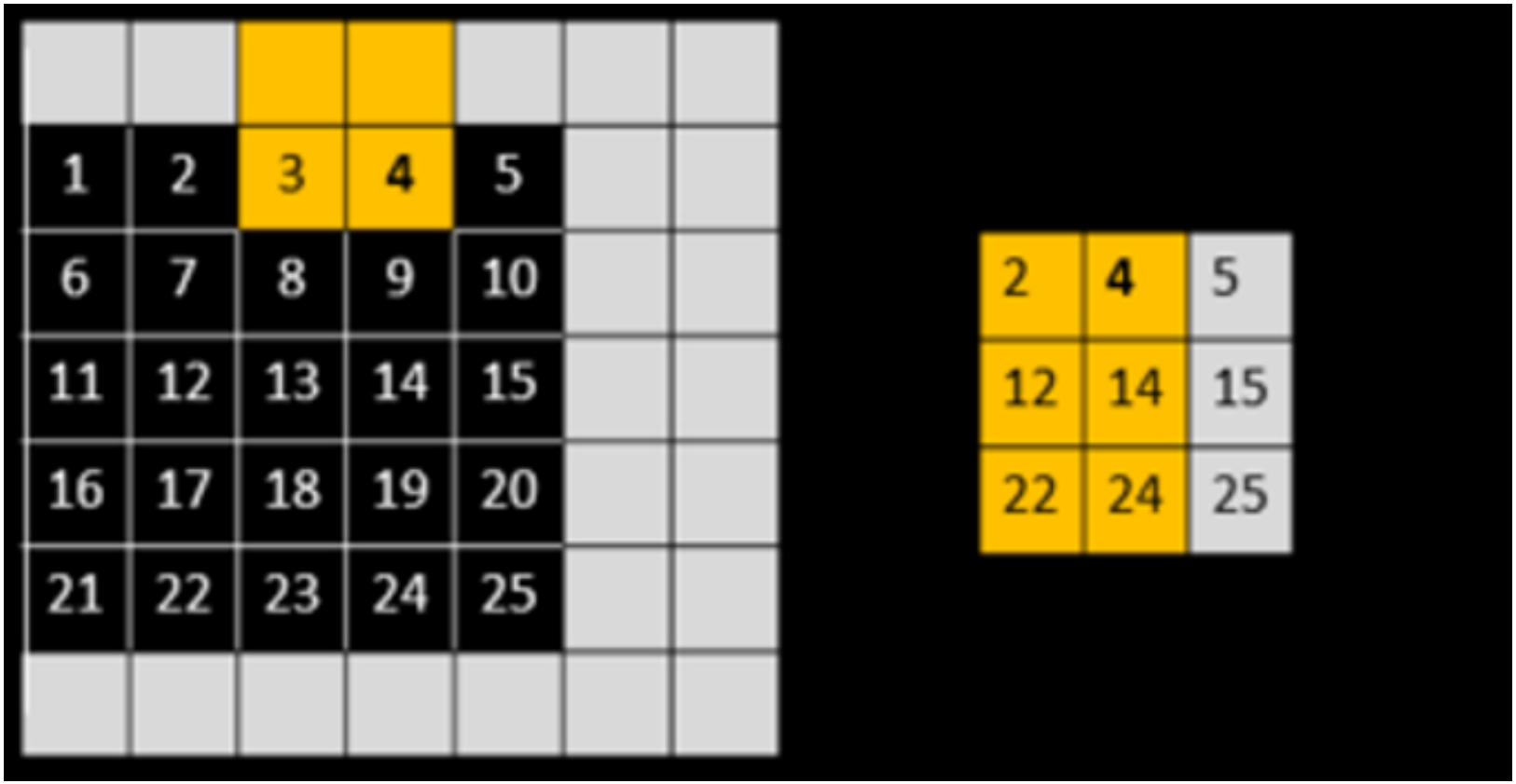
COMPUTER VISION LAB.,
School of Artificial Intelligence,
Inha University

- **Shift-and-Stich is Filter Rarefaction**
  - *Input Shifting & Output Stitching* : Coarse input → Dense pixel Output
  - Example : Consider 2×2 Maxpool with Stride 2
    - Shifting

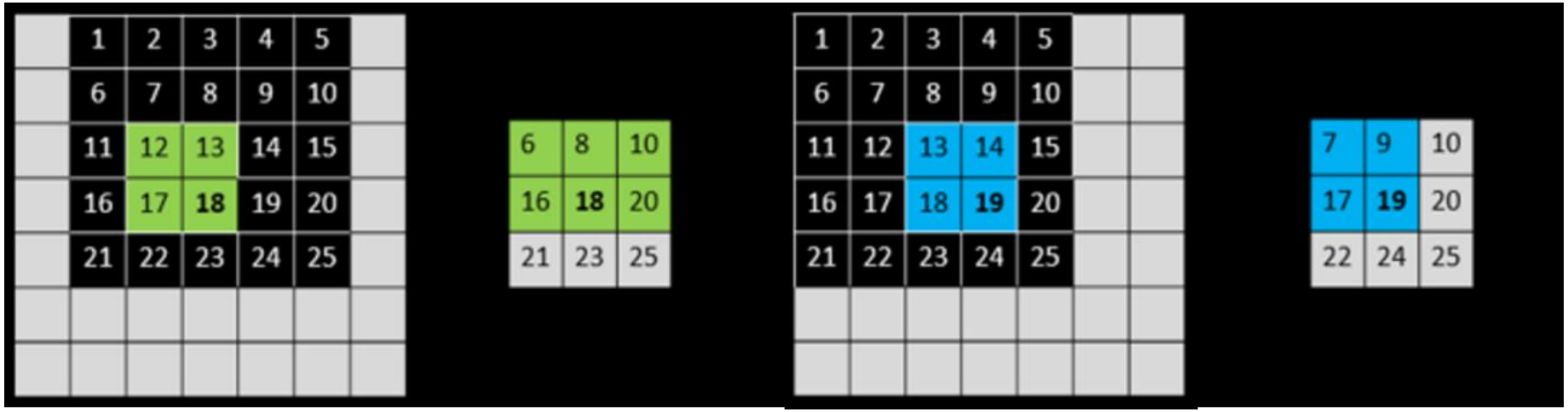


COMPUTER VISION LAB.,
School of Artificial Intelligence,
Inha University

- **Shift-and-Stich is Filter Rarefaction**
  - *Input Shifting & Output Stitching* : Coarse input → Dense pixel Output
  - Example : Consider 2×2 Maxpool with Stride 2
    - Shifting



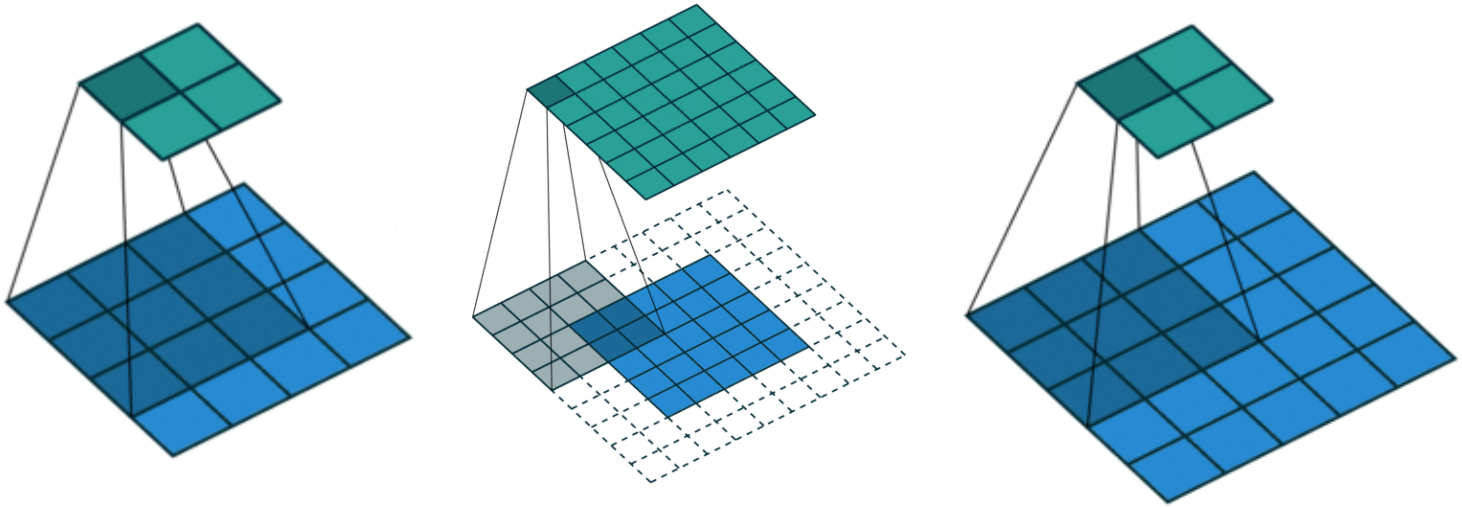
- **Shift-and-Stich is Filter Rarefaction**
  - *Input Shifting & Output Stitching* : Coarse input → Dense pixel Output
  - Example : Consider 2×2 Maxpool with Stride 2
    - Shifting



- **Shift-and-Stich is Filter Rarefaction**
  - *Input Shifting & Output Stitching* : Coarse input → Dense pixel Output
  - Example : Consider 2×2 Maxpool with Stride 2
    - Stiching

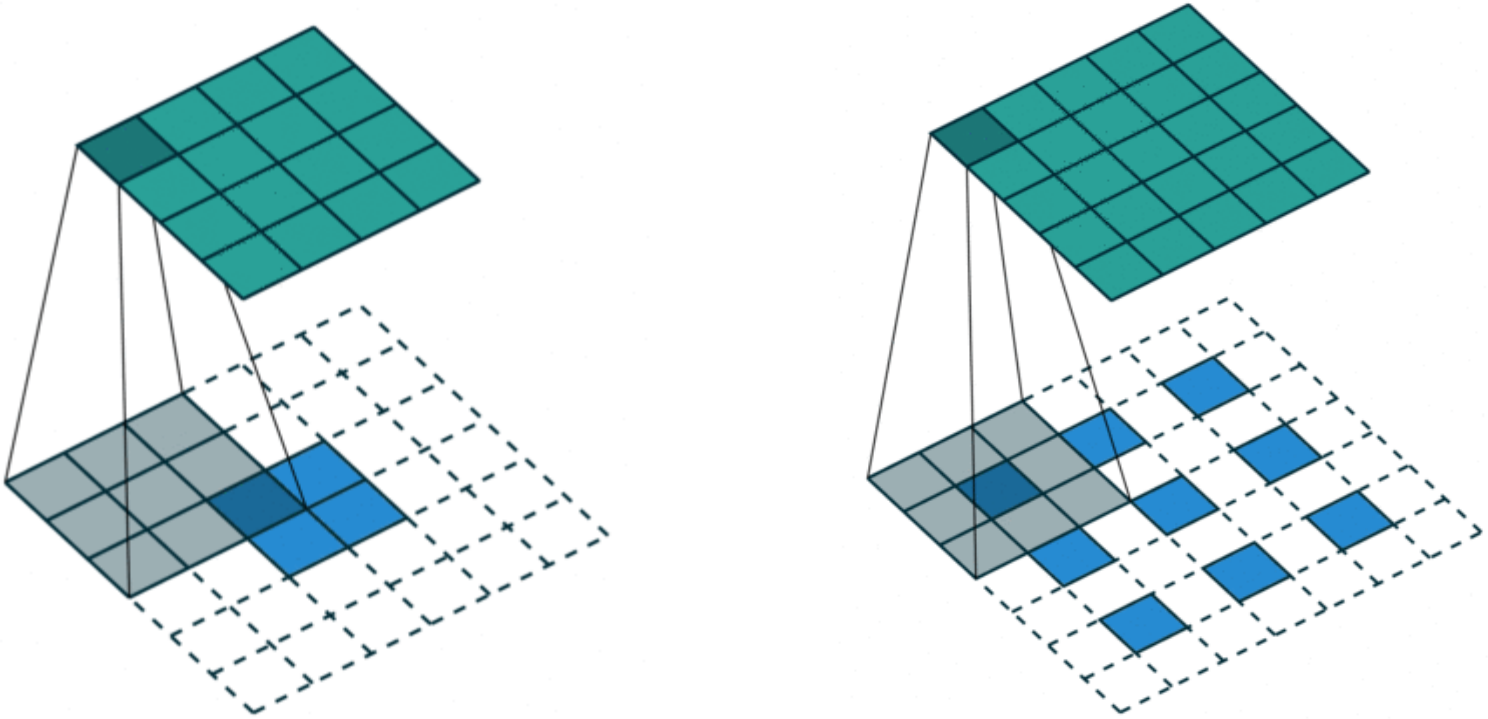
- **Upsampling is Backwards Strided Convolution**
  - How to upsample feature map? Interpolation
    - Nearest Interpolation, Bilinear Interpolation, Bicubic Interpolation
  - Can the model learn to upsample while preserving the amount of information as much as possible? Backward Convolution, Deconvolution, Transposed Convolution
    - Learning-based upsampling method
    - There are many convolution layer
      - Traditional Convolution, Transposed Convolution, Atrous Convolution, …

- **Upsampling is Backwards Strided Convolution**
  - Various Convolution Layer
    - Traditional Convolution

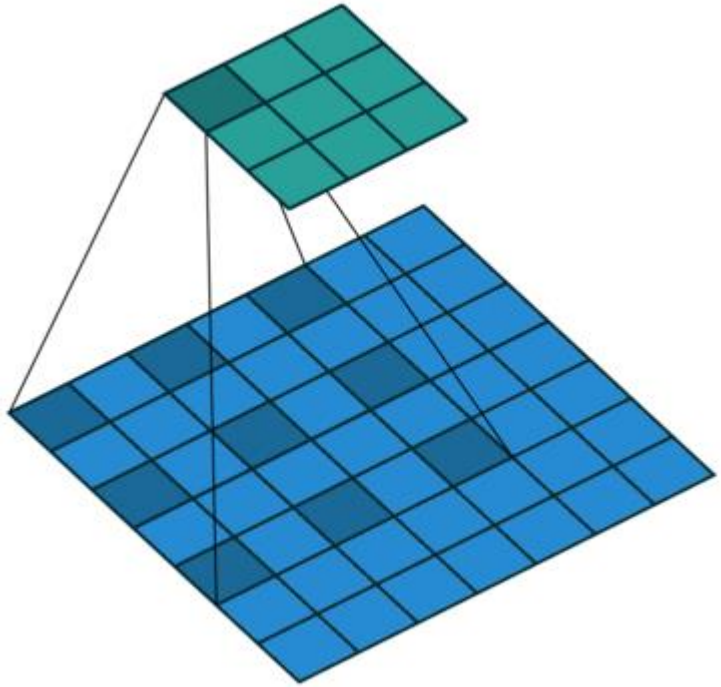


COMPUTER VISION LAB.,
School of Artificial Intelligence,
Inha University

- **Upsampling is Backwards Strided Convolution**
  - Various Convolution Layer
    - Transposed Convolution



- **Upsampling is Backwards Strided Convolution**
  - Various Convolution Layer
    - Atrous Convolution



- **Patchwise Training is Loss Sampling**
  - "*Whole image fully convolutional training is identical to patchwise training where each batch consists of all the receptive fields of the units below the loss for an image*."
  - Sampling in patchwise training
    - 1. Class imbalance ↓
    - 2. Spatial correlation of dense patches ↓

- **From Classifier to Dense FCN**
  - Apply *Convolutionalization* into AlexNet, VGG, GoogLeNet.
  - 1×1 Conv with channel dimension "21" → #PASCAL VOC class
  - Fine Tuning(Classification → Segmentation)
  - Performance : ~75%
  - SOTA : FCN-VGG16 → mIoU : 56.0%

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

- **Combining What and Where**

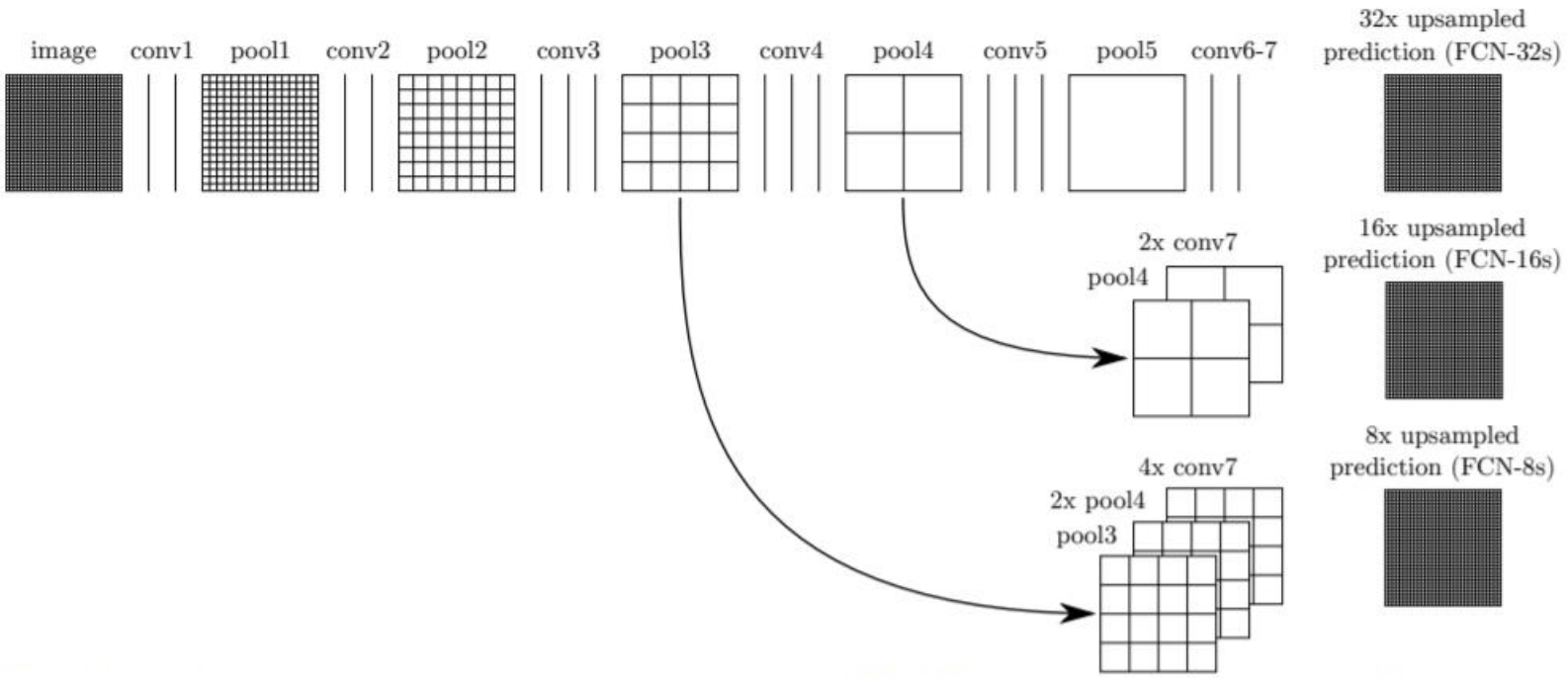


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (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. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

- **Combining What and Where**

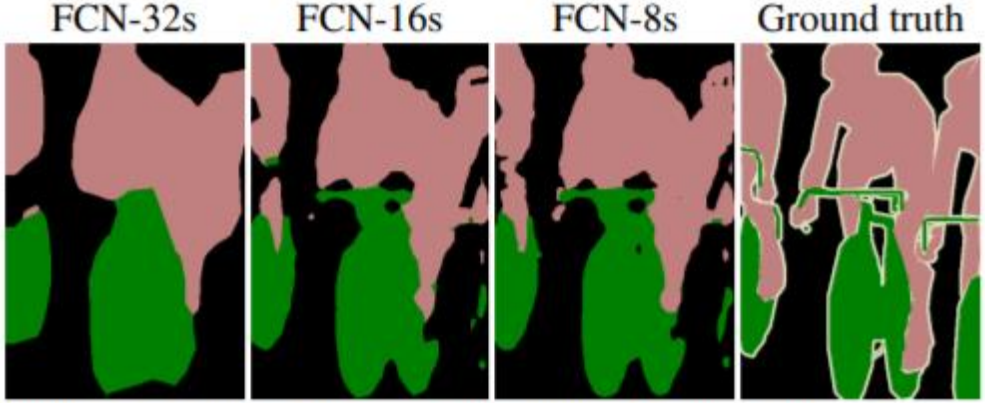


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

- **Experimental Framework**
  - Optimization : SGD with momentum 0.9
    - Batch size : 20
    - Fixed Learning rate : $10^{-3} \rightarrow 10^{-4} \rightarrow 5^{-5}$
    - Weight decay : 5^{-4} or 2^{-4}
  - Fine-tuning
  - Patch-Sampling
  - Class Balancing
  - Dense Prediction
  - Augmentation

- **Metrics**
  - pixel accuracy
  - mean accuracy
  - mean IoU
  - frequency weighted IoU
- **Dataset**
  - PASCAL VOC
  - NYUDv2
  - SIFT Flow

Table 3. Our fully convolutional net gives a 20% relative improvement over the state-of-the-art on the PASCAL VOC 2011 and 2012 test sets and reduces inference time.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [10]	47.9	-	-
SDS [15]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

Table 4. Results on NYUDv2. *RGBD* is early-fusion of the RGB and depth channels at the input. *HHA* is the depth embedding of [13] as horizontal disparity, height above ground, and the angle of the local surface normal with the inferred gravity direction. *RGB-HHA* is the jointly trained late fusion model that sums RGB and HHA predictions.

	pixel acc.	mean acc.	mean IU	f.w. IU
Gupta *et al.* [13]	60.3	-	28.6	47.0
FCN-32s RGB	60.0	42.2	29.2	43.9
FCN-32s RGBD	61.5	42.4	30.5	45.5
FCN-32s HHA	57.1	35.2	24.2	40.4
FCN-32s RGB-HHA	64.3	44.9	32.8	48.0
FCN-16s RGB-HHA	**65.4**	**46.1**	**34.0**	**49.5**

Table 5. Results on SIFT Flow[9] with class segmentation (center) and geometric segmentation (right). Tighe [33] is a non-parametric transfer method. Tighe 1 is an exemplar SVM while 2 is SVM + MRF. Farabet is a multi-scale convnet trained on class-balanced samples (1) or natural frequency samples (2). Pinheiro is a multi-scale, recurrent convnet, denoted RCNN$_3$ (o^3). The metric for geometry is pixel accuracy.

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al.* [23]	76.7	-	-	-	-
Tighe *et al.* [33]	-	-	-	-	90.8
Tighe *et al.* [34] 1	75.6	41.1	-	-	-
Tighe *et al.* [34] 2	78.6	39.2	-	-	-
Farabet *et al.* [7] 1	72.3	50.8	-	-	-
Farabet *et al.* [7] 2	78.5	29.6	-	-	-
Pinheiro *et al.* [28]	77.7	29.8	-	-	-
FCN-16s	**85.2**	**51.7**	39.5	76.1	**94.3**

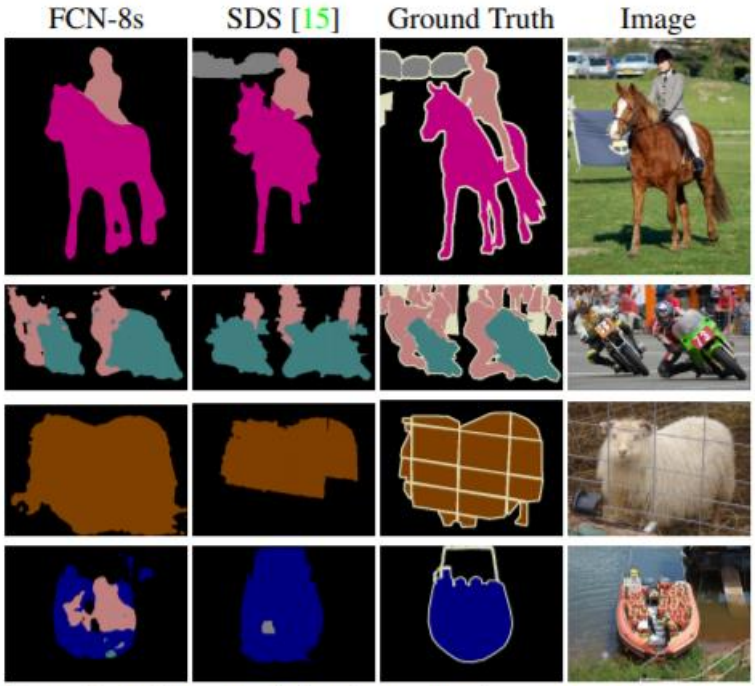


Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system by Hariharan *et al.* [15]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fourth row shows a failure case: the net sees lifejackets in a boat as people.