Part 5: Convolutional neural networks and regularization

anton.korosov@nersc.no

November 2021

NERSC

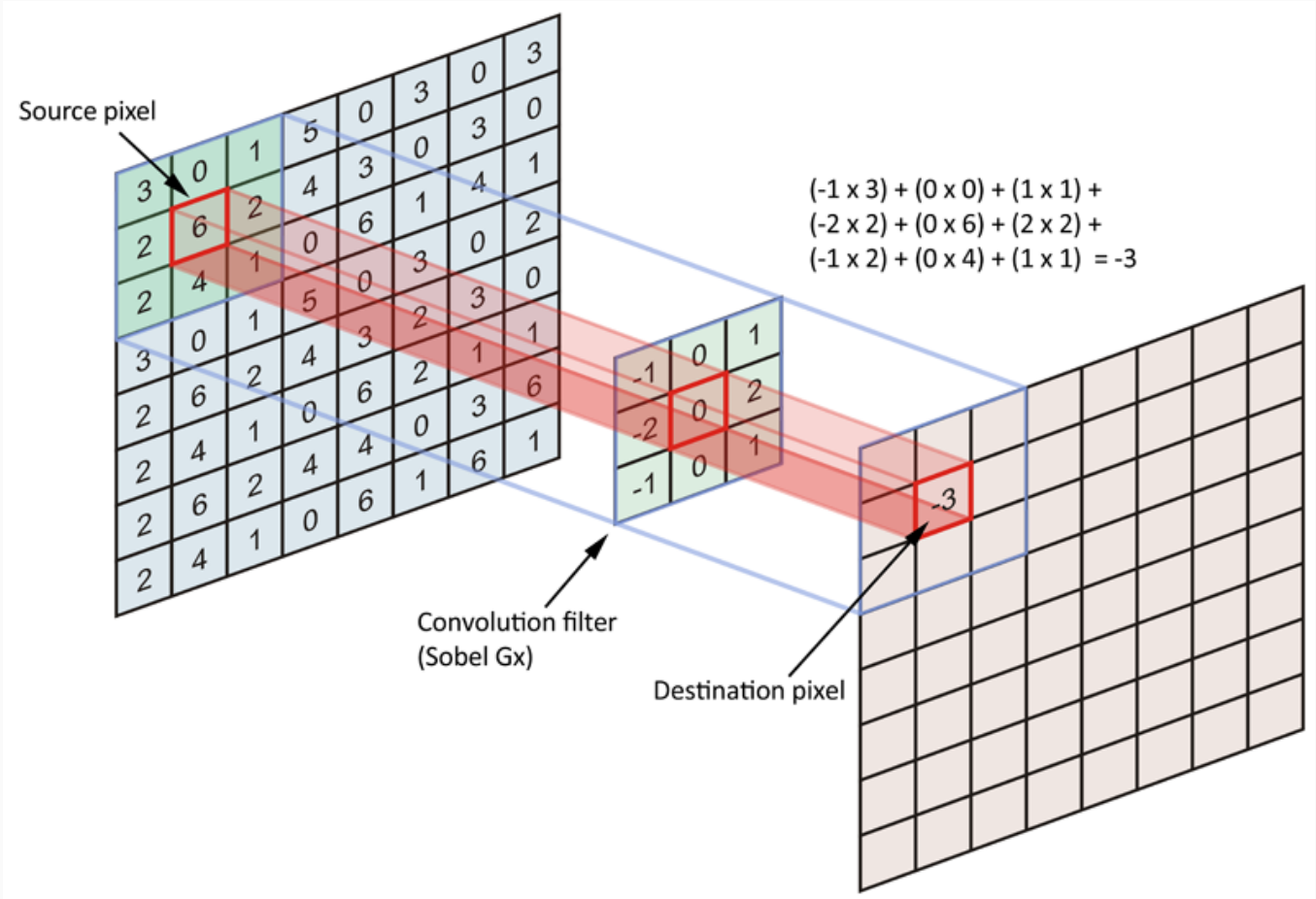
slides+notebook:`https://github.com/nansencenter/nersc_ml_course`

Table of contents

What is convolutional layer?

Convolution filter: scheme

"The convolution operation, simply put, is combination of element-wise product of two matrices."

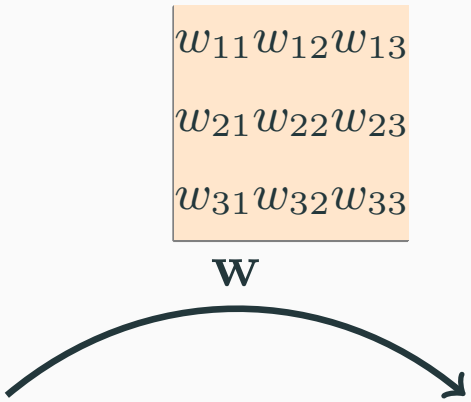


Images from Dynamic Stardust

Convolution filter: formula

X: an image

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}



h: first feature

h_{11}	h_{12}	h_{13}	h_{14}
h_{21}	h_{22}	h_{23}	h_{24}
h_{31}	h_{32}	h_{33}	h_{34}
h_{41}	h_{42}	h_{43}	h_{44}

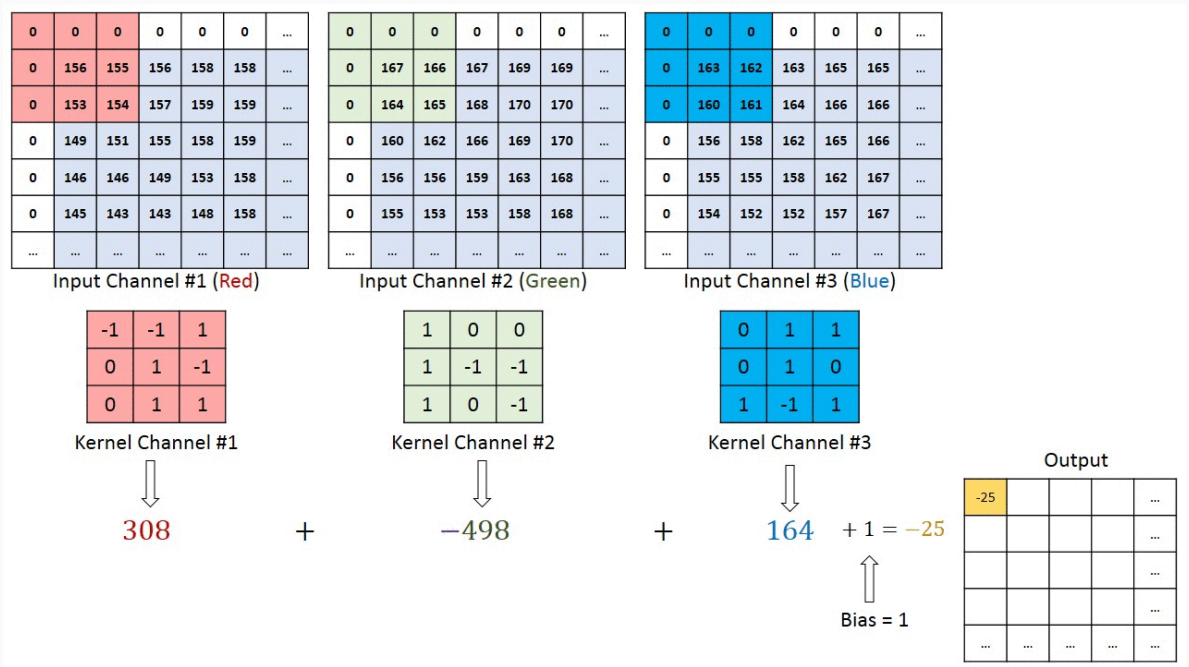
Perform a standard convolution

$$h_{i,j} = \sum_{k=1}^{3} \sum_{l=1}^{3} x_{i+k-1,j+l-1} \cdot w_{k,l}$$

Convolution filter of 3D input

Convolution of input 2D matrix with several bands is a sum of convolutions of each band $\equiv$ convolution of 3D matrix with 3D filter but only in 2 dimensions:

Input image: $7 \times 7 \times 3$
Filter: $3 \times 3 \times 3$
Output image: $5 \times 5 \times 1$



Images from Sumit Saha

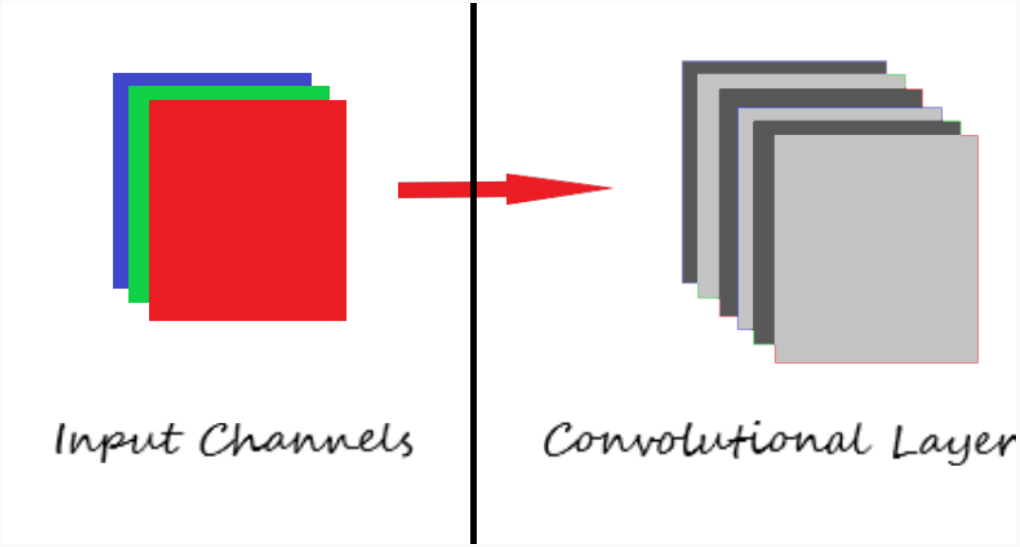
Main parameters of a convolutional layer

- Size of the filter K

Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p

  A convolutional layer is composed of p convolutions (size of layer) extracting p features from the data.

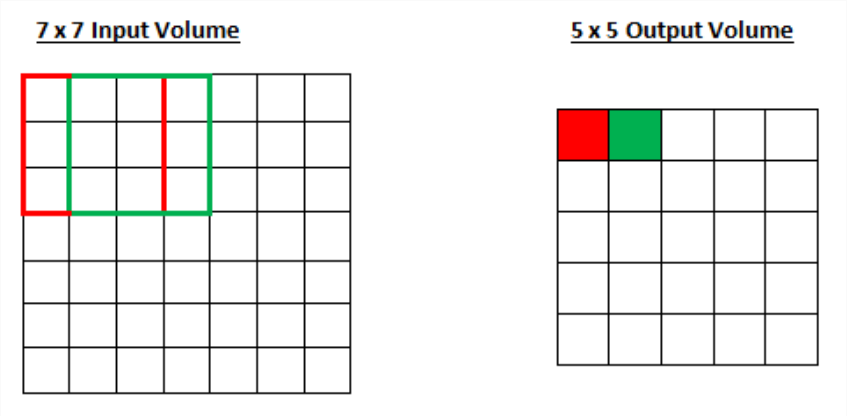


$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

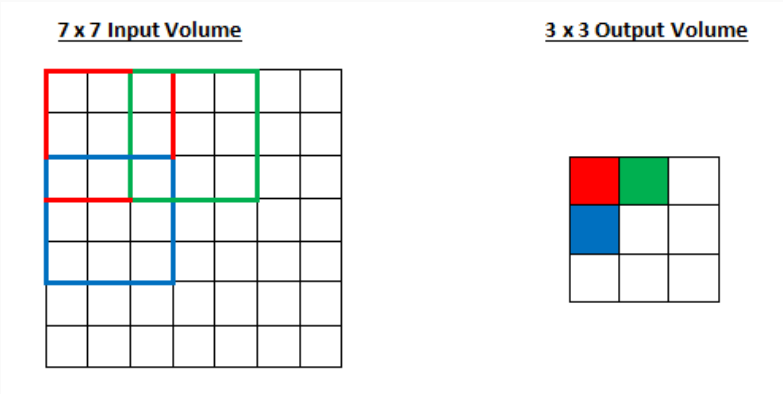
Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p
- Strides S

$S = 1$



$S = 2$

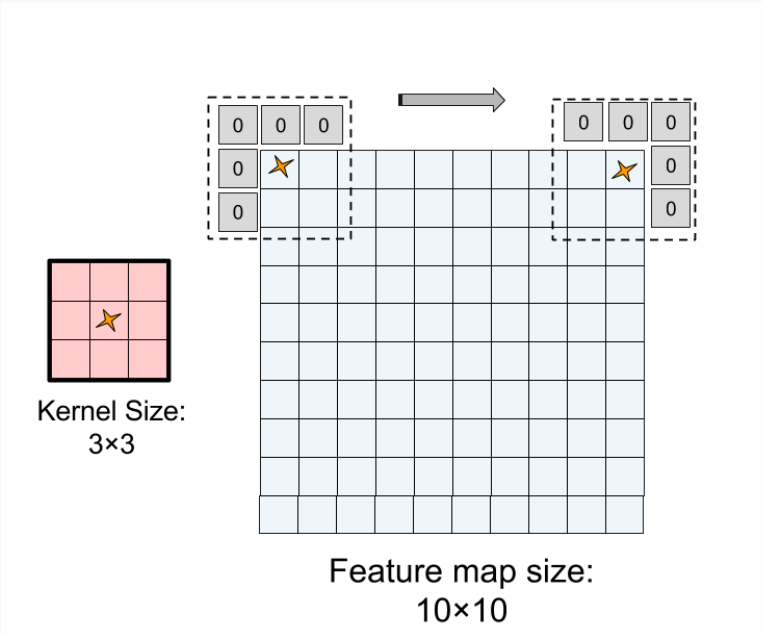


$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

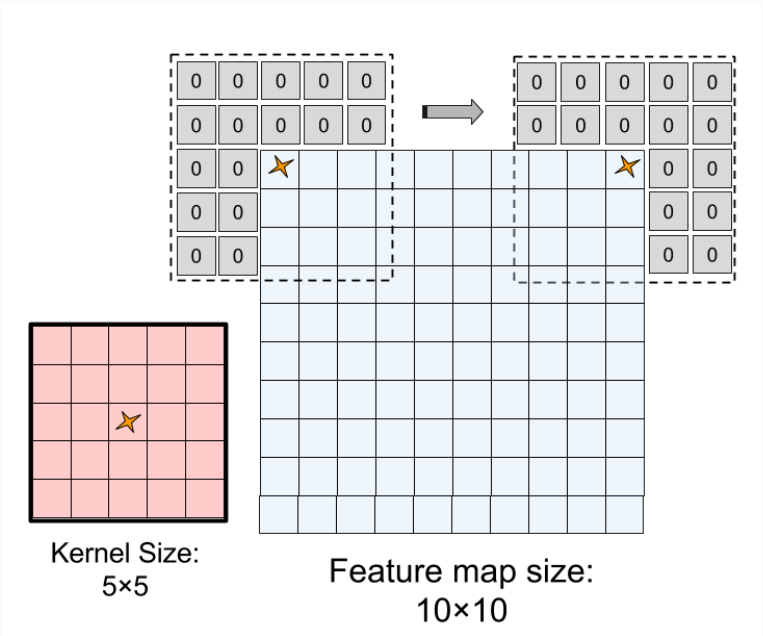
Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p
- Strides S
- Padding P

$P = 1$



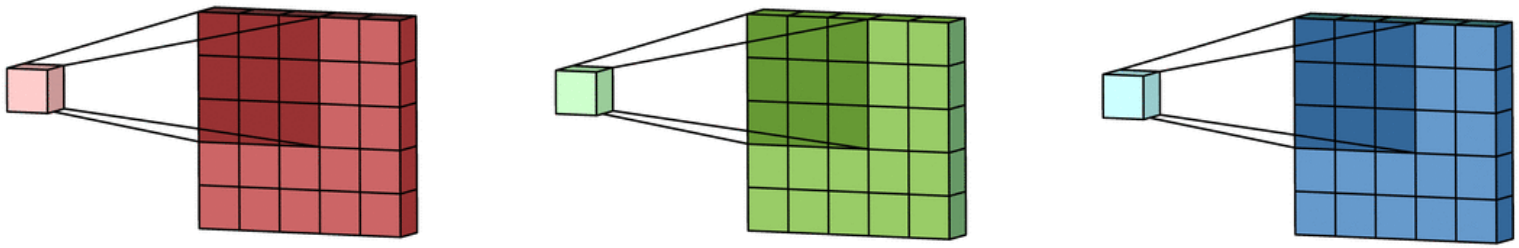
$P = 2$



$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

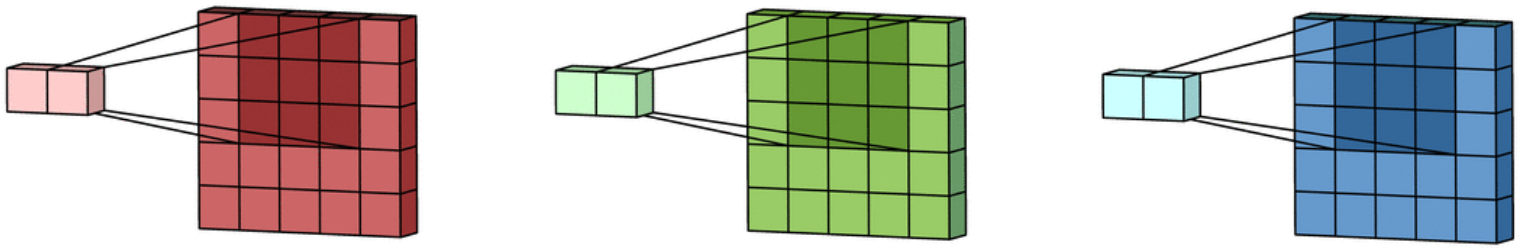
Summary of Convolutional layer steps

1. Convolution



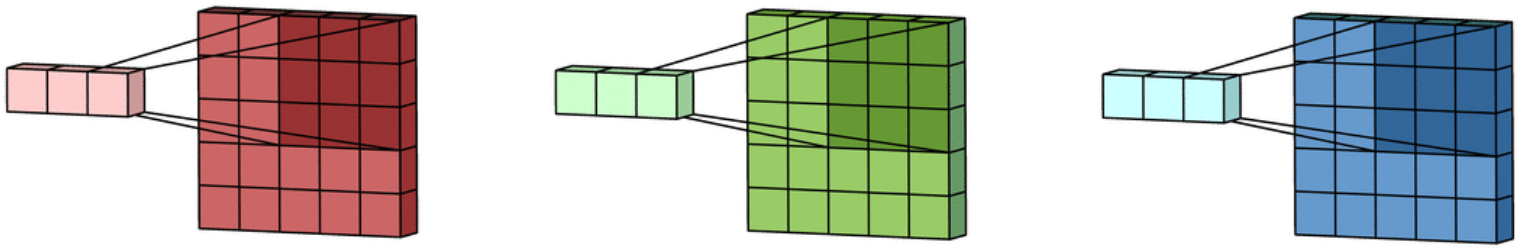
Summary of Convolutional layer steps

1. Convolution



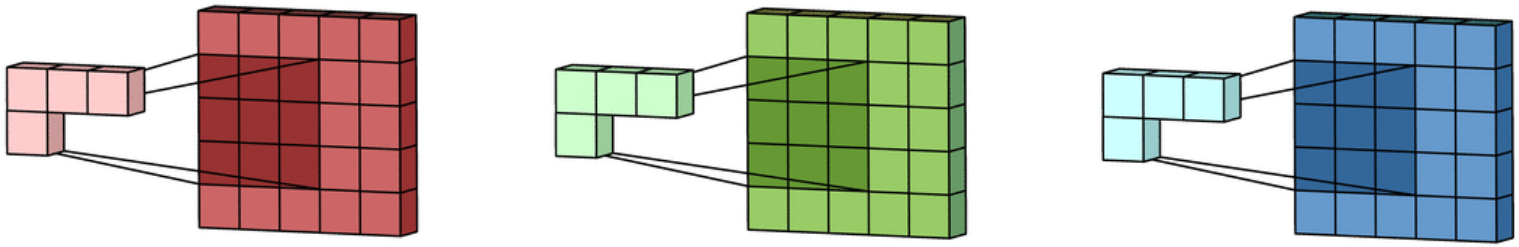
Summary of Convolutional layer steps

1. Convolution



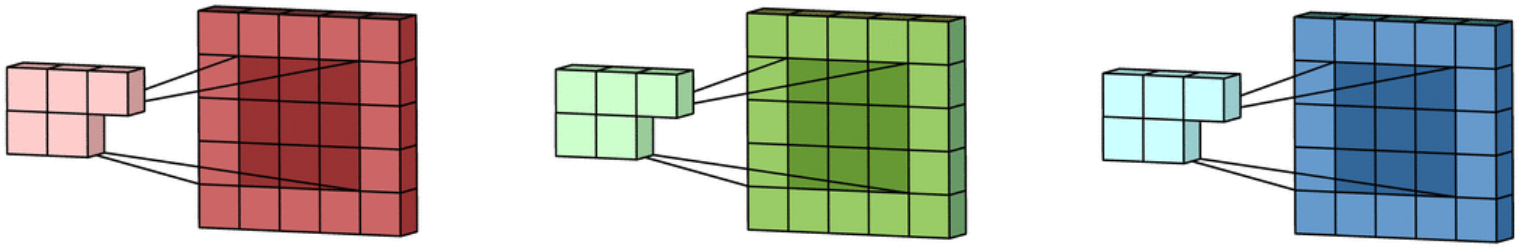
Summary of Convolutional layer steps

1. Convolution



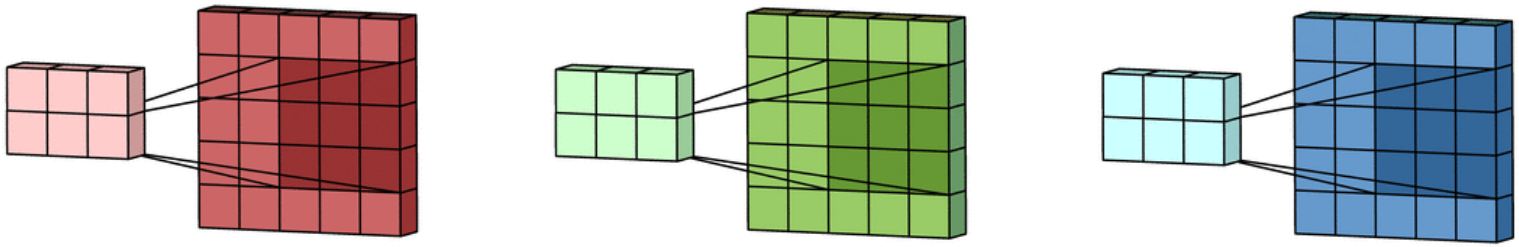
Summary of Convolutional layer steps

1. Convolution



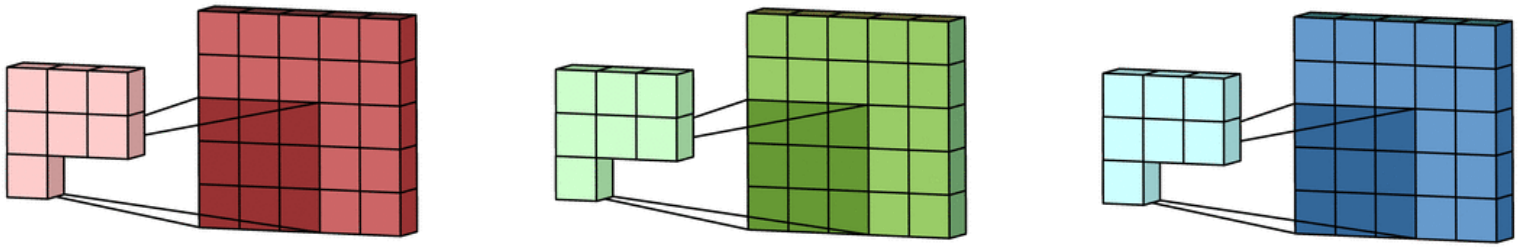
Summary of Convolutional layer steps

1. Convolution



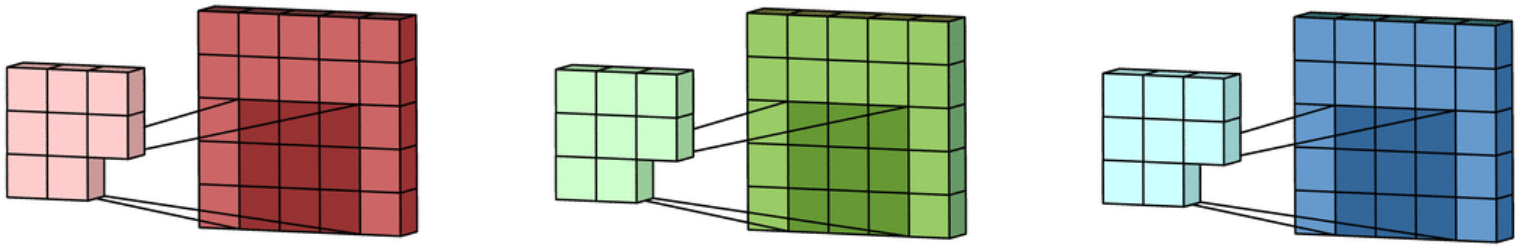
Summary of Convolutional layer steps

1. Convolution



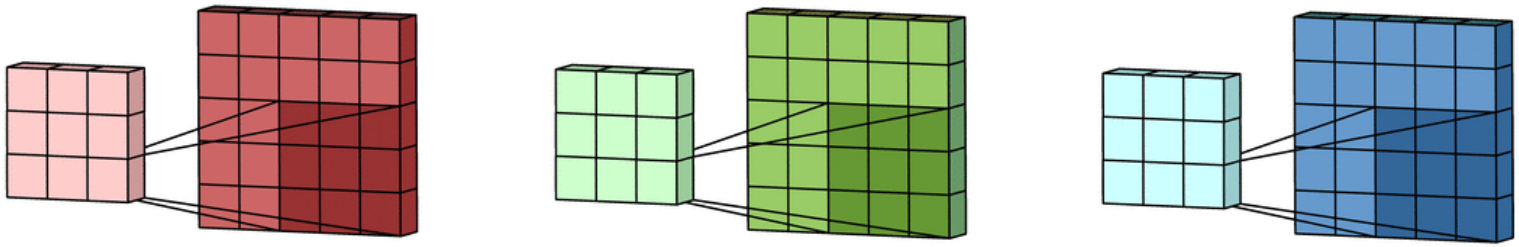
Summary of Convolutional layer steps

1. Convolution



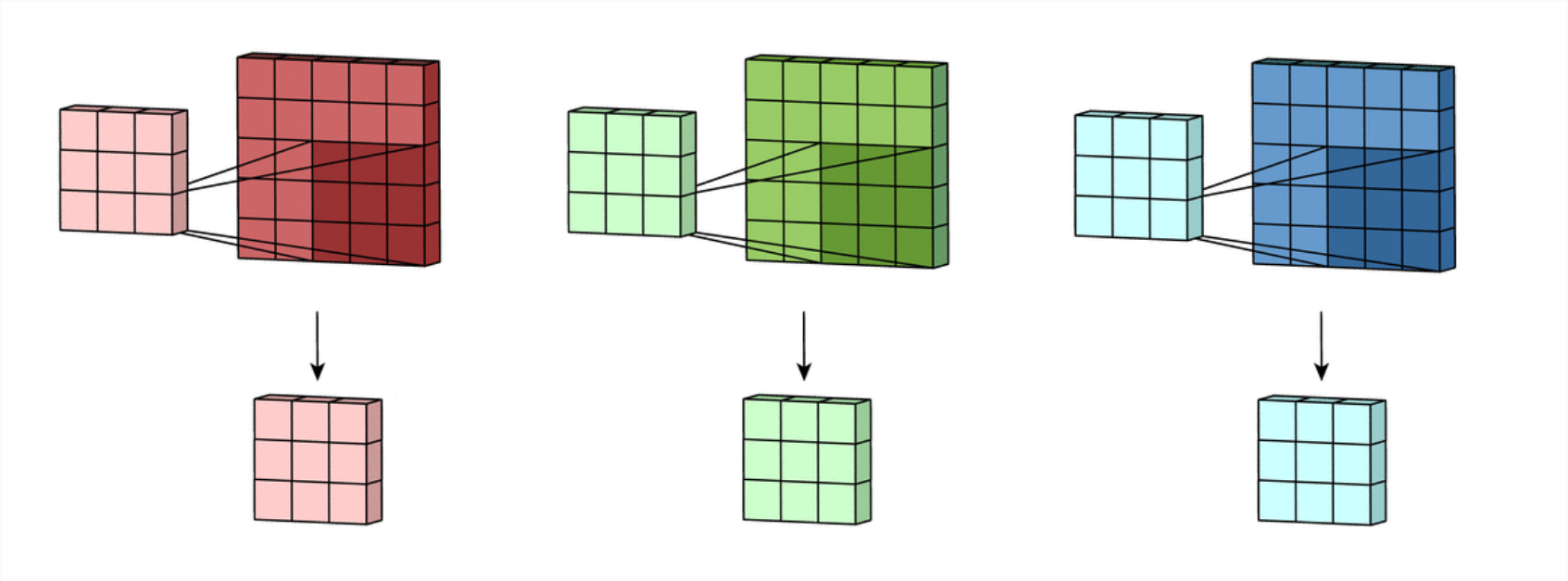
Summary of Convolutional layer steps

1. Convolution



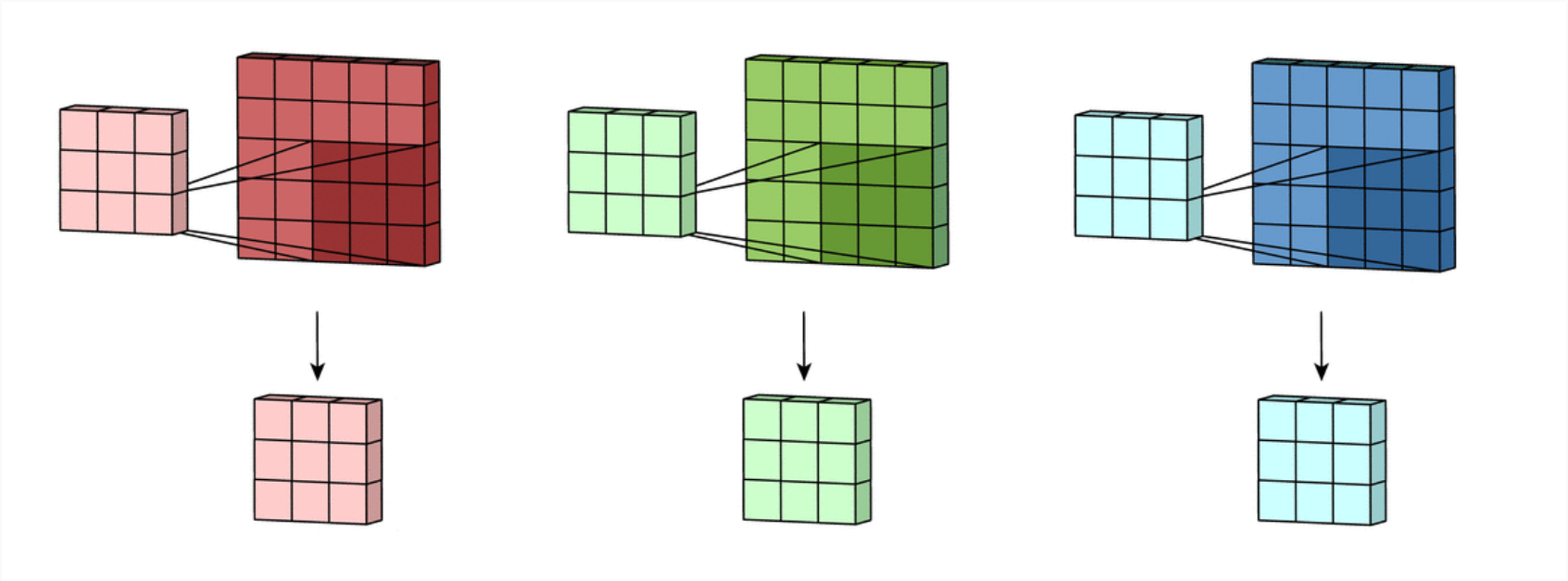
Summary of Convolutional layer steps

1. Convolution



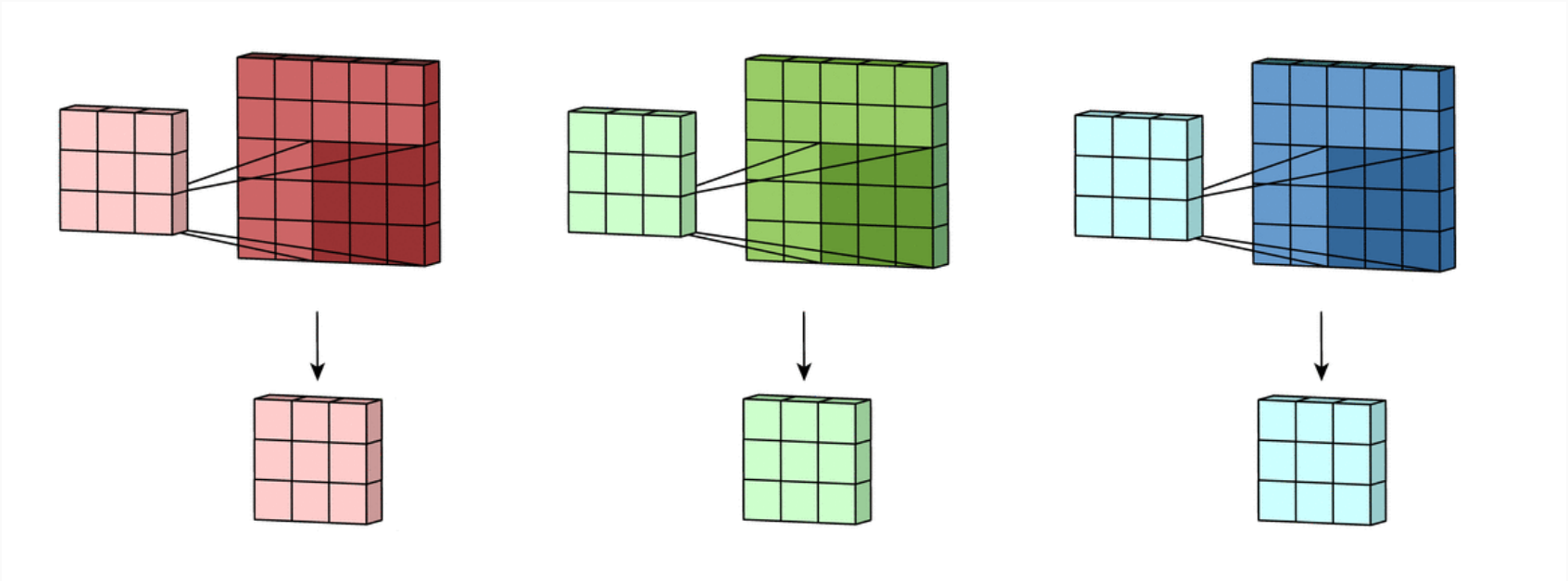
Summary of Convolutional layer steps

1. Convolution

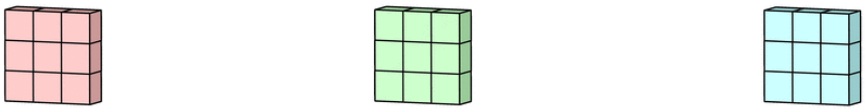


Summary of Convolutional layer steps

1. Convolution

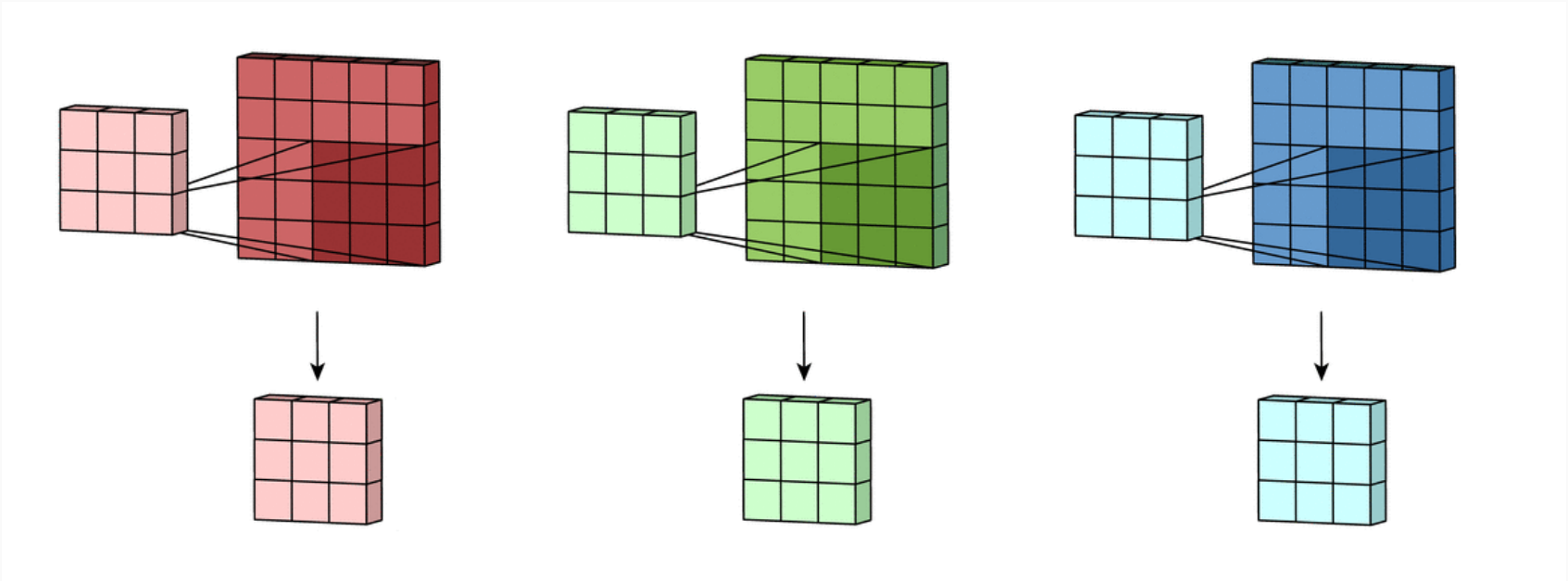


2. Addition

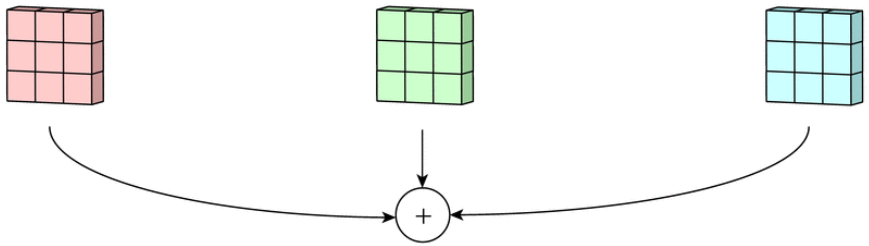


Summary of Convolutional layer steps

1. Convolution

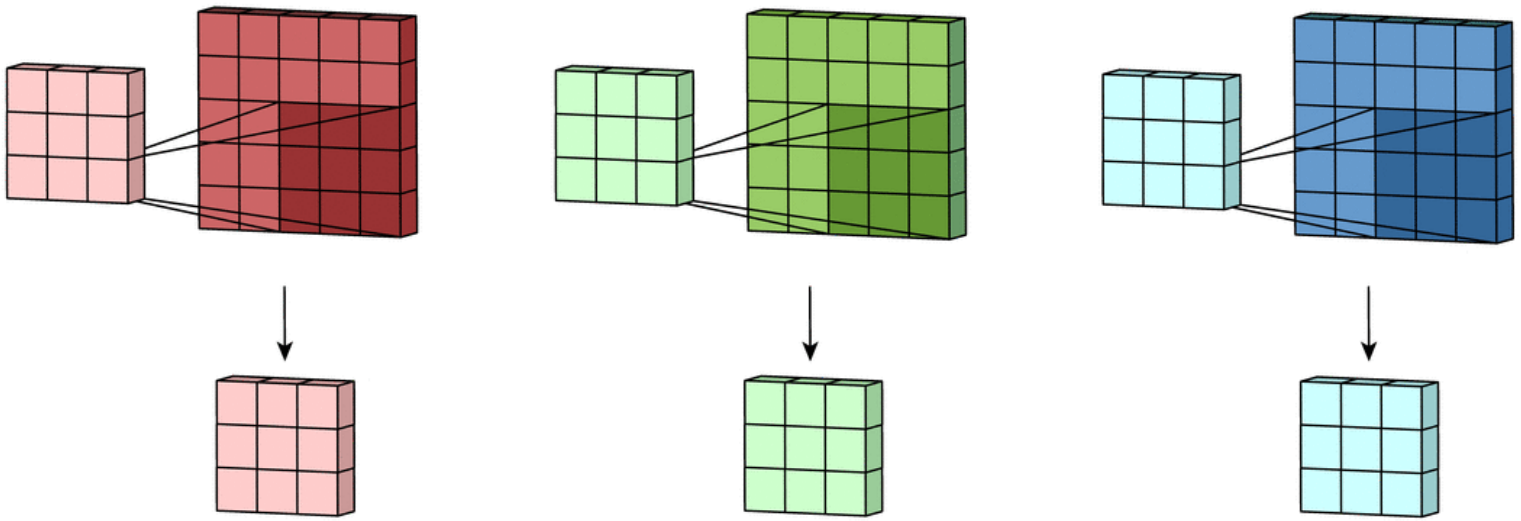


2. Addition

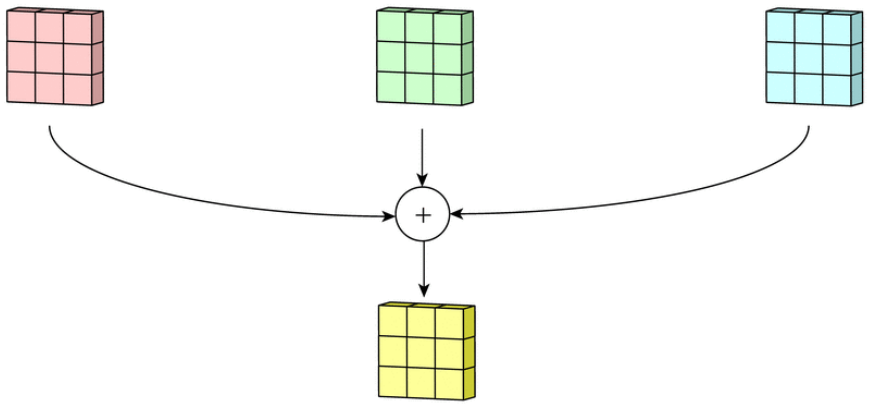


Summary of Convolutional layer steps

1. Convolution

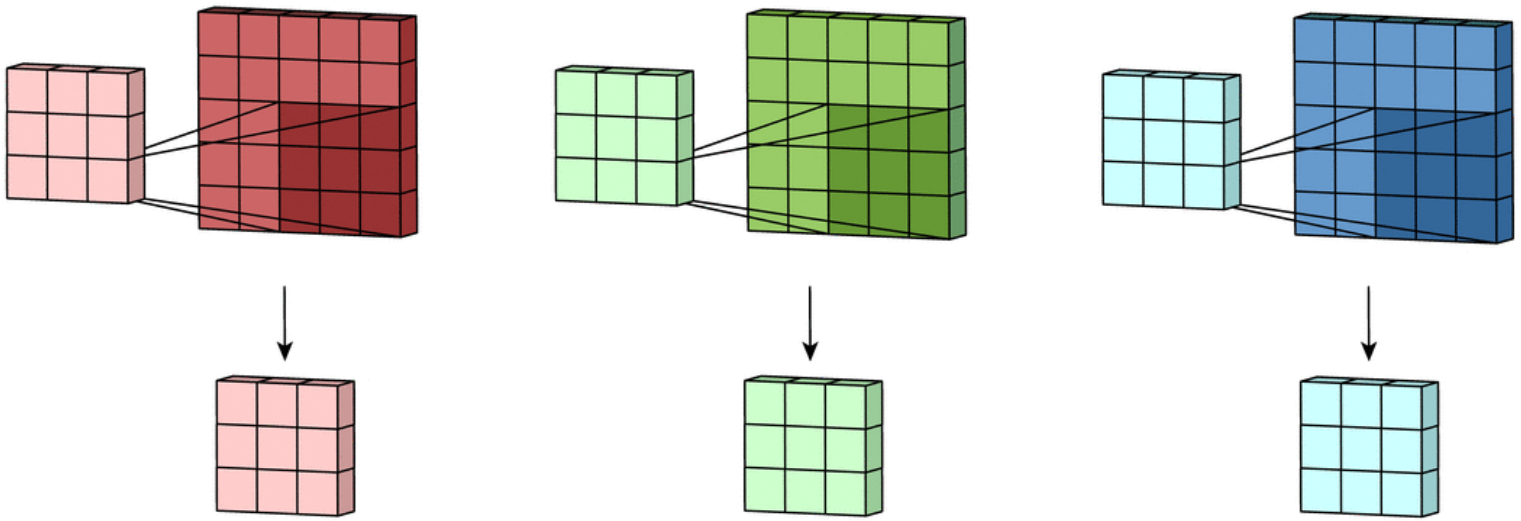


2. Addition

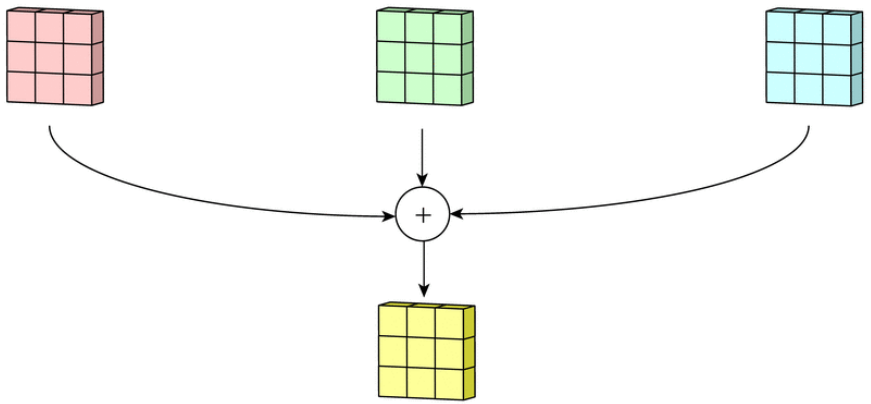


Summary of Convolutional layer steps

1. Convolution

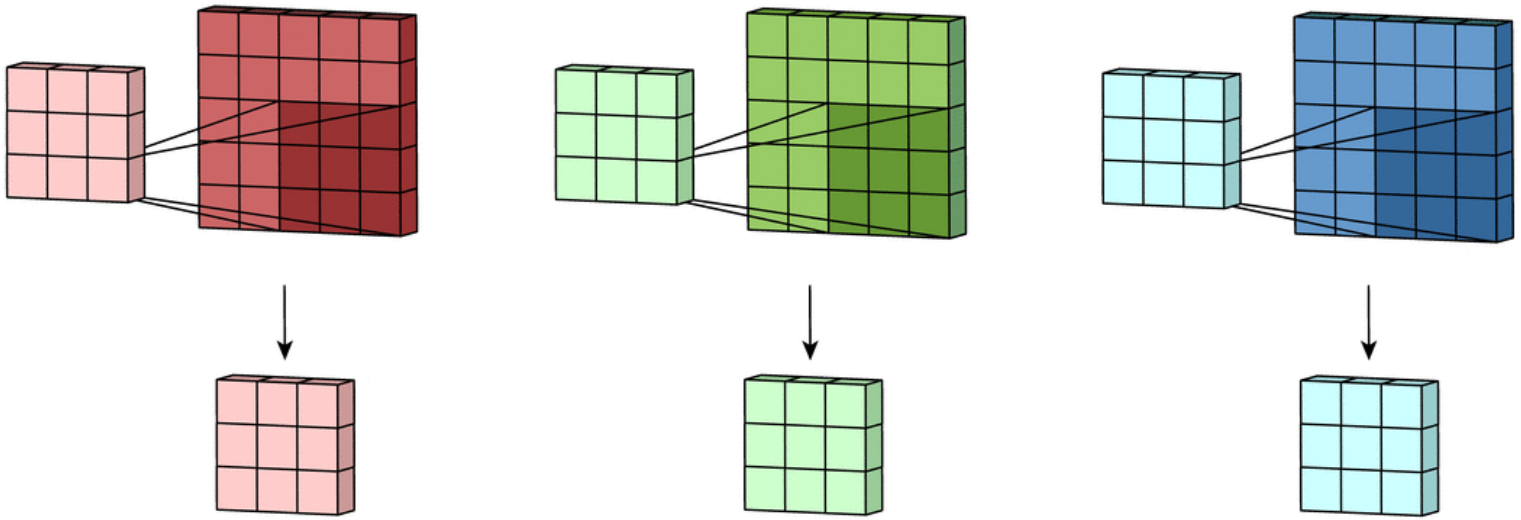


2. Addition

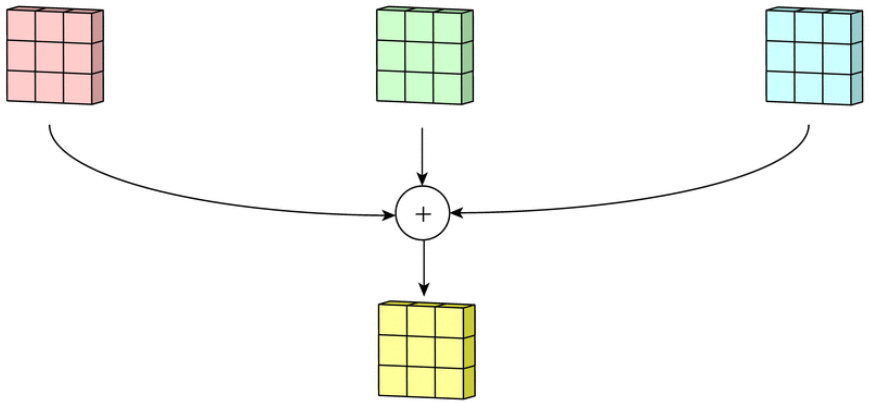


Summary of Convolutional layer steps

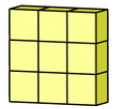
1. Convolution



2. Addition

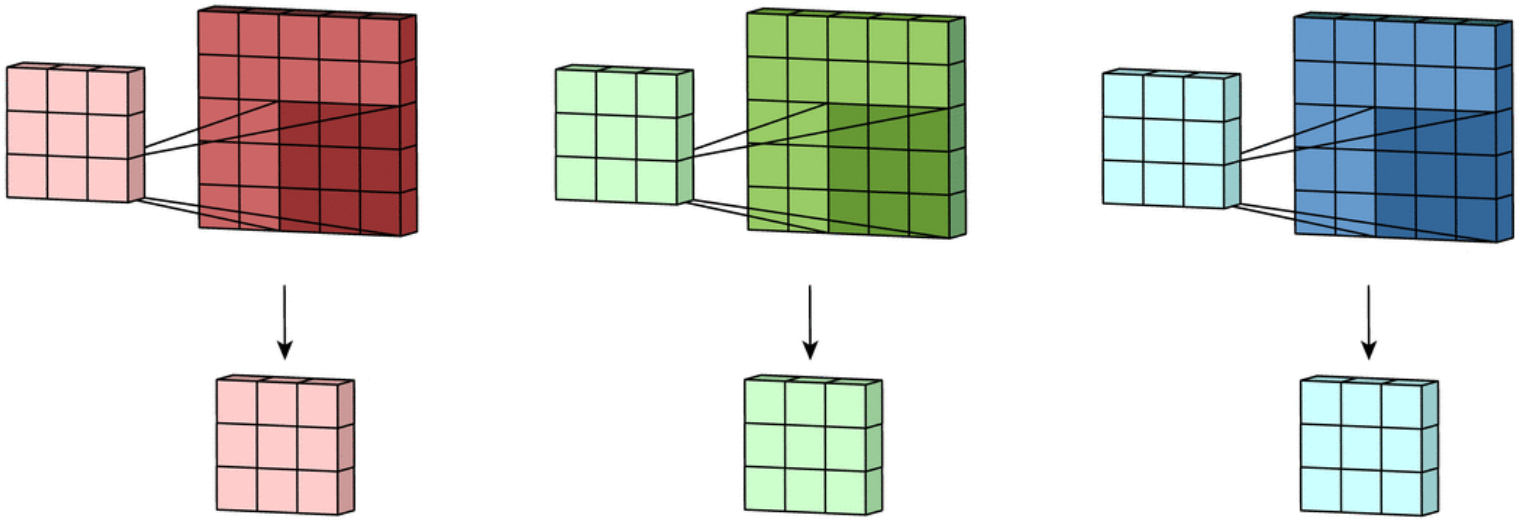


3. Bias

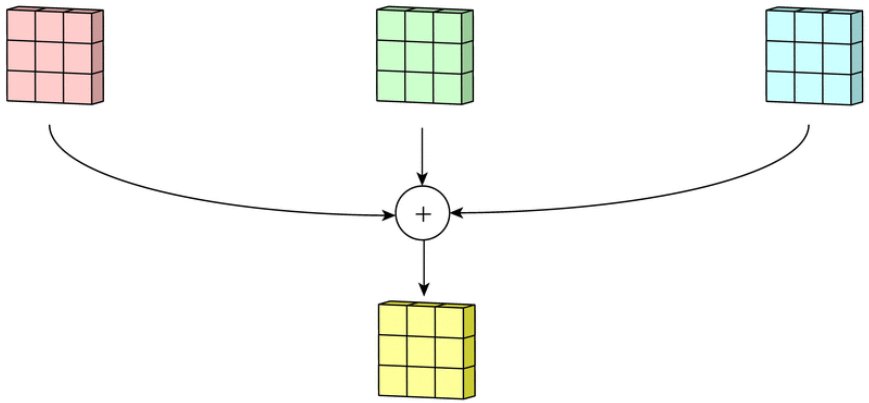


Summary of Convolutional layer steps

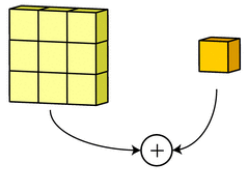
1. Convolution



2. Addition

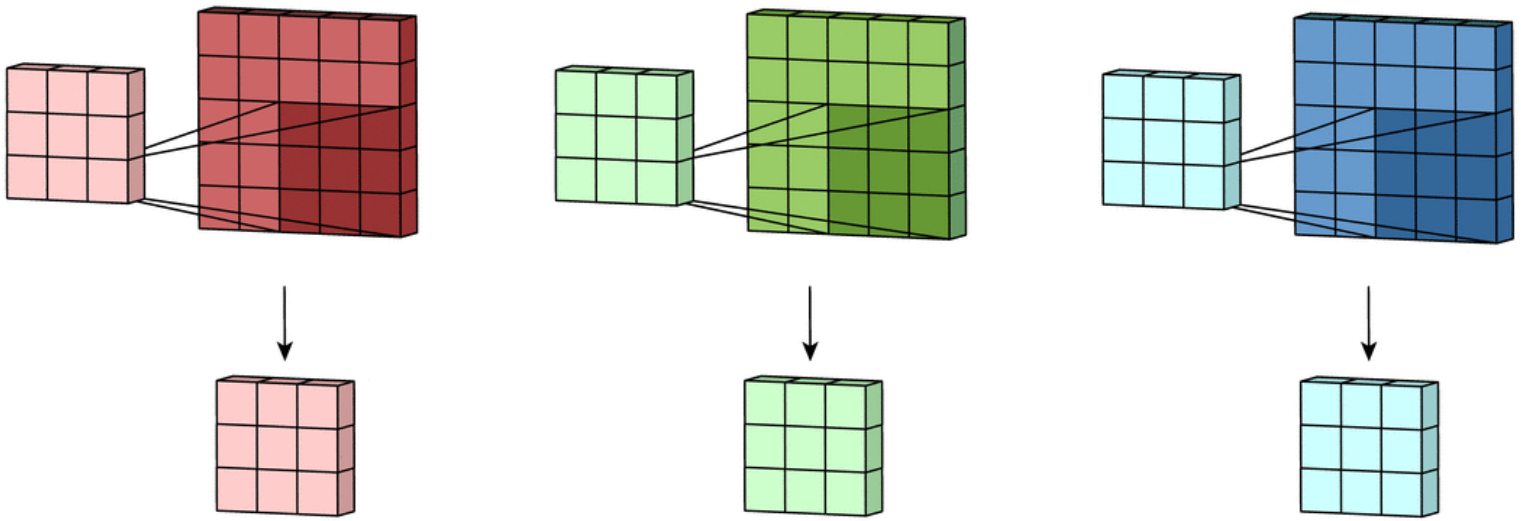


3. Bias

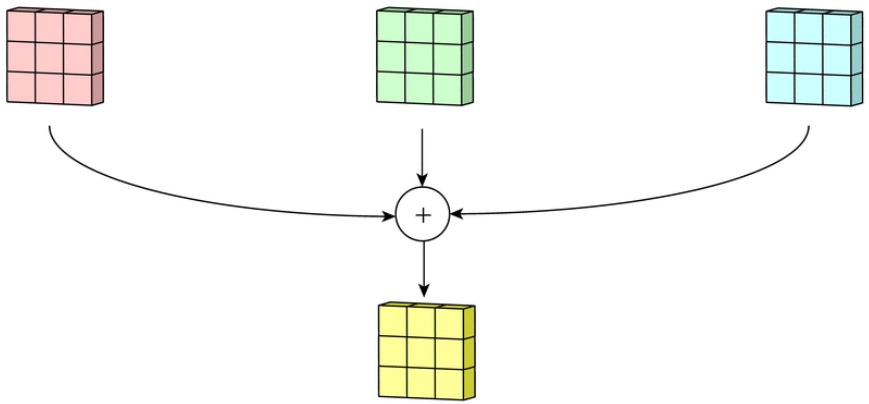


Summary of Convolutional layer steps

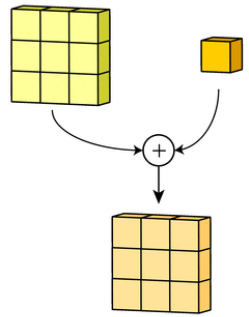
1. Convolution



2. Addition



3. Bias

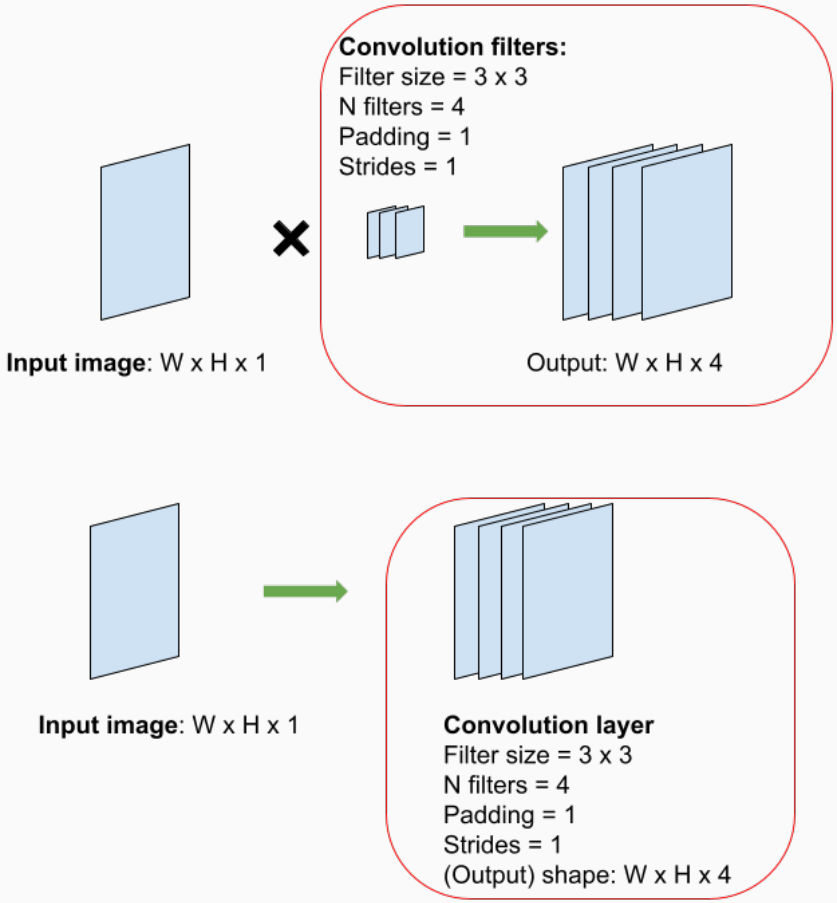


Remarks on Convolutional layers

- Convolutional layers are acting locally on the image (But you can still use large scale information by adding more layers)
- Convolutions are invariant by translation (the weights do not depend on the location on the image).
- They can handle images of different sizes.

Convolution layer notation

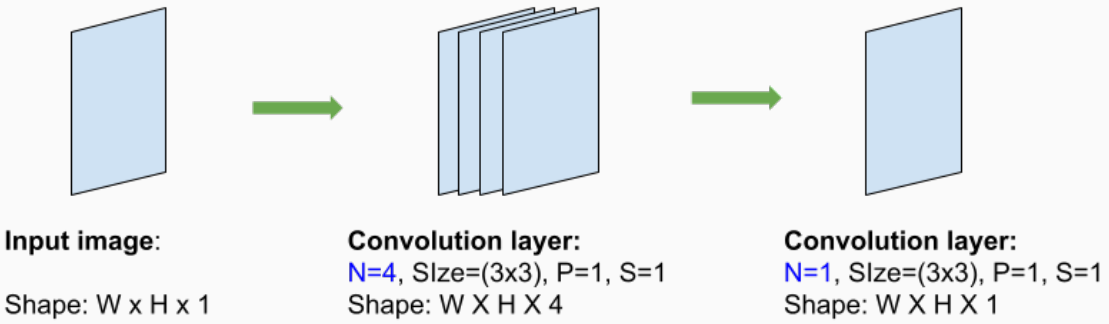
The operation of convolution, the number of filters and their size, stride and padding, and the output shape is often shown on schemes as just one "Convolution layer":

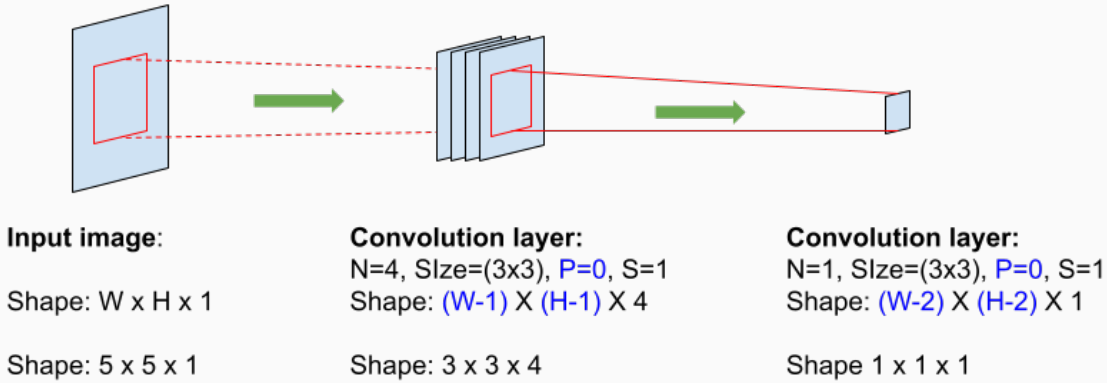


Convolutional neural networks

Combining several convolution layers

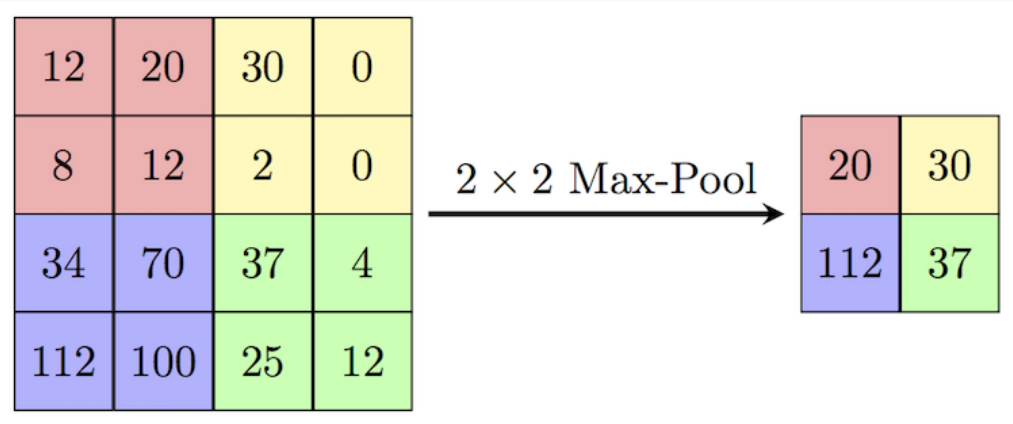
Several convolution layers can be stacked to created a "deep" network. The shape of input can be preserved or reduced.



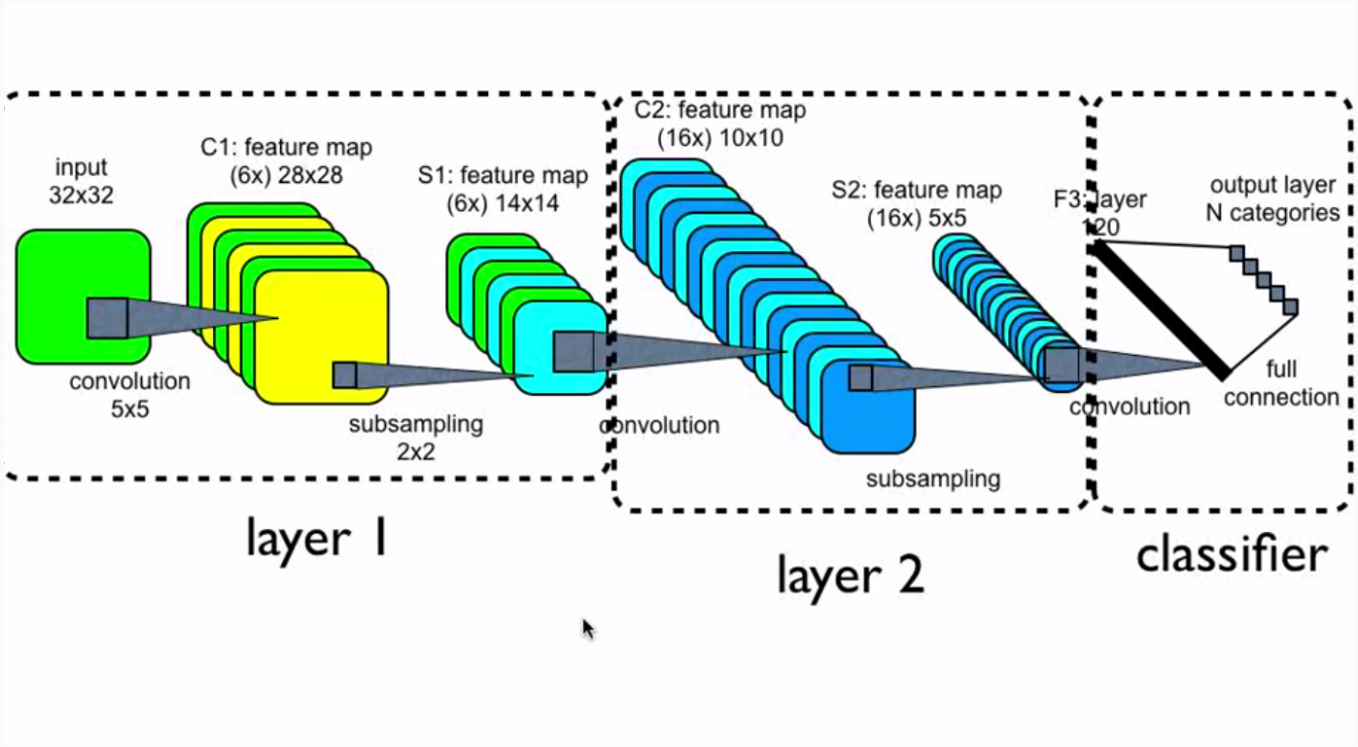


Max-Pooling

In order to reduce the size of the feature space (to enhance the gradients), but to keep information from the entire input image a common operation to perform is "max-pooling".



A traditional CNN architecture



Such network can be used for "image classification" tasks (is it a dog or a cat on the input image?) or "image regression" tasks (how much is the fish on the image?)

Example of AlexNet

AlexNet is the first Deep architecture used on ImageNet challenge in 2012 and achieved an error of 15.3% (10% better than the previous best classifier). The paper was cited more than 34,000 times.

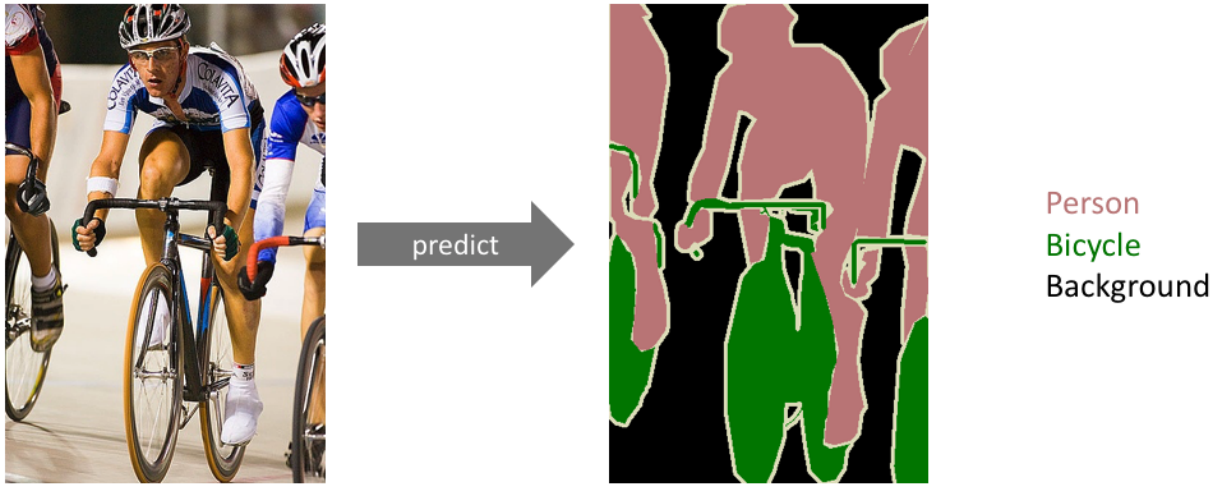
Alex Krizhevsky and Geoffrey E Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Neural Information Processing Systems (2012), 1–9.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Image segmentation

What is image segmentation?

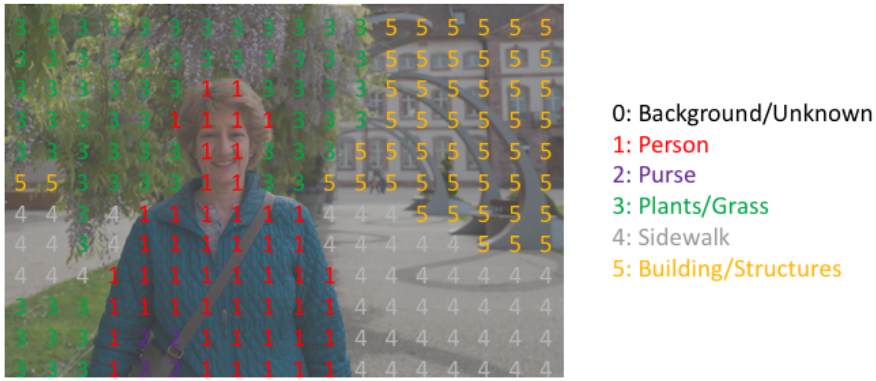
When a 2D output with labels should be produced the task is called "image segmentation":



Images from JEREMY JORDAN

Semantic labels

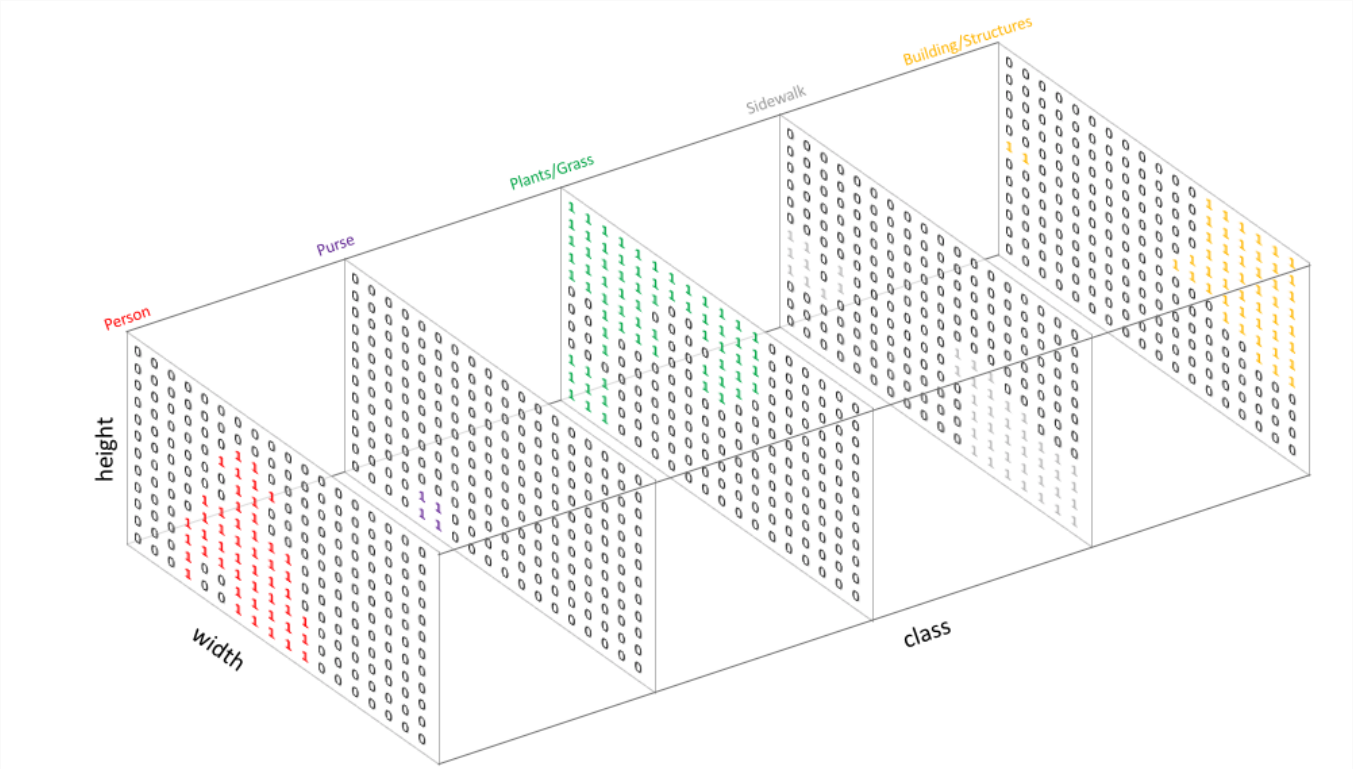
On a segmented image each pixel (or each patch with several pixels) is labeled with a number.



Images from JEREMY JORDAN

Image segmentation: one-hot-encoding

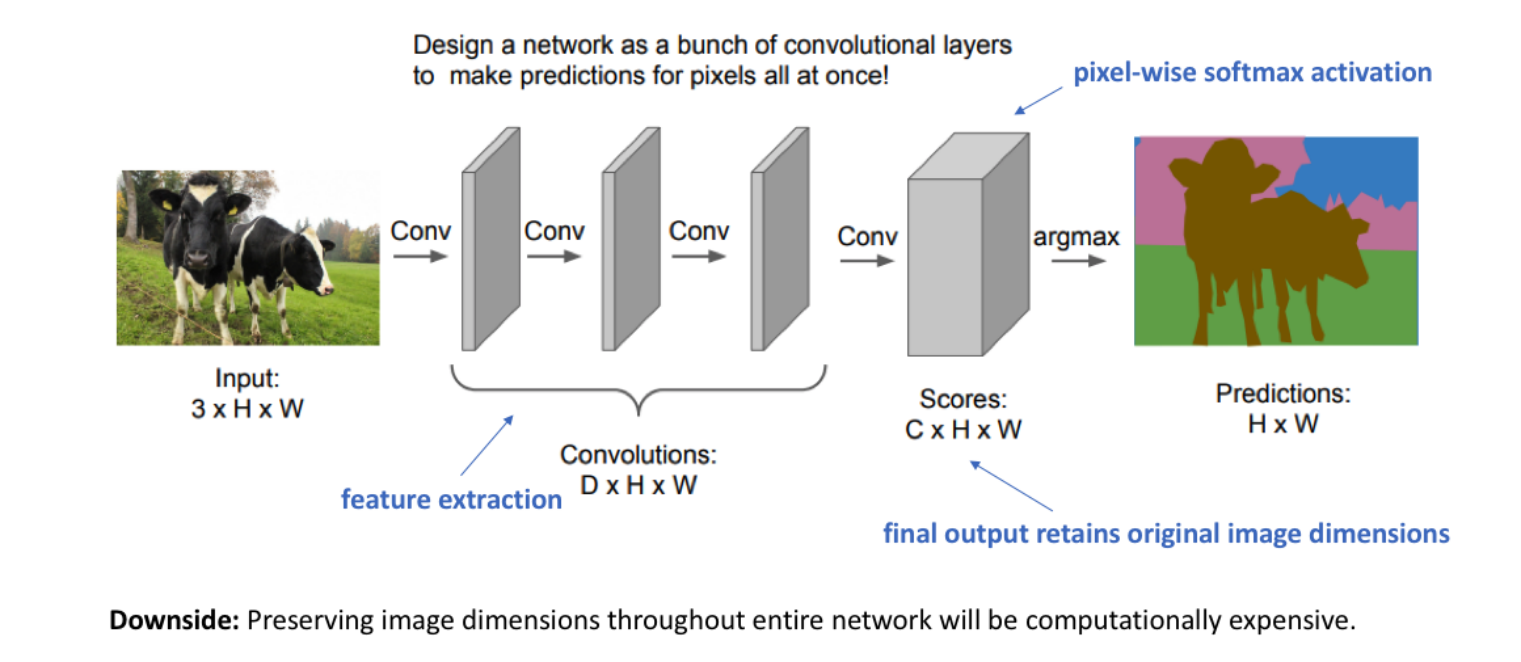
Operation of "on-hot encoding" replaces class digits with probability vectors.



Images from JEREMY JORDAN

Naive CNN for image segmentation

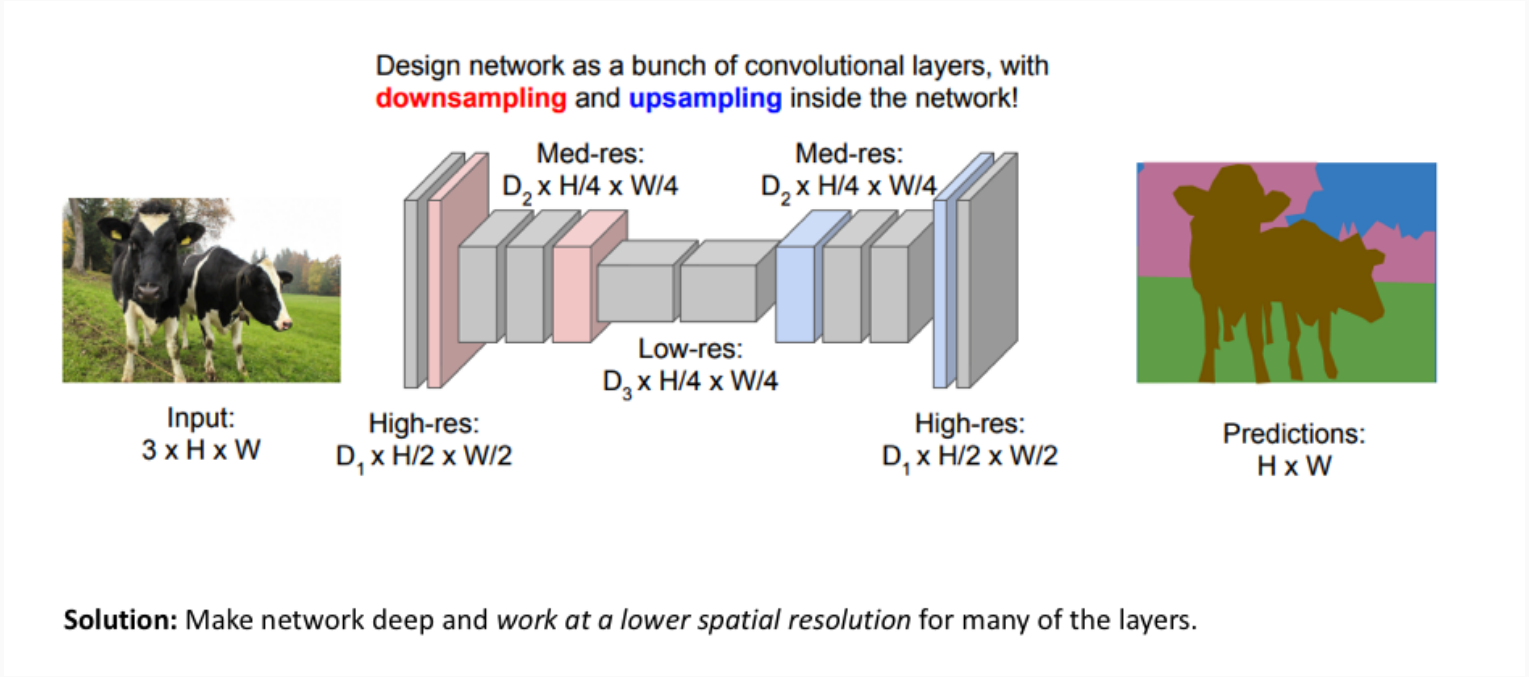
Size of convolution filters and padding is selected such that the input image size is preserved (padding='same').



Images from JEREMY JORDAN

Deep CNN for image segmentation

Another approach: convolution + DOWN-sampling + convolution + UP-sampling.

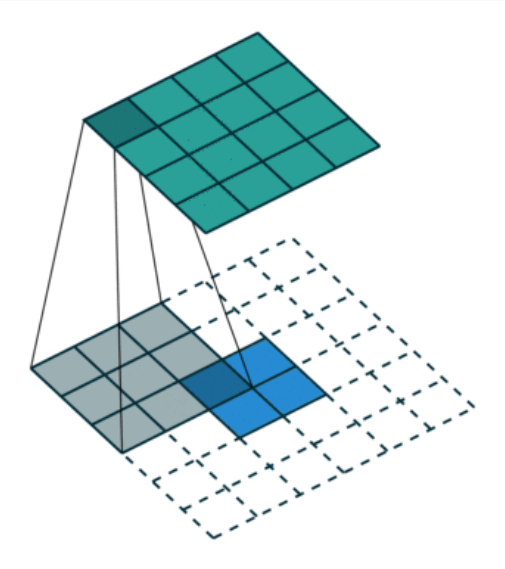


Images from JEREMY JORDAN

Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

Blue - input image, gray - filter, green - output image.

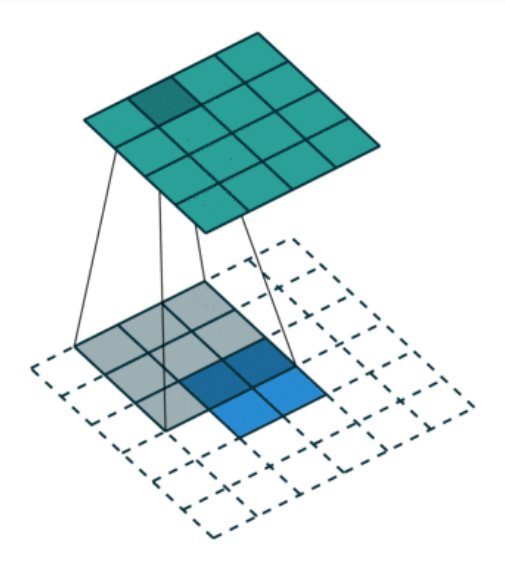


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

Blue - input image, gray - filter, green - output image.

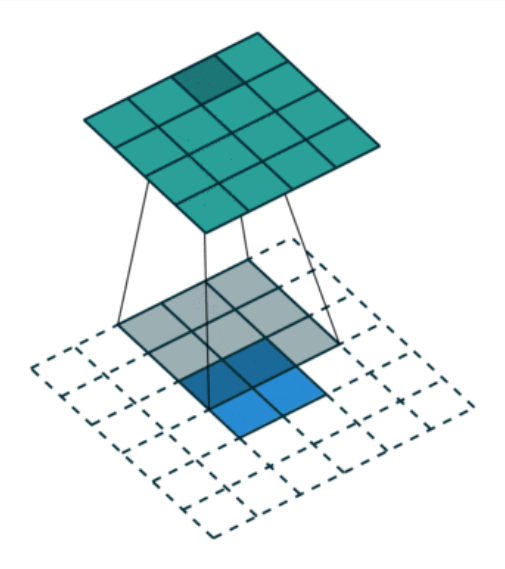


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

Blue - input image, gray - filter, green - output image.

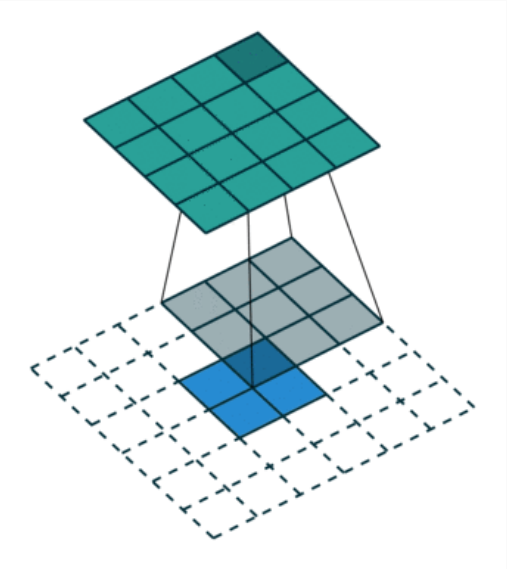


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

Blue - input image, gray - filter, green - output image.

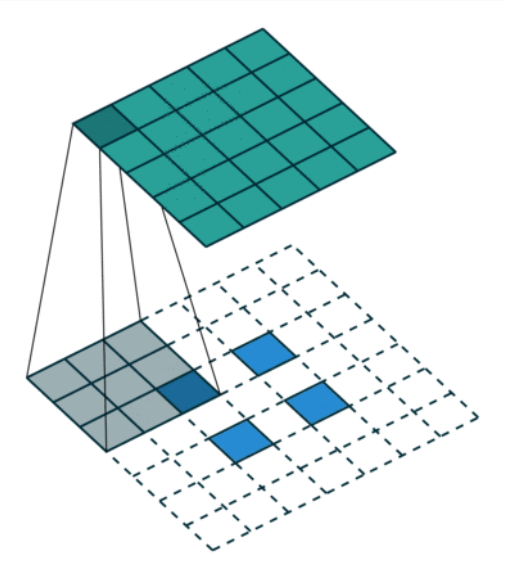


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

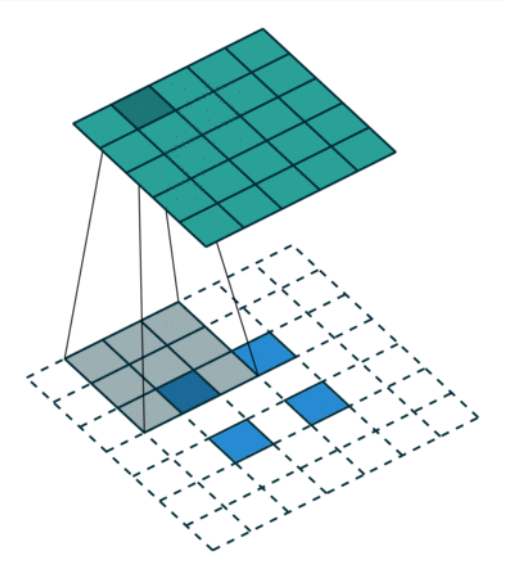


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

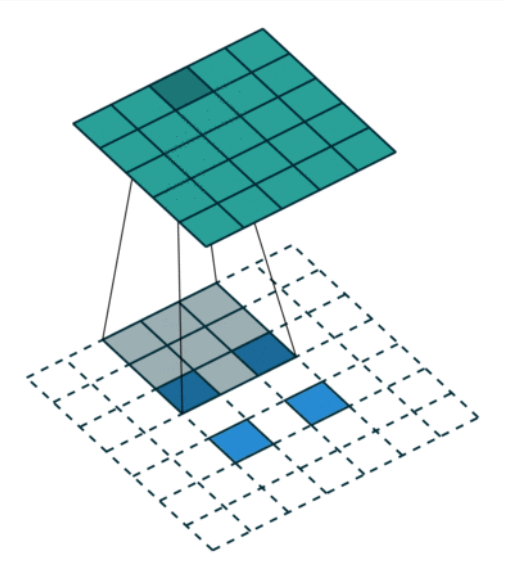


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

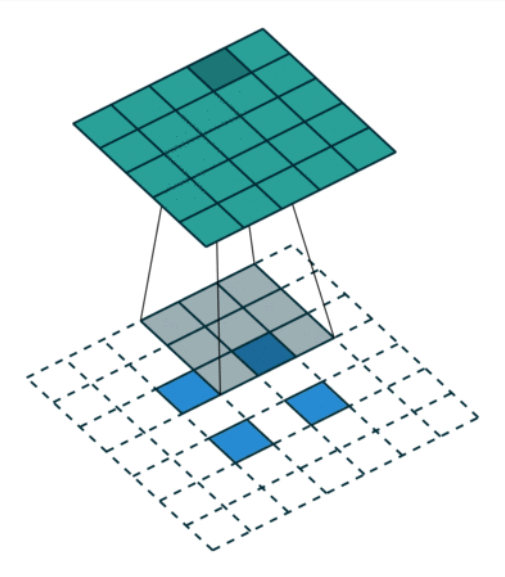


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

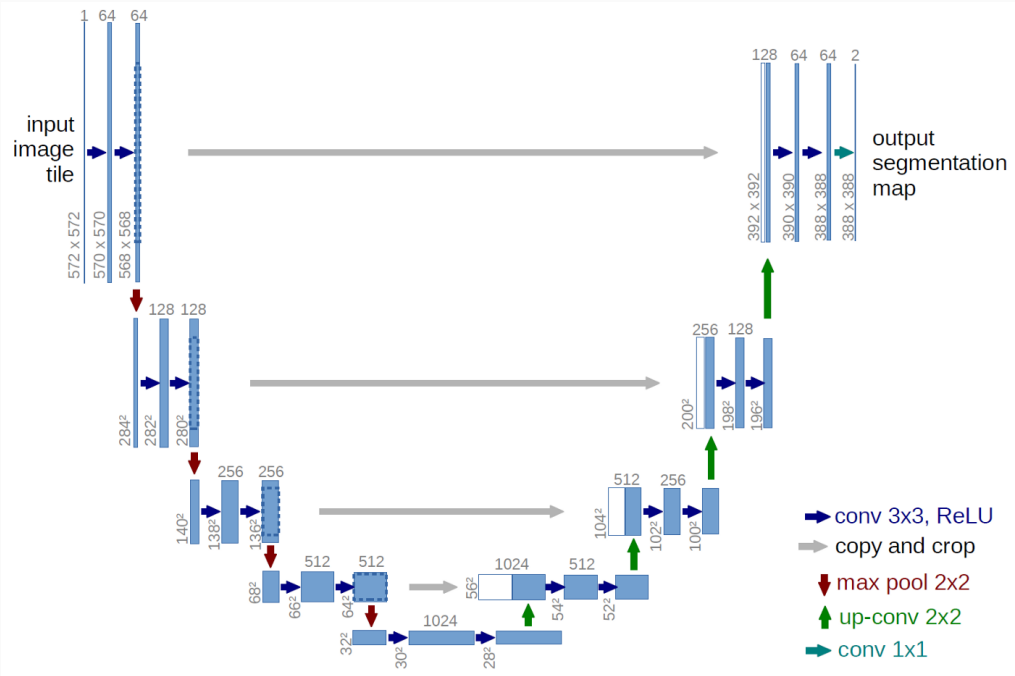


Images from fvisin Francesco

U-net architecture

U-net is very well known CNN architecture for semantic segmentation (34K citations).

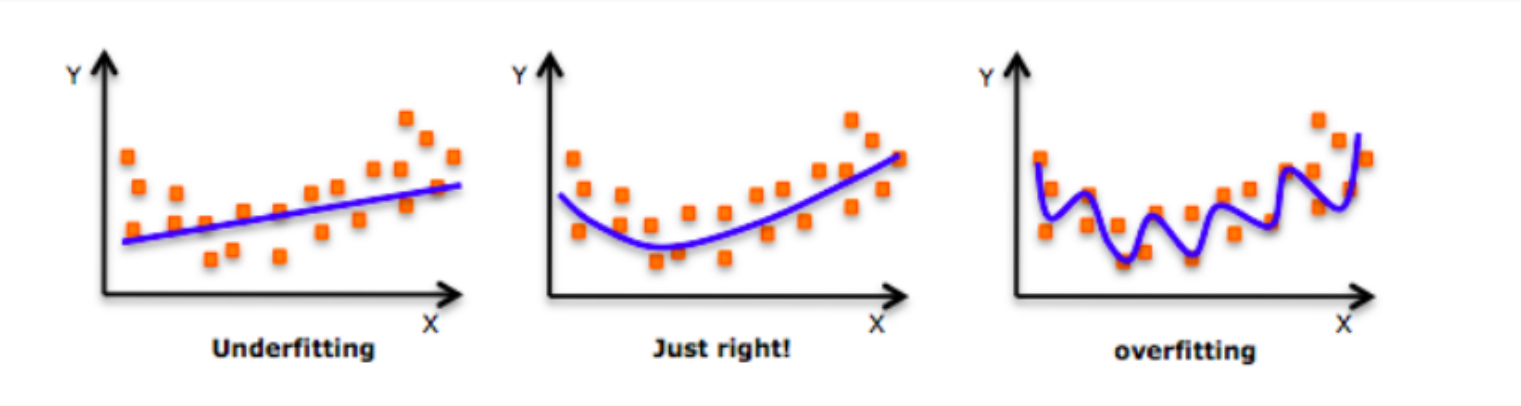
Olaf Ronneberger, Philipp Fischer and Thomas Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, arXiv, 2015.



Regularization of neural network training

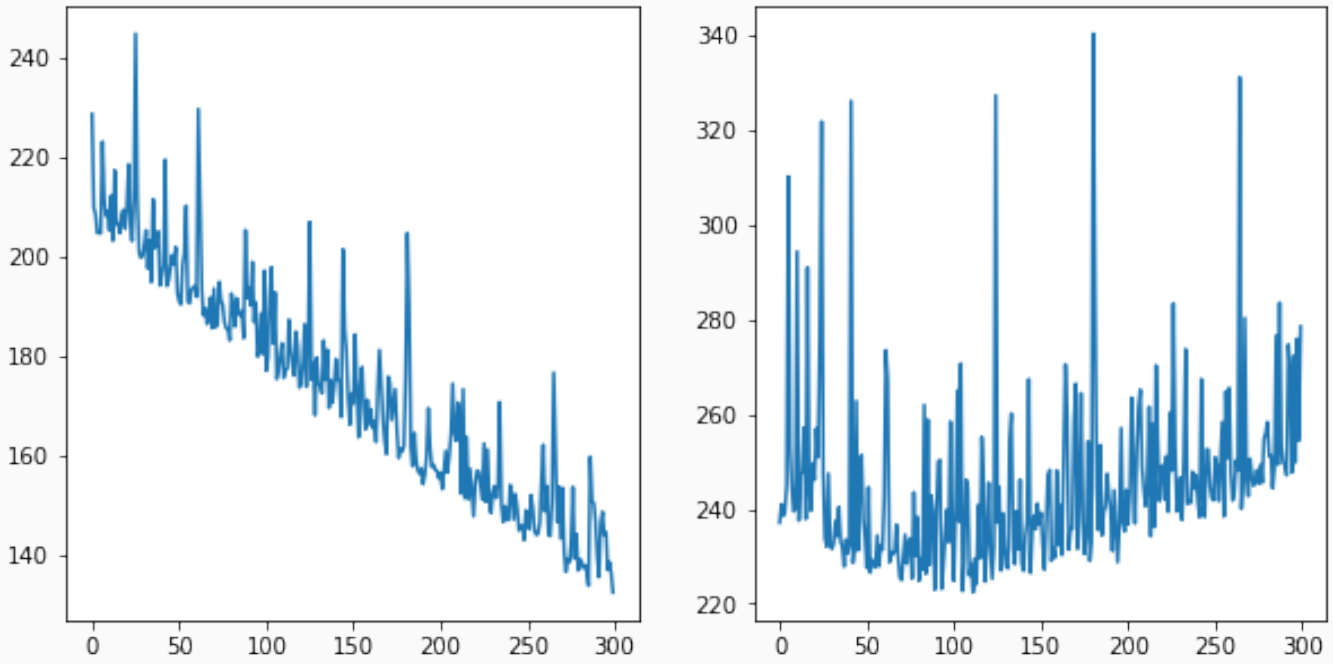
Under-fitting, Over-fitting

From the regression lecture we remember that we can under-fit or over-fit our model when we have too few or too many parameters.



Over-fitting of CNN

Training of a network (also CNN) is an iterative process. At each iteration we compute a train and a validation loss. Observing dynamics of train and validation loss can help detect over-fitting.



The figure shows dynamics of train loss (left) and validation loss (right).

L2-regularization

L2-regularization is a term in the loss calculation that penalizes not only error of prediction but also square of the model parameters. L2-regularization is used quite often as it reduces too high values of parameters.

$$L(x,y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$$

where $h_\theta x_i = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

L1-regularization

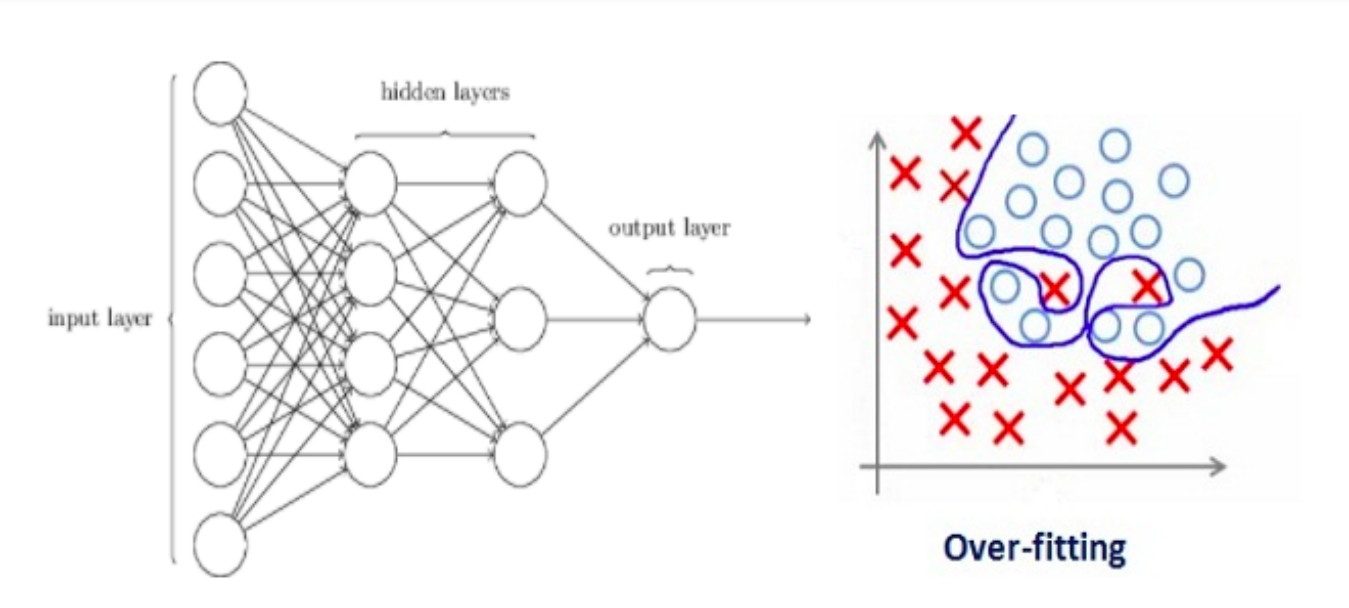
L1-regularization penalizes modulus of the model parameters.

L1-regularization is less famous as too many parameters may turn to zero.

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$

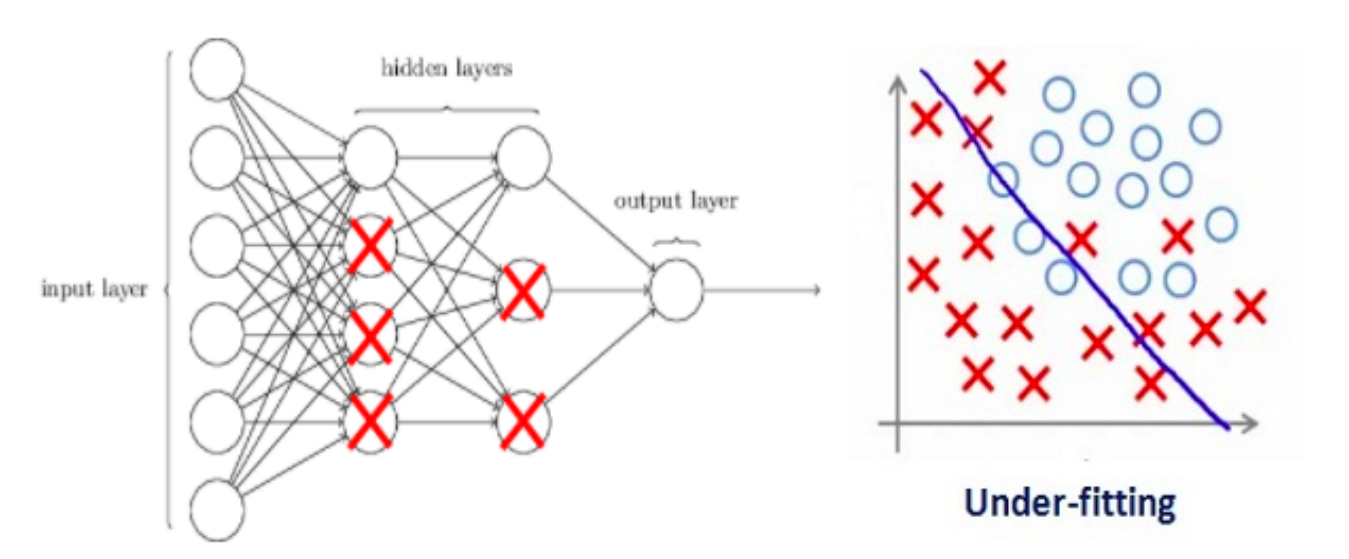
Too complex network

When the network is too complex (too many neurons and connections between them) the classification task can be over-fitted.



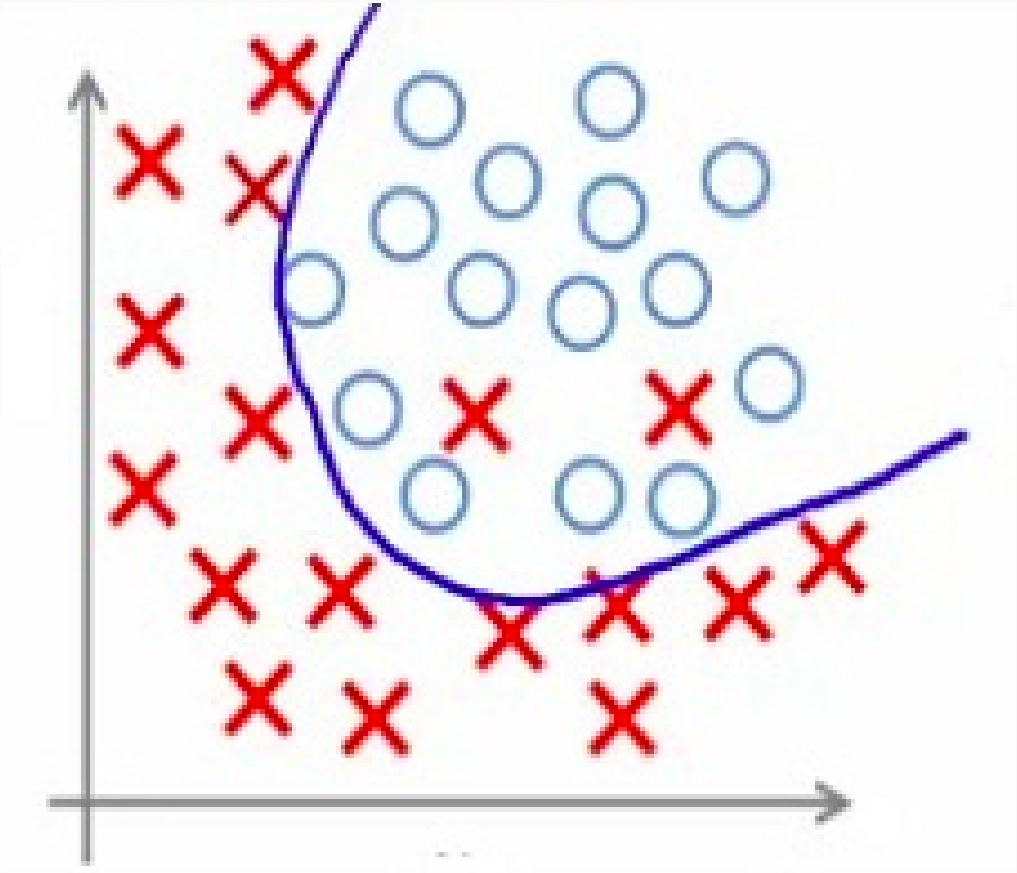
Too simple network

When the network is too simple (too few neurons and connections between them) the classification task can be under-fitted.



Optimal network?

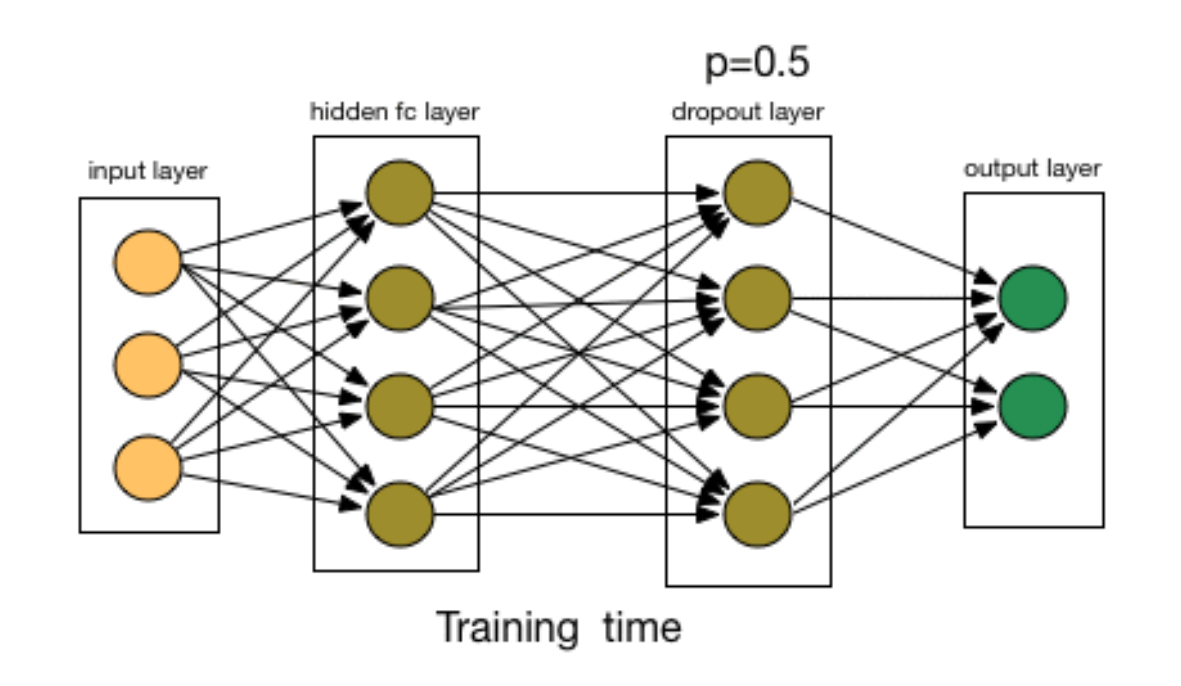
What is the optimal network for finding optimal solution and not over-fitting.



Appropriate-fitting

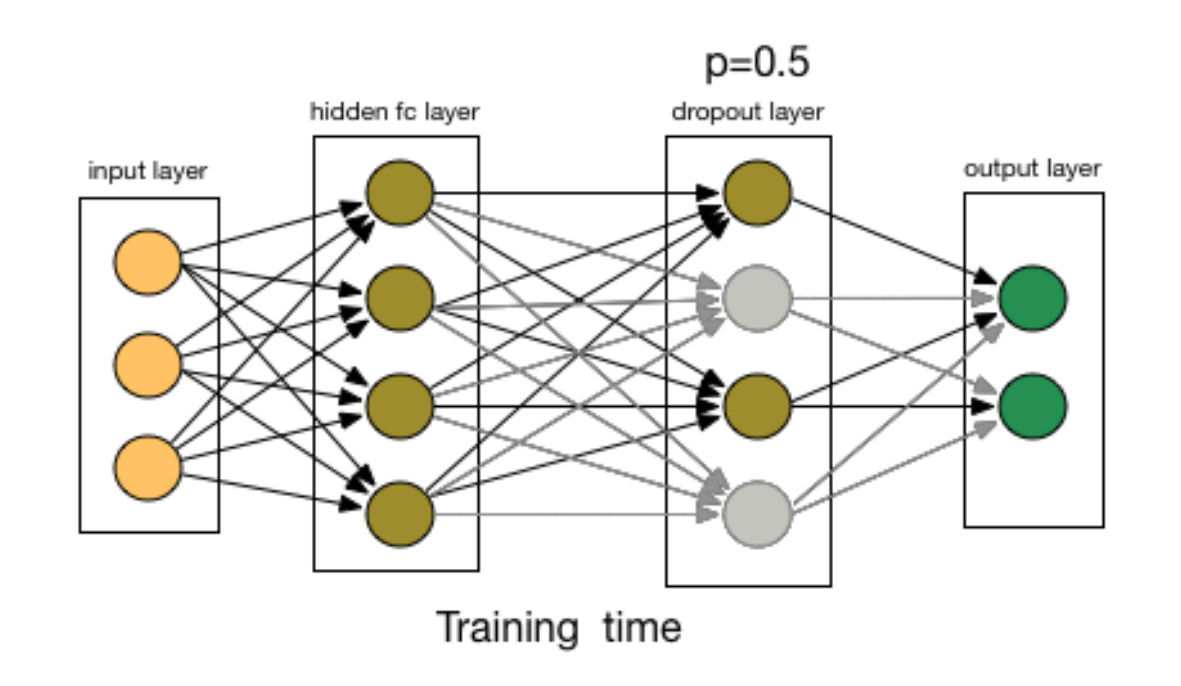
Dropout

A Dropout layer is added to the model. Random neurons are removed from this layer at each epoch of training. The fraction of remove neurons is called dropout rate.



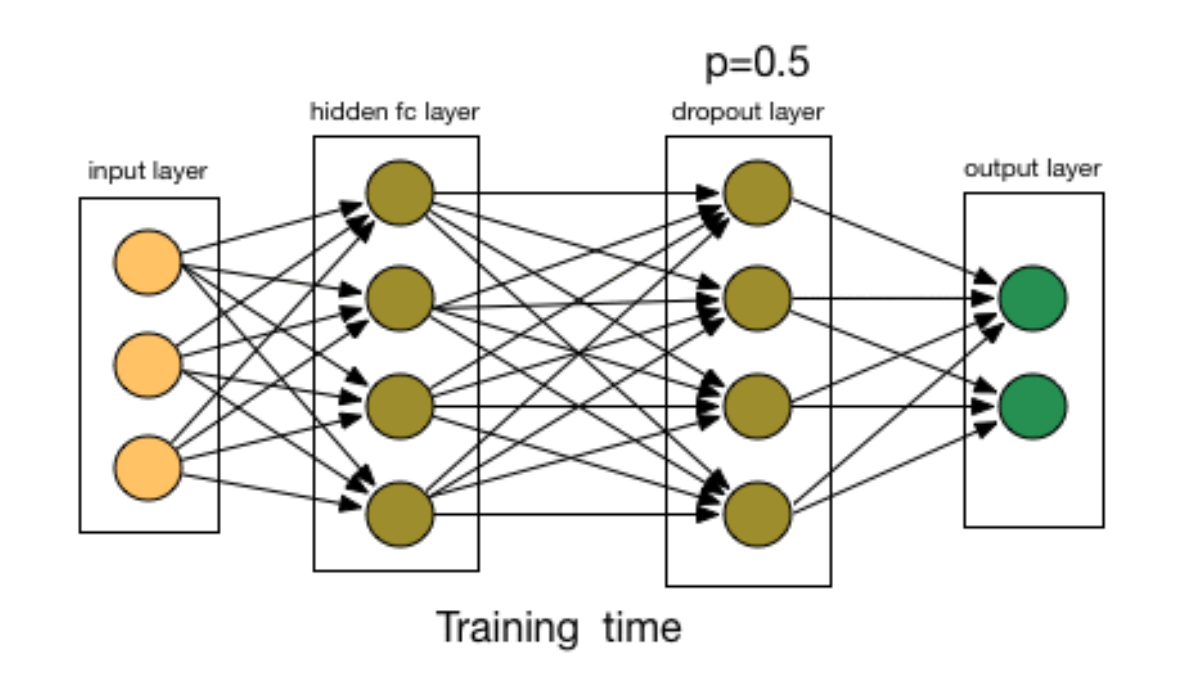
Dropout

A Dropout layer is added to the model. Random neurons are removed from this layer at each epoch of training. The fraction of remove neurons is called dropout rate.



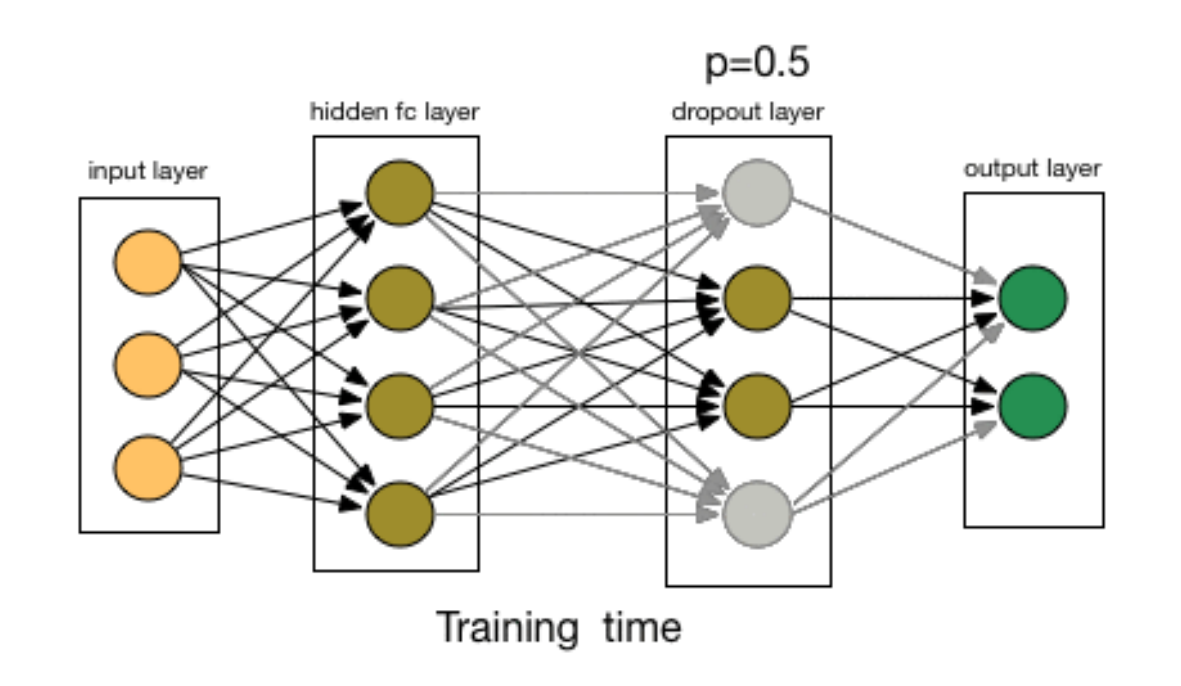
Dropout

A Dropout layer is added to the model. Random neurons are removed from this layer at each epoch of training. The fraction of remove neurons is called dropout rate.



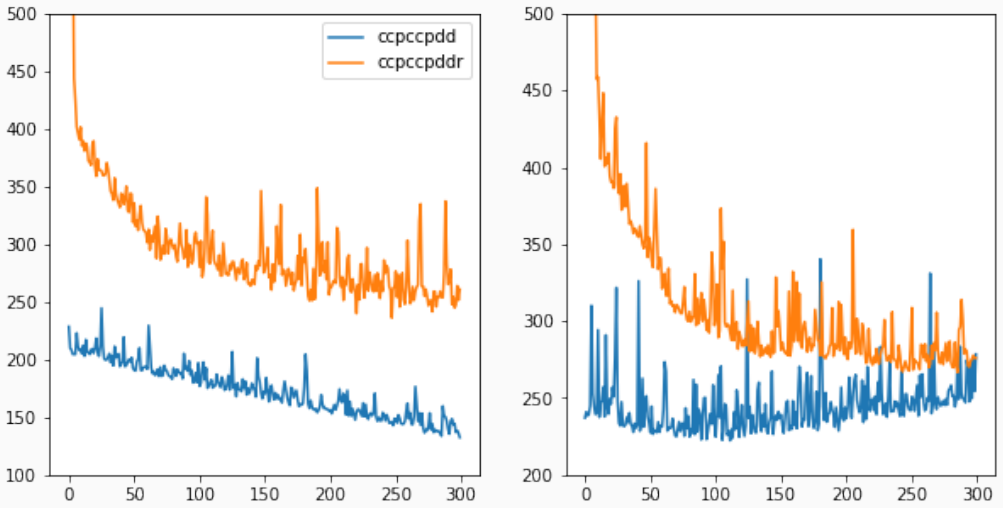
Dropout

A Dropout layer is added to the model. Random neurons are removed from this layer at each epoch of training. The fraction of remove neurons is called dropout rate.



Results of model regularization

After the regularization options are added the train loss is higher, but the model is not overfitted.

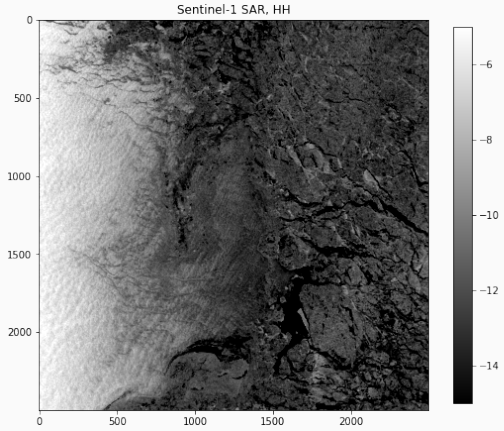


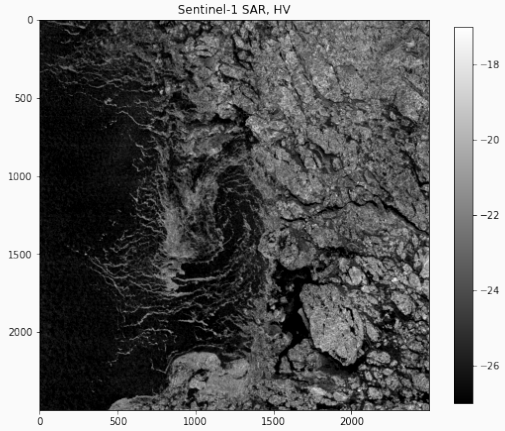
Data for the practical exercise

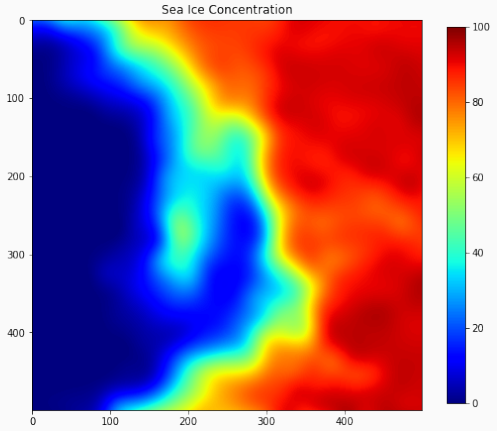
Data sources and task

Data sources:

- Sentinel-1 SAR. 1 pixel: 40m. Two polarisations: HH and HV.
- Sea ice concentration. 1 pixel: 12 km. Range of SIC: 0 - 100







Collocated HV and SIC