MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

CENTRE LUDMER CENTER
NEUROINFORMATIQUE & SANTÉ MENTALE | NEUROINFORMATICS & MENTAL HEALTH

neuro
Institut et hôpital neurologiques de Montréal
Montreal Neurological Institute and Hospital

Jewish General Hospital
Lady Davis Institute for Medical Research

Intro to Brain Segmentation with Keras

MAIN 2018 Educational Course

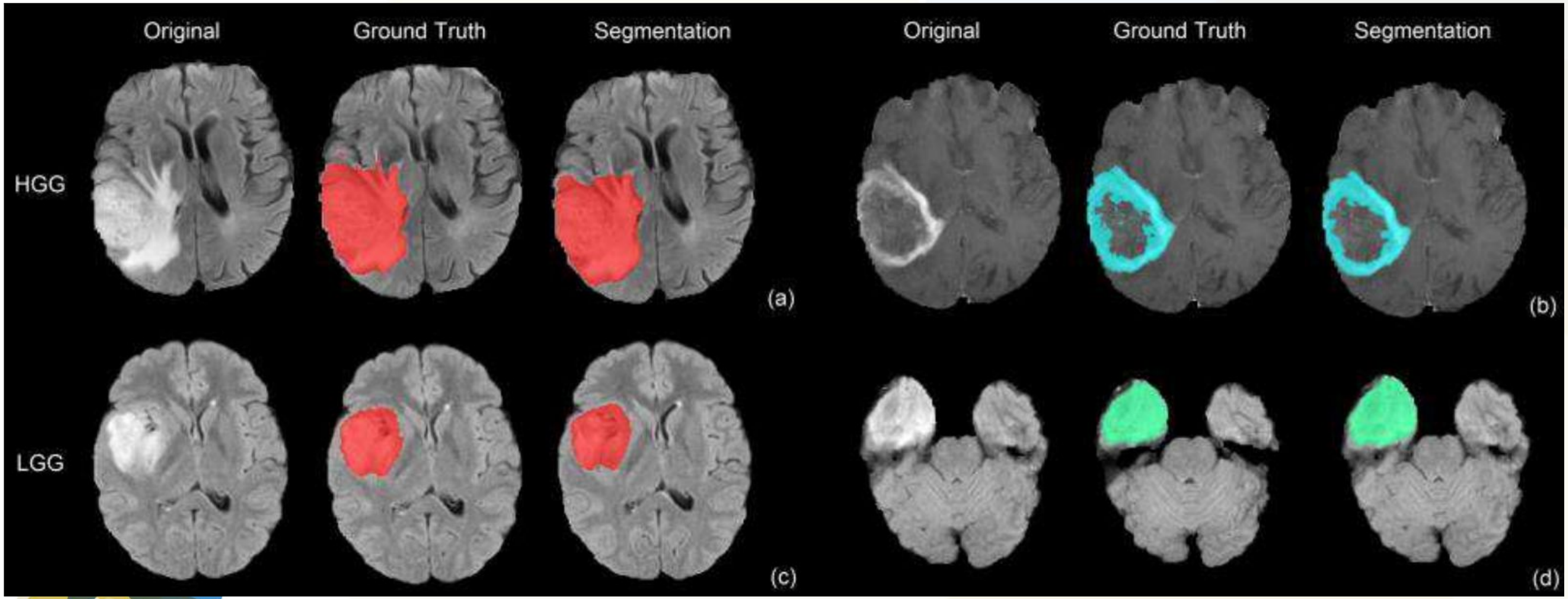
Thomas Funck, McGill University

Email: tffunck@gmail.com; Twitter: @tffunck

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Why CNNs for segmentation?

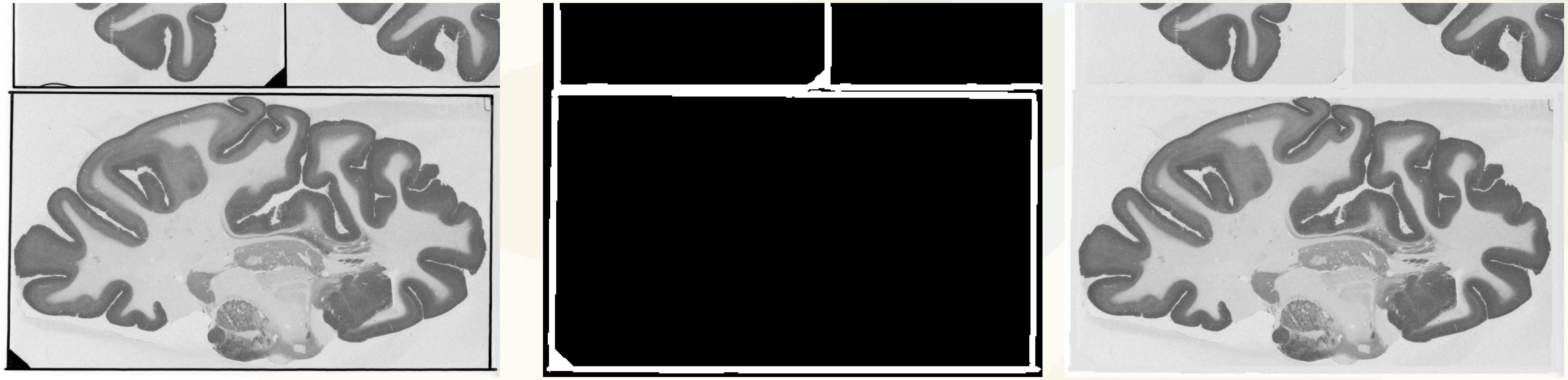
- Can solve complicated problems
  - ...if you have the data + labels



Dong, et al. (2017) **Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks.** https://arxiv.org/abs/1705.03820

Why CNNs for segmentation?

- Can solve complicated problems
- Can also solve simple, but annoying problems



Why CNNs for segmentation?

- Can solve complicated problems
- Can also solve simple, but annoying problems
- Goal: Dive into nitty-gritty to building a CNN
  - So that you can go back to the lab and create your own

Outline

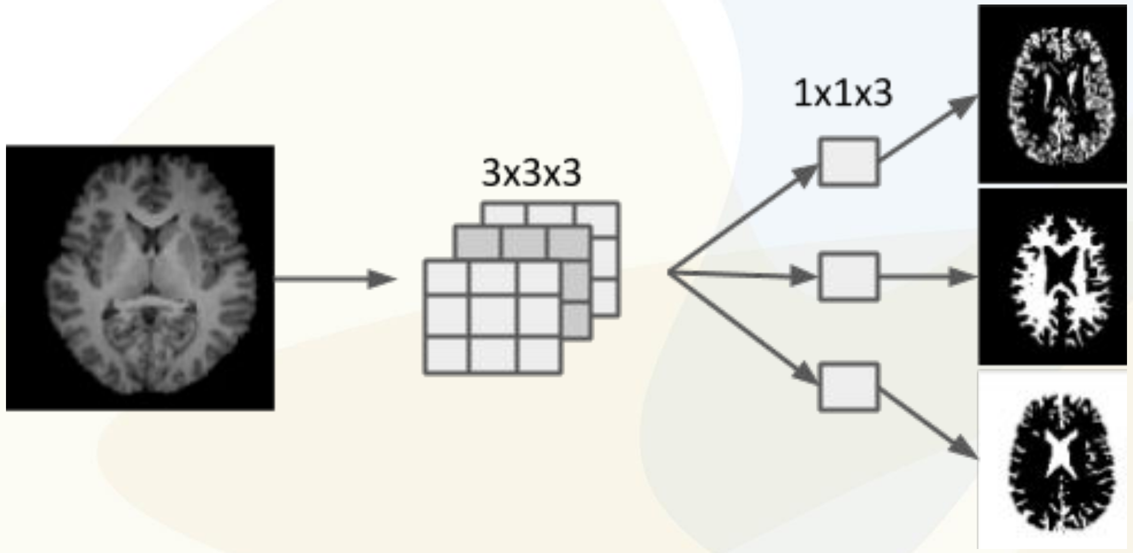
1. Basic Concepts
2. Simple Networks
3. U-Net

Outline

1. Basic Concepts
    a. Kernels
    b. Receptive field & dilations
    c. Upsampling & downsampling
    d. Final activation functions
    e. Loss functions
    f. Metrics
    g. Cross-validation
    h. Feature extraction

Example of a simple convolutional network

- 1 Layer, 3 Kernels of 3x3 size

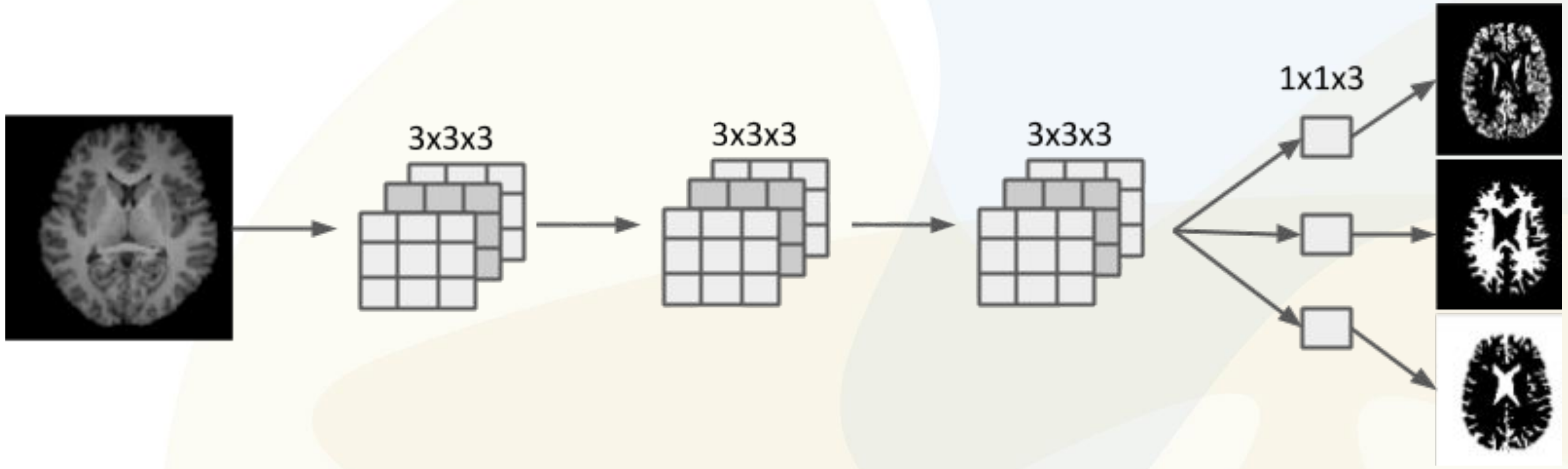


Example of a simple convolutional network

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 110, 92, 1)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 110, 92, 3)        30
_________________________________________________________________
dropout_1 (Dropout)          (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 92, 3)        12
=================================================================
Total params: 42
Trainable params: 42
Non-trainable params: 0
```

Example of a simple convolutional network

- 3 Layer, 3 Kernels of 3x3 size



Example of a simple convolutional network

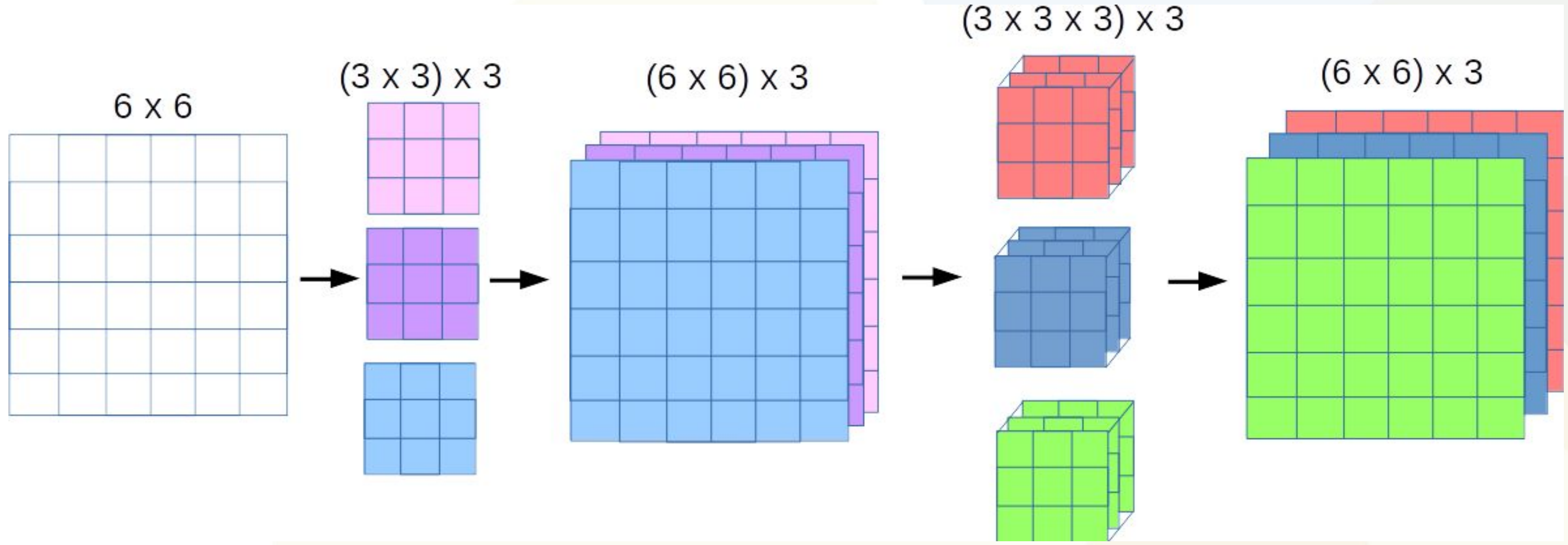
- 3 Layer, 3 Kernels of 3x3 size

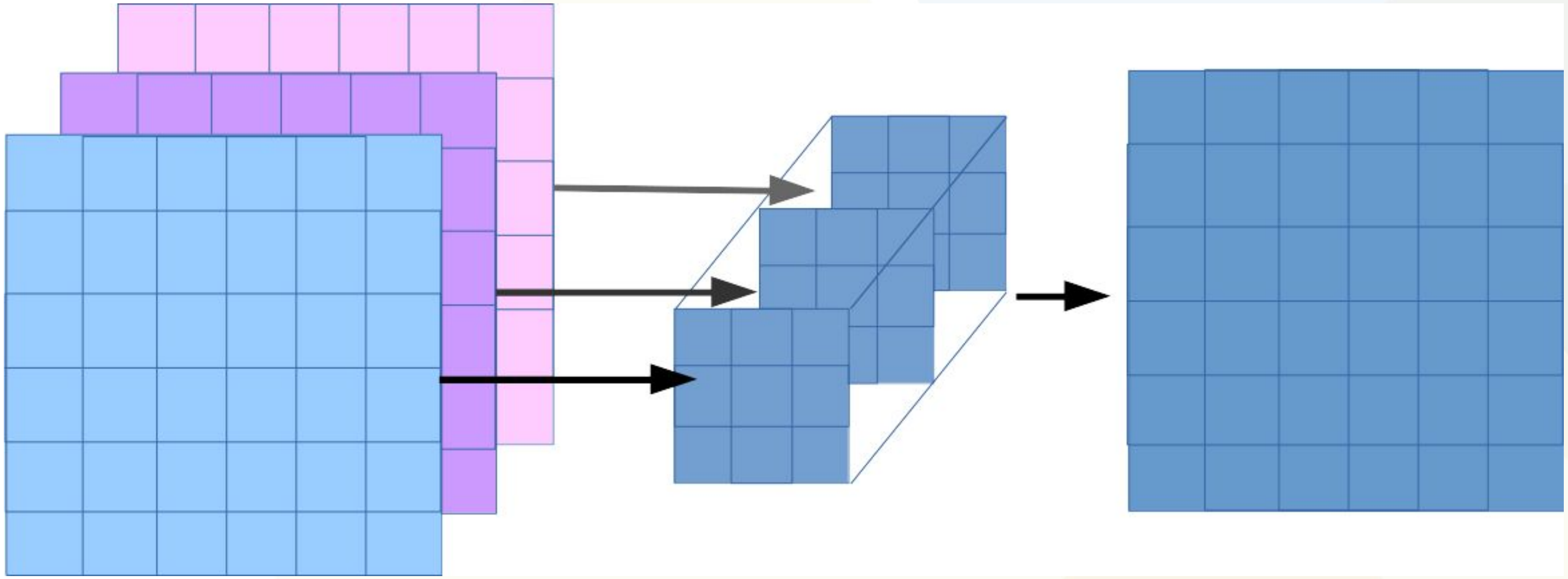
```
Layer (type)                Output Shape            Param #
=================================================================
input_2 (InputLayer)        (None, 110, 92, 1)        0
_________________________________________________________________
conv2d_3 (Conv2D)           (None, 110, 92, 3)        30
_________________________________________________________________
dropout_2 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_4 (Conv2D)           (None, 110, 92, 3)        84
_________________________________________________________________
dropout_3 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_5 (Conv2D)           (None, 110, 92, 3)        84
_________________________________________________________________
dropout_4 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_6 (Conv2D)           (None, 110, 92, 3)        12
=================================================================
Total params: 210
Trainable params: 210
Non-trainable params: 0
```

Different number of parameters. What gives?

- 3 Layer, 3 Kernels of 3x3 size

```
Layer (type)                Output Shape             Param #
=================================================================
input_2 (InputLayer)        (None, 110, 92, 1)        0
_________________________________________________________________
conv2d_3 (Conv2D)           (None, 110, 92, 3)        30
_________________________________________________________________
dropout_2 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_4 (Conv2D)           (None, 110, 92, 3)        84
_________________________________________________________________
dropout_3 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_5 (Conv2D)           (None, 110, 92, 3)        84
_________________________________________________________________
dropout_4 (Dropout)         (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_6 (Conv2D)           (None, 110, 92, 3)        12
=================================================================
Total params: 210
Trainable params: 210
Non-trainable params: 0
```

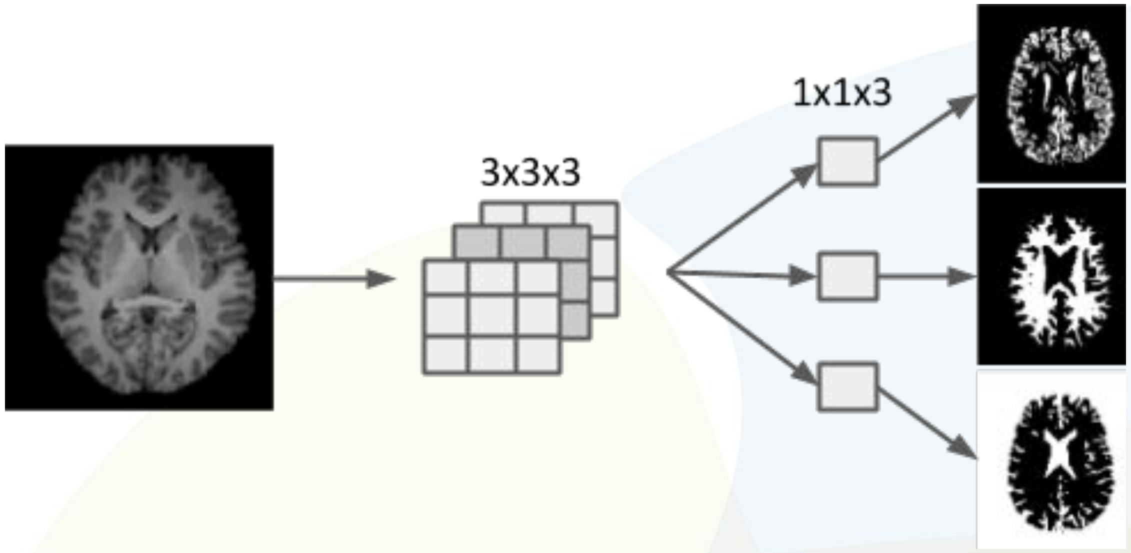

6 x 6
(3 x 3) x 3
(6 x 6) x 3
(3 x 3 x 3) x 3
(6 x 6) x 3

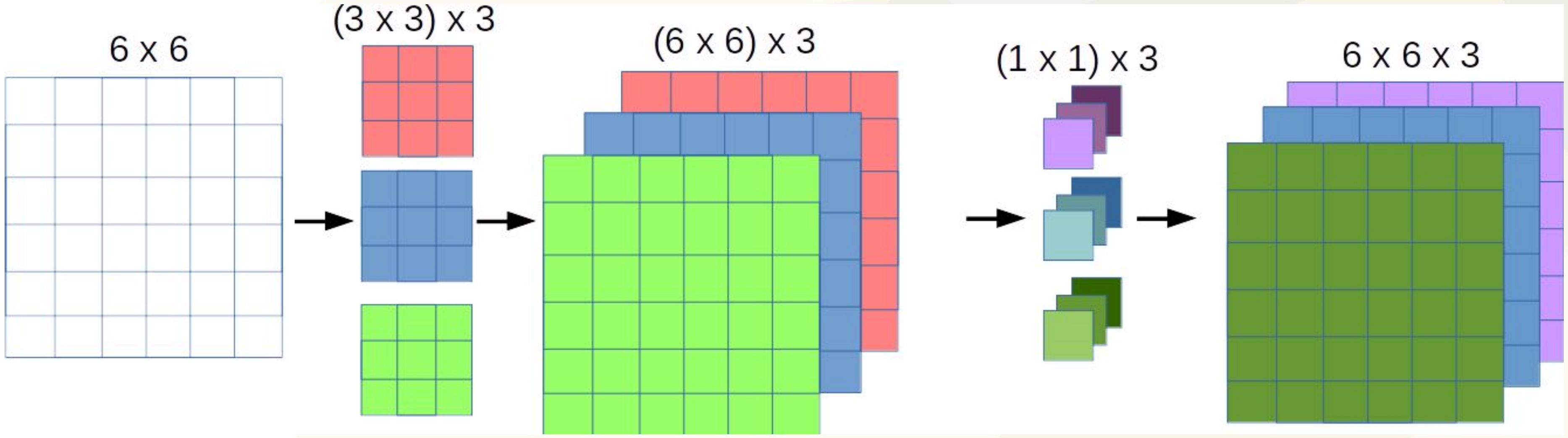


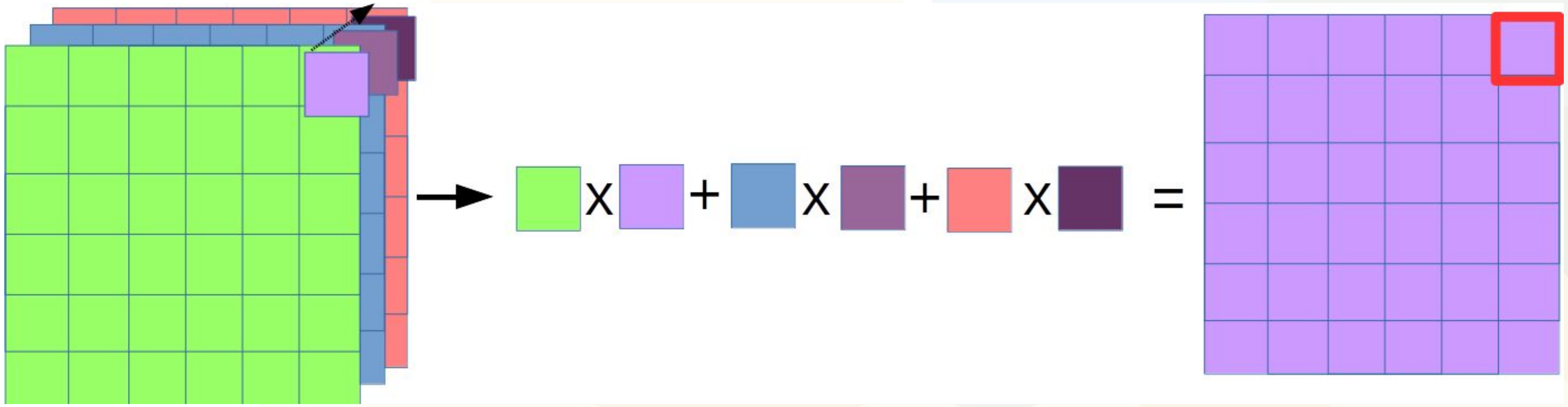
$$N_{parameters} = N_{kernels} \times Kernel_{Dimension1} \times Kernel_{Dimension2} \times Kernel_{Channels} + N_{kernels}$$

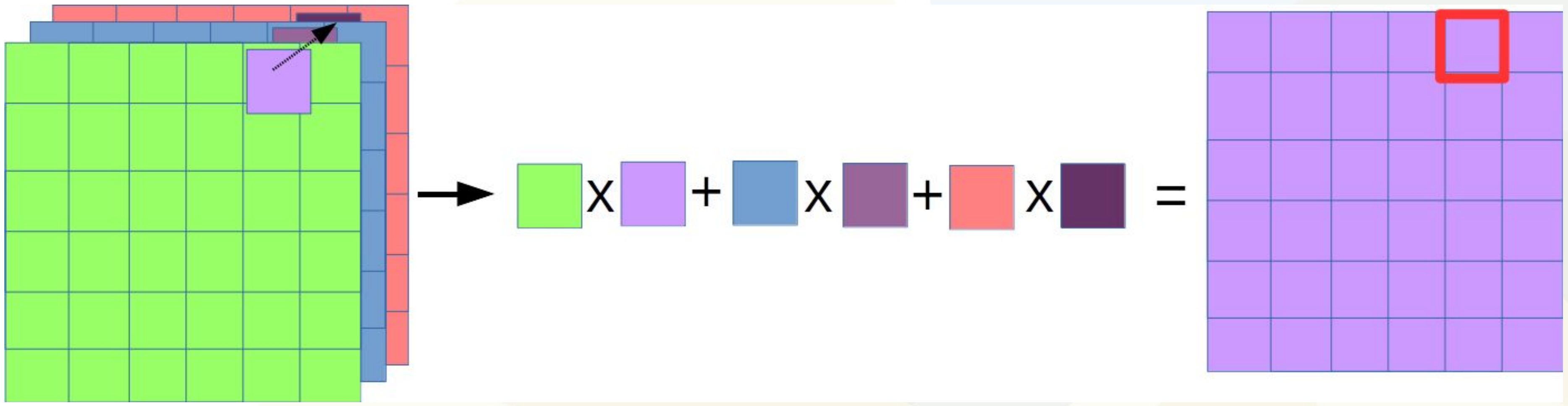
$$84 = 3 \times 3 \times 3 \times 3 + 3$$

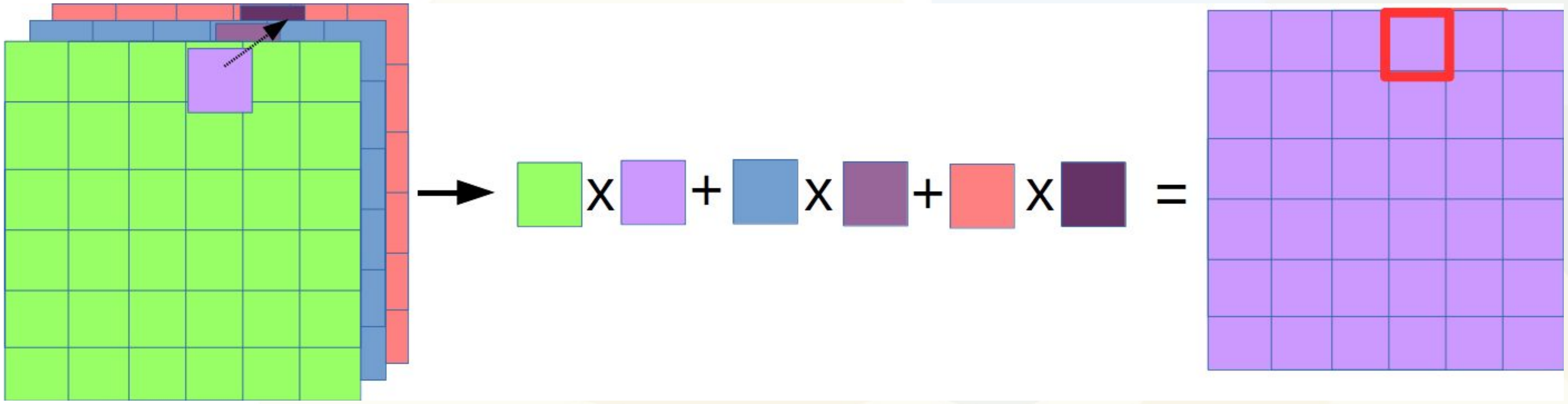
```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 110, 92, 1)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 110, 92, 3)        30
_________________________________________________________________
dropout_2 (Dropout)          (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 110, 92, 3)        84
_________________________________________________________________
dropout_3 (Dropout)          (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 110, 92, 3)        84
_________________________________________________________________
dropout_4 (Dropout)          (None, 110, 92, 3)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 110, 92, 3)        12
=================================================================
Total params: 210
Trainable params: 210
Non-trainable params: 0
```


3x3x3
1x1x3


6 x 6
(3 x 3) x 3
(6 x 6) x 3
(1 x 1) x 3
6 x 6 x 3

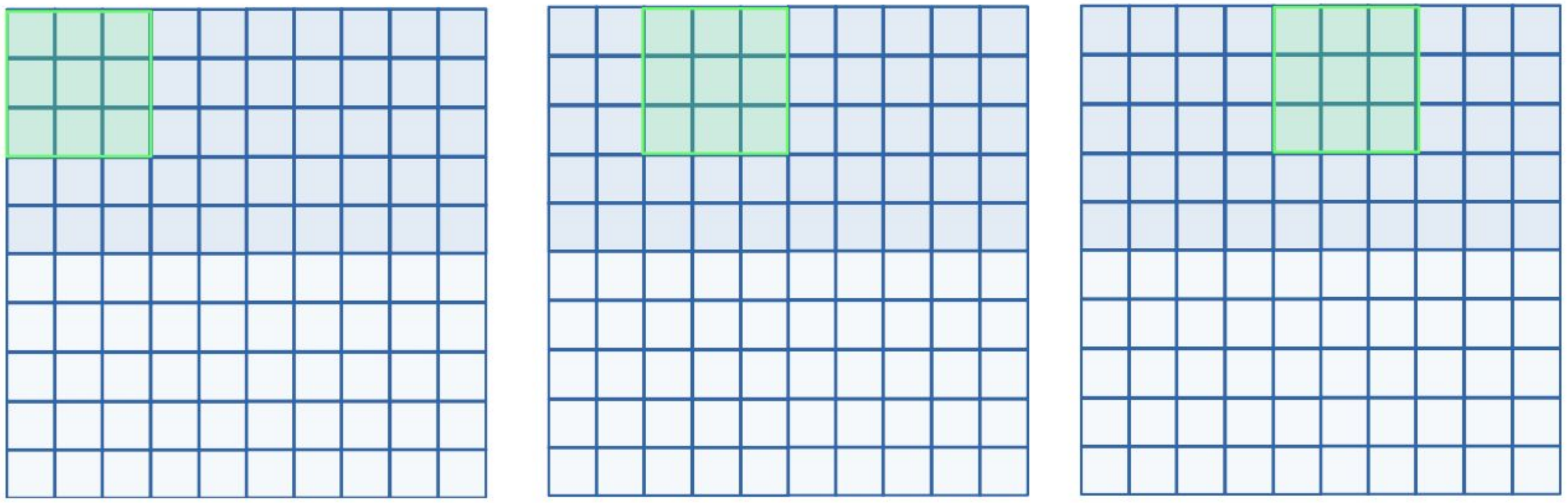

X
+
X
+
X
=


X
+
X
+
X
=


X
+
X
+
X
=

Kernel Stride

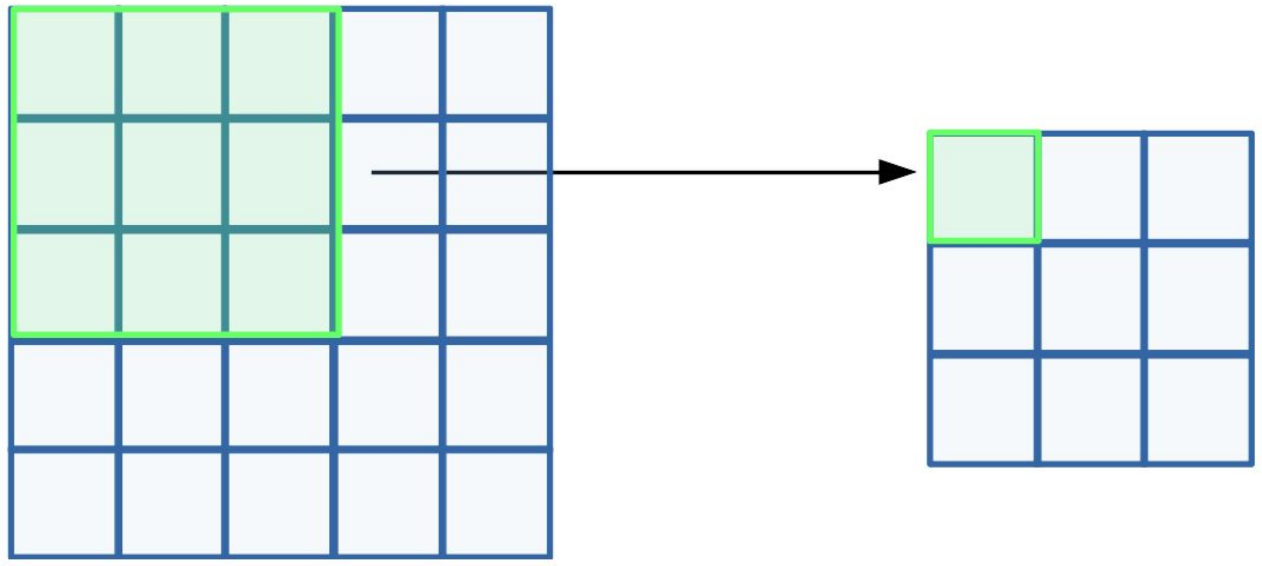
- must divide the image evenly
  - you can pad the image if necessary



kernel stride = 2

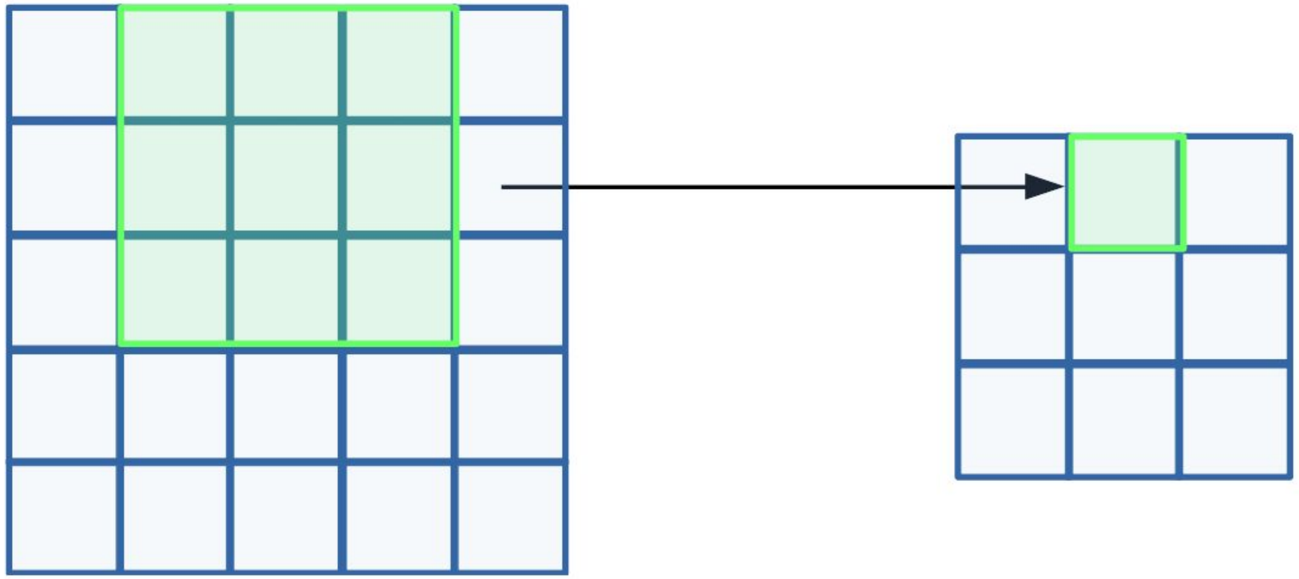
Kernel Padding

padding = “valid”



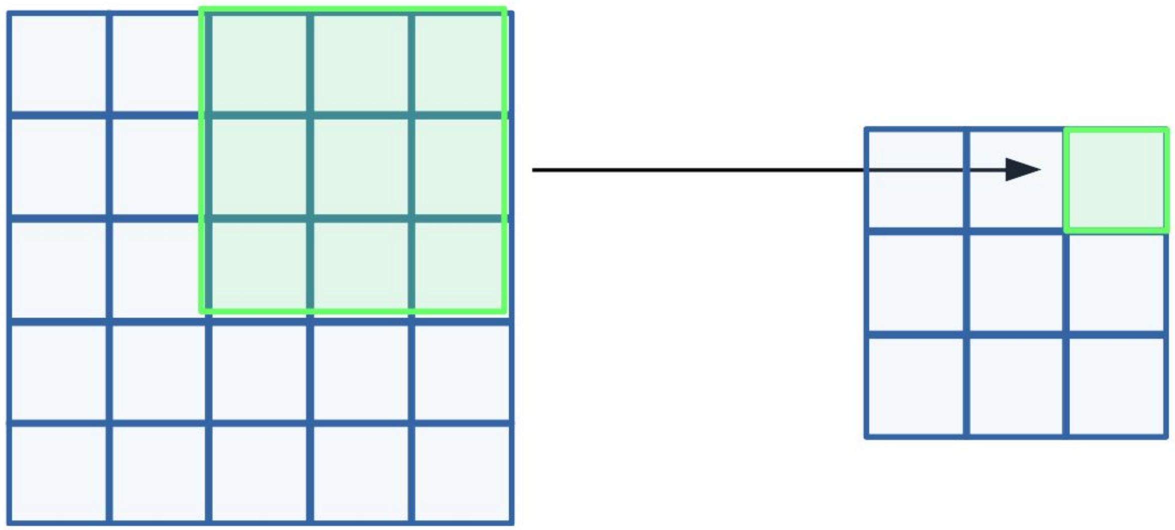
Kernel Padding

padding = "valid"



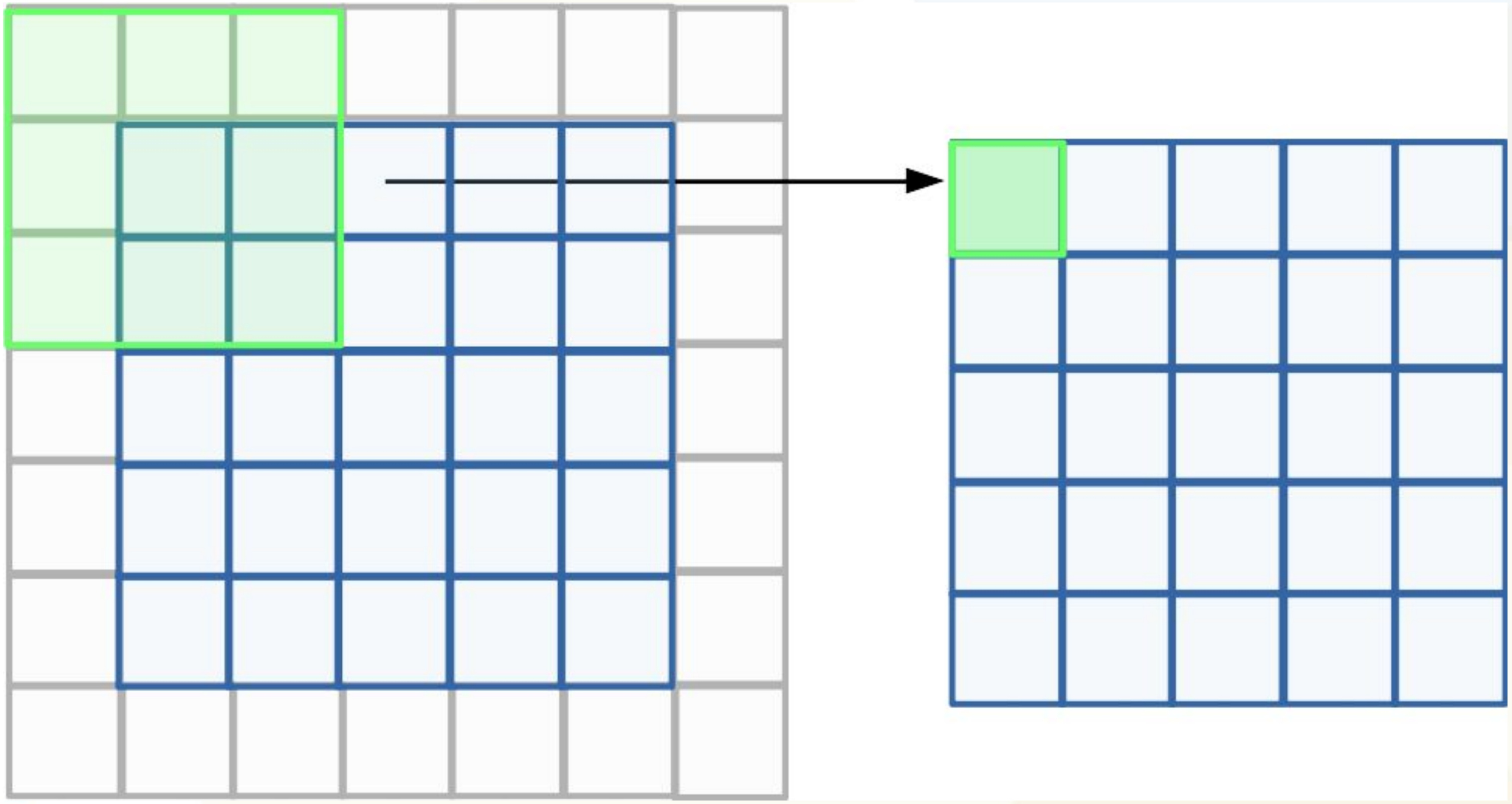
Kernel Padding

padding = "valid"



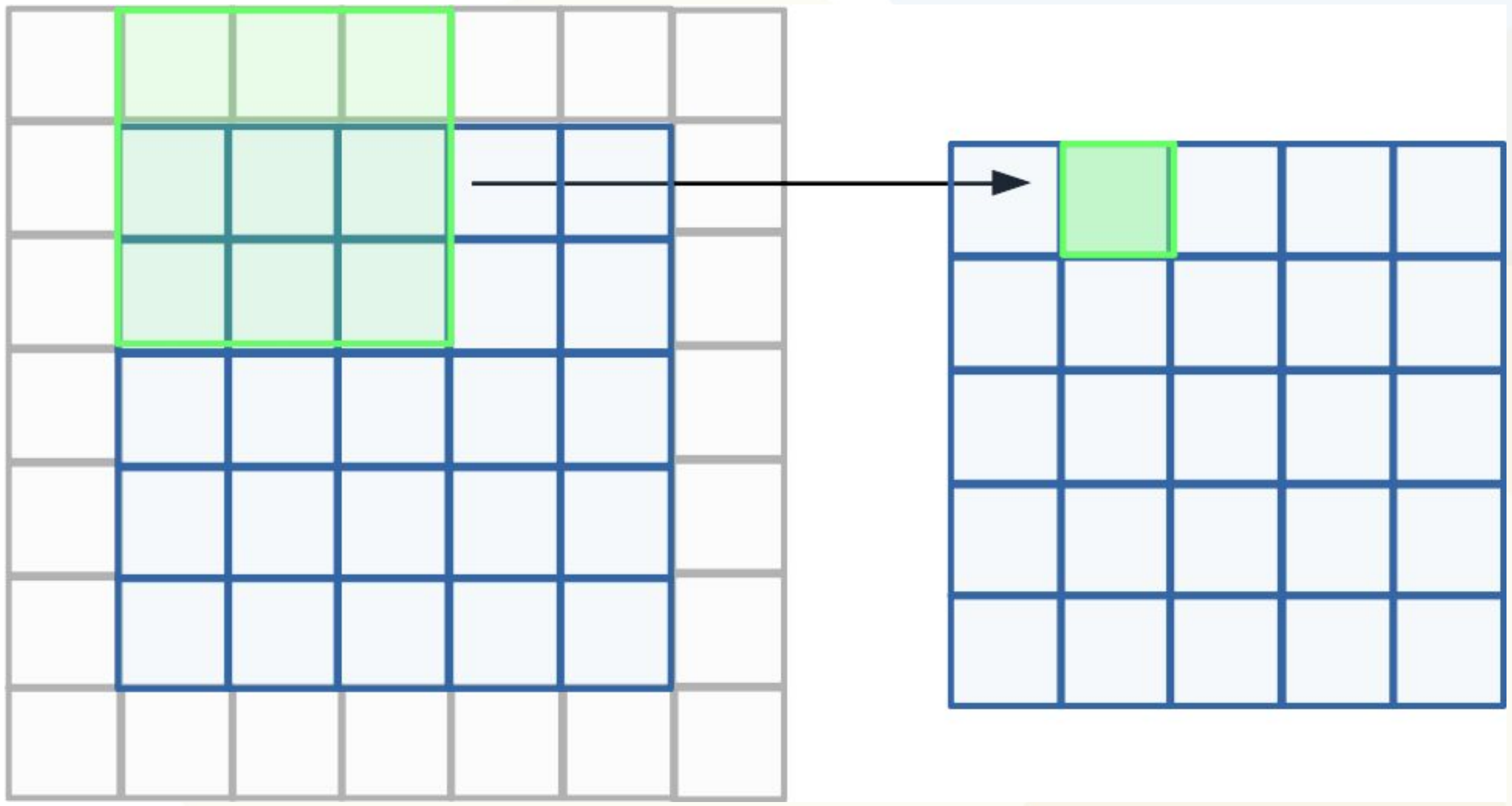
Kernel Padding

padding = "same"



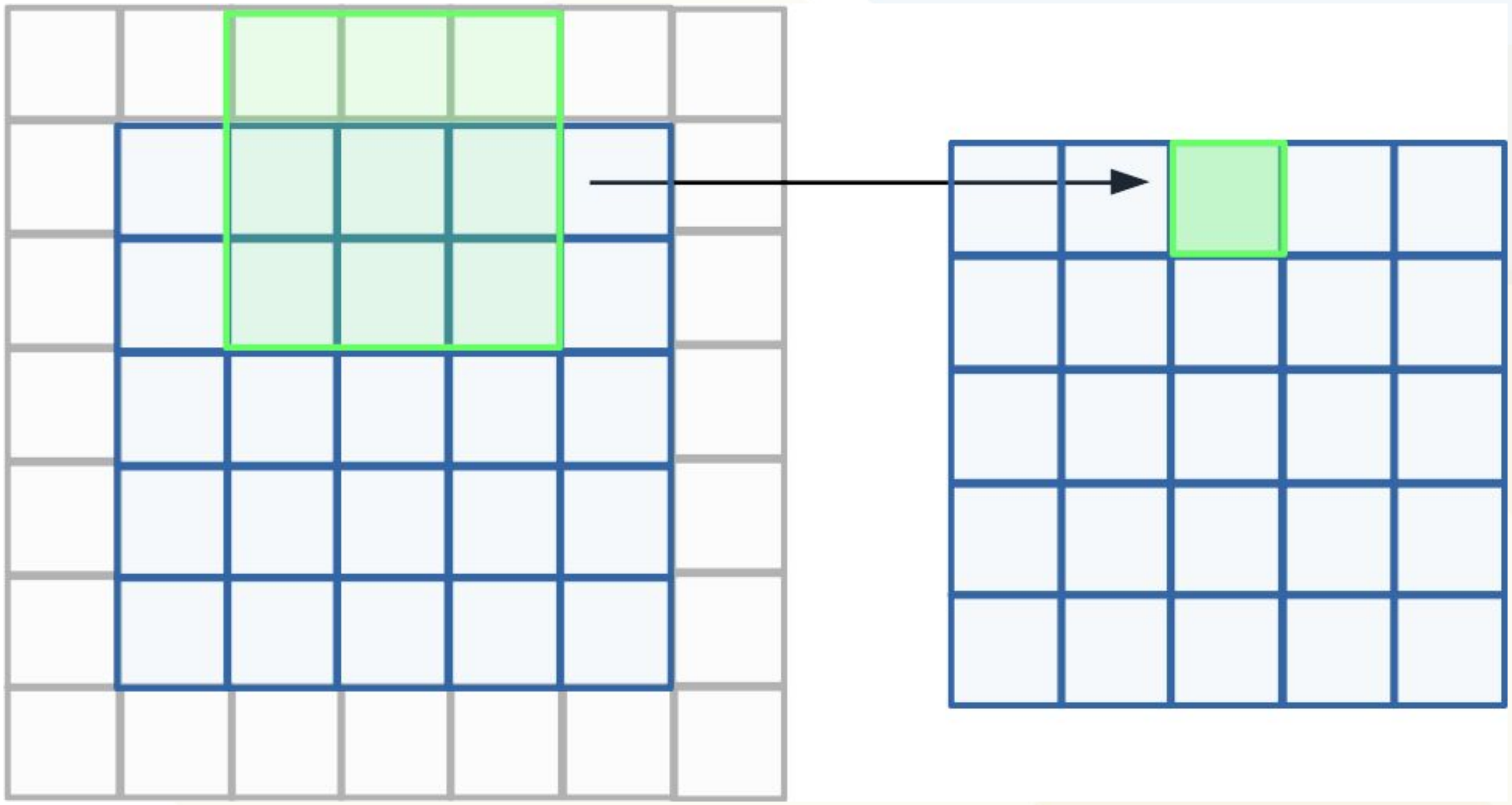
Kernel Padding

padding = "same"



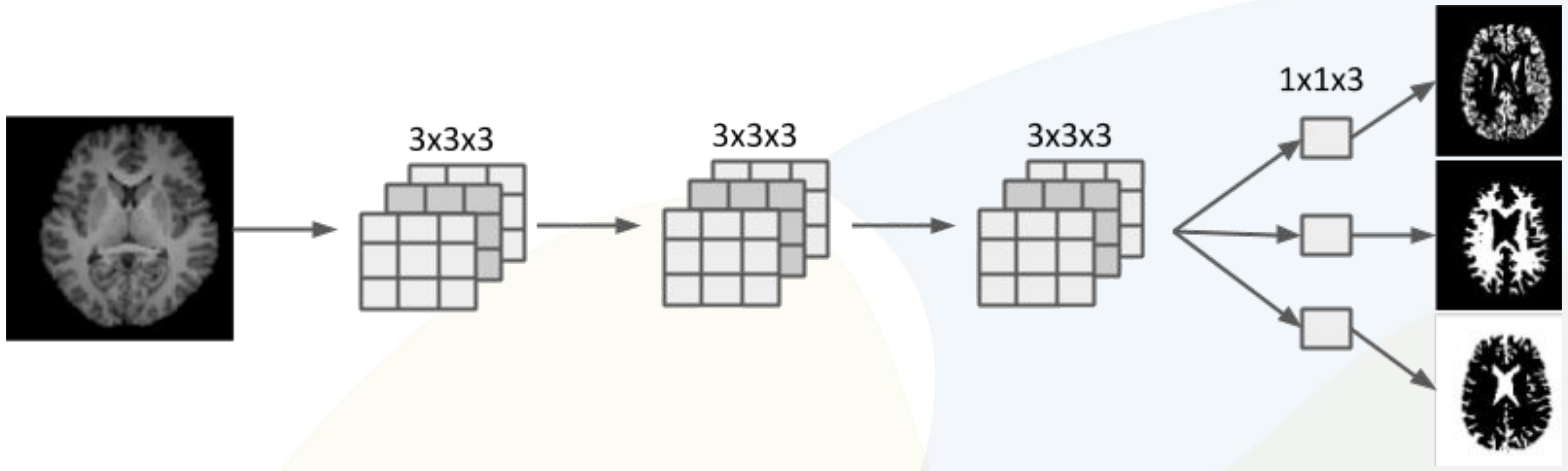
Kernel Padding

padding = "same"



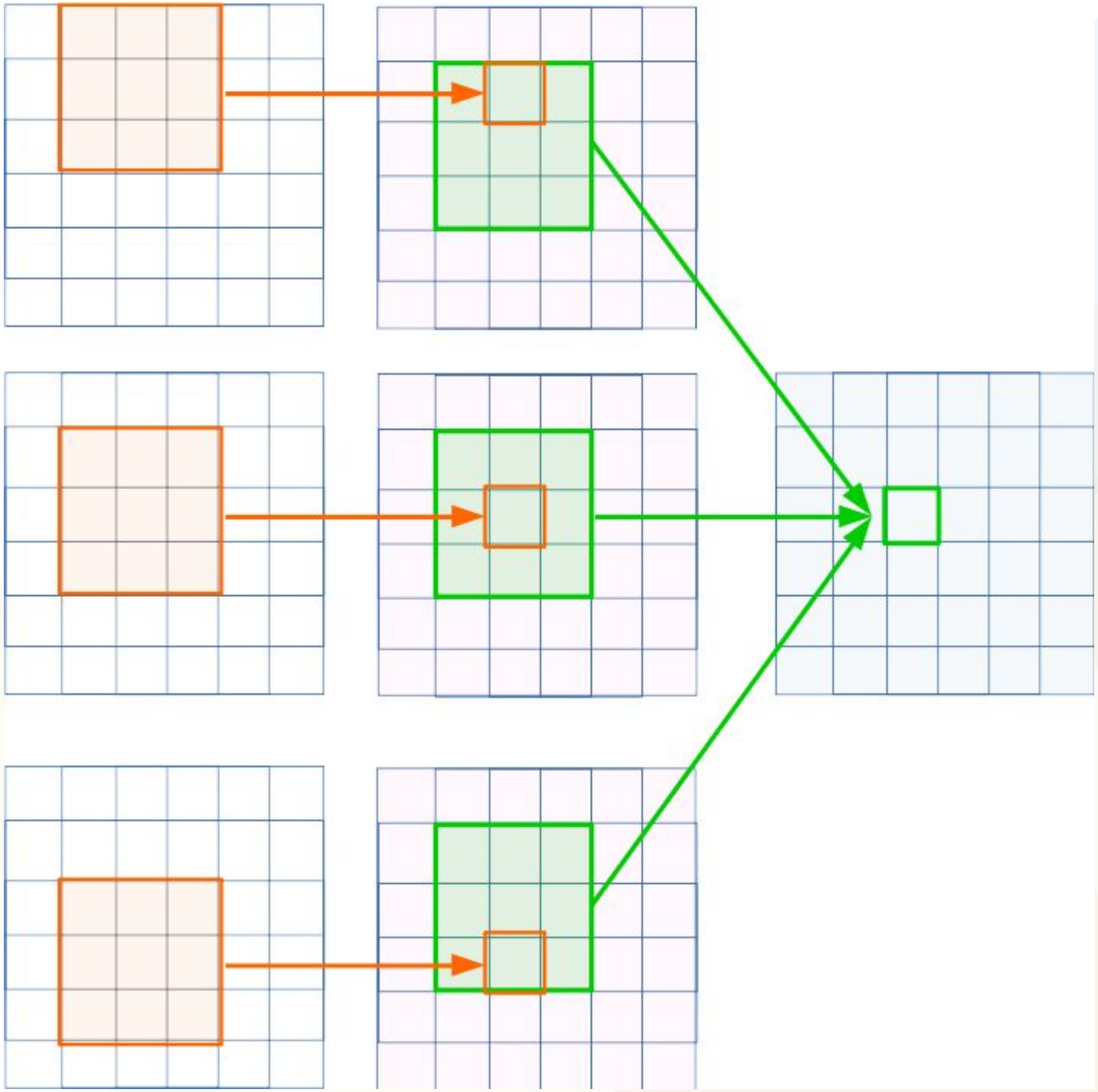
Outline

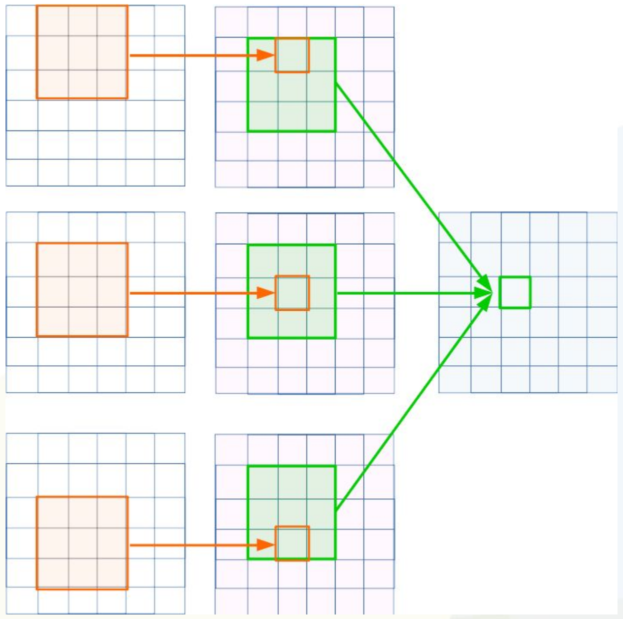
1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction

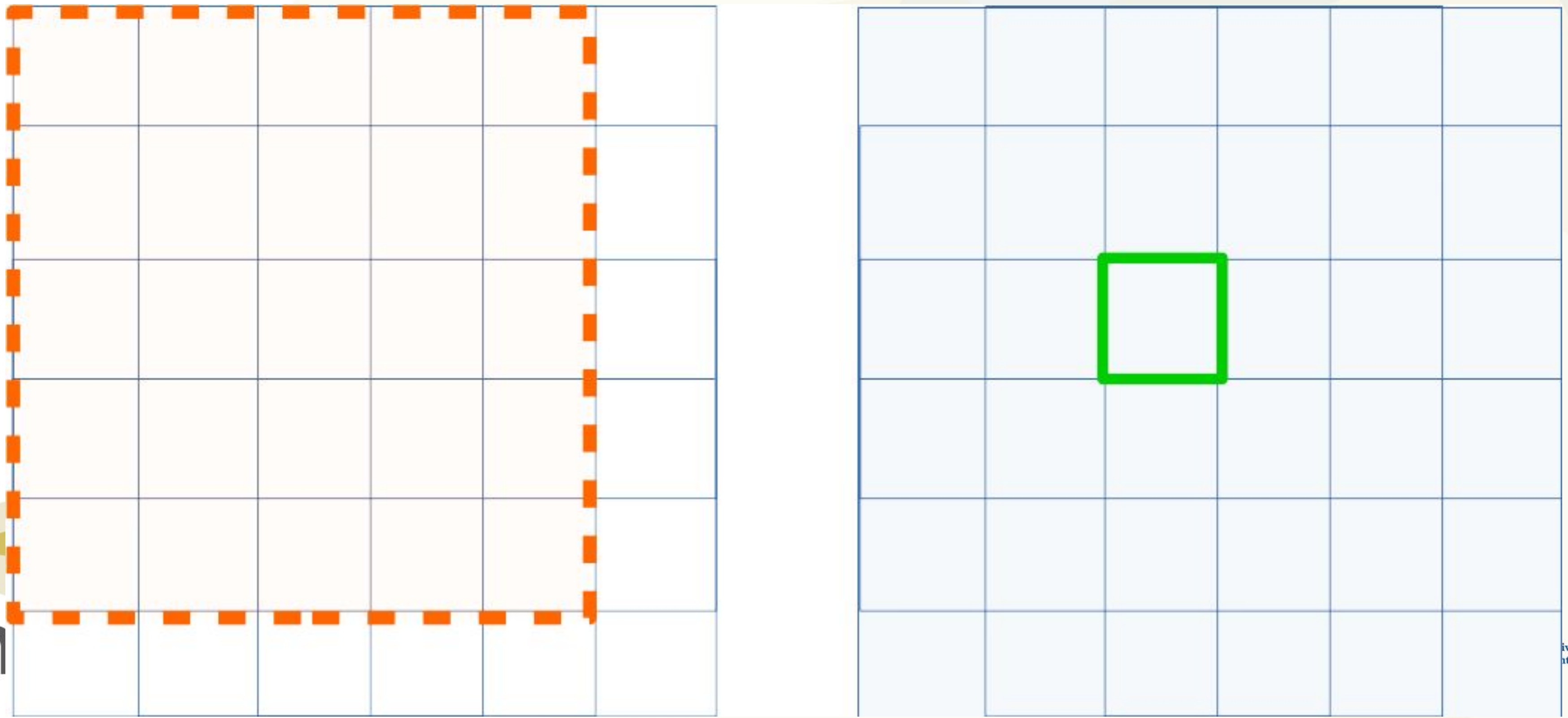


“The ***receptive field*** *of an individual sensory neuron is the particular region of the sensory space in which a stimulus will modify the firing of that neuron.”*

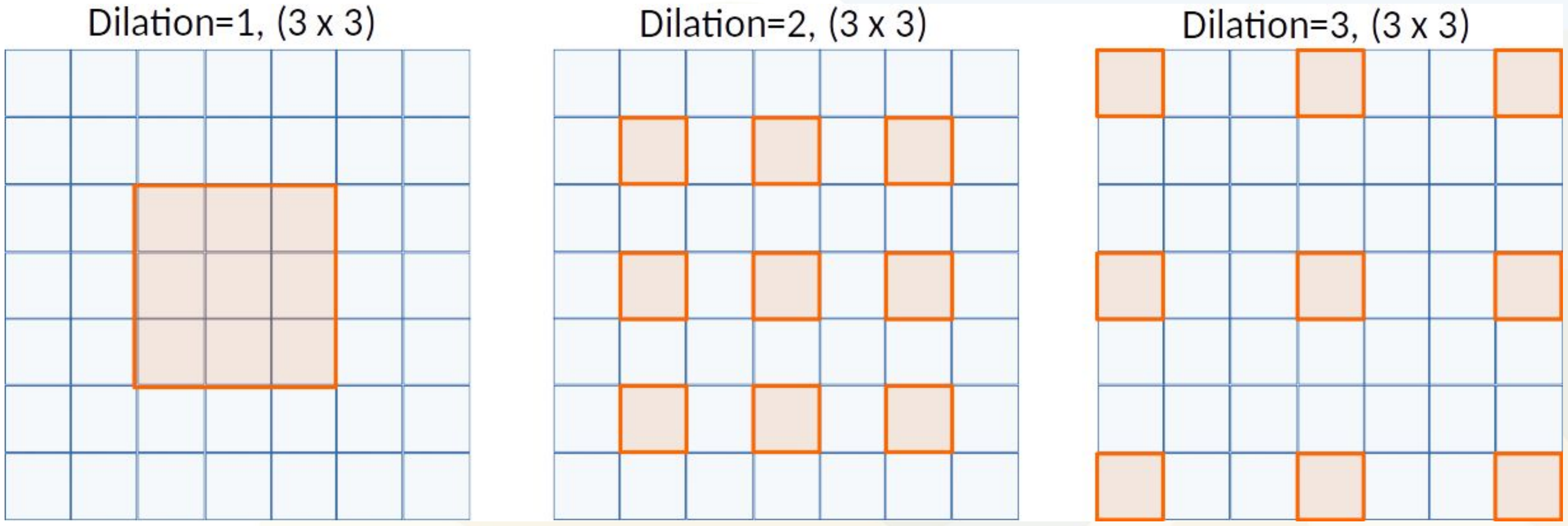
https://en.wikipedia.org/wiki/Receptive_field







Dilations

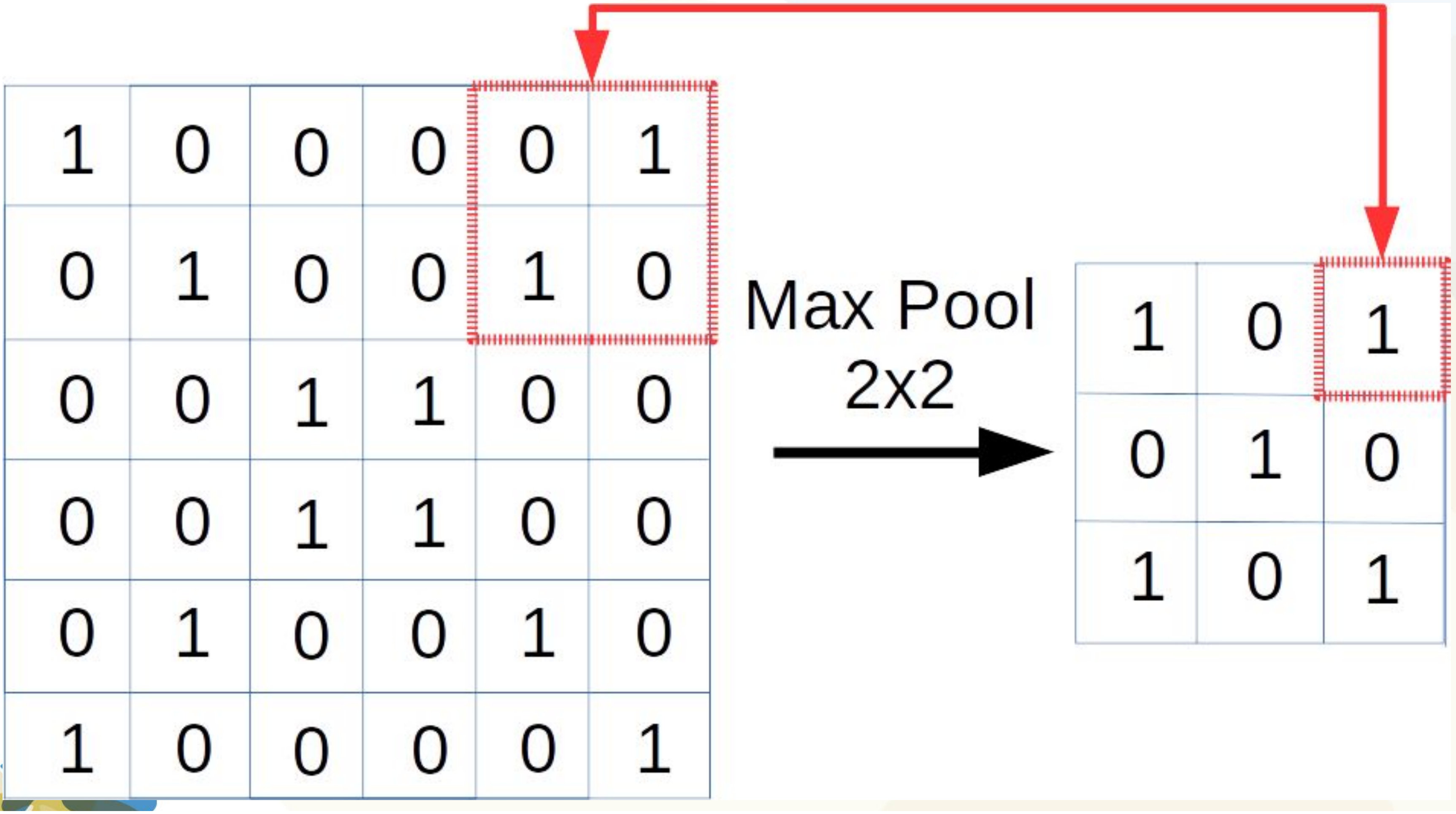


- Increase receptive field without increasing parameters

Outline

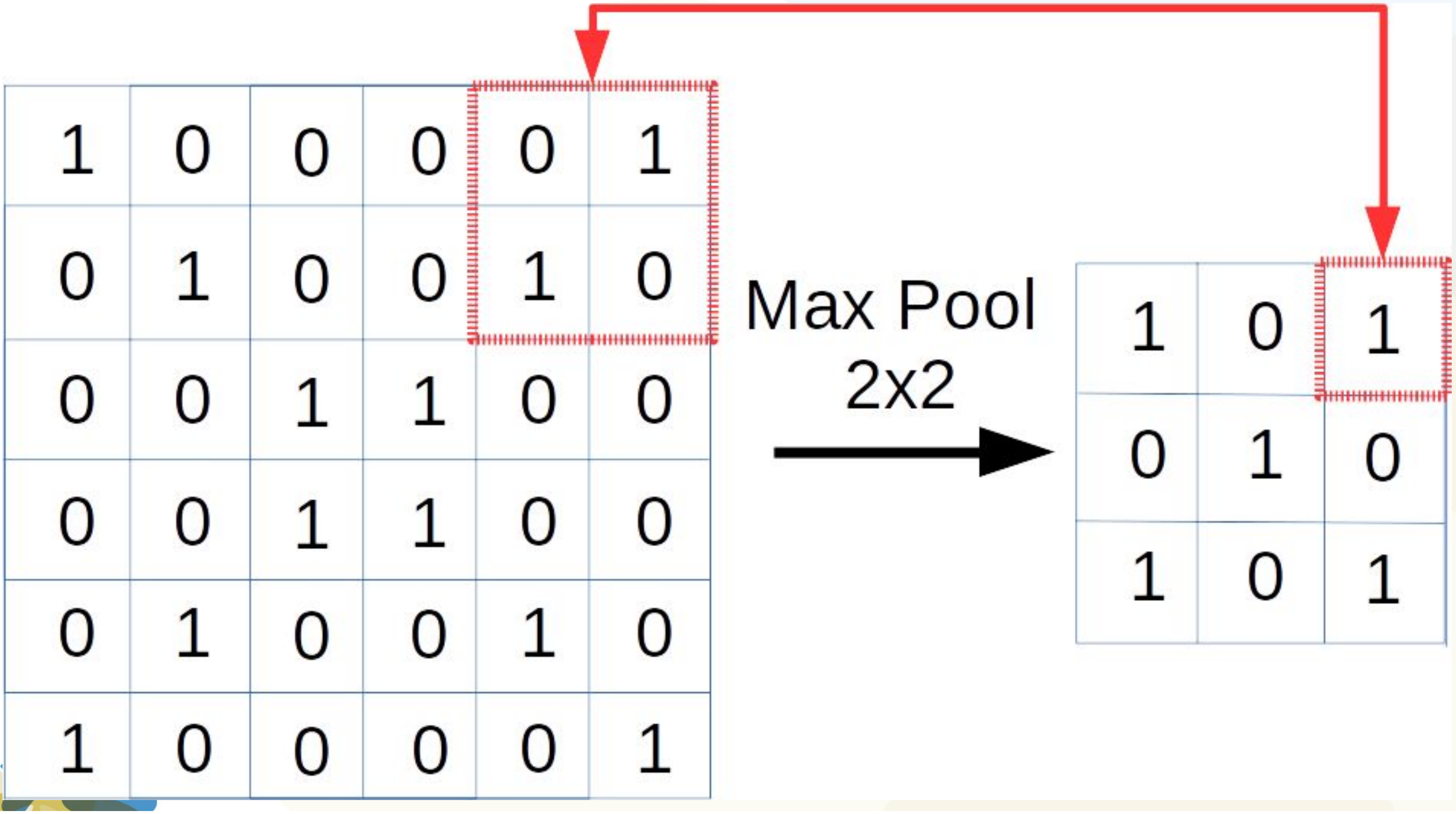
1. Basic Concepts
    a. Kernels
    b. Receptive field & dilations
    c. Upsampling & downsampling
    d. Final activation functions
    e. Loss functions
    f. Metrics
    g. Cross-validation
    h. Feature extraction

Downsampling



Summarizes essential features of image at lower spatial resolution

Downsampling



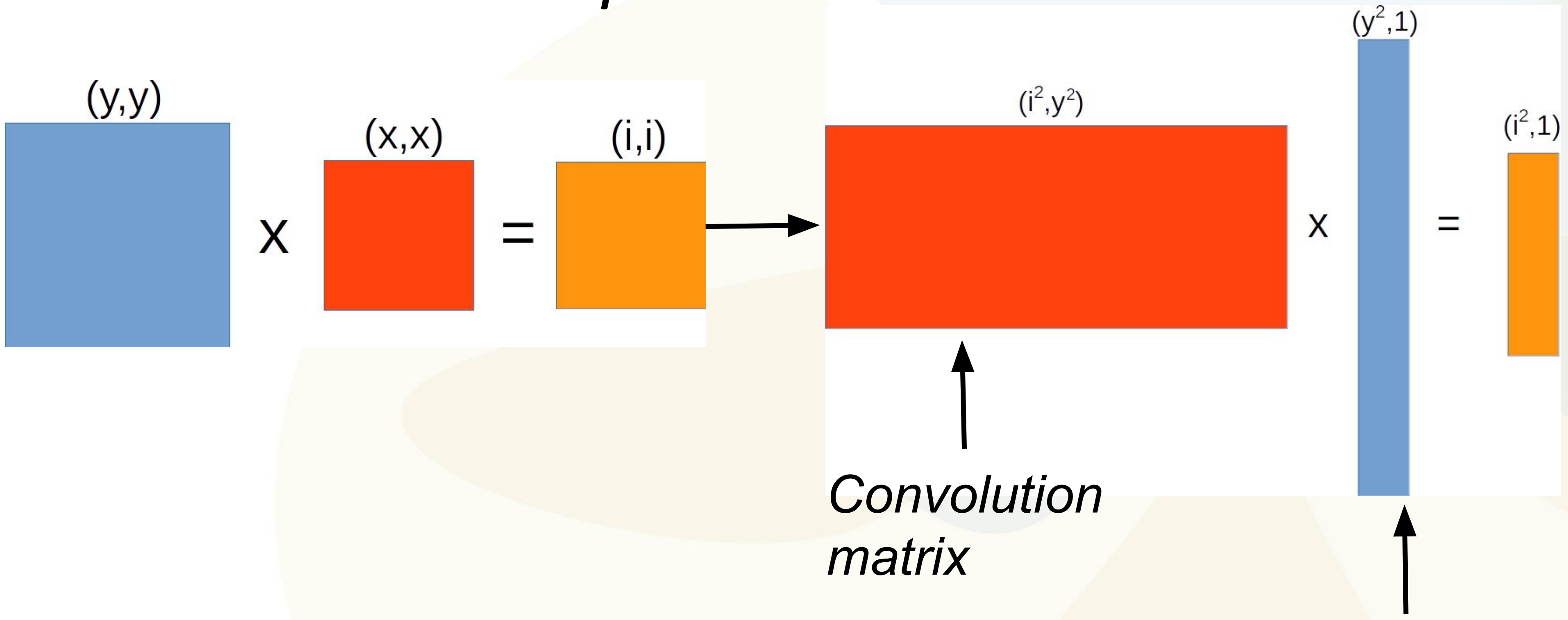
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
0	0	1	1	0	0
0	1	0	0	1	0
1	0	0	0	0	1

1	0	1
0	1	0
1	0	1

<u>Decreases the image size by half!</u>

Upsampling

Transpose Convolution

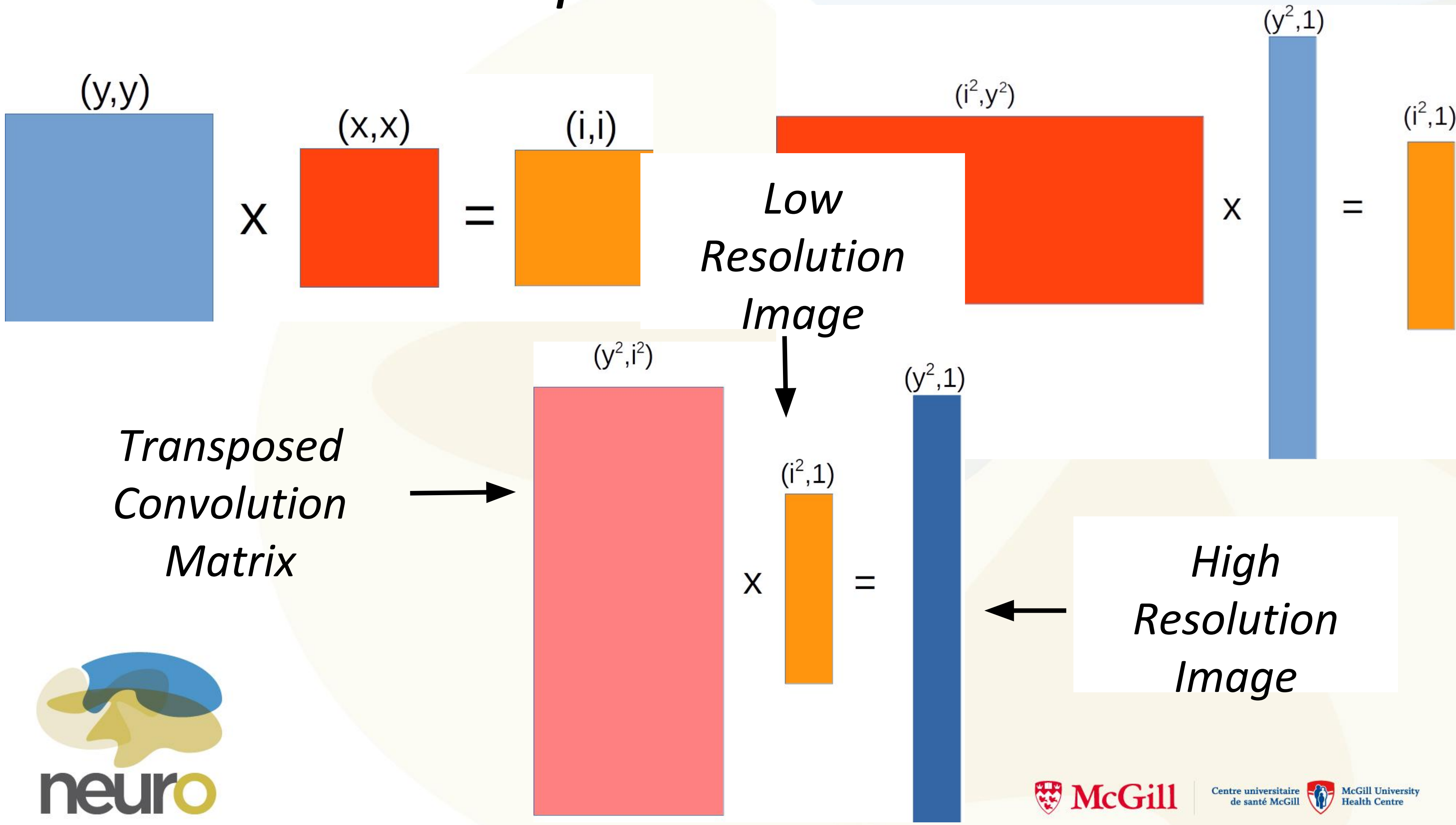


Convolution matrix

Vectorized image

Upsampling

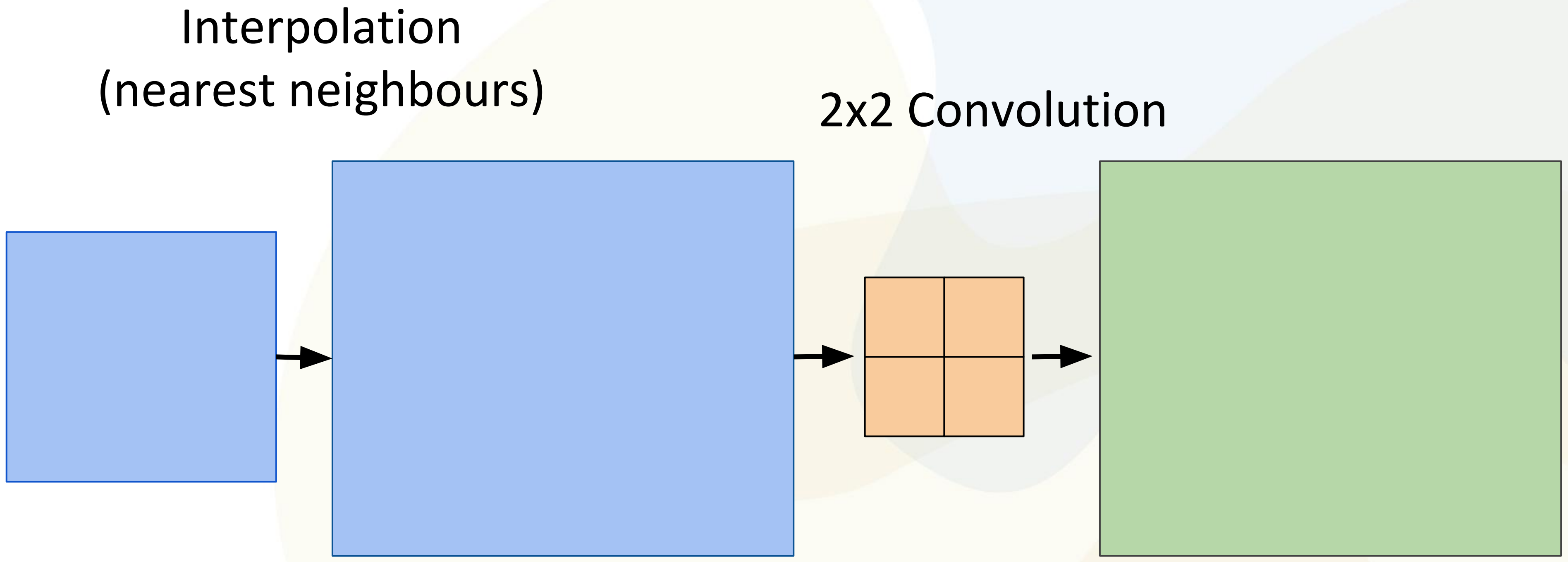
Transpose Convolution



McGill | Centre universitaire de santé McGill | McGill University Health Centre

Upsampling

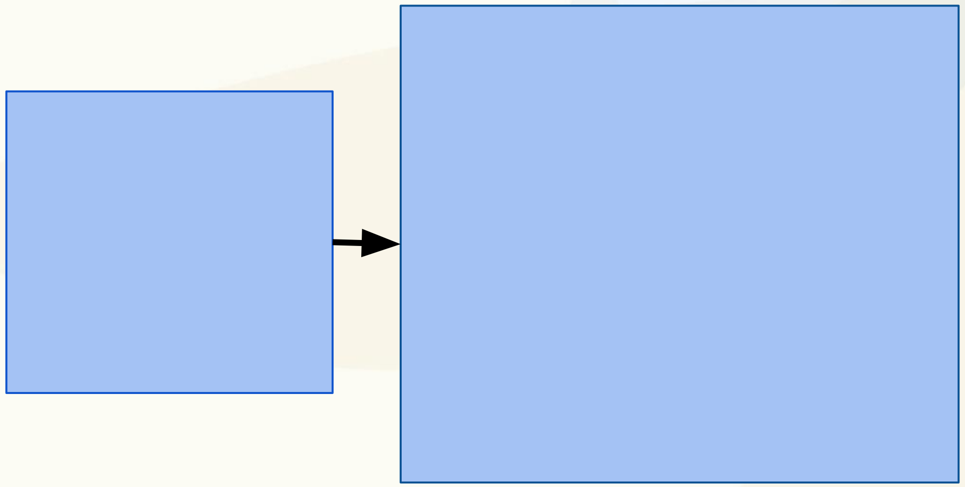
Interpolation + 2x2 Convolution



Upsampling

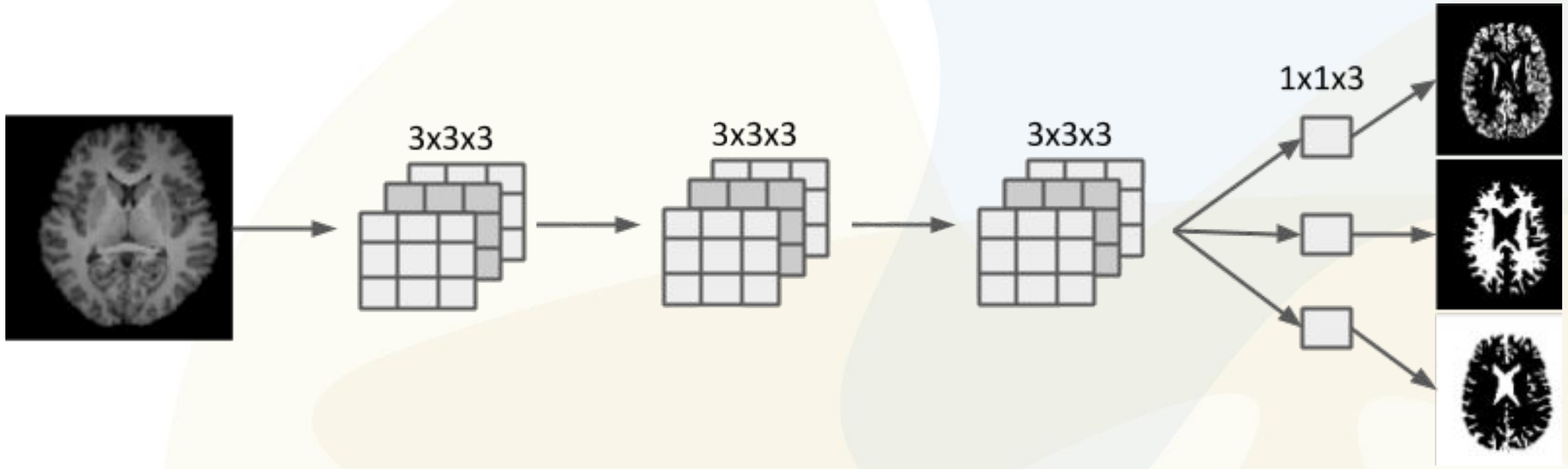
Interpolation

Interpolation
(nearest neighbours)

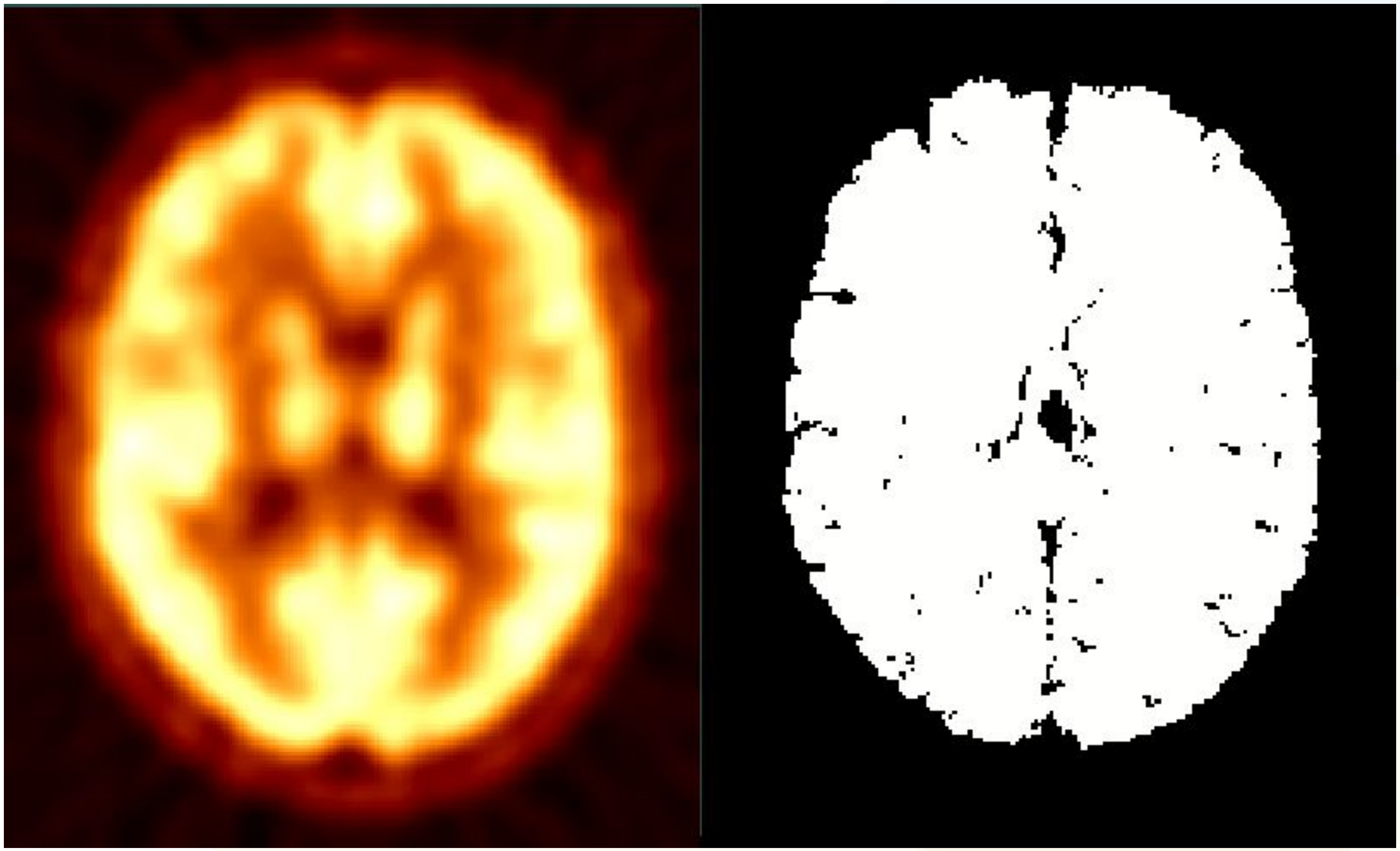


Outline

1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction


3x3x3
3x3x3
3x3x3
1x1x3

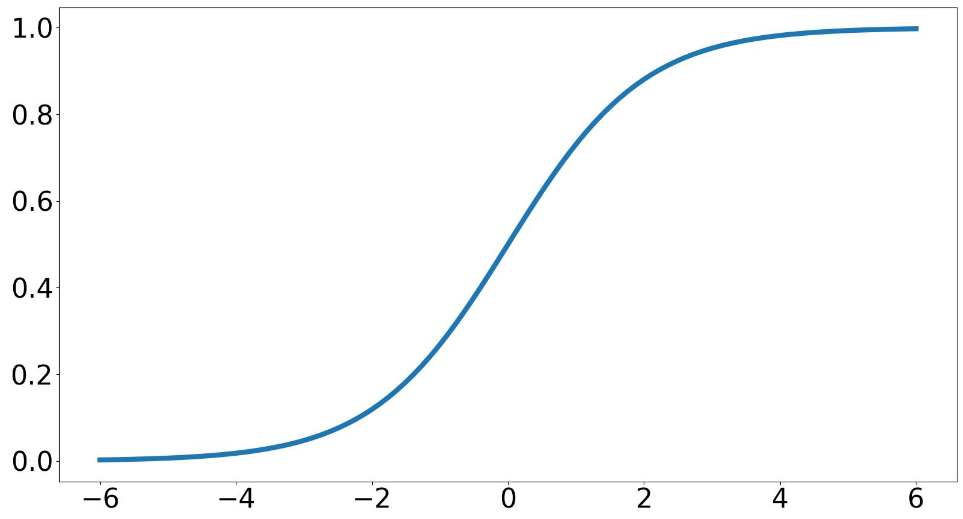
Binary Classification



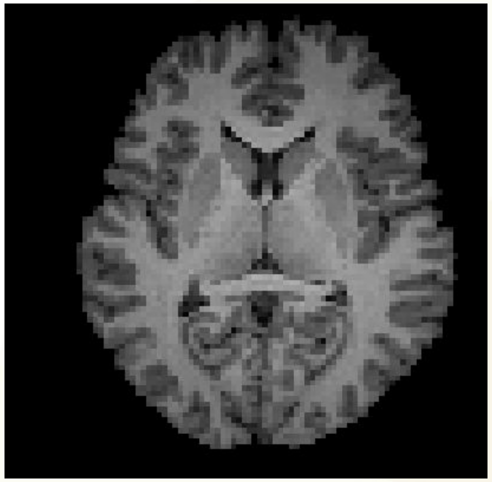
Binary Classification

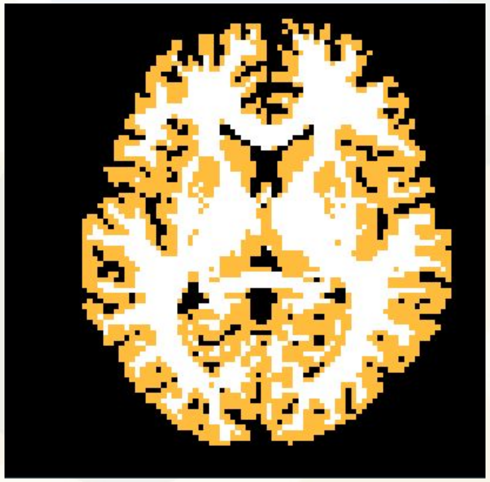
Sigmoid :

$$S(x) = \frac{1}{1+e^{-x}}$$



Multi-category Classification





Multi-category Classification

Softmax :

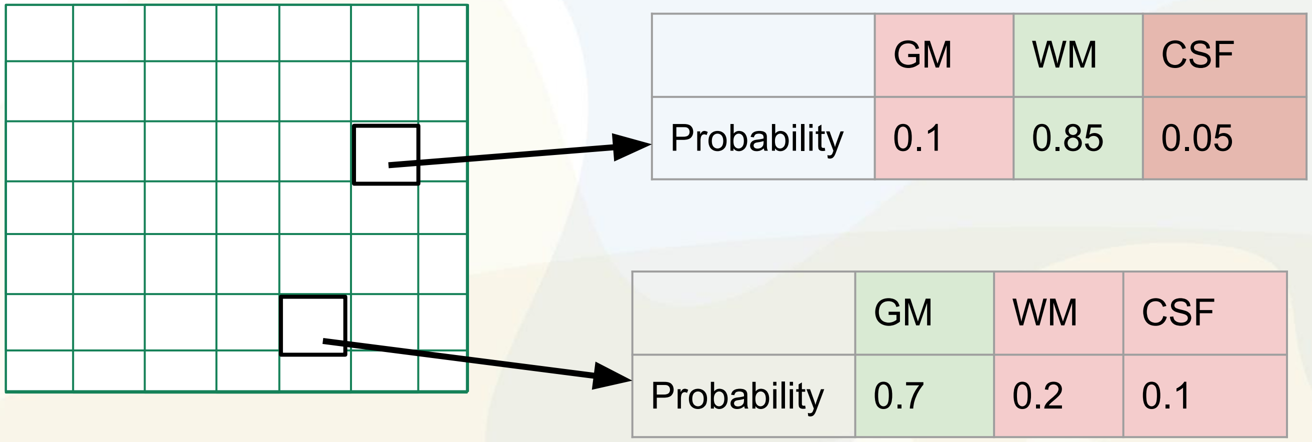
- Generalization of binary sigmoid
- Creates pseudo-probability distribution

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

Softmax Output of Network

Softmax

- Each pixel = [P(GM) P(WM), P(BG)]
- 3D output array = (Width, Height, 3)



	GM	WM	CSF
Probability	0.1	0.85	0.05

	GM	WM	CSF
Probability	0.7	0.2	0.1

- Transform to 2D image by finding class with max probability at each pixel

Outline

1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction

Choosing loss function

- **Cross Entropy :**
  - From information theory
  - Quantifies difference between two probability distributions

$$CE = \frac{1}{N}\sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

Choosing loss function

Cross Entropy :

$$CE = \frac{1}{N}\sum_i^N \sum_c^C \underbrace{True_i^c}_{\text{Ground truth probability from labels}} \times \underbrace{log_2(Predicted_i^c)}_{\text{Bits of information for predicted label}}$$

Ground truth probability from labels

Bits of information for predicted label

Cross Entropy

True Distribution

	GM	WM	CSF
data_0	1	0	0
data_1	0	1	0
data_2	0	1	0

Predicted Distribution

	GM	WM	CSF
data_0	.5	.2	.3
data_1	.2	.4	.4
data_2	.1	.5	.4

Predicted distribution produced with softmax / sigmoid final activation

Cross Entropy

True

$$\begin{array}{c|ccc} & GM & WM & CSF \\ \hline \textit{data\_0} & 1 & 0 & 0 \\ \textit{data\_1} & 0 & 1 & 0 \\ \textit{data\_2} & 0 & 1 & 0 \end{array}$$

Predicted

$$\begin{array}{c|ccc} & GM & WM & CSF \\ \hline \textit{data\_0} & .5 & .2 & .3 \\ \textit{data\_1} & .2 & .4 & .4 \\ \textit{data\_2} & .1 & .5 & .4 \end{array}$$

$$CE = \frac{1}{N} \sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

$1 \times -\log_2(0.5) + 0 \times \log_2(.2) + 0 \times \log_2(.3) +$

Cross Entropy

True

$$\begin{array}{c|ccc} & GM & WM & CSF \\ \hline \textit{data\_0} & 1 & 0 & 0 \\ \textit{data\_1} & 0 & 1 & 0 \\ \textit{data\_2} & 0 & 1 & 0 \end{array}$$

Predicted

$$\begin{array}{c|ccc} & GM & WM & CSF \\ \hline \textit{data\_0} & .5 & .2 & .3 \\ \textit{data\_1} & .2 & .4 & .4 \\ \textit{data\_2} & .1 & .5 & .4 \end{array}$$

$$CE = \frac{1}{N} \sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

$1 \text{ x } -\log_2(0.5) + 0 \text{ x } \log_2(.2) + 0 \text{ x } \log_2(.3) +$

$0 \text{ x } -\log_2(0.2) + 1 \text{ x } \log_2(.4) + 0 \text{ x } \log_2(.4) +$

Cross Entropy

$$
\begin{array}{c|ccc}
\textit{True} & GM & WM & CSF \\
\hline
data\_0 & 1 & 0 & 0 \\
data\_1 & 0 & 1 & 0 \\
data\_2 & 0 & 1 & 0
\end{array}
\qquad
\begin{array}{c|ccc}
\textit{Predicted} & GM & WM & CSF \\
\hline
data\_0 & .5 & .2 & .3 \\
data\_1 & .2 & .4 & .4 \\
data\_2 & .1 & .5 & .4
\end{array}
$$

$$CE = \frac{1}{N}\sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

$1 \text{ x } -\log_2(0.5) + 0 \text{ x } \log_2(.2) + 0 \text{ x } \log_2(.3) +$

$0 \text{ x } -\log_2(0.2) + 1 \text{ x } \log_2(.4) + 0 \text{ x } \log_2(.4) +$

$0 \text{ x } -\log_2(0.1) + 1 \text{ x } \log_2(.5) + 0 \text{ x } \log_2(.4)$

neuro

McGill Centre universitaire de santé McGill McGill University Health Centre

Cross Entropy

True

	GM	WM	CSF
data_0	1	0	0
data_1	0	1	0
data_2	0	1	0

Predicted

	GM	WM	CSF
data_0	.5	.2	.3
data_1	.2	.4	.4
data_2	.1	.5	.4

$$CE = -\frac{1}{N}\sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

$1 \text{ x } -\log_2(0.5) + 0 \text{ x } \log_2(.2) + 0 \text{ x } \log_2(.3) +$

$0 \text{ x } -\log_2(0.2) + 1 \text{ x } \log_2(.4) + 0 \text{ x } \log_2(.4) +$

$0 \text{ x } -\log_2(0.1) + 1 \text{ x } \log_2(.5) + 0 \text{ x } \log_2(.4)$

$= 1 \text{ x } -\log_2(0.5) + 1 \text{ x } \log_2(.4) + 1 \text{ x } \log_2(.4)$

= 3.64

Cross Entropy

True

	GM	WM	CSF
data_0	1	0	0
data_1	0	1	0
data_2	0	1	0

Predicted

	GM	WM	CSF
data_0	.8	.1	.1
data_1	.1	.8	.1
data_2	.1	.8	.1

Cross Entropy

True

	GM	WM	CSF
data_0	1	0	0
data_1	0	1	0
data_2	0	1	0

Predicted

	GM	WM	CSF
data_0	.8	.1	.1
data_1	.1	.8	.1
data_2	.1	.8	.1

$$CE = -\frac{1}{N}\sum_i^N \sum_c^C True_i^c \times log_2(Predicted_i^c)$$

$= 1 \text{ x } -\log_2(0.8) + 1 \text{ x } \log_2(.8) + 1 \text{ x } \log_2(.8)$

= 0.97

By improving our prediction, we've decreased the value of the cross entropy function

Outline

1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction

Metrics

1. Classification labels are integers
2. Dice Metric
   a. Binary classification
   b. Custom metric, not included by default in Keras
   c. Perfect overlap = (2 x |1|) / (1+1) = 1

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

Metrics

1. Classification labels are integers
   a. $\rightarrow$ Can't use MSE or similar metrics
2. Dice Metric
3. Categorical Accuracy
   a. Binary and multi-class labels
   b. Use 'acc' in Keras

$$\frac{1}{N}\sum_i I(Predicted_i = Label_i)$$

Metrics

1. Classification labels are integers
2. Dice Metric
3. Categorical Accuracy
4. Baseline for metrics is not usually not 0
    a. Ex. Guess all 0 $\longrightarrow$ metric = 0.6

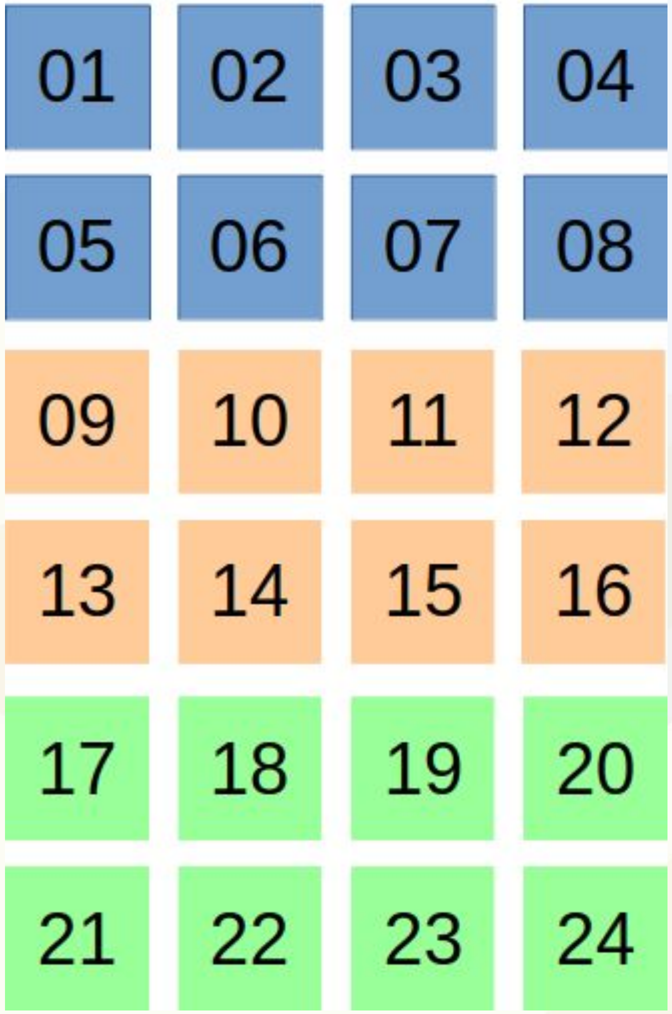
Outline

1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction

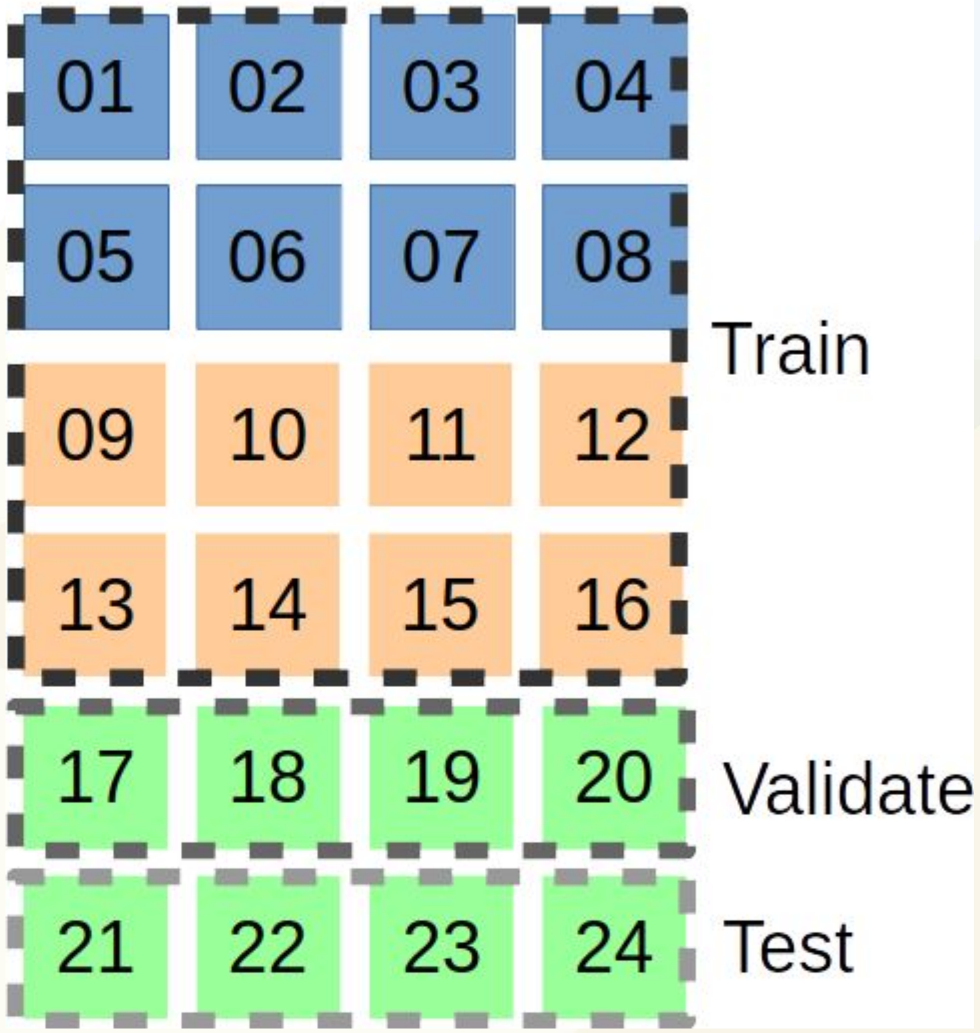
K-folds cross validation

🟥 01	🟥 02	🟥 03	🟥 04	🟩 05	🟩 06	🟩 07	🟩 08	🟦 09	🟦 10	🟦 11	🟦 12	🟦 13	🟦 14	🟦 15	🟦 16
🟦 01	🟦 02	🟦 03	🟦 04	🟥 05	🟥 06	🟥 07	🟥 08	🟩 09	🟩 10	🟩 11	🟩 12	🟦 13	🟦 14	🟦 15	🟦 16
🟦 01	🟦 02	🟦 03	🟦 04	🟦 05	🟦 06	🟦 07	🟦 08	🟥 09	🟥 10	🟥 11	🟥 12	🟩 13	🟩 14	🟩 15	🟩 16
🟩 01	🟩 02	🟩 03	🟩 04	🟦 05	🟦 06	🟦 07	🟦 08	🟦 09	🟦 10	🟦 11	🟦 12	🟥 13	🟥 14	🟥 15	🟥 16

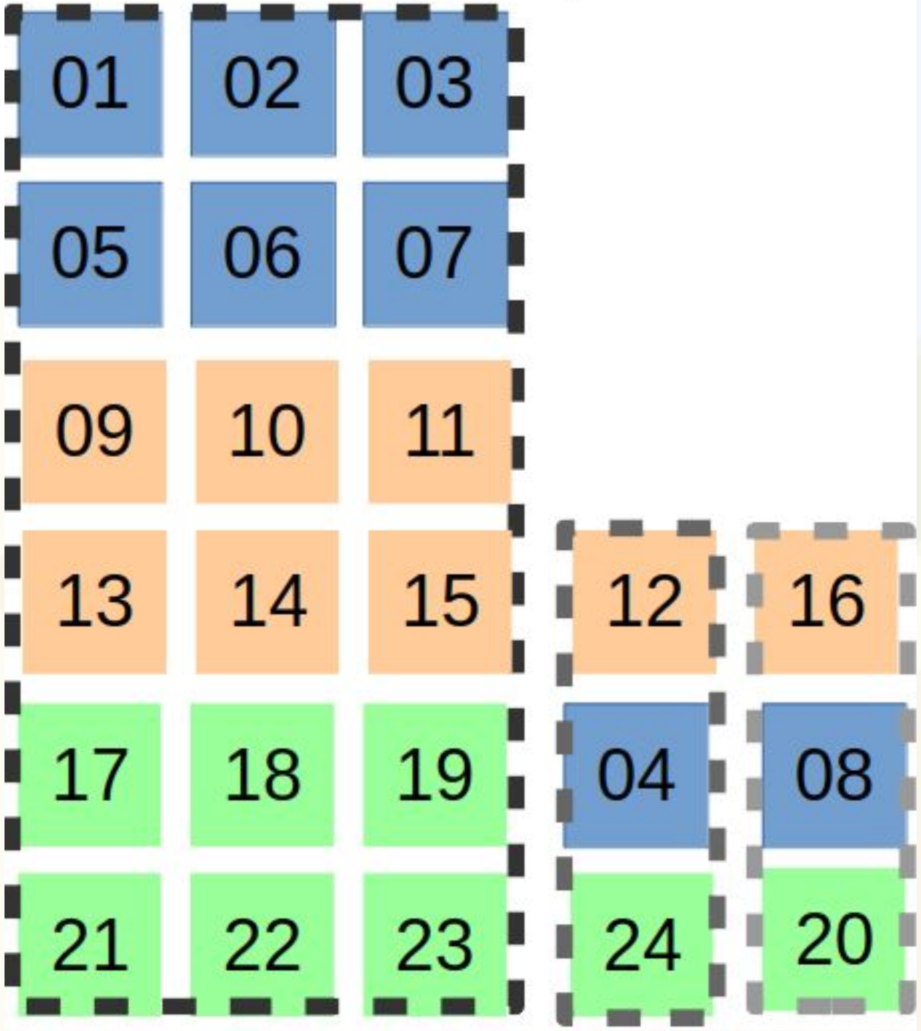
Data

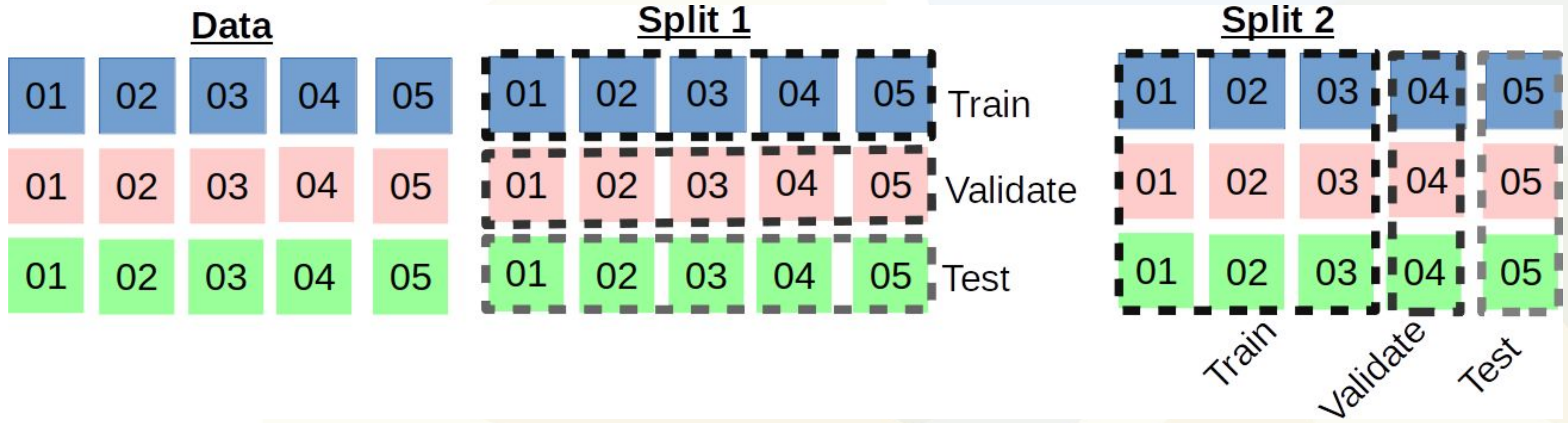


Split 1


01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
Train
17
18
19
20
Validate
21
22
23
24
Test

Split 2



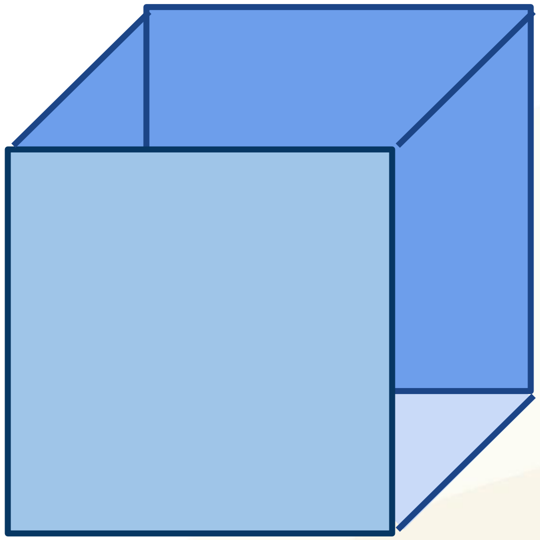

Data
01
02
03
04
05
01
02
03
04
05
01
02
03
04
05
Split 1
Train
Validate
Test
Split 2
Train
Validate
Test

Outline

1. Basic Concepts
   a. Kernels
   b. Receptive field & dilations
   c. Upsampling & downsampling
   d. Final activation functions
   e. Loss functions
   f. Metrics
   g. Cross-validation
   h. Feature extraction

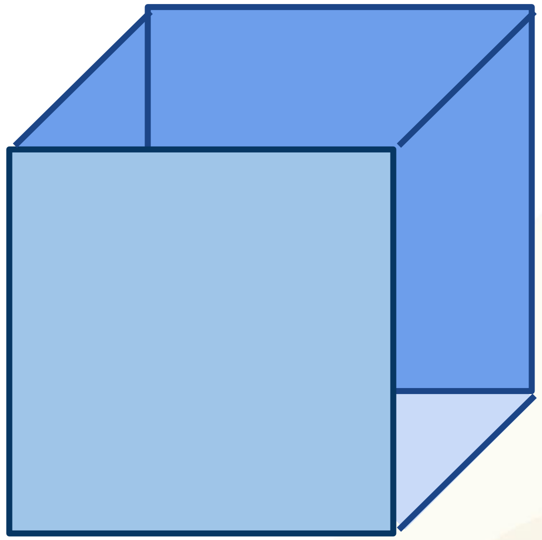
Many ways to analyze volumes

Full 3D Volume

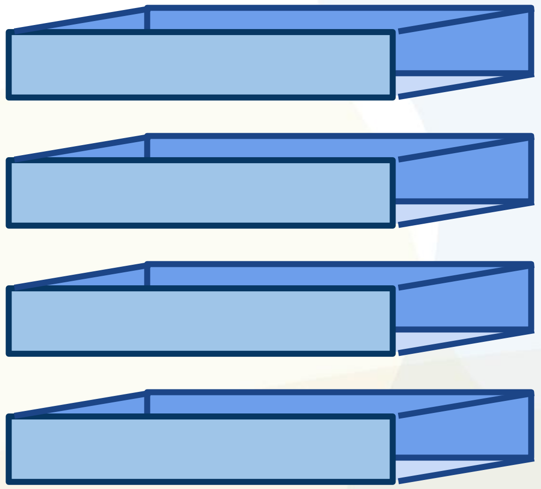


Many ways to analyze volumes

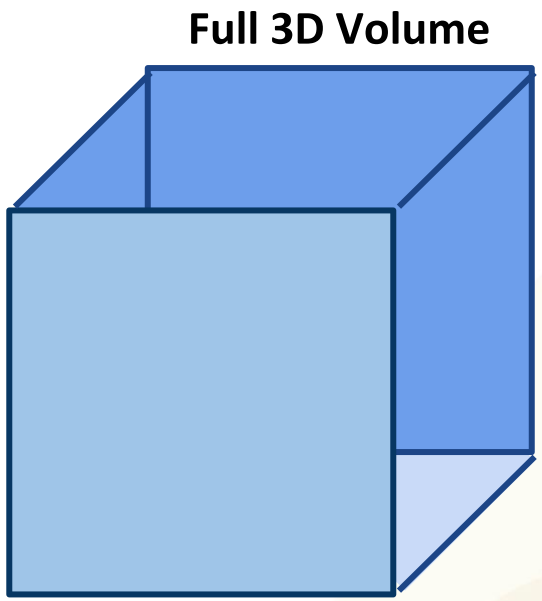
Full 3D Volume

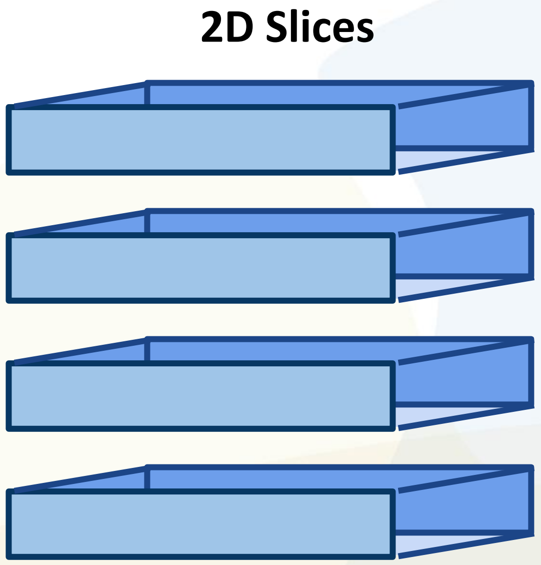


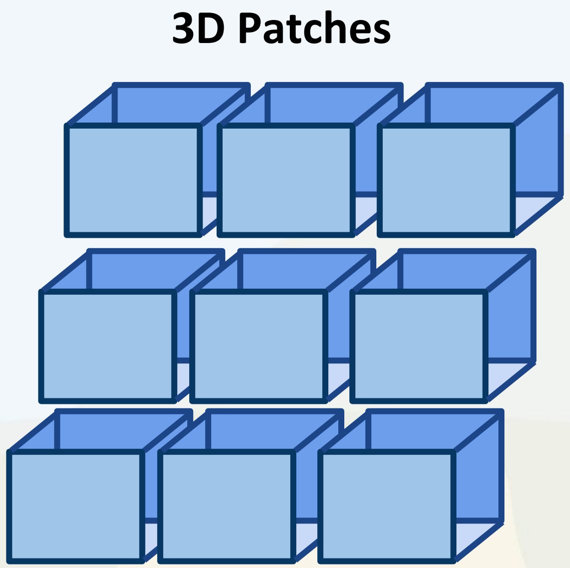
2D Slices



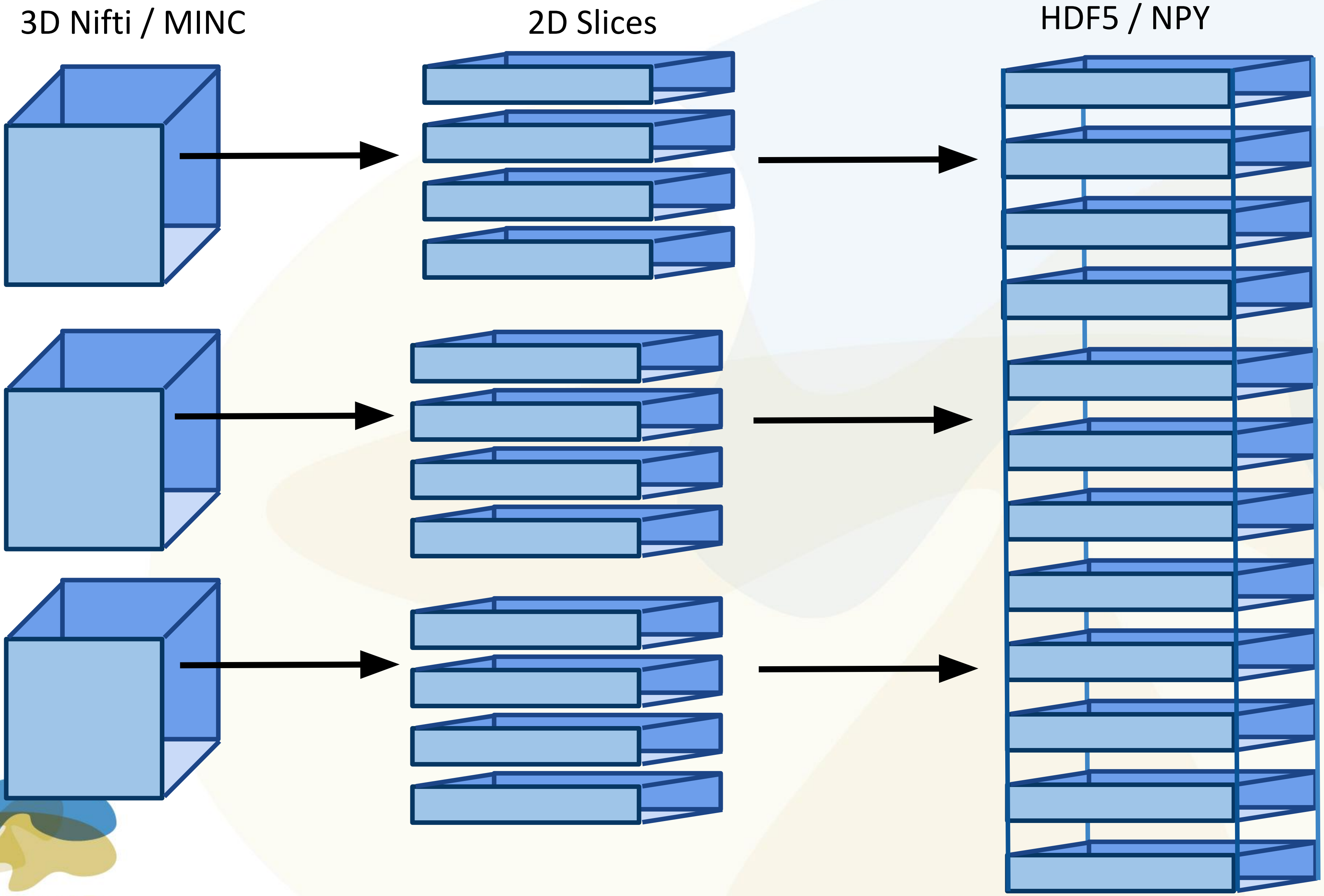
Many ways to analyze volumes







Formatting data to feed into Keras



More info

1. Overview
   a. "**Convolutional Neural Nets for Visual Recognition**", http://cs231n.github.io/convolutional-networks/, karpathy@cs.stanford.edu
      i. *This is a very good overview. Definitely read!*
   b. **"Deep Learning"** *http://neuralnetworksanddeeplearning.com/chap6.html* , Michael Nielsen
2. Kernels
   a. "**Convolutional Neural Networks (CNN, or ConvNets)**", https://medium.com/@phidaouss/convolutional-neural-networks-cnn-or-convnets-d7c688b0a207, Firdaouss Doukkali.
3. Softmax
   a. "**Difference Between Softmax Function and Sigmoid Function**", http://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/ , Saimadhu Polamuri .
4. Cross Entropy
   a. **"A Short Introduction to Entropy, Cross-Entropy and KL-Divergence",** Aurélien Géron**.** https://www.youtube.com/watch?v=ErfnhcEV1O8 .
   b. **A Friendly Introduction to Cross-Entropy Loss**, Rob DiPietro. https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/.