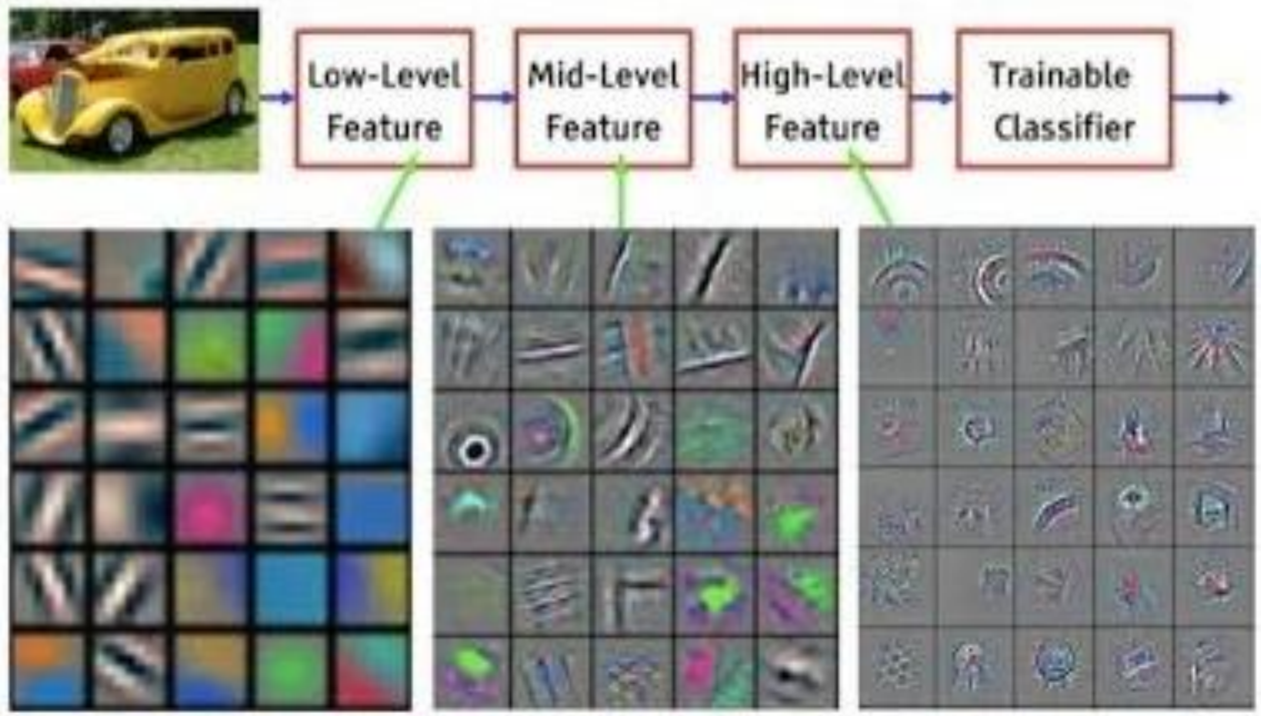
CMPT 743 Practices for Visual Computing

Ali Mahdavi Amiri

Convolutional Neural Network

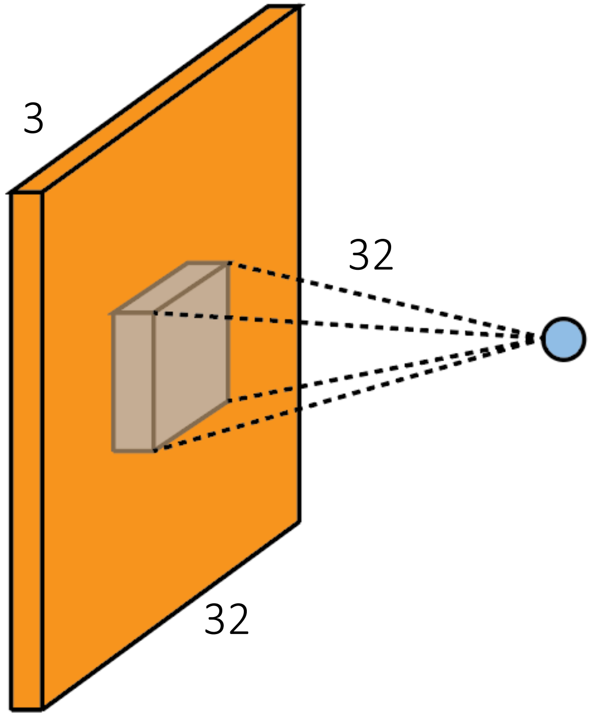
- Provide unique hierarchical features of the image



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

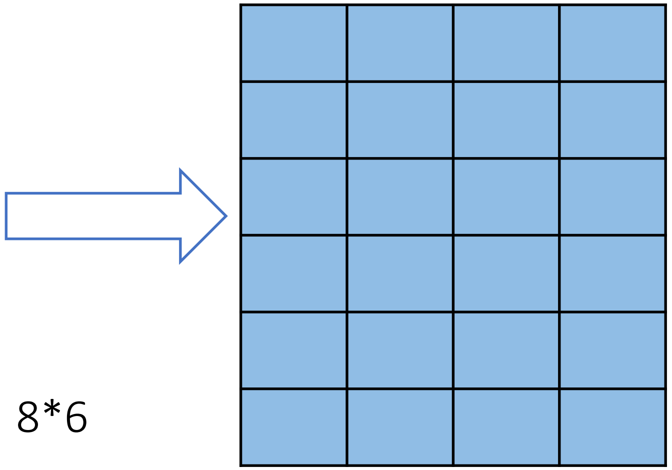
Convolution Layers

- Keep sliding the filter produces a new images (feature) with a new dimension



2	2	3	4	5	6
2	2	1	2	2	7
1	2	4	6	2	1
1	2	54	54	5	4
5	3	2	8	9	0
1	2	4	6	8	0
43	21 3	1	54	54	8
32	2	21	34	32	1

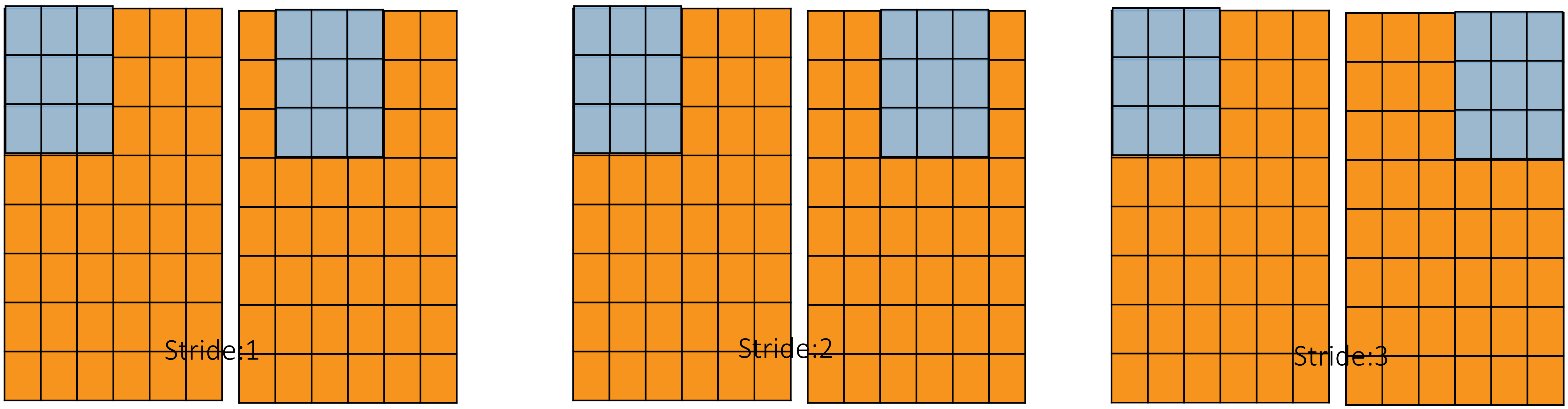
8*6



6*4

Stride

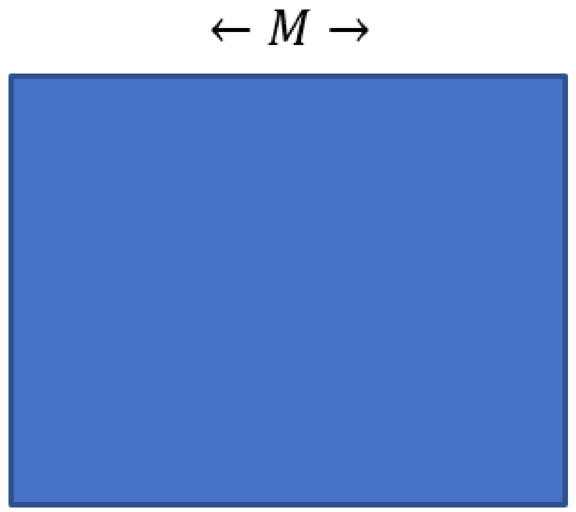
- Jump between pixels



Padding

- To bring boundary pixels to the game and lose less information, we can apply padding

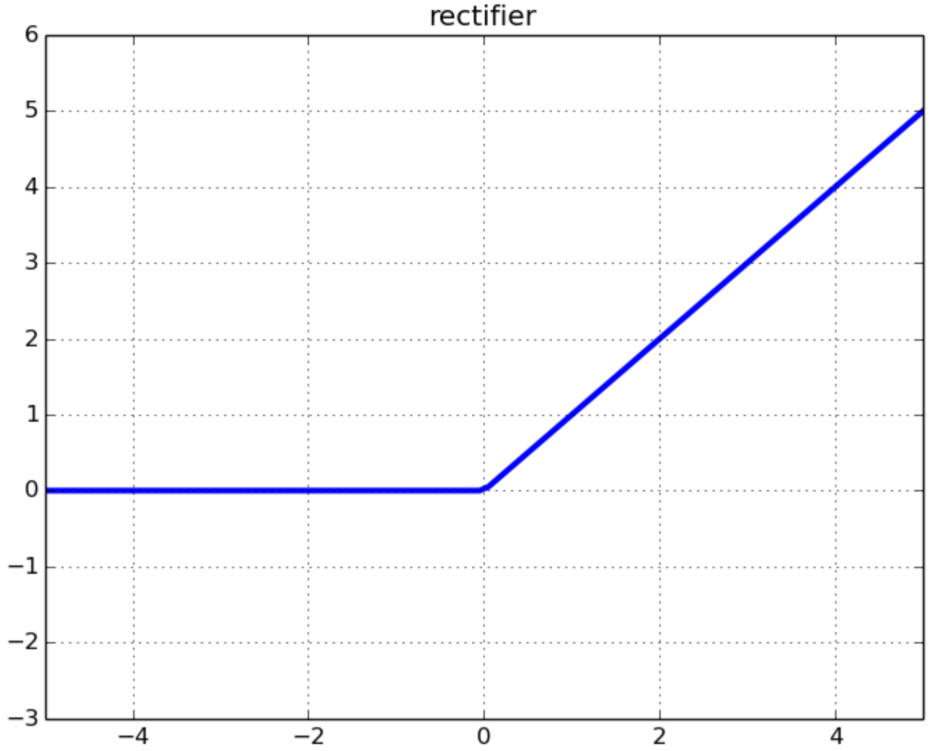
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	3	4	5	6	0	0
0	0	2	2	1	2	2	7	0	0
0	0	1	2	4	6	2	1	0	0
0	0	1	2	54	54	5	4	0	0
0	0	5	3	2	8	9	0	0	0
0	0	1	2	4	6	8	0	0	0
0	0	43	213	1	54	54	8	0	0
0	0	32	2	21	34	32	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



$$M = \frac{N - F + 2P}{S} + 1$$

Activation Function

- Relu is the most common activation function for CNN



$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

Max Pooling

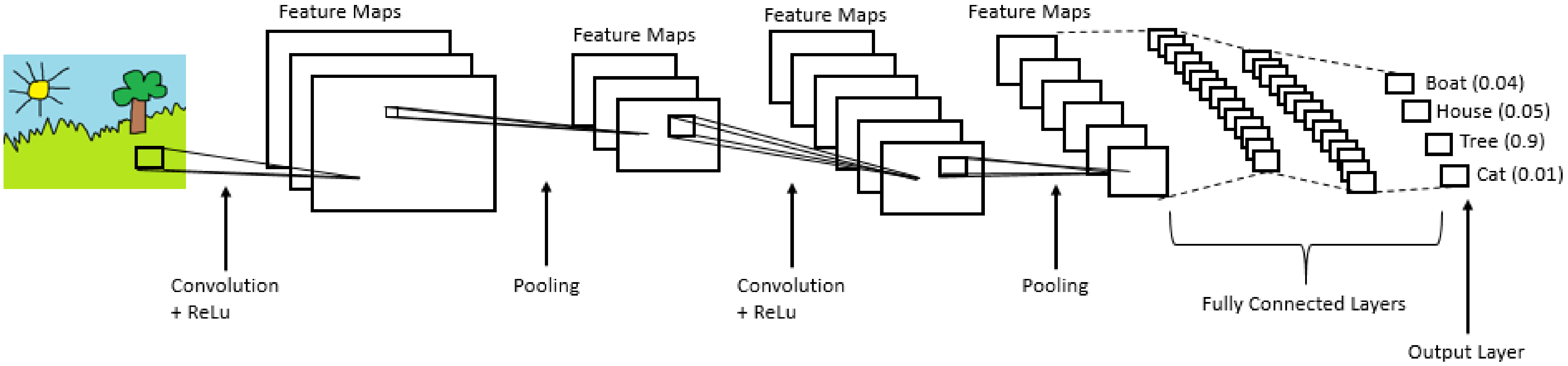
- Pooling is very similar to convolution. Instead of dot product, we find the maximum of elements (no parameter to learn).

7	4	6	2
3	2	3	5
1	2	1	0
5	4	2	9

7	6
5	9

Image Classification

- Overall Structure



AlexNet

- Designed by Alex Krizhevsky, Ilya Sustkever and Geoffrey Hinton

AlexNet

- Designed by Alex Krizhevsky, Ilya Sustkever and Geoffrey Hinton

- Rank 1 in 2012 ImageNet Large Scale Visual Recognition Challenge

ImageNet

- 14 Million images that are hand annotated in more that 20K categories.

Image Classification

- AlexNet

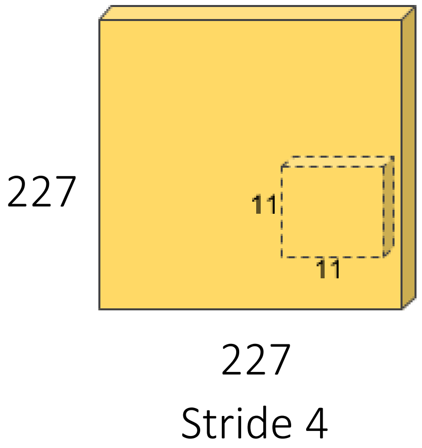


Image Classification

- AlexNet

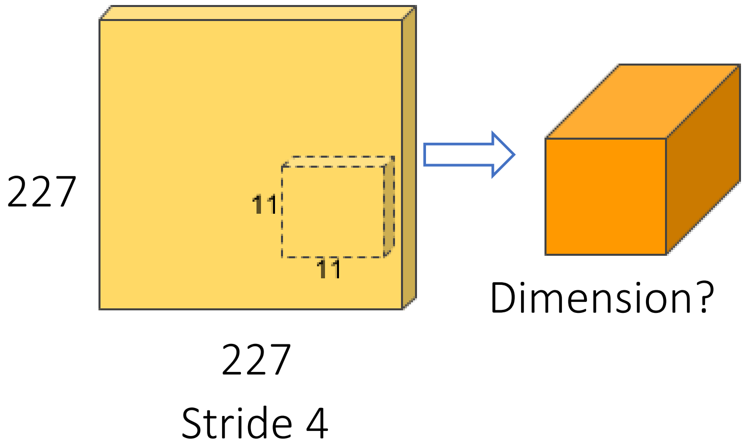
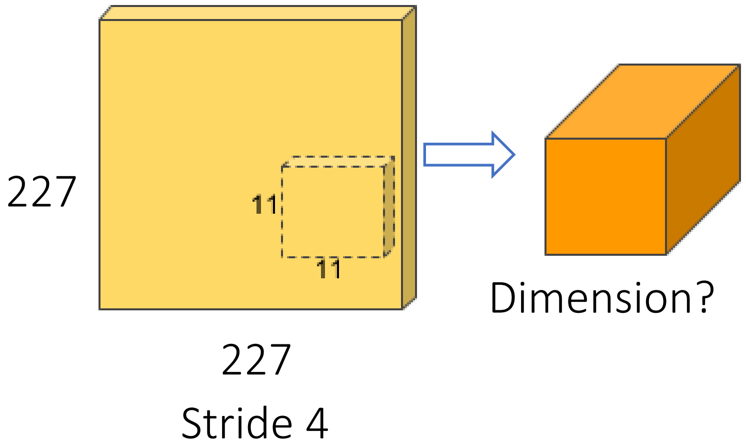


Image Classification

- AlexNet



$$(N - F)/s + 1 = (227 - 11)/4 + 1 = 55$$

Image Classification

- AlexNet

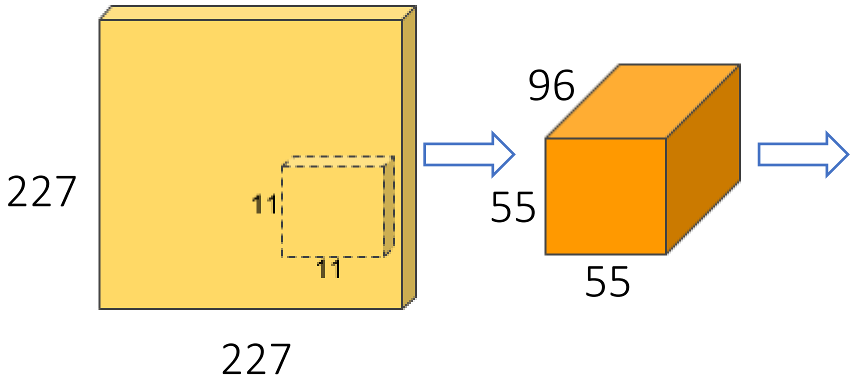


Image Classification

- AlexNet

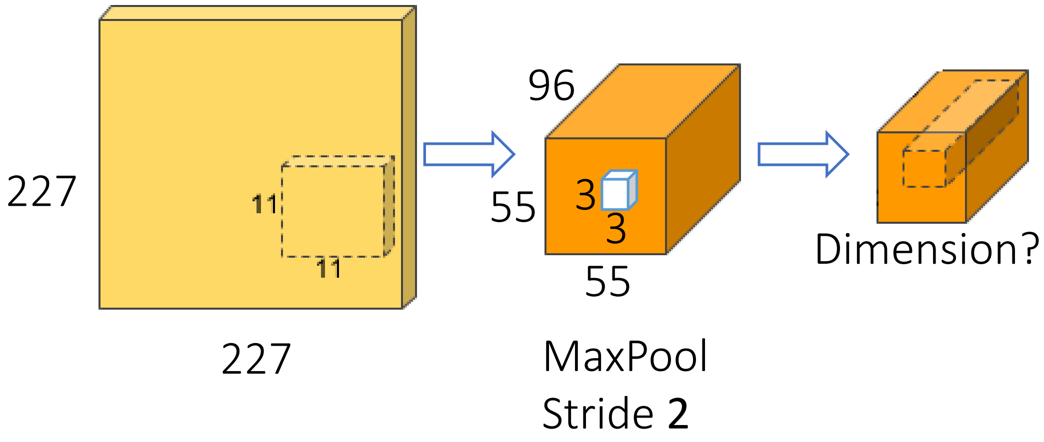
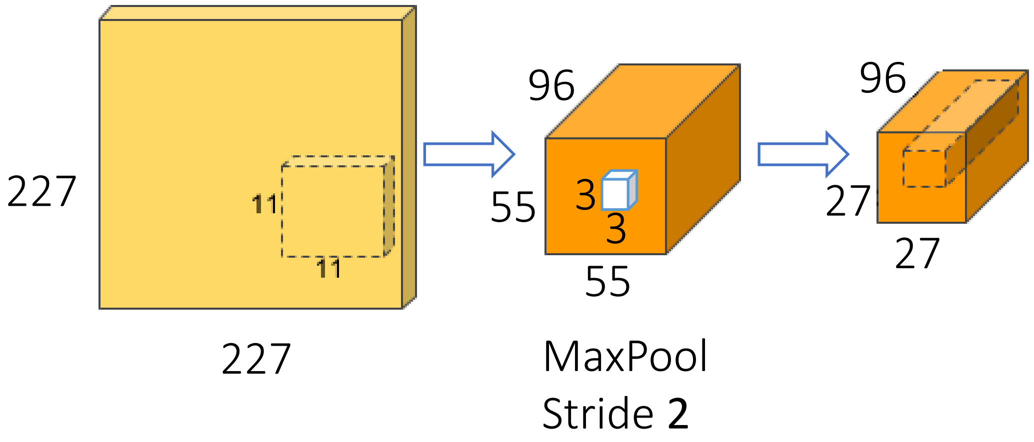


Image Classification

- AlexNet



$$(N - F)/s + 1 = (55 - 3)/2 + 1 = 27$$

Image Classification

- AlexNet

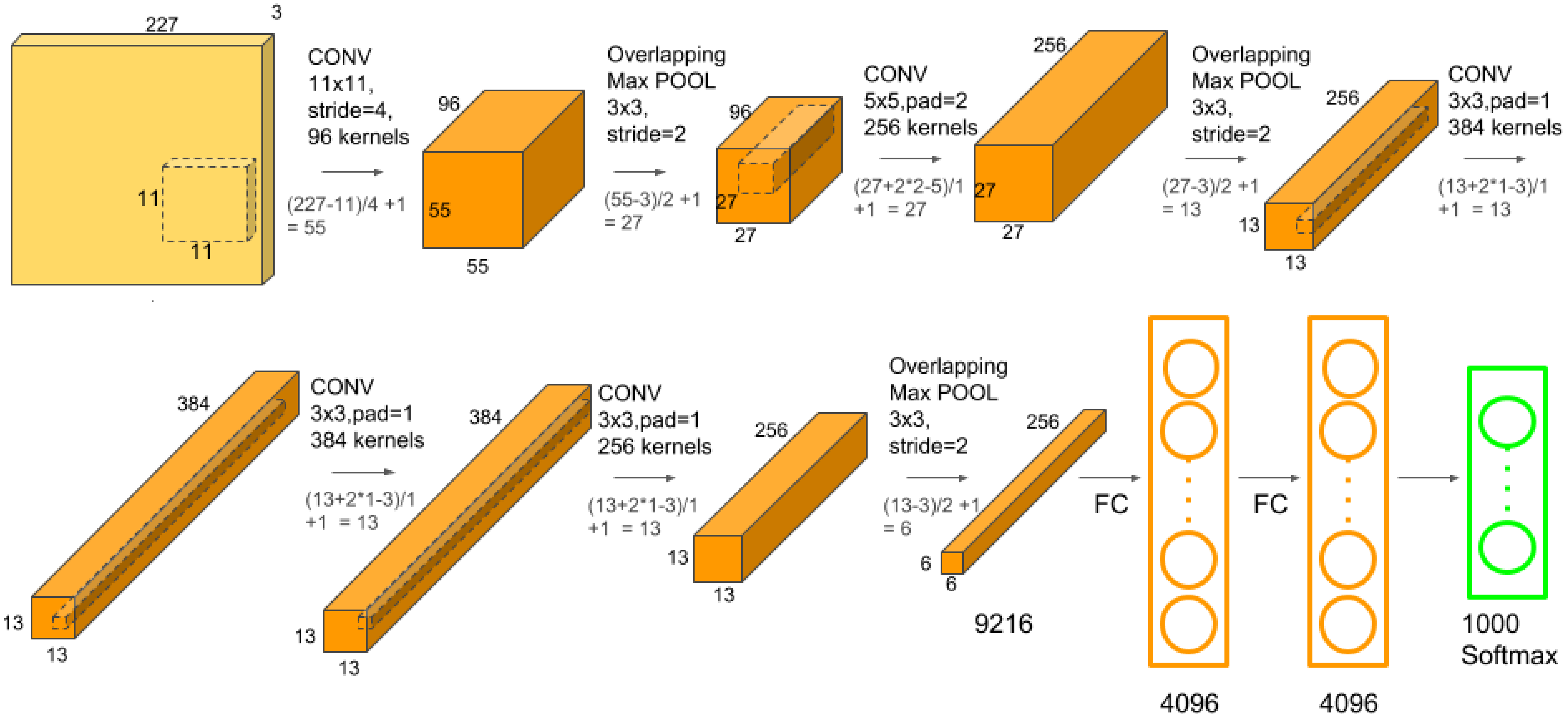
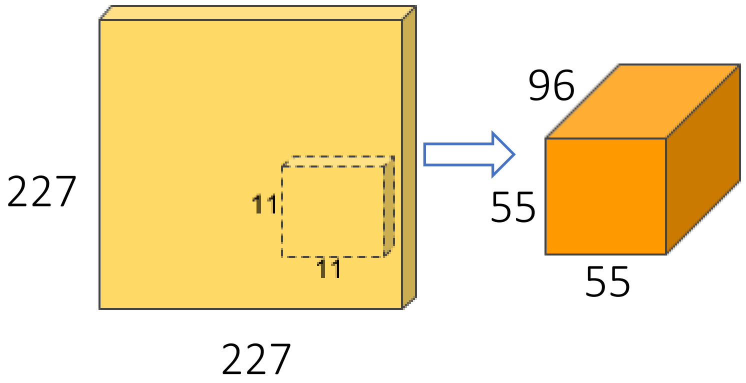


Image Classification

- In CNN therefore we have usually three main components
  - Convolutional layer
  - MaxPooling
  - Fully Connected Layer

Number of Parameters

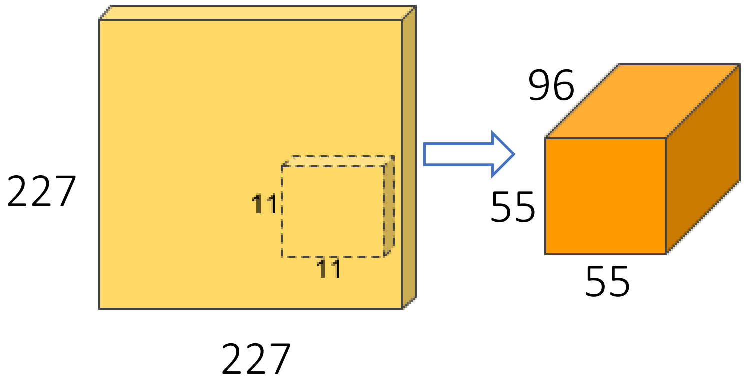
- Convolution layer
  - Example: first layer of AlexNet

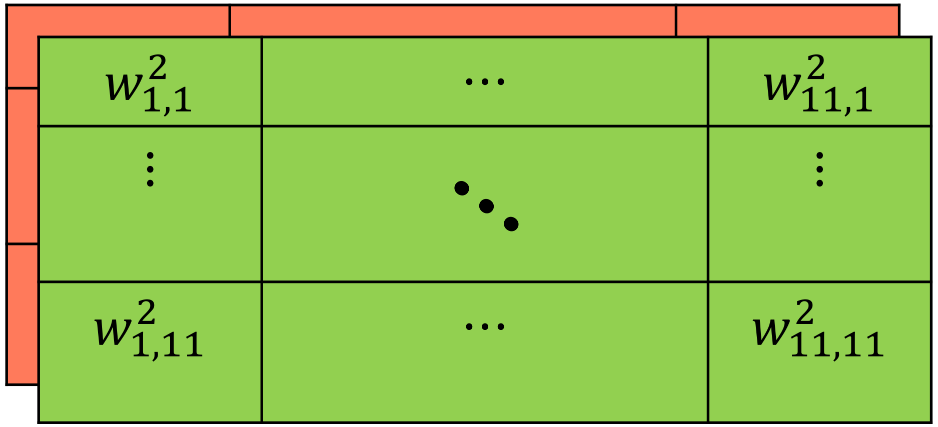


$w^1_{1,1}$	$\cdots$	$w^1_{11,1}$
$\vdots$	$\ddots$	$\vdots$
$w^1_{1,11}$	$\cdots$	$w^1_{11,11}$

Number of Parameters

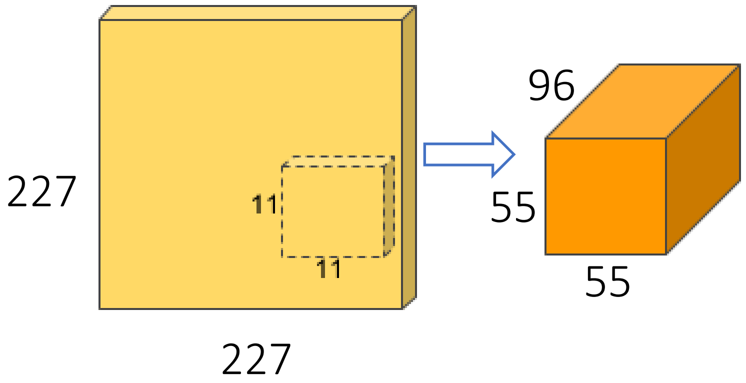
- Convolution layer
  - Example: first layer of AlexNet

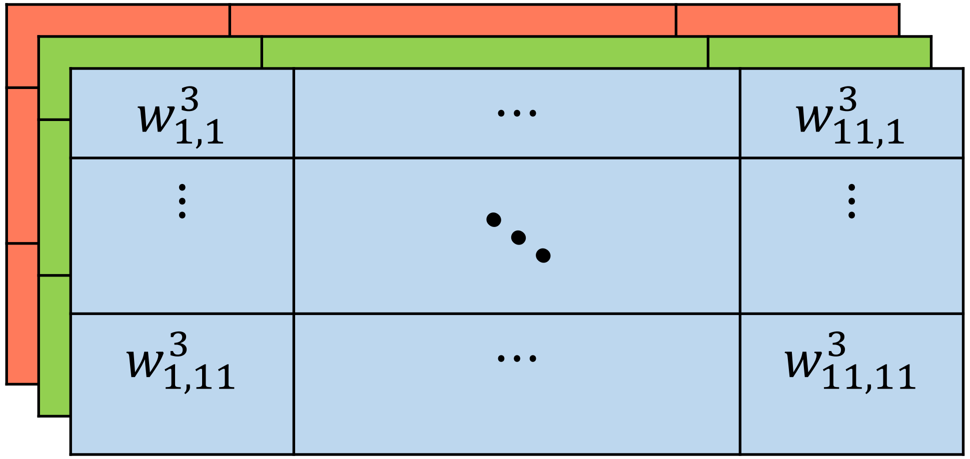




Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet

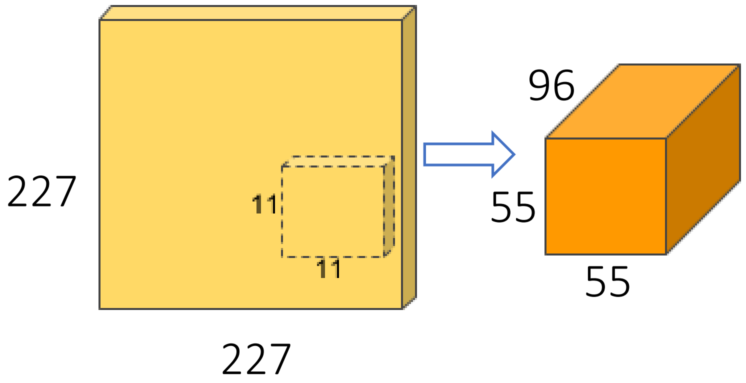


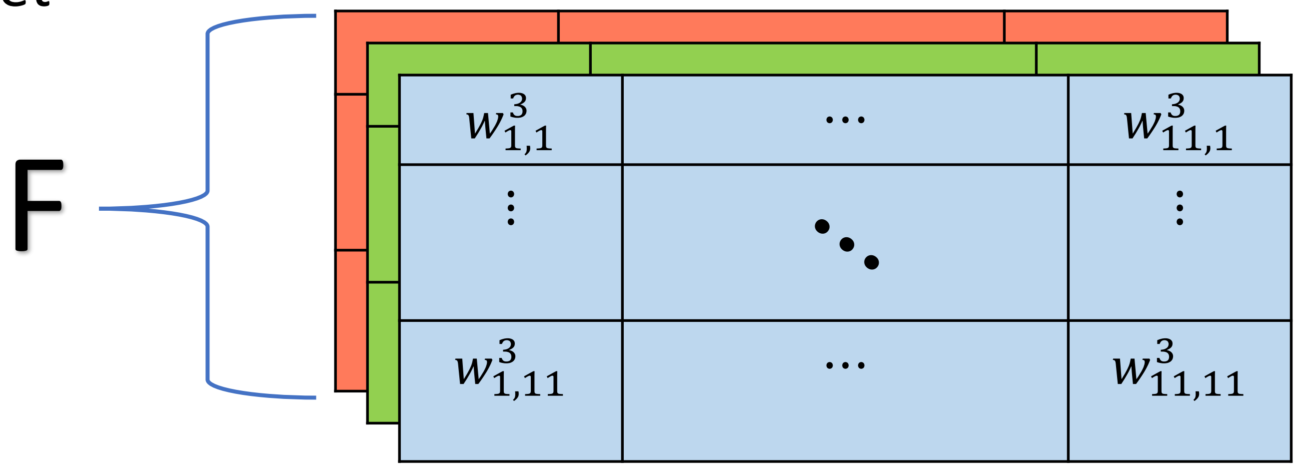


How many parameters?

Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet



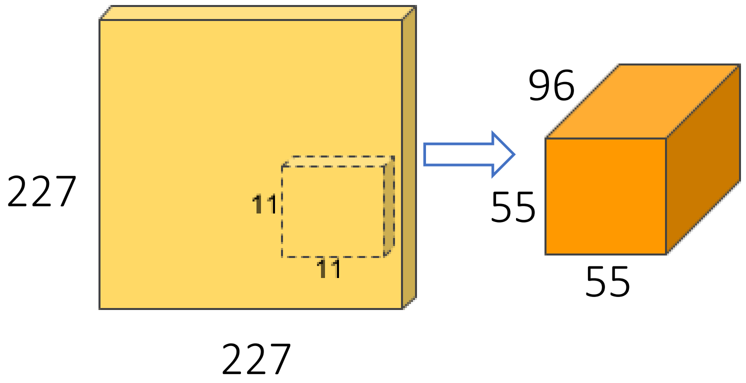


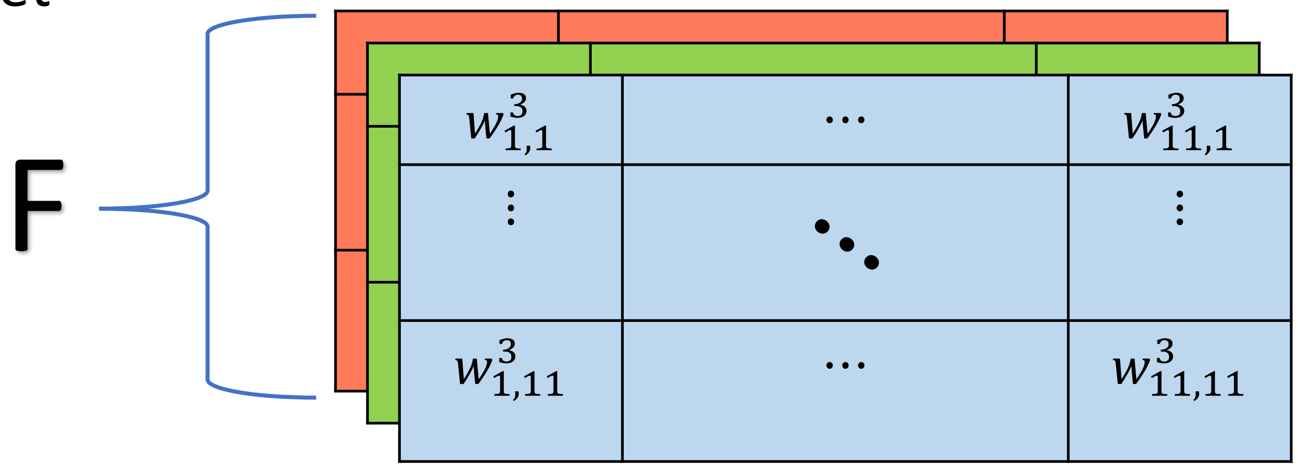
How many parameters?

#weights per out-put channel=F*F*number of input channels

Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet





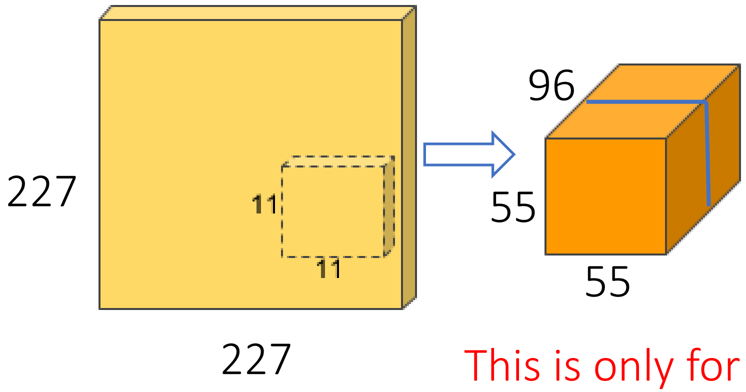
How many parameters?

#weights per out-put channel=F*F*number of input channels

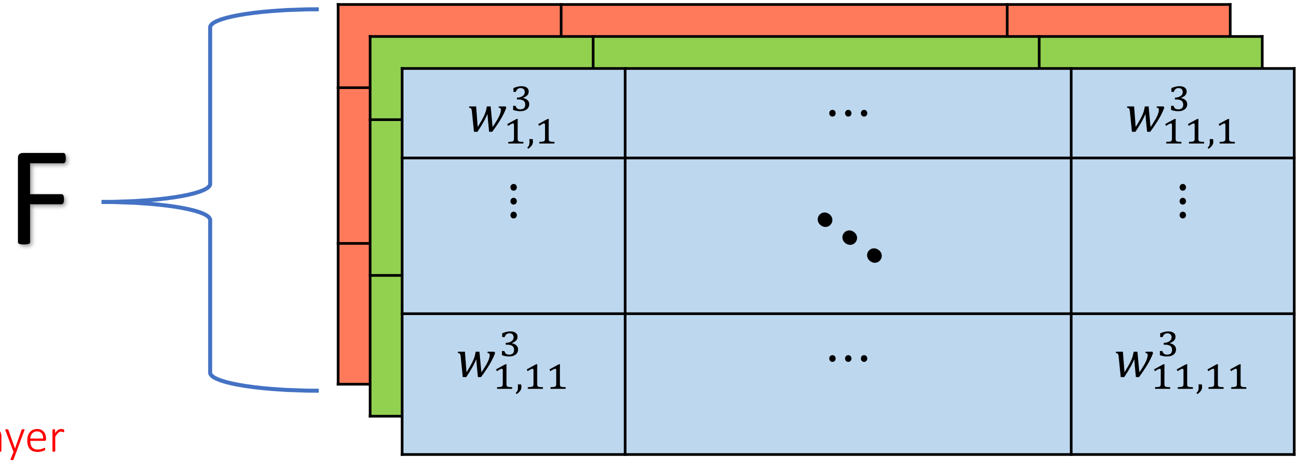
=11*11*3

Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet



This is only for one output layer



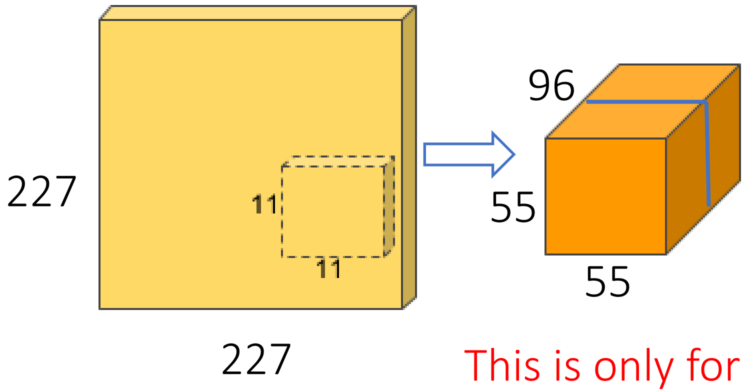
How many parameters?

#weights per out-put channel=F*F*number of input channels

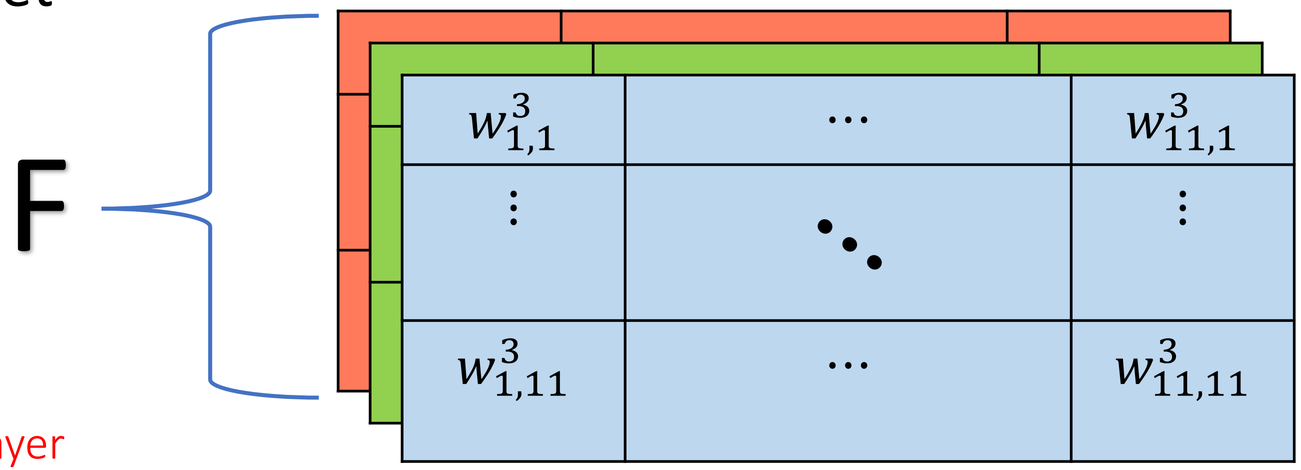
=11*11*3

Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet



This is only for one output layer



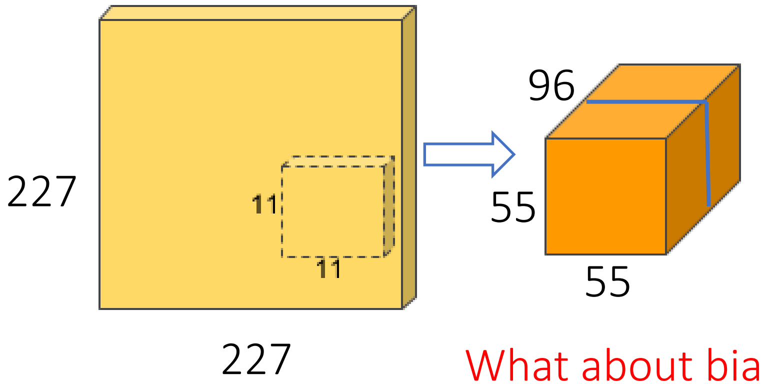
How many parameters?

#total weights=F*F*number of input channels*number of output channels

=11*11*3*96

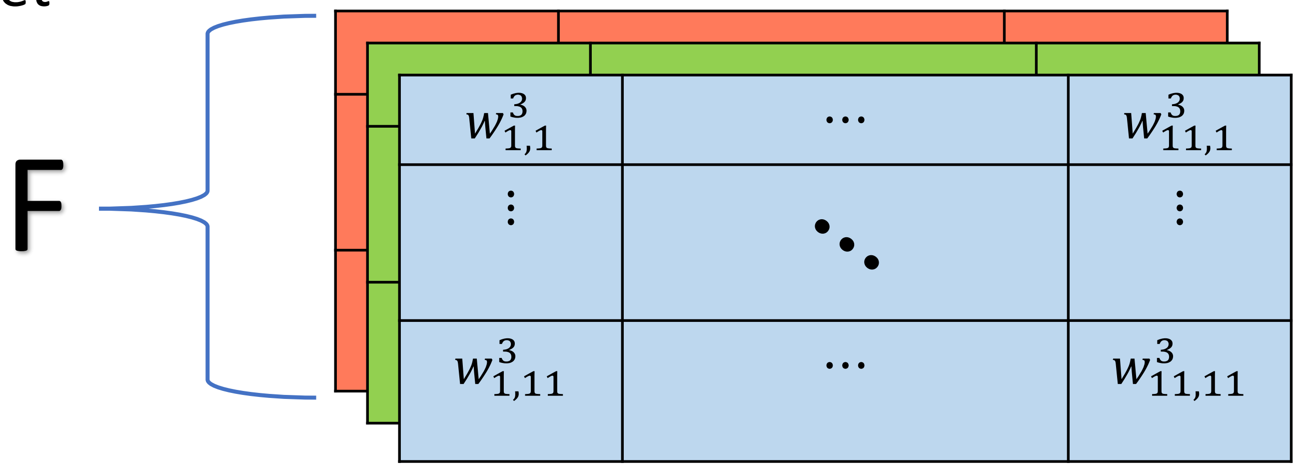
Number of Parameters

- Convolution layer
  - Example: first layer of AlexNet



What about biases?

For each output channel we have a bias



How many parameters?

#total parameters=F*F*number of input channels*number of output channels+biases

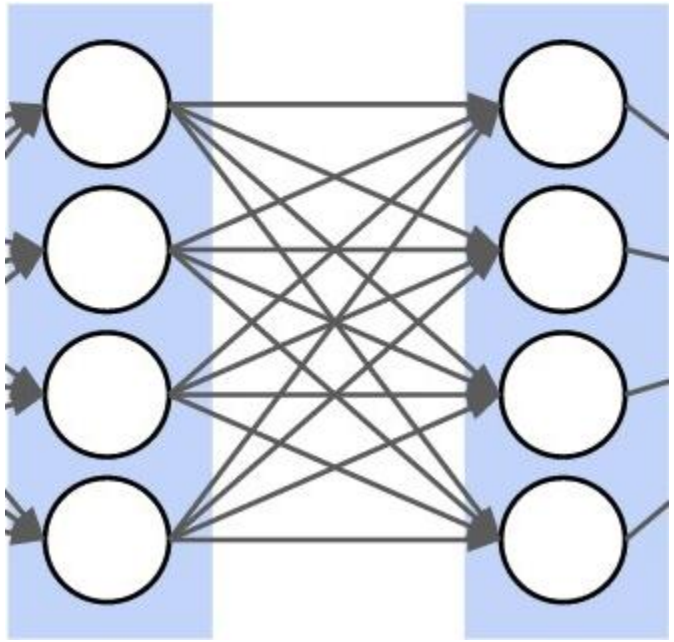
=11*11*3*96+96=34944

Number of Parameters

- Max-pool layers
  - No parameter is being learned (very cheap)
  - The pool size, stride, and padding are hyperparameters.

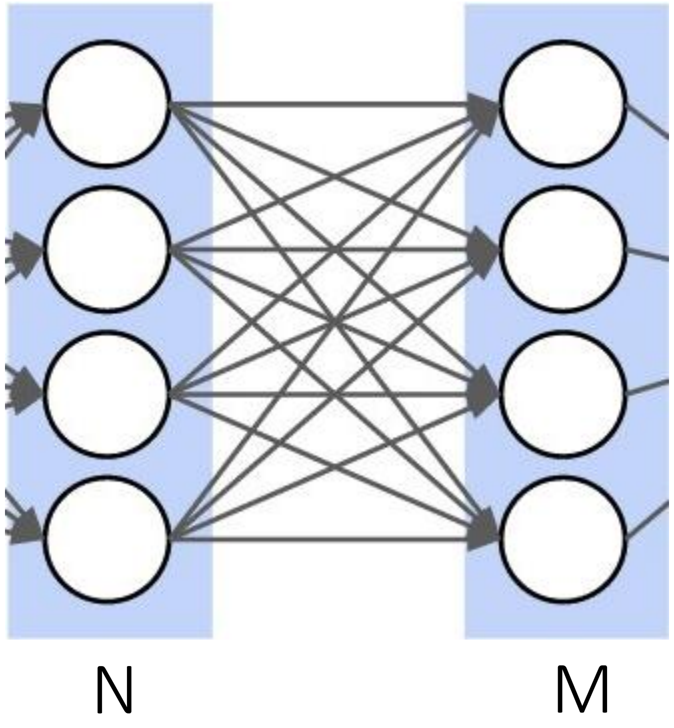
Number of Parameters

- Fully connected layers, all the neurons are connected to the neurons of the previous layer



Number of Parameters

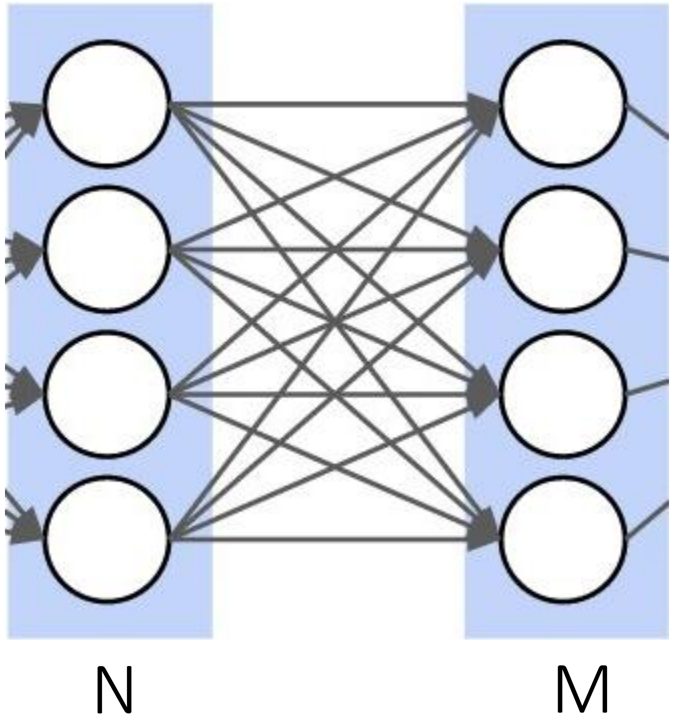
- Fully connected layers, all the neurons are connected to the neurons of the previous layer



Number of Parameters

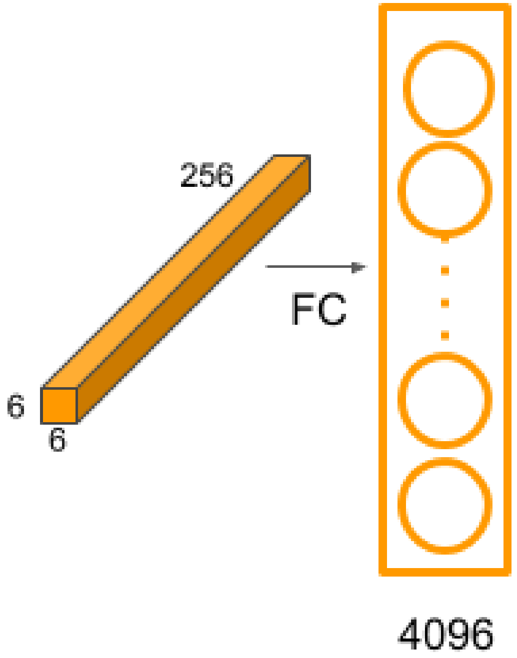
- Fully connected layers, all the neurons are connected to the neurons of the previous layer

#parameters=N*M+bias



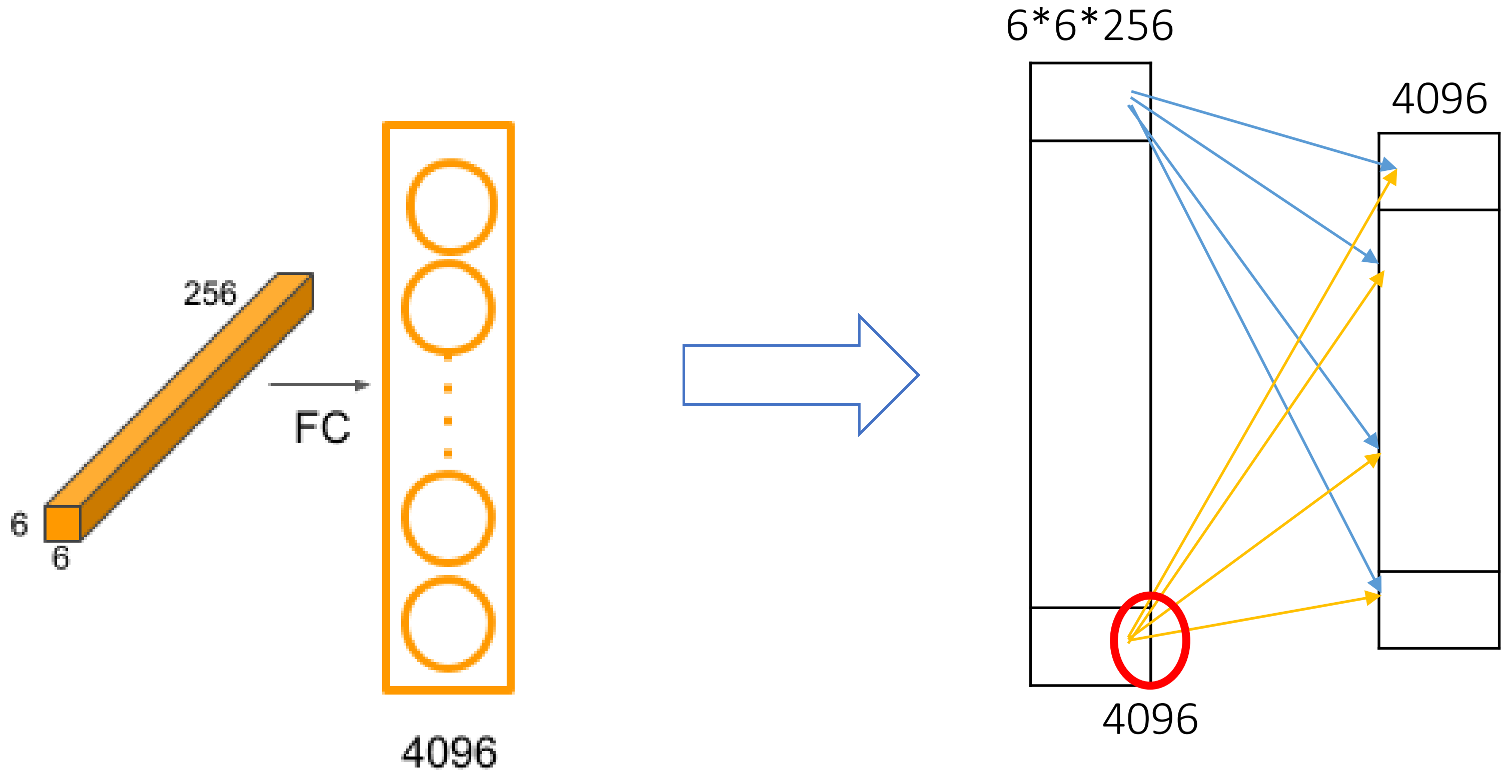
Number of Parameters

- Example:
  - What is the number of parameters in the first fully connected layer of the AlexNet?



Number of Parameters

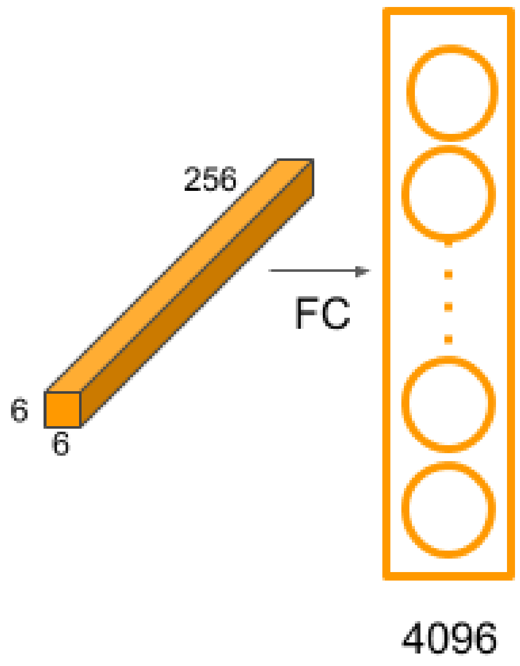
- Example:
  - What is the number of parameters in the first fully connected layer of the AlexNet?

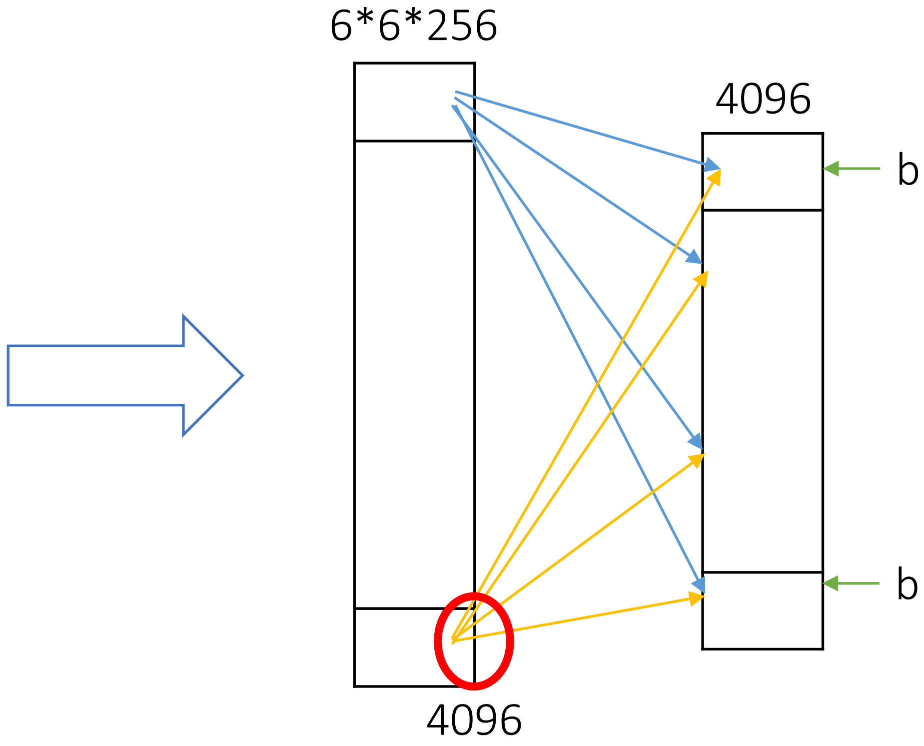


6*6*256*4096

Number of Parameters

- Example:
  - What is the number of parameters in the first fully connected layer of the AlexNet?





6*6*256*4096+4096

Summary

- Reviewed Neural Network
- Linear Classifier
- Convolutional Neural Network
- Layers of CNN
- Main terminologies
- Number of Parameters
- AlexNet

Image Segmentation

- Apply a classification algorithm on each pixel

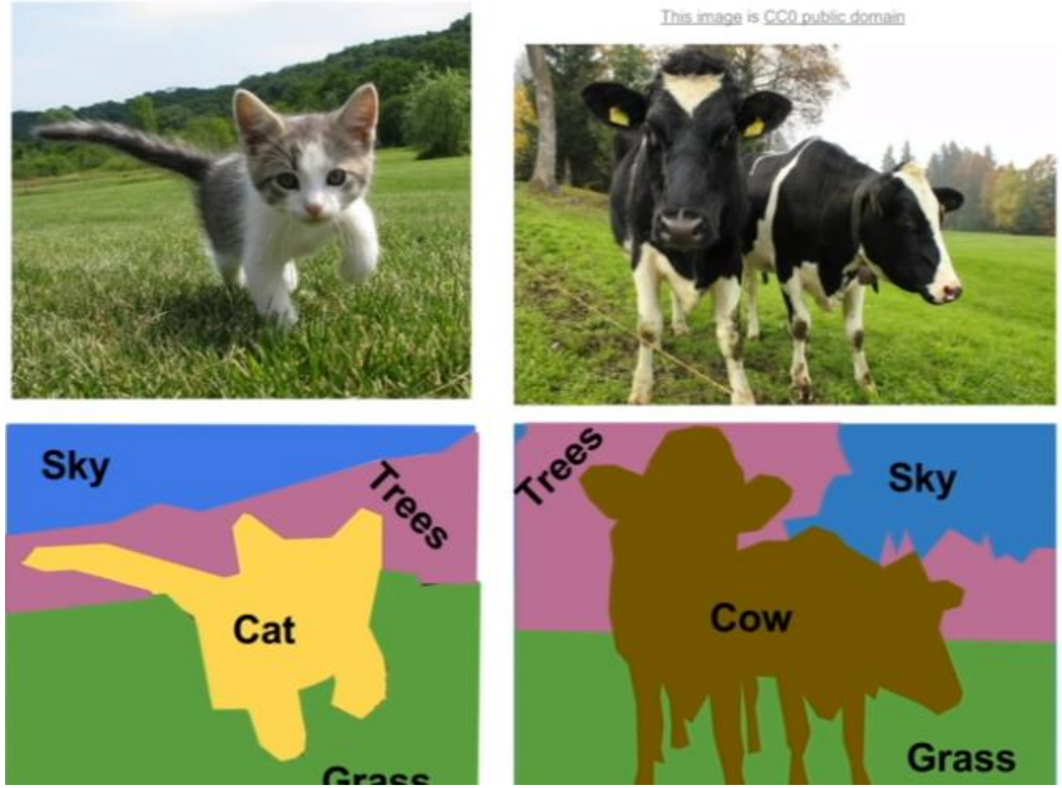


Image Segmentation

- Many little patches and then classify it using a classifier, consider the centre pixels as that particular class

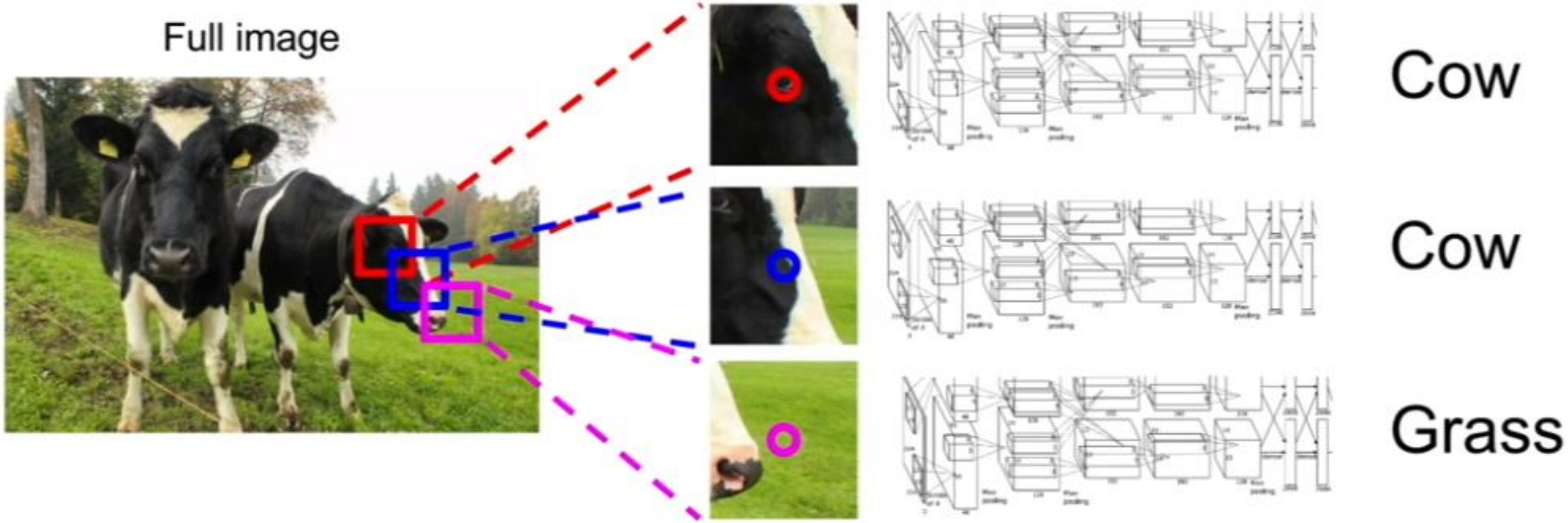


Image Segmentation

- Many little patches and then classify it using a classifier, consider the centre pixels as that particular class
- Expensive

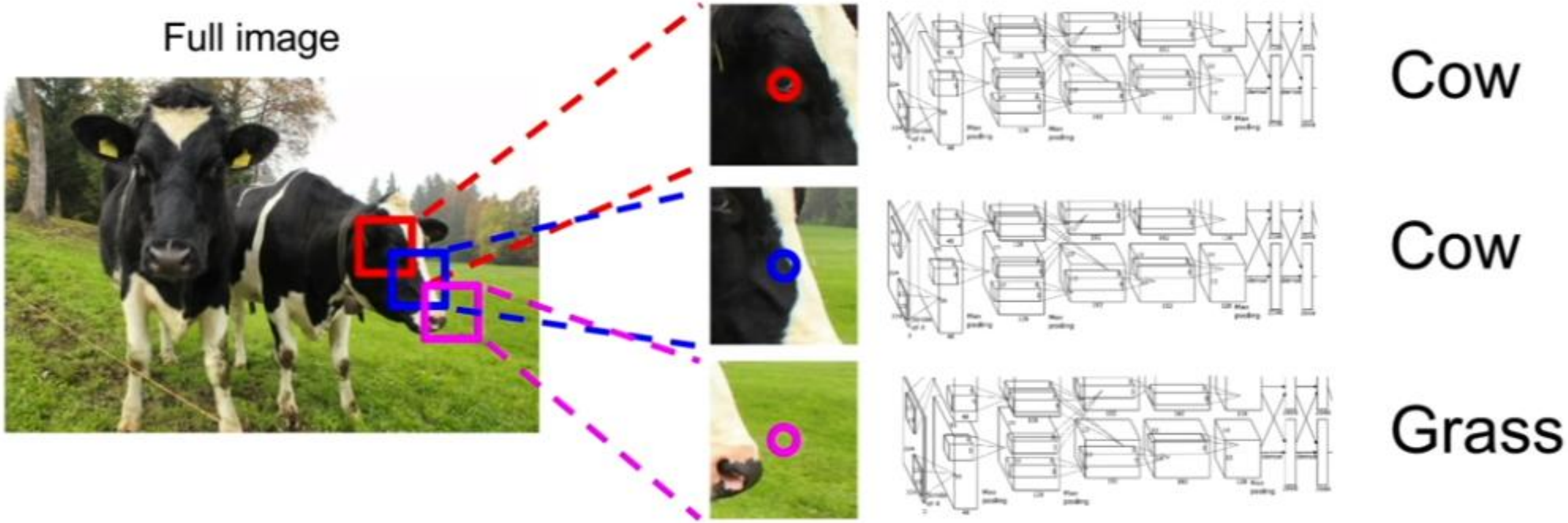


Image Segmentation

- End to end fully convolutional layer

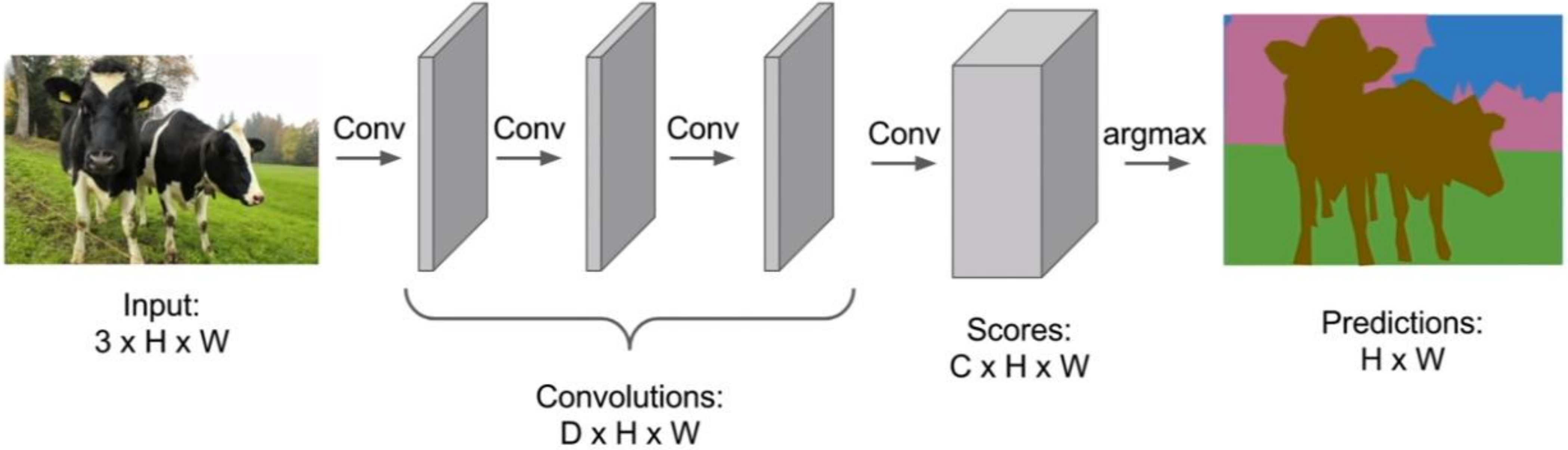


Image Segmentation

- How can we get image with the same dimension using CNN?

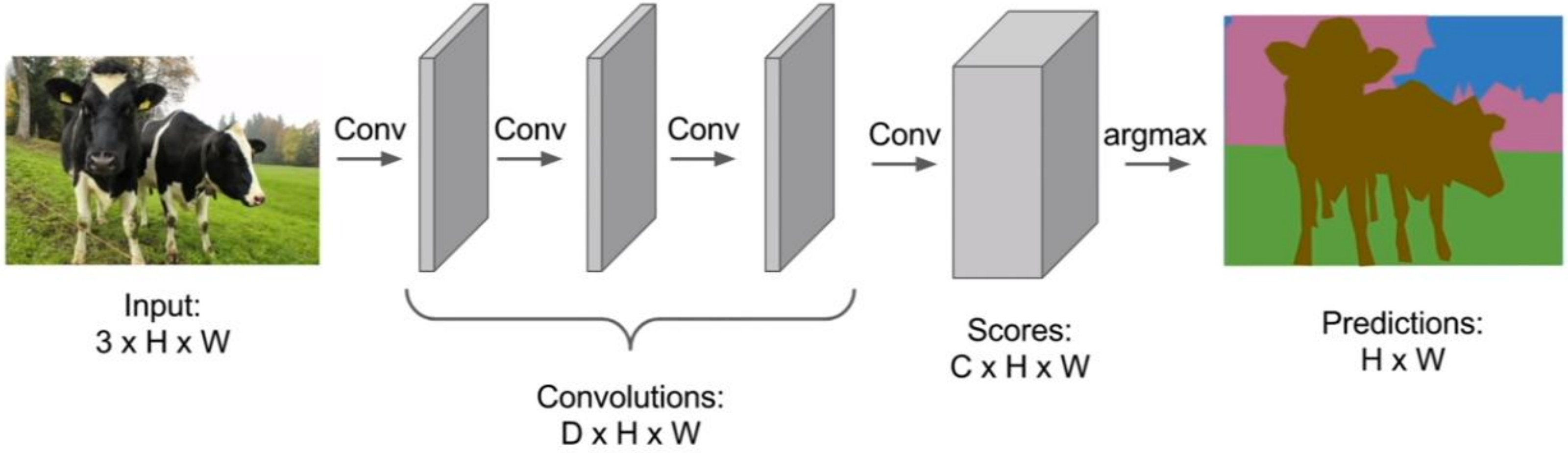


Image Segmentation

- We need many ground truth annotated data.

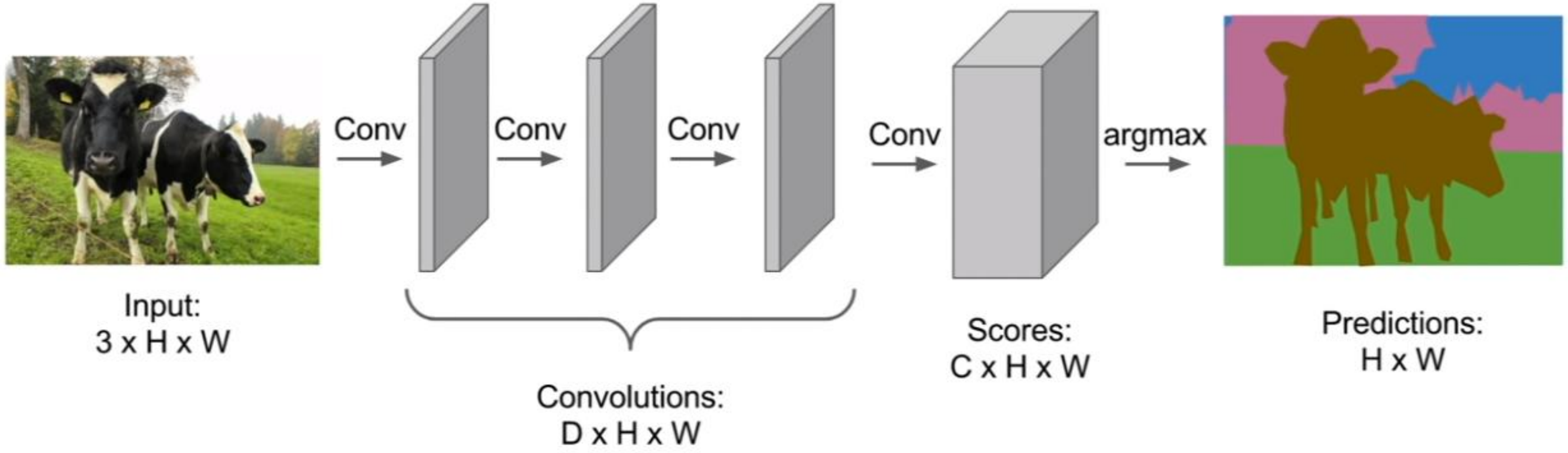
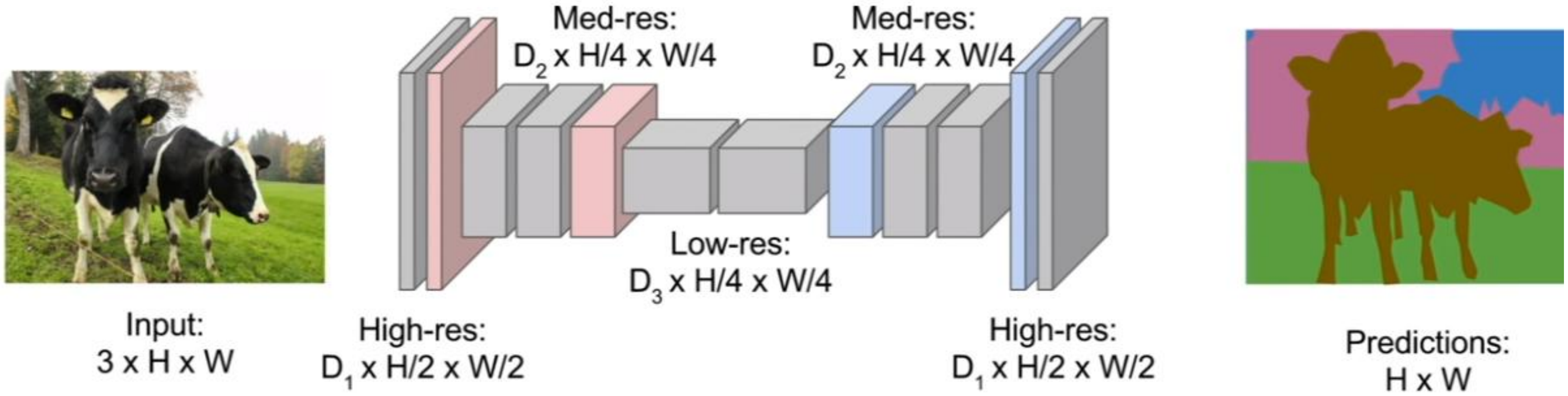


Image Segmentation

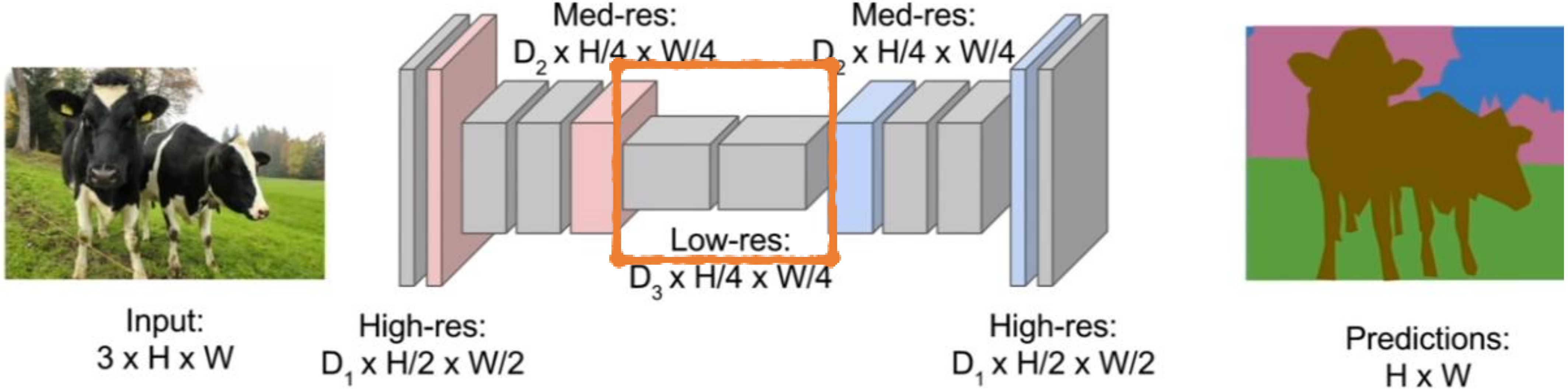
- Make a bottleneck
- Computationally cheaper



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

Image Segmentation

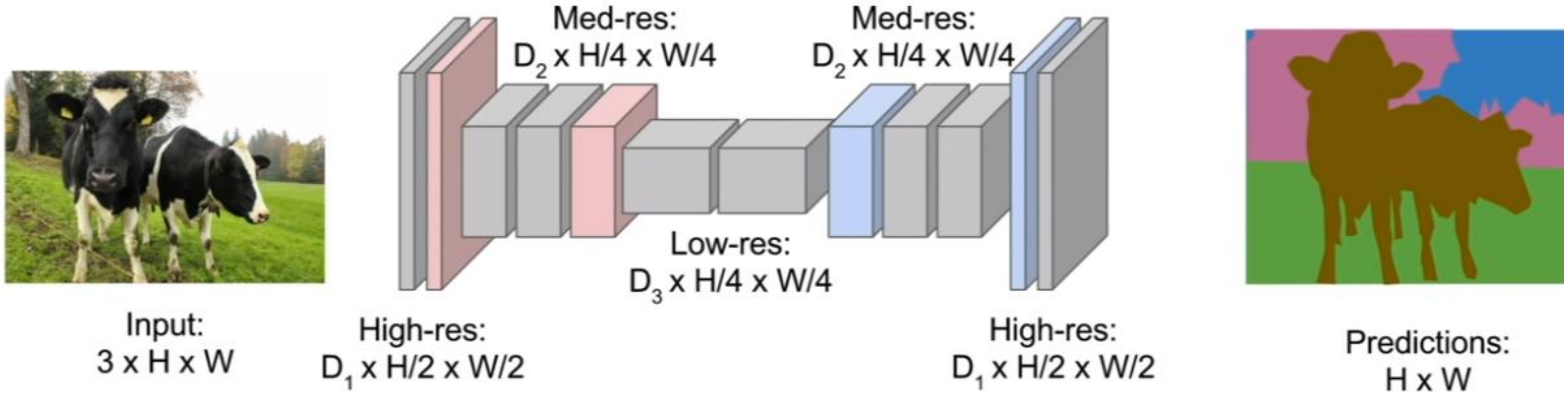
- Network learns high level features common between among similar data sets.



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

Image Segmentation

- We have usually K number of features at the end of the segmentation network, each feature representing a class of objects assigned to pixels



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

Image Segmentation

- We use cross-entropy loss function combined with Softmax

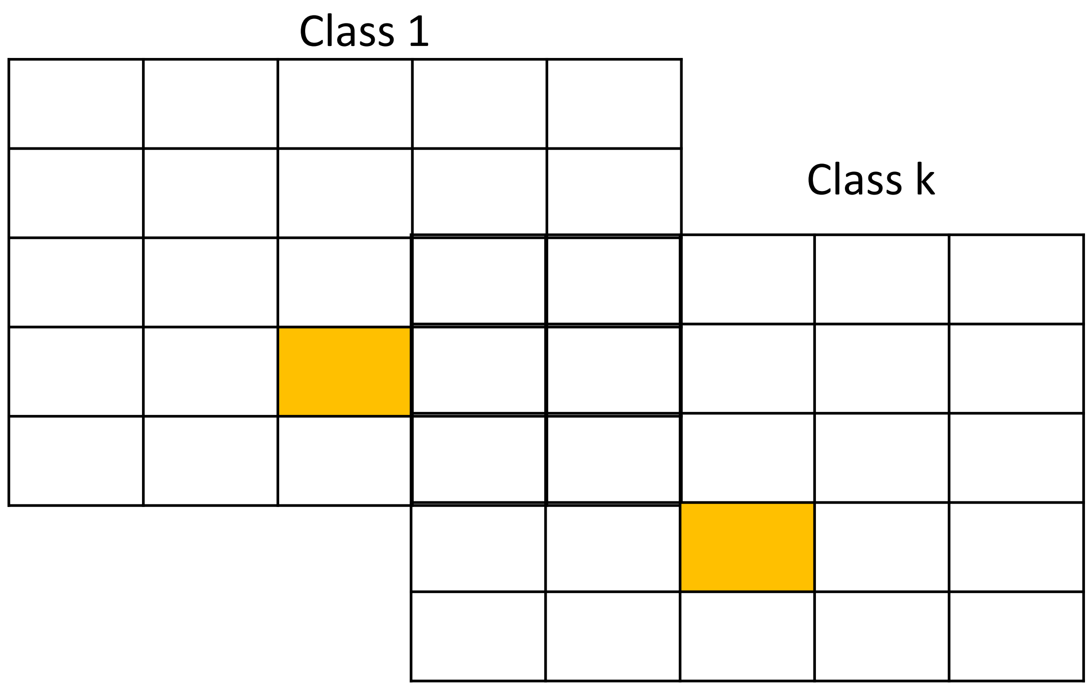


Image Segmentation

- $a_k(x)$: activation in feature channel k

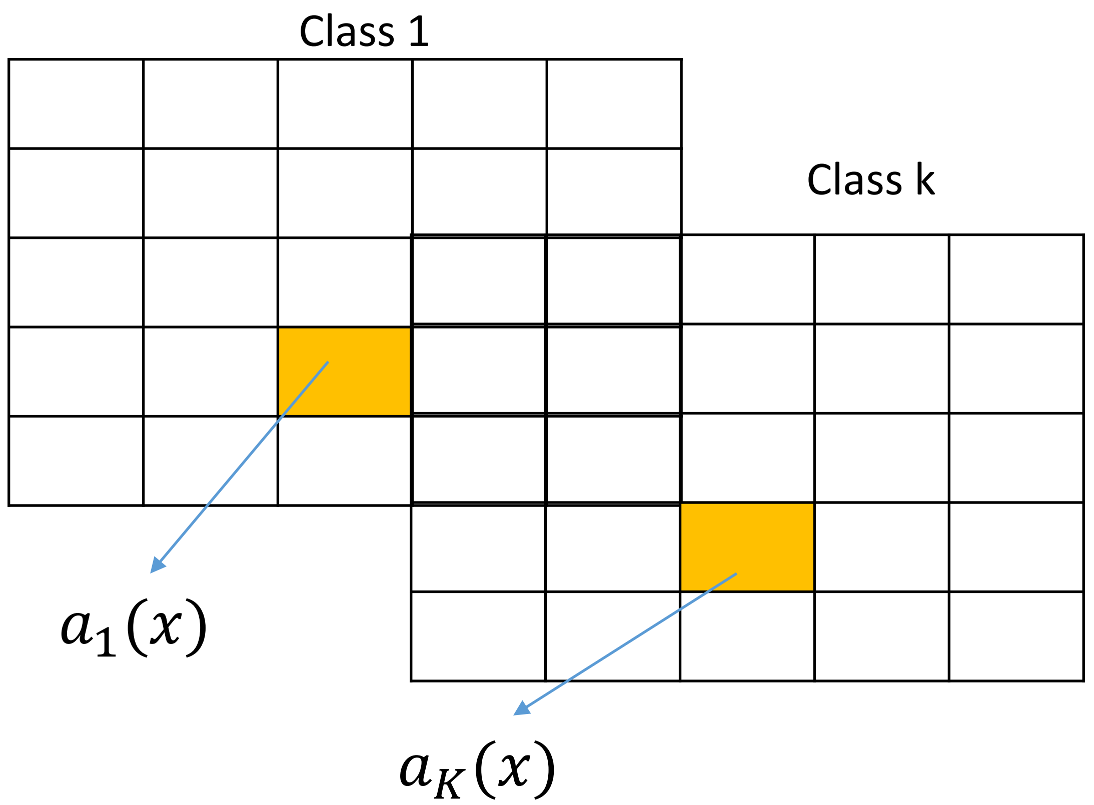
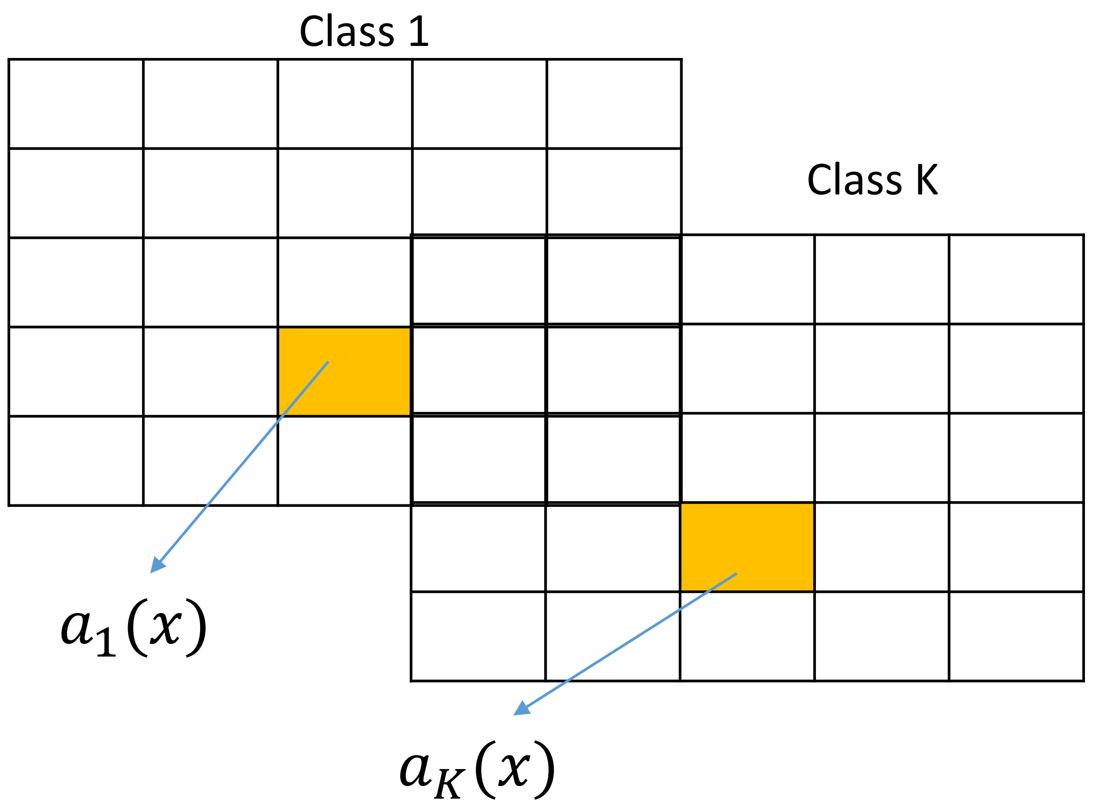


Image Segmentation

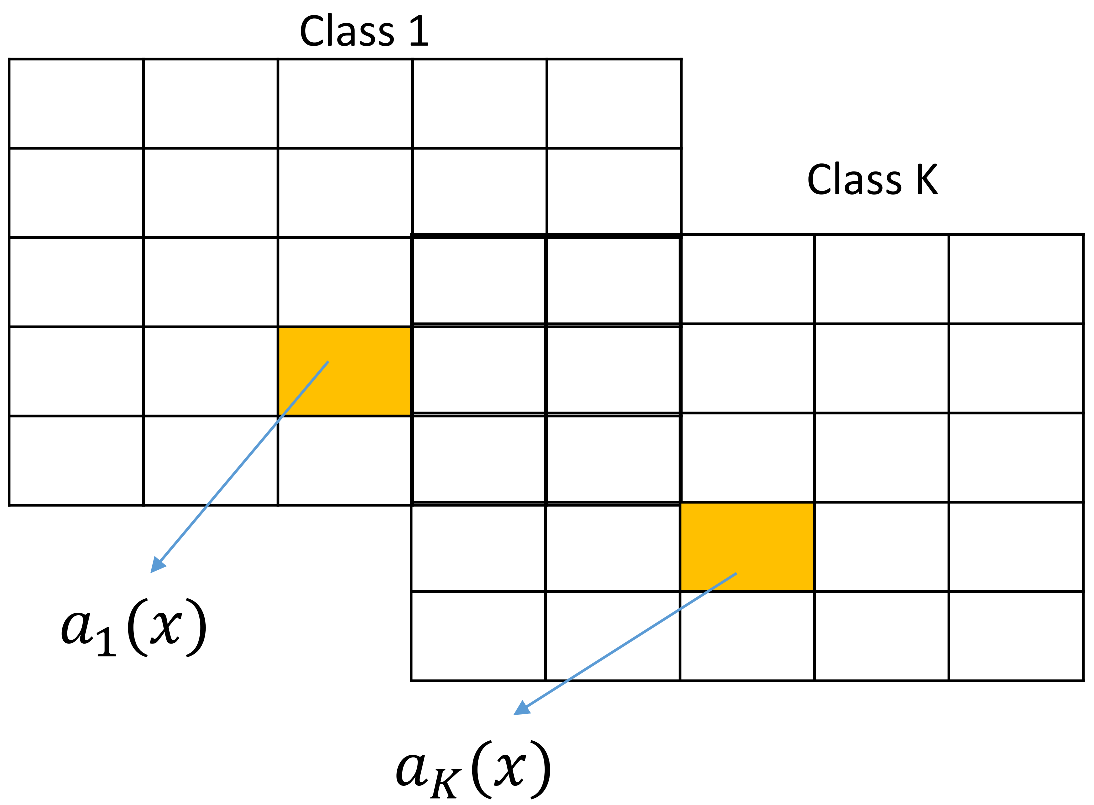
- Softmax is defined as



$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{j=1}^{K} e^{a_j(x)}}$$

Image Segmentation

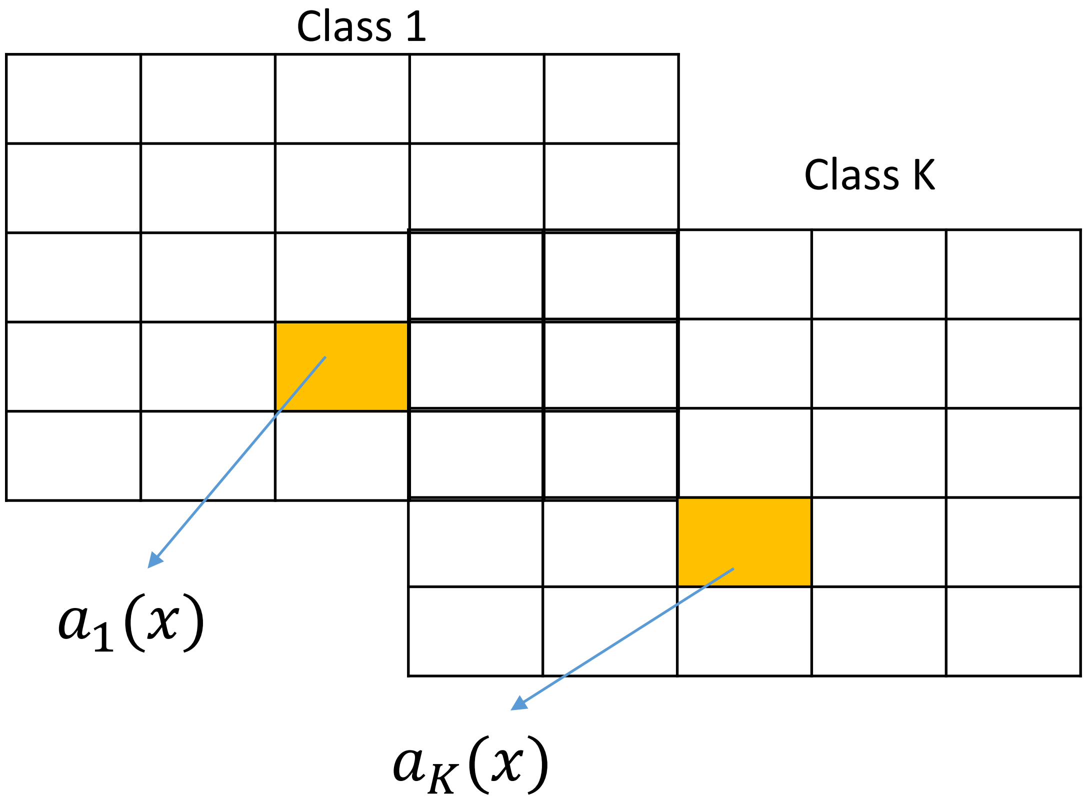
- $p_k(x) \approx 1$ for k that has the maximum activation $a_k(x)$ and $p_k(x) \approx 0$ for others.



$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{j=1}^{K} e^{a_j(x)}}$$

Image Segmentation

- Cross entropy penalizes at each pixel, the deviation of $p_{l(x)}(x)$ from 1. $l(x)$ is the true label of x.

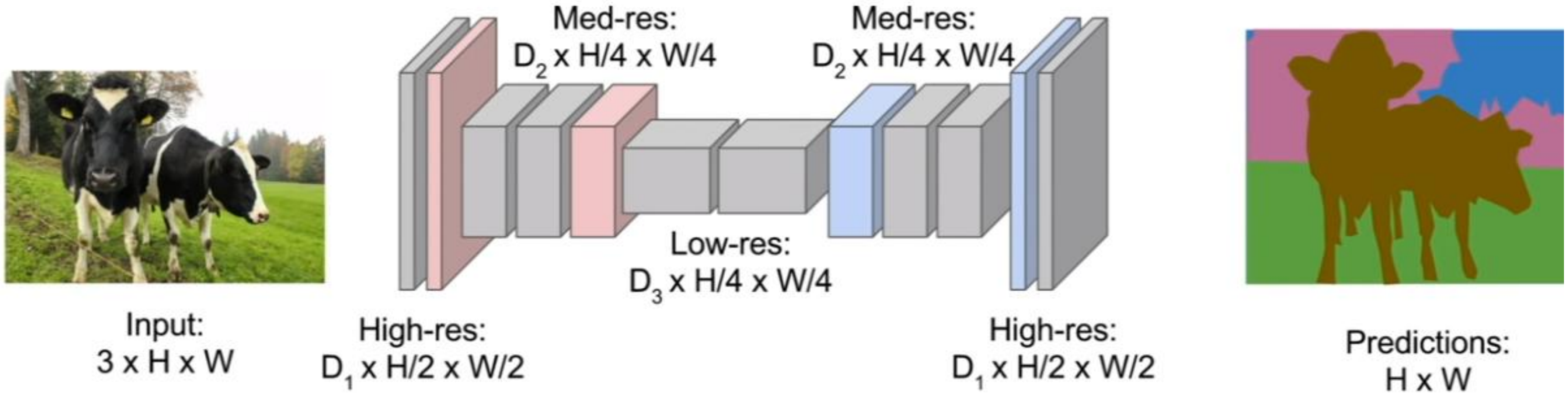


$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{j=1}^{K} e^{a_j(x)}}$$

$$E = \sum_{x \in \{1,2,\ldots,K\}} \log(p_{l(x)}(x))$$

Image Segmentation

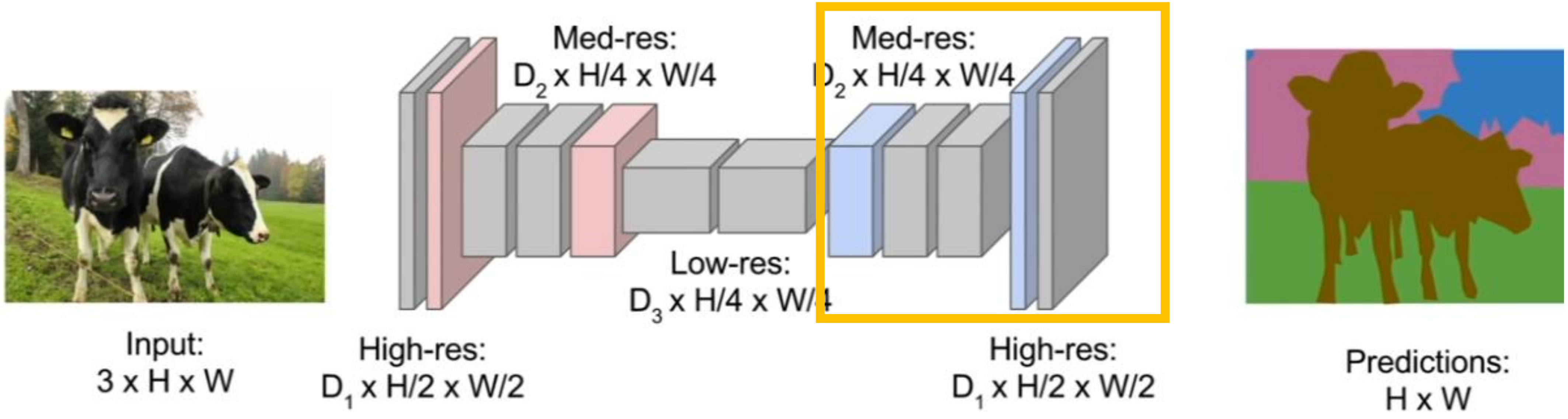
- We use cross-entropy loss function combined with Softmax



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

Image Segmentation

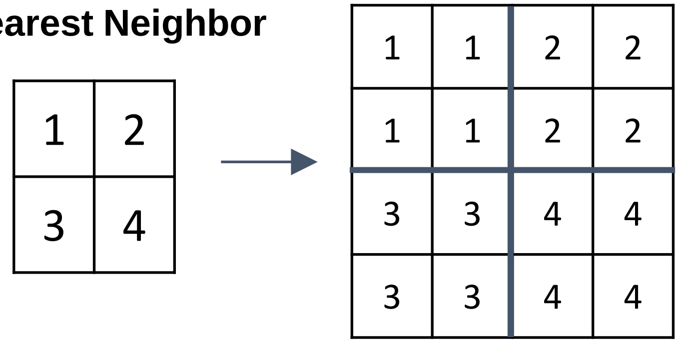
- Upsampling



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

upsampling: “Unpooling”

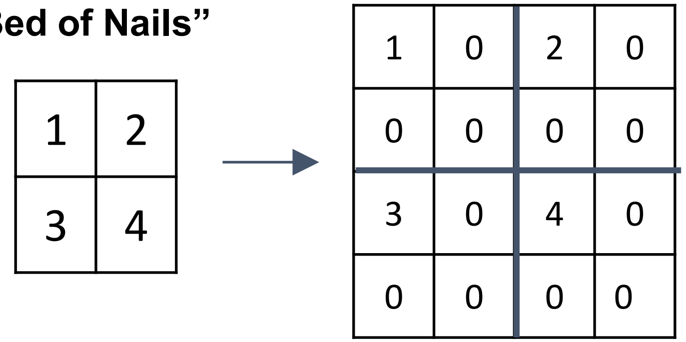
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

upsampling: “Unpooling”

Bilinear

1	2
3	4

Input: 2 x 2

1	1.33	1.66	2
1.66	2	2.33	2.66
2.33	2.66	3	3.33
3	3.33	3.66	4

Output: 4 x 4

upsampling: “Unpooling”

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

…

Rest of the network

Max Unpooling
Use positions from pooling layer

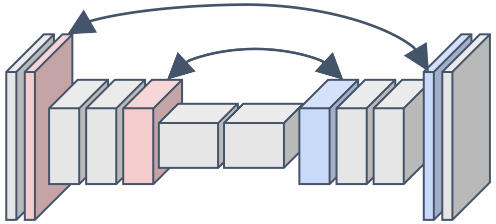
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

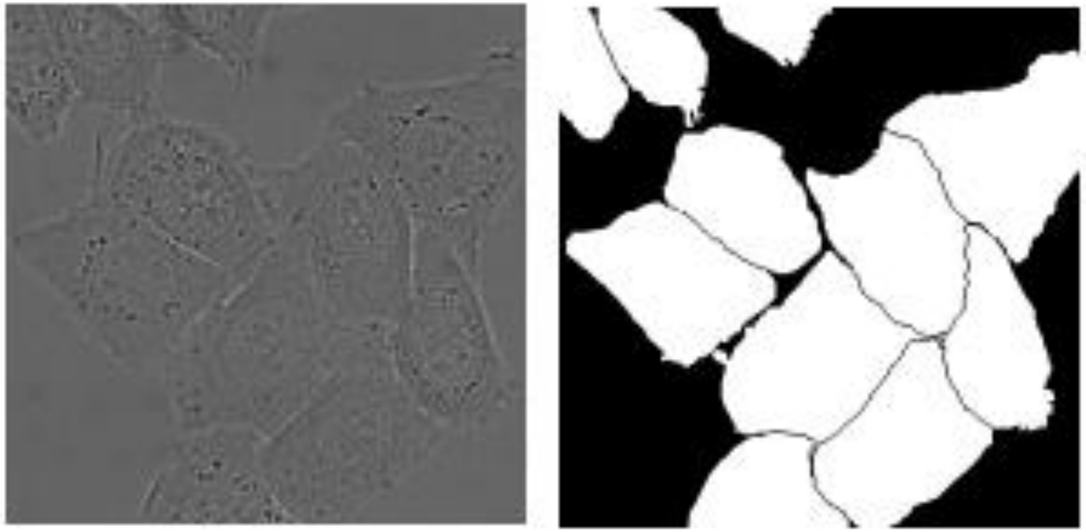
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



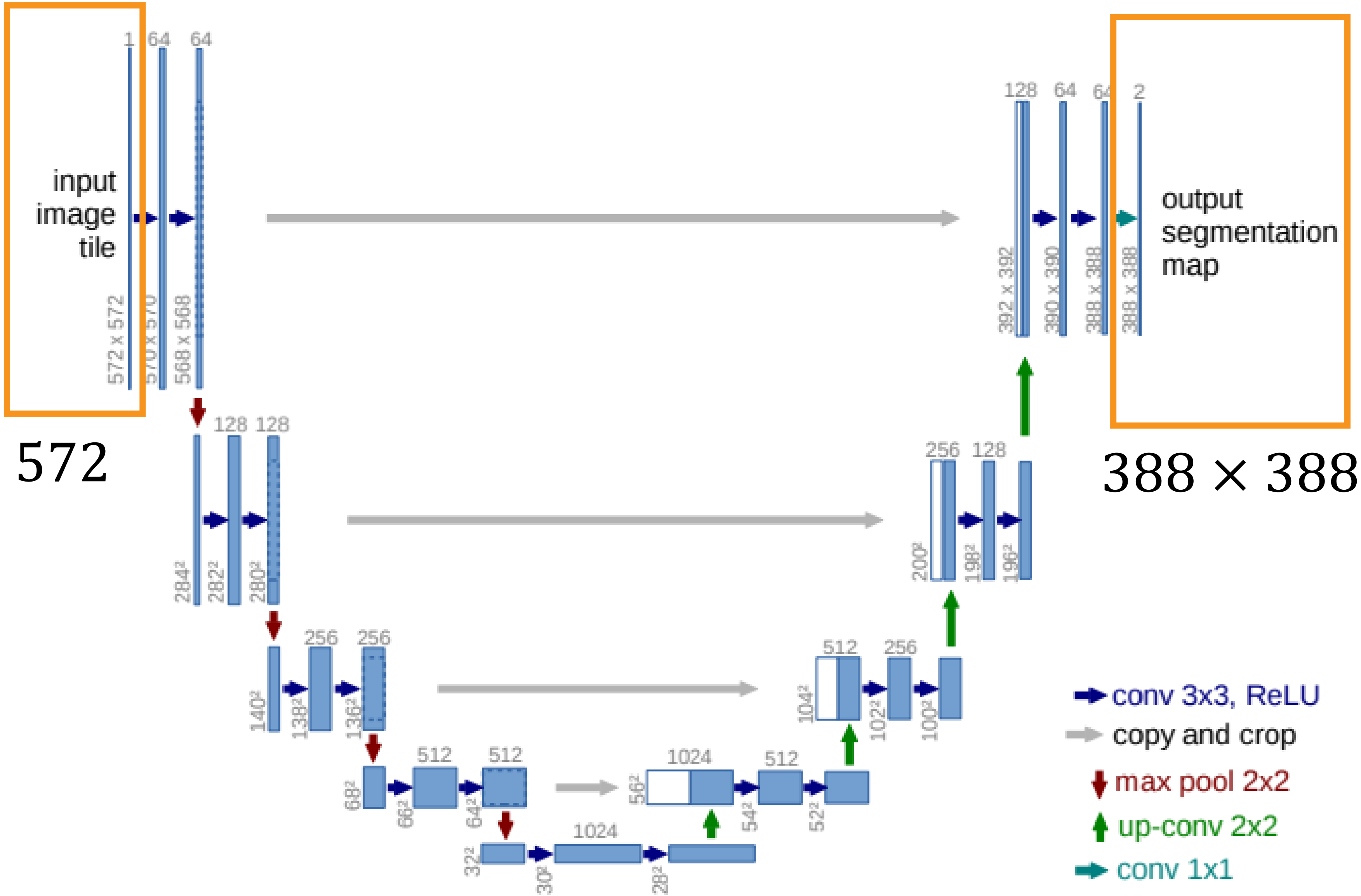
UNet

- UNet (your first assignment) is designed along the same direction to segment medical images.



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

UNet



572×572

388×388

Input

- A bigger window is chosen to be aware of the neighborhood at boundary to provide a seamless connection between patches.

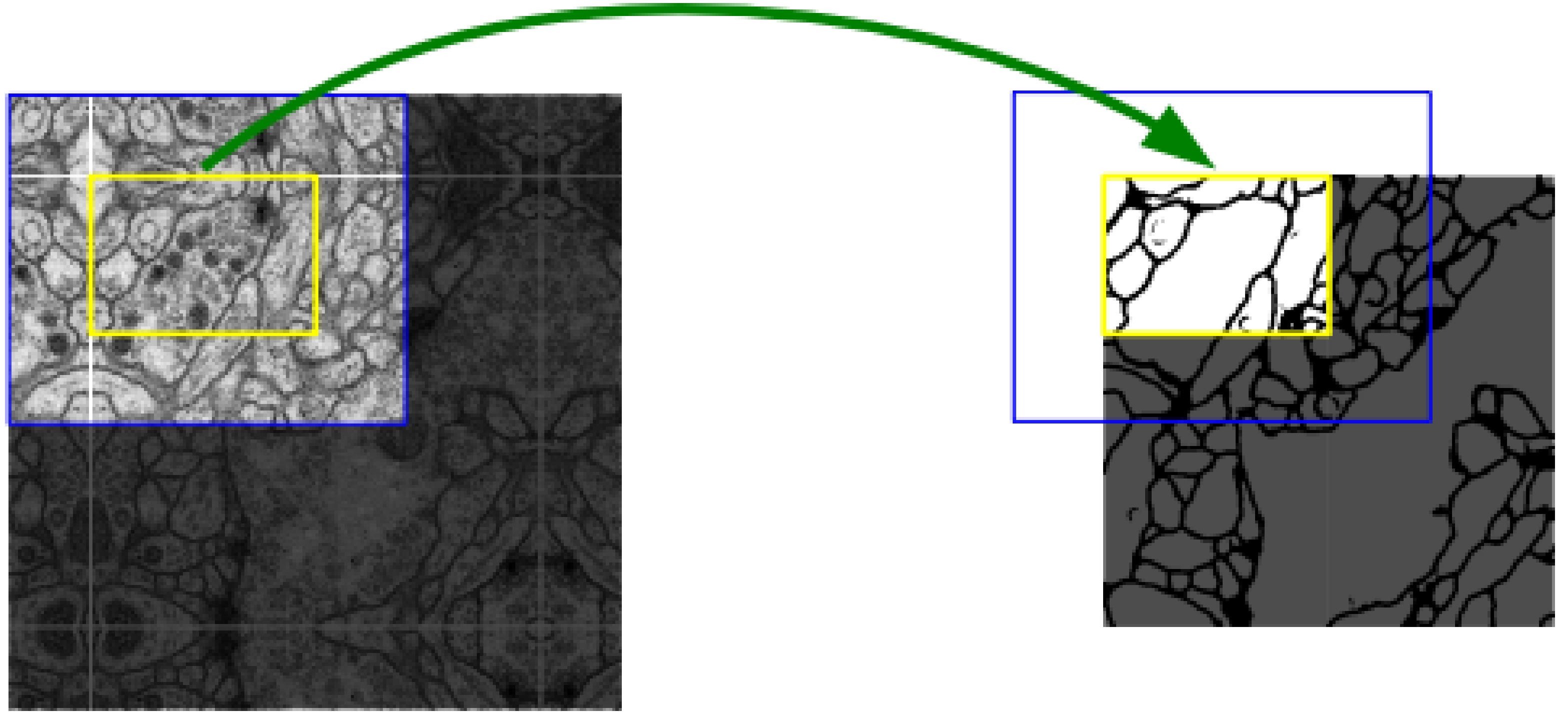
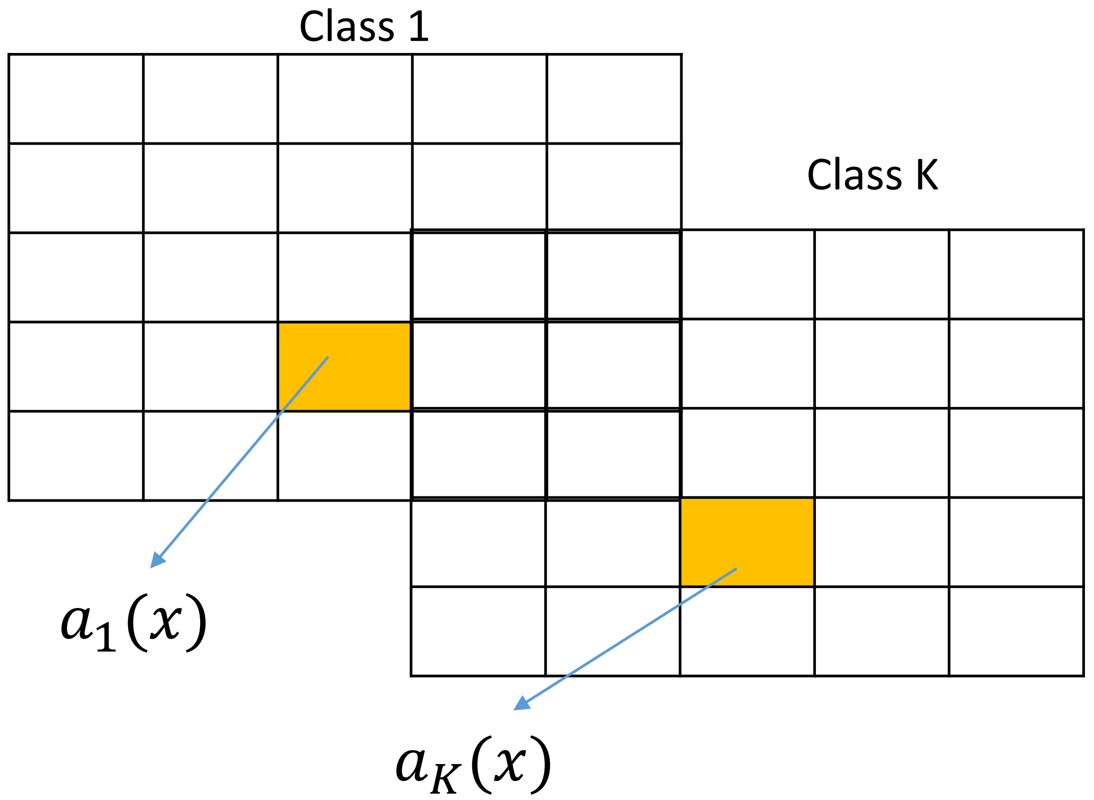


Image Segmentation

- $w(x)$ is defined to give more importance to pixels close to boundaries in ground truth.



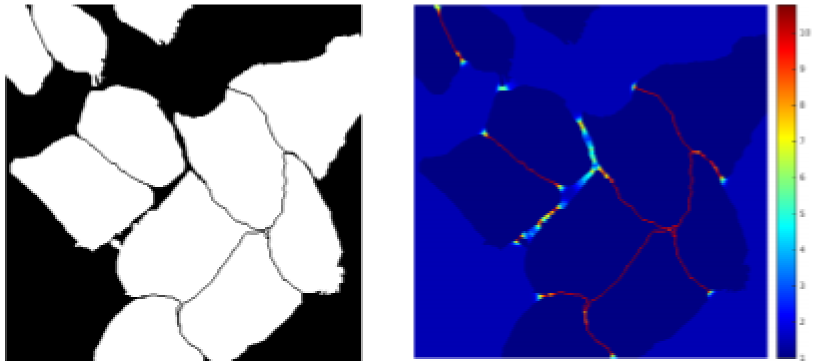
$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{j=1}^{K} e^{a_j(x)}}$$

$$E = \sum_{x \in \{1,2,\ldots,K\}} \boxed{w(x)} \log(p_{l(x)}(x))$$

Image Segmentation

- $w(x)$ is defined to give more importance to pixels close to boundaries in ground truth.

$$E = \sum_{x \in \{1,2,\ldots,K\}} \boxed{w(x)} \log(p_{l(x)}(x))$$



$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

Image Segmentation

- $d_1(x)$ and $d_2(x)$ are distances to the first and second nearest cell boundaries.

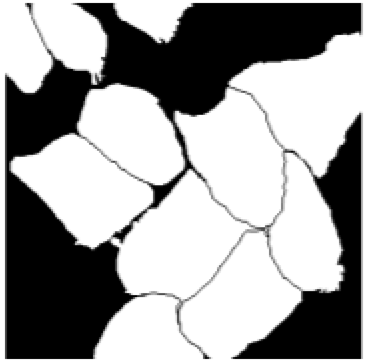
$$E = \sum_{x \in \{1,2,\ldots,K\}} w(x) \log(p_{l(x)}(x))$$

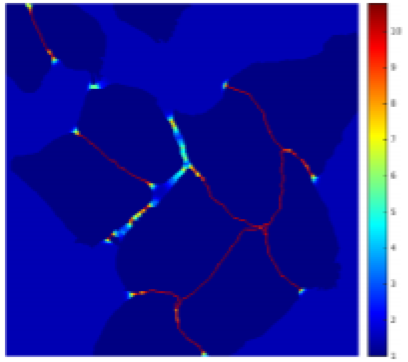
$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(\boxed{d_1(\mathbf{x})} + \boxed{d_2(\mathbf{x})})^2}{2\sigma^2}\right)$$

Image Segmentation

- $w_c(x)$ is class probability map, $w_0 = 10$, $\sigma = 5$.

$$E = \sum_{x \in \{1,2,\ldots,K\}} w(x) \log(p_{l(x)}(x))$$



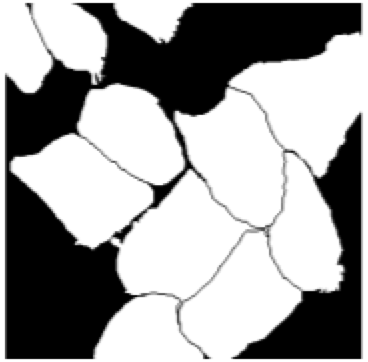


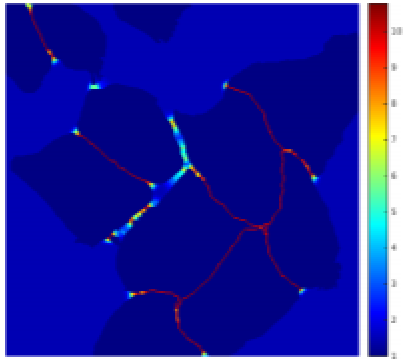
$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(\boxed{d_1(\mathbf{x})} + \boxed{d_2(\mathbf{x})})^2}{2\sigma^2}\right)$$

Image Segmentation

- This is a good example of feature engineering.

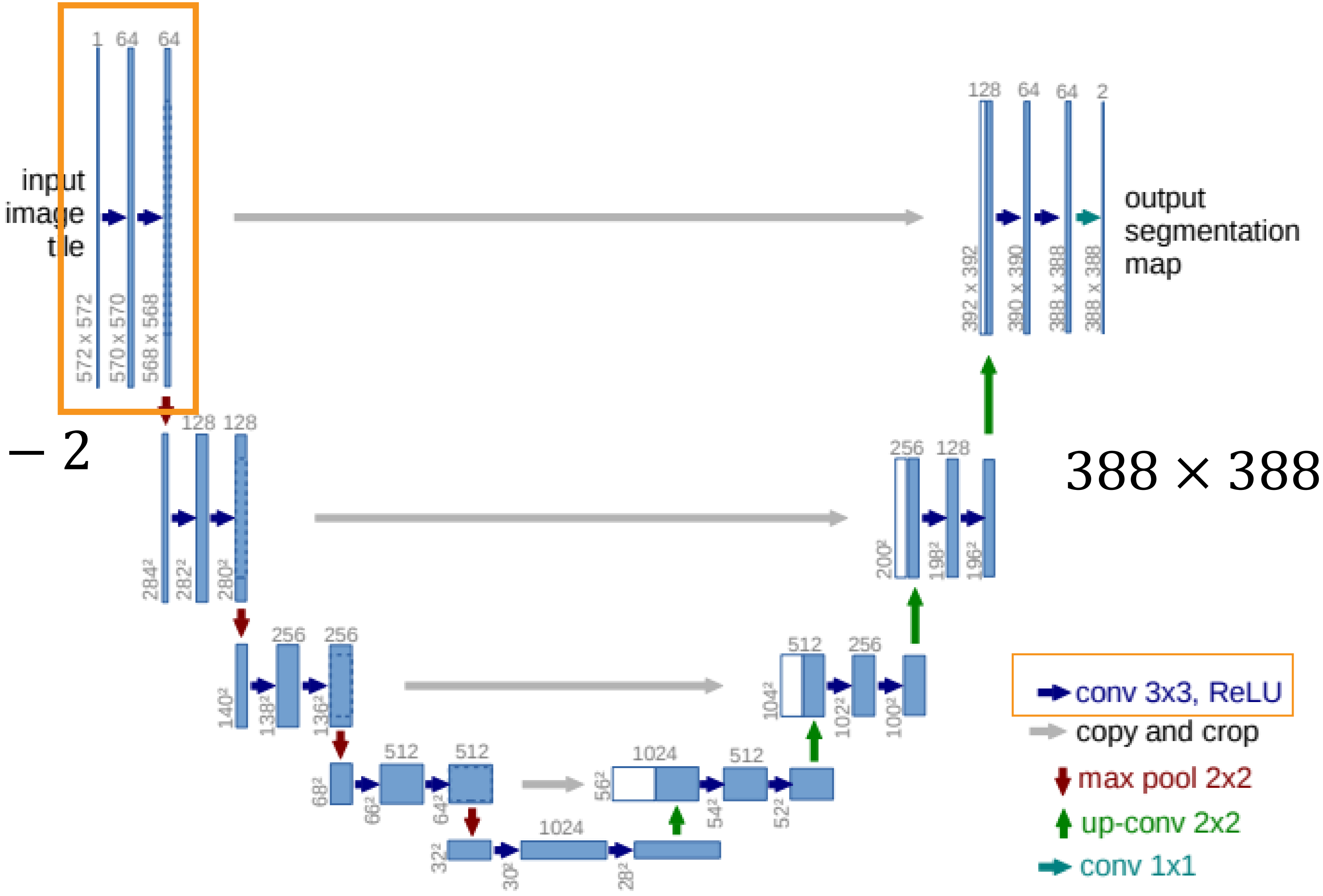
$$E = \sum_{x \in \{1,2,\ldots,K\}} w(x) \log(p_{l(x)}(x))$$





$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(\boxed{d_1(\mathbf{x})} + \boxed{d_2(\mathbf{x})})^2}{2\sigma^2}\right)$$

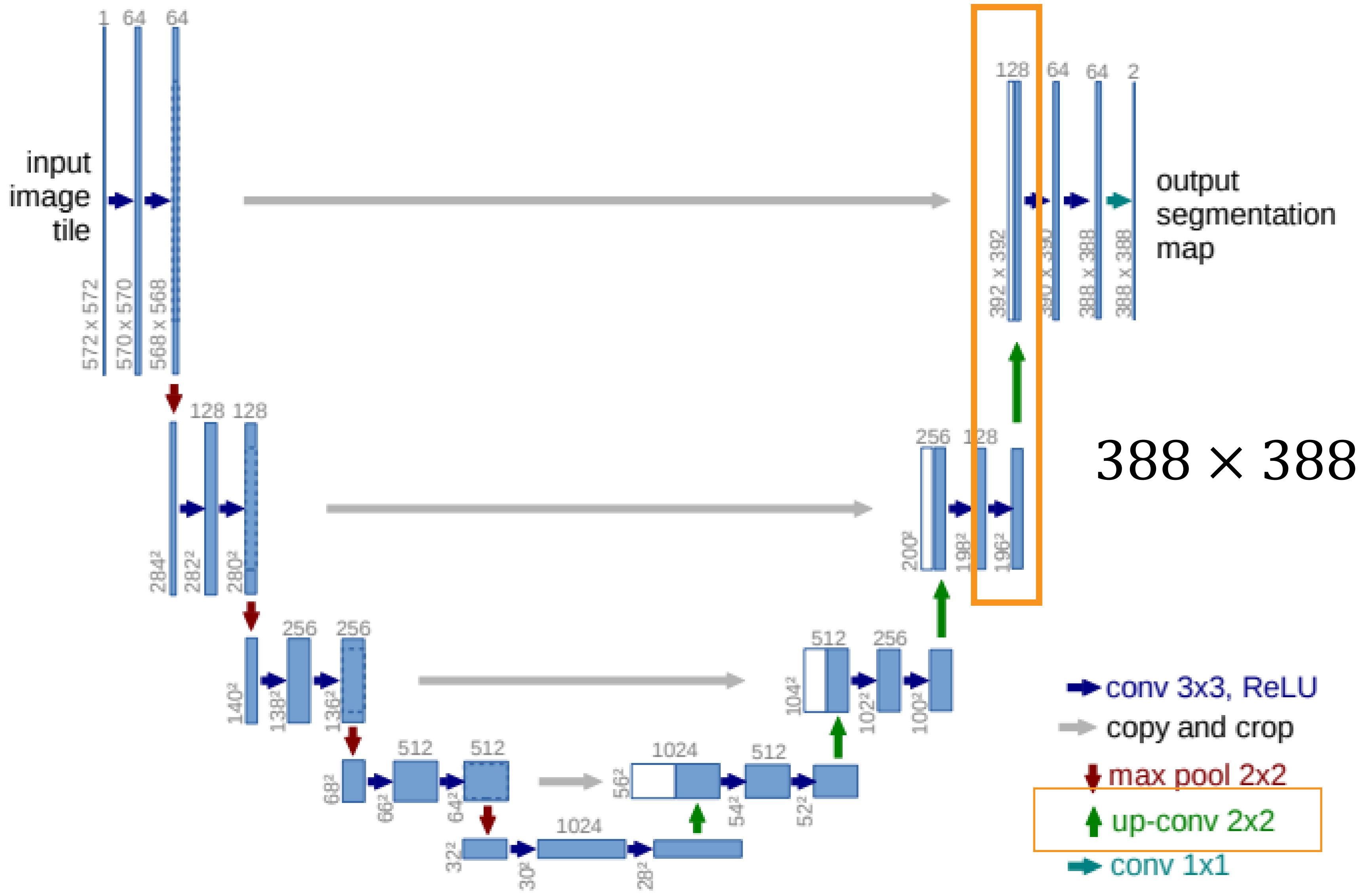
UNet



$n \to n - 2$

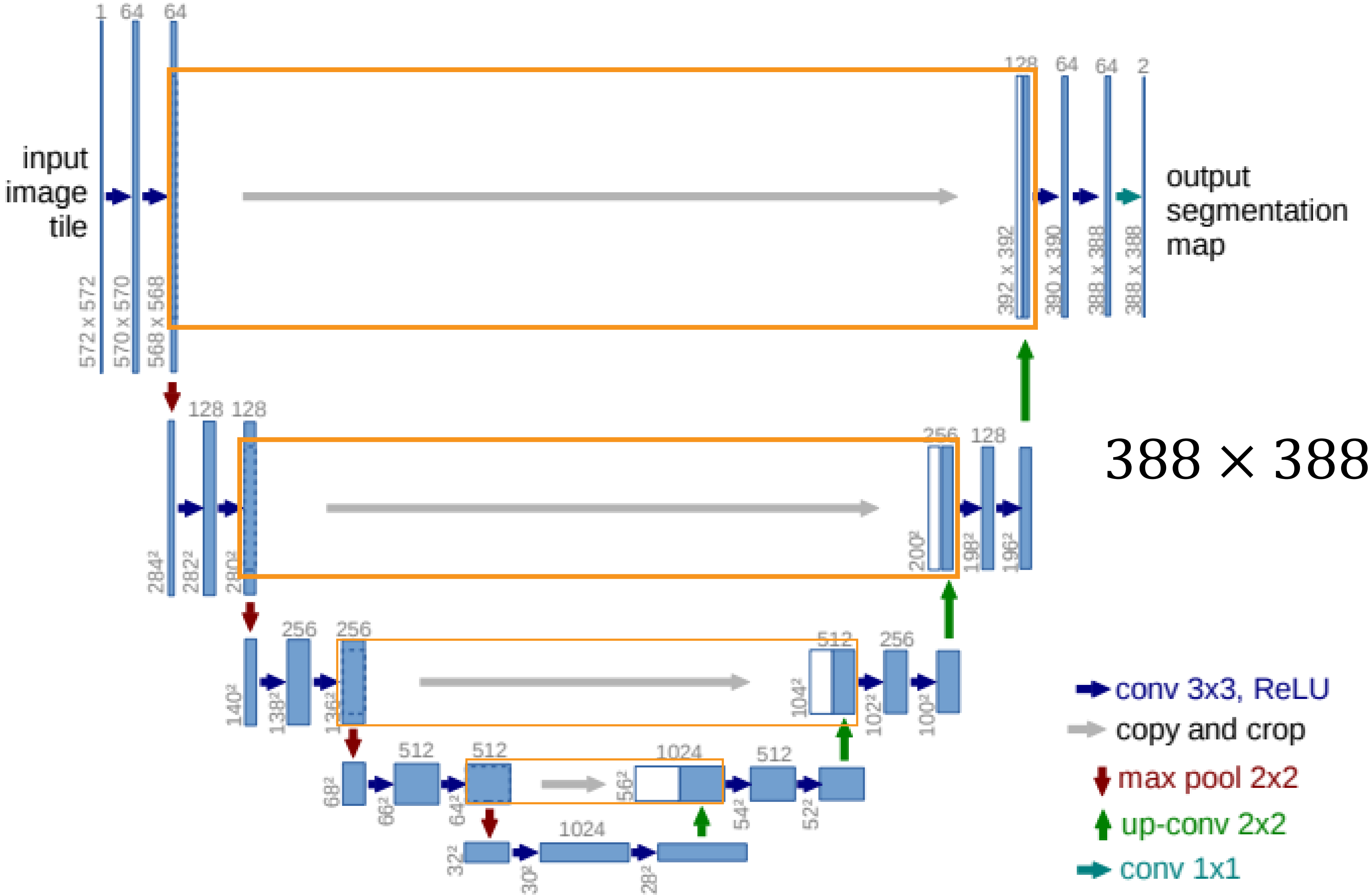
388×388

UNet


input image tile
572 x 572
570 x 570
568 x 568
1
64
64
128
128
256
256
512
512
1024
1024
512
512
256
256
128
128
64
64
2
output segmentation map
388 × 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

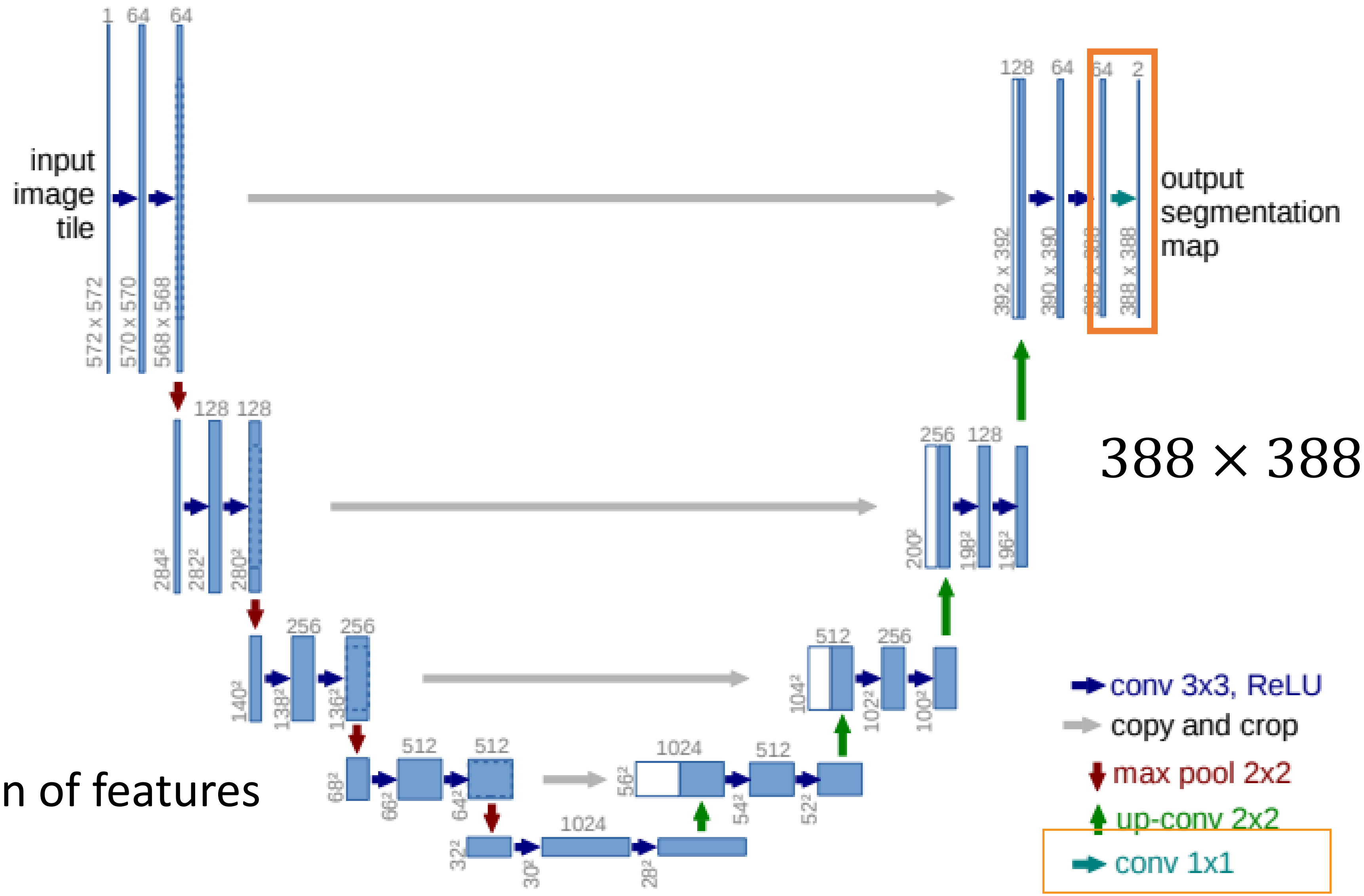
UNet

Part of the features are copied and transferred to next layers
It makes the network remember about early features



388×388

UNet

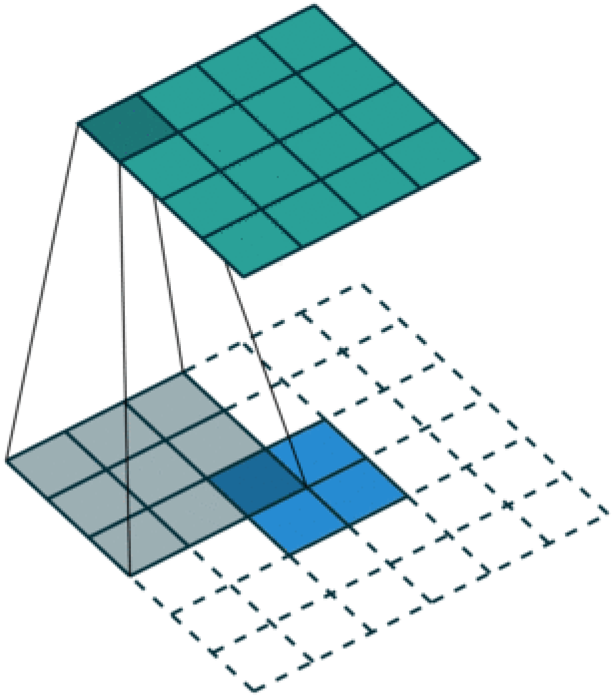


388×388

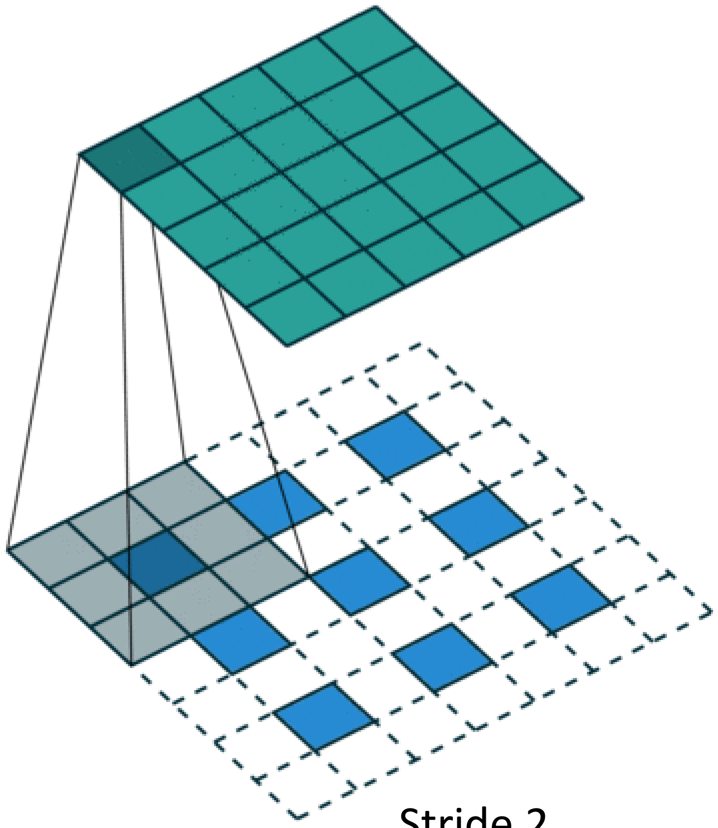
Linear combination of features into one number

Deconvolution

- Very similar to convolution. You just add padding both to corners and also around pixels to make the dimension bigger.



Stride 1



Stride 2

Data Augmentation

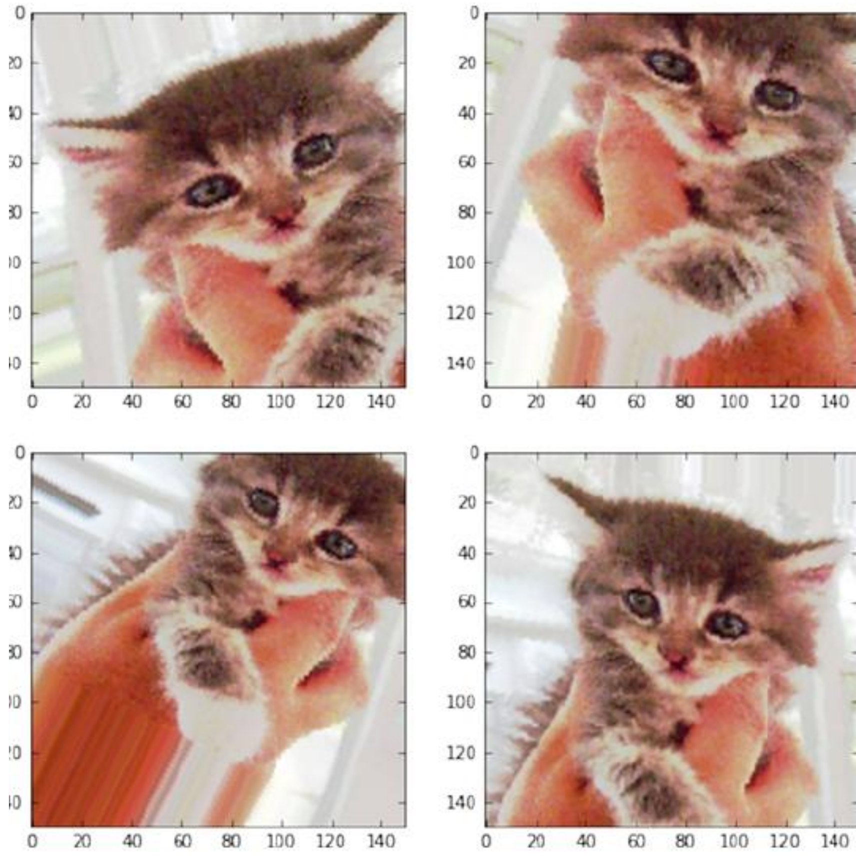
- Sometimes, your data is not enough.
- To do so, you can scale, transpose, and rotate training data to provide more data to your network:

Data Augmentation

- In Keras, it is very easy to do so:

```
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```



Data Augmentation

- In your assignment, you are going to augment the data.

The End