University of Antwerp

Artificial Neural Networks

[2500WETANN]

José Oramas

Convolutional Neural Networks

[Part 1 – Foundations]

José Oramas

University of Antwerp

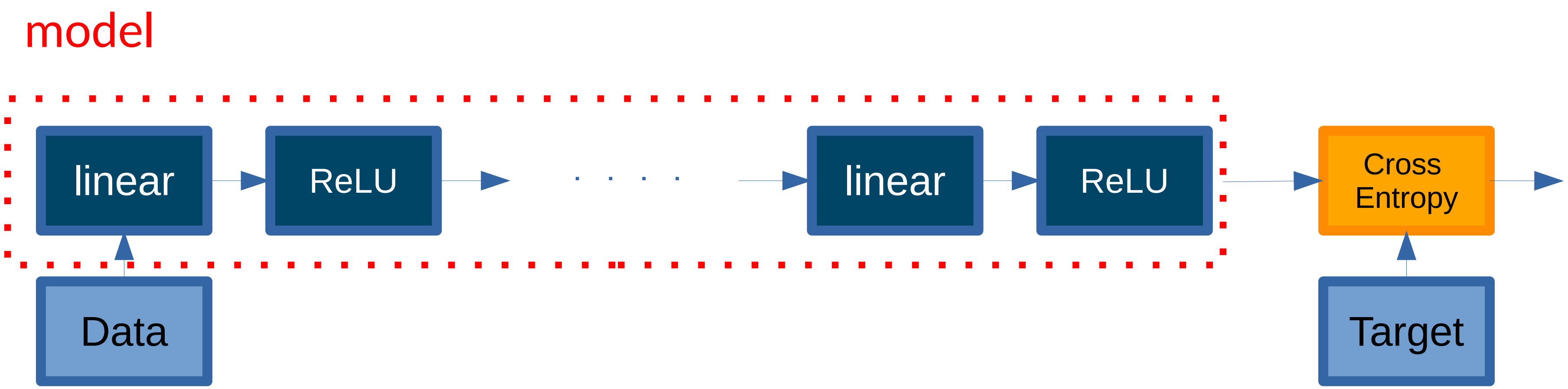
Previous lecture

[Shallow / Deep Neural Networks]

Recap Previous Lecture

$$f(x)$$

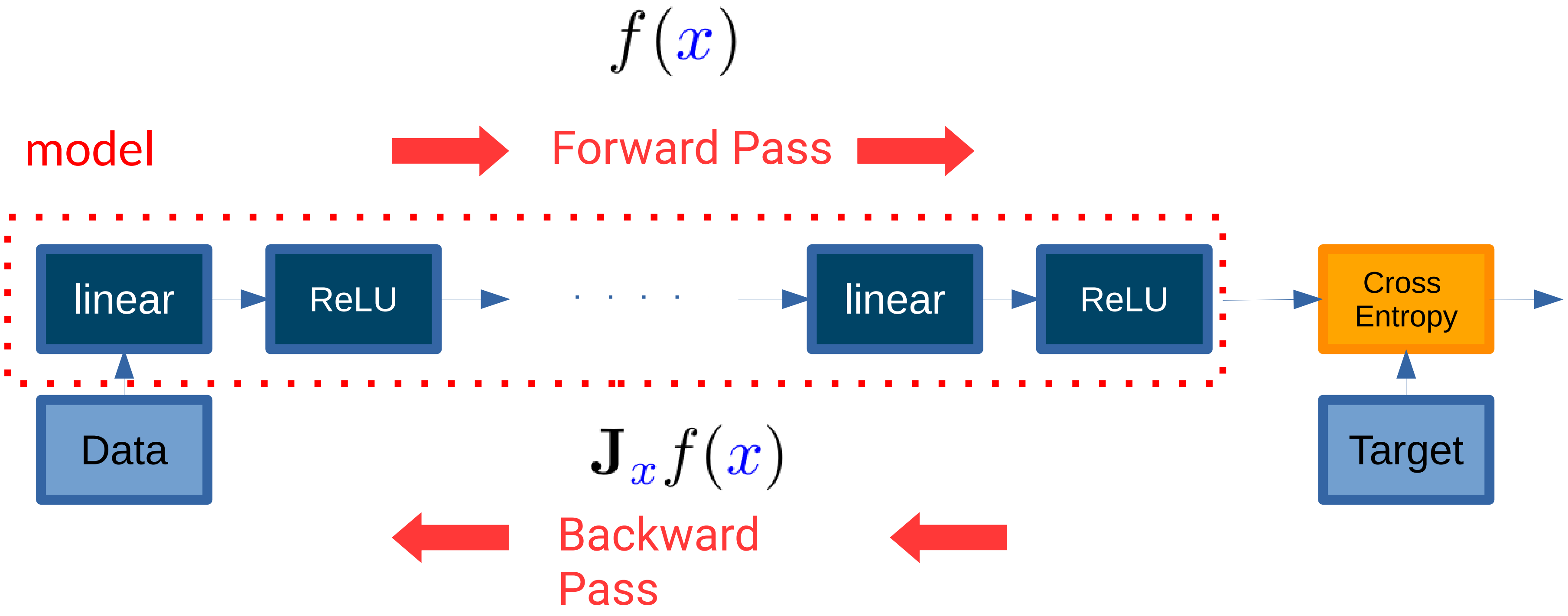
Forward Pass



$$\mathbf{J}_x f(x)$$

Backward Pass

Recap Previous Lecture



- Universal Approximation Theorem
- Use gradients to optimize weights w.r.t. prediction *(loss function)*
- Use of ReLU instead of sigmoid activation functions when going deeper
- In the early days → provide forward/backward operations

Convolutional Neural Networks

[Part 1 – Foundations]

Let’s consider the case of visual data

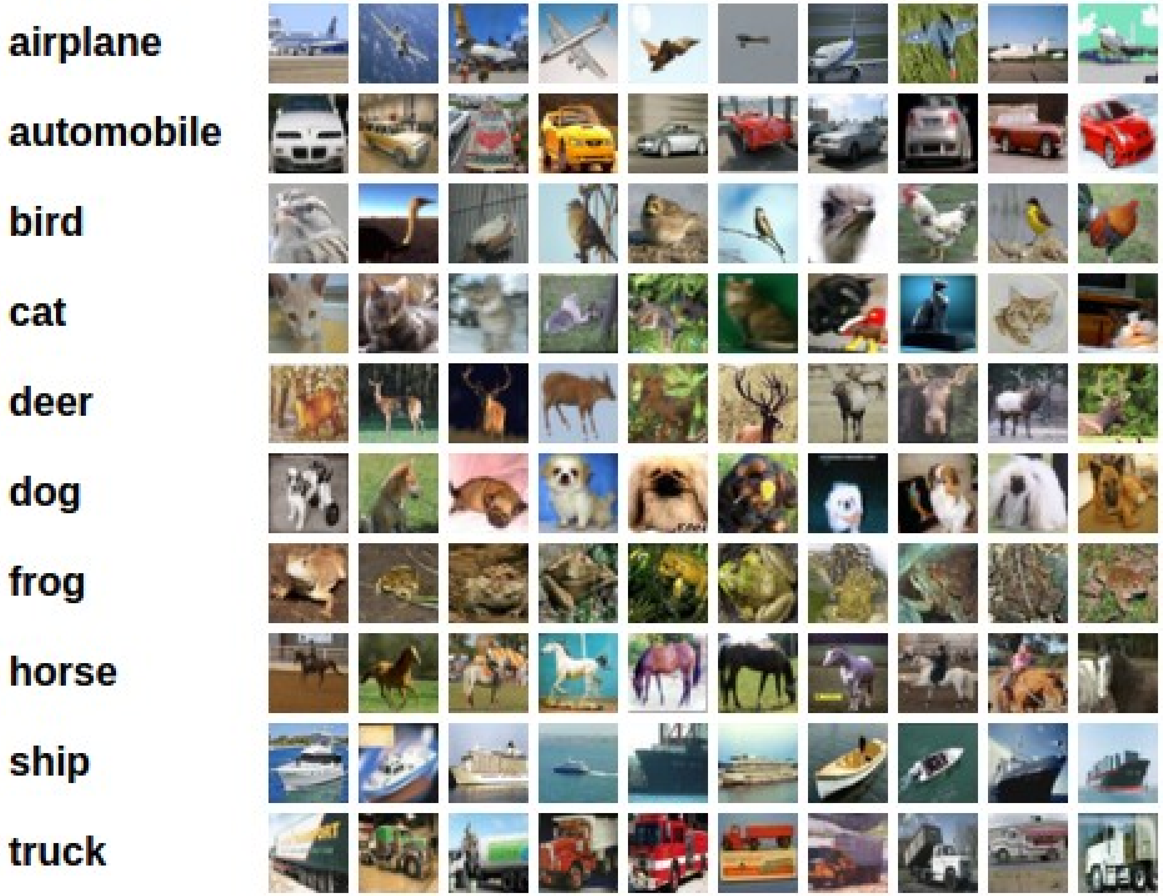


Image Recognition task

Given:

- an input image *x*

Do:

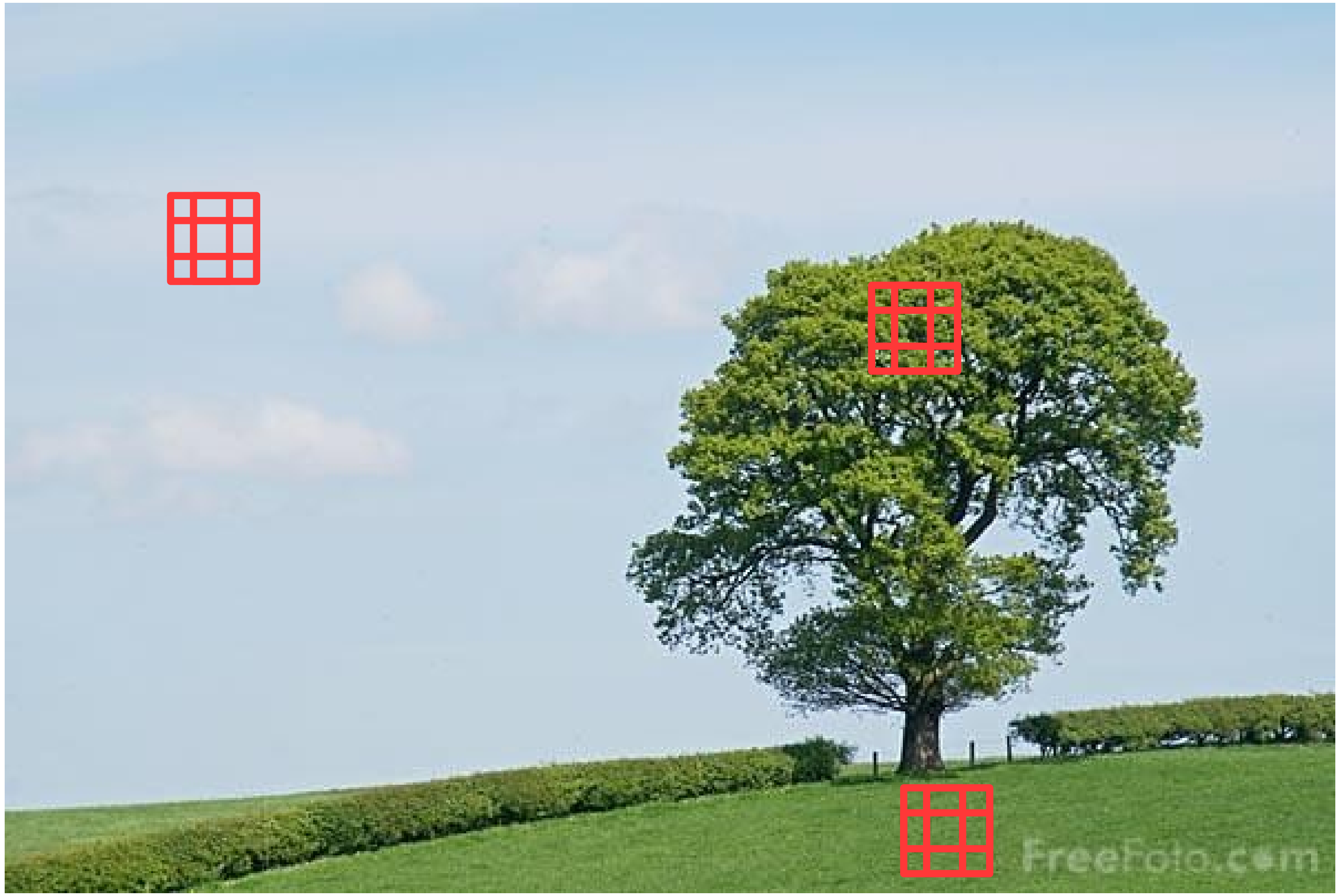
- predict a label *y*

  (out of a set of class labels)

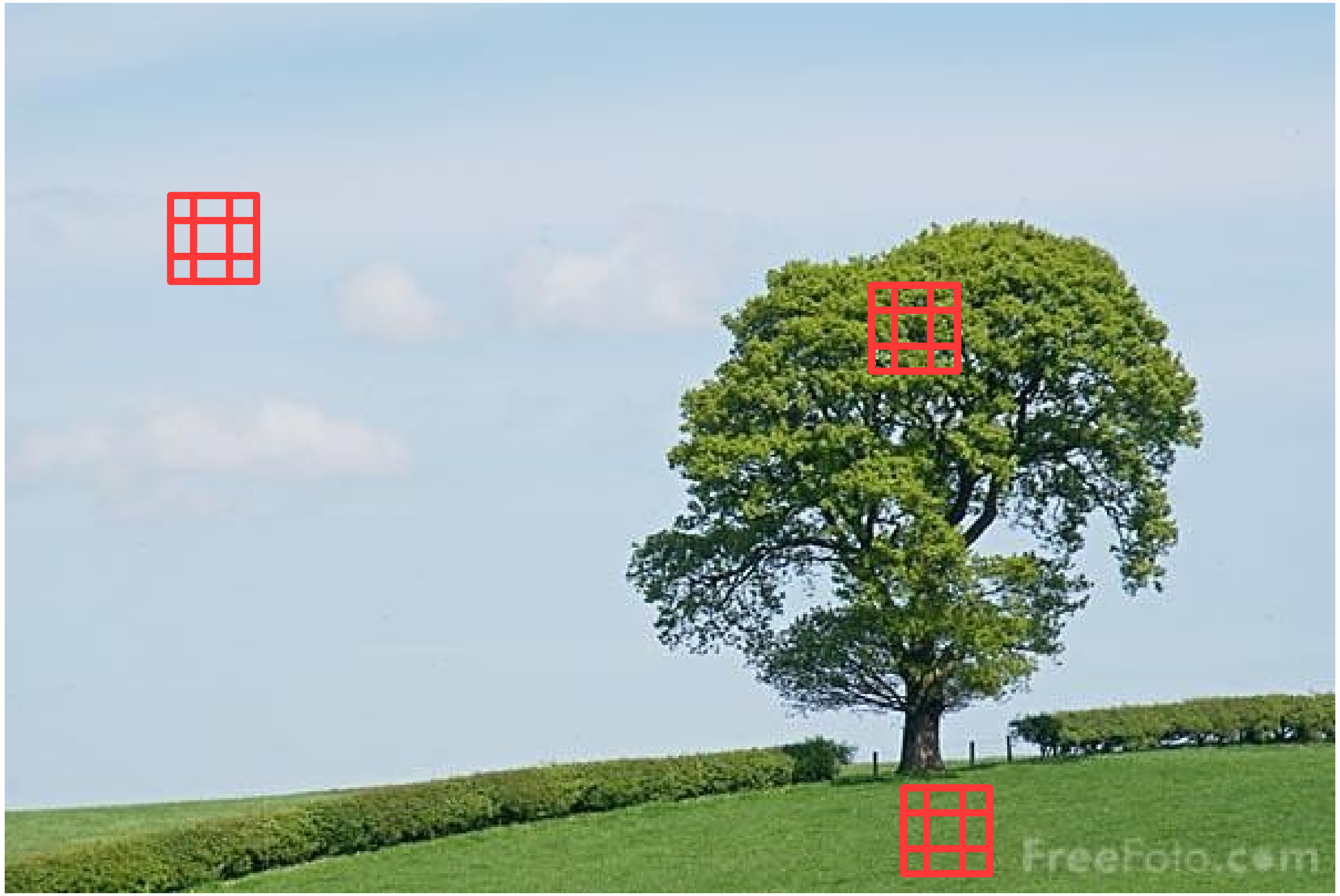
Some Motivations

Some Characteristics of Visual Data

Some Characteristics of Visual Data



Some Characteristics of Visual Data



Locality

- Neighbooring pixels are highly correlated.

Some Characteristics of Visual Data

Some Characteristics of Visual Data

Translation Invariance

- Meaningful patterns can appear anywhere

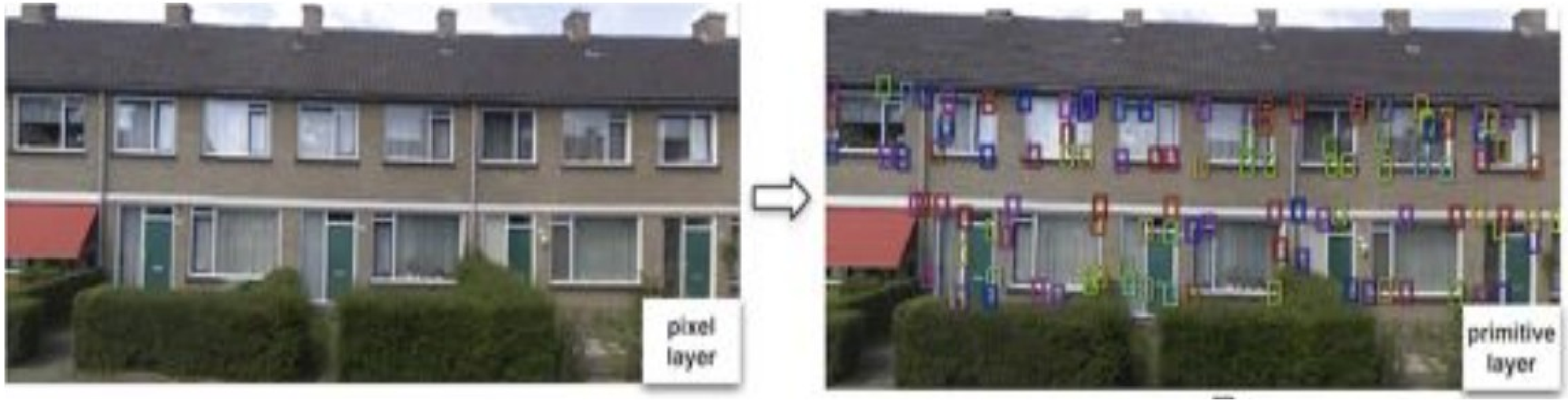
Some Characteristics of Visual Data

Translation Invariance

- Meaningful patterns can appear anywhere

Some Characteristics of Visual Data

Some Characteristics of Visual Data



Some Characteristics of Visual Data

[Antanas et al., 2014]

Some Characteristics of Visual Data

[Antanas et al., 2014]

Compositionality

- Learning feature hierarchies

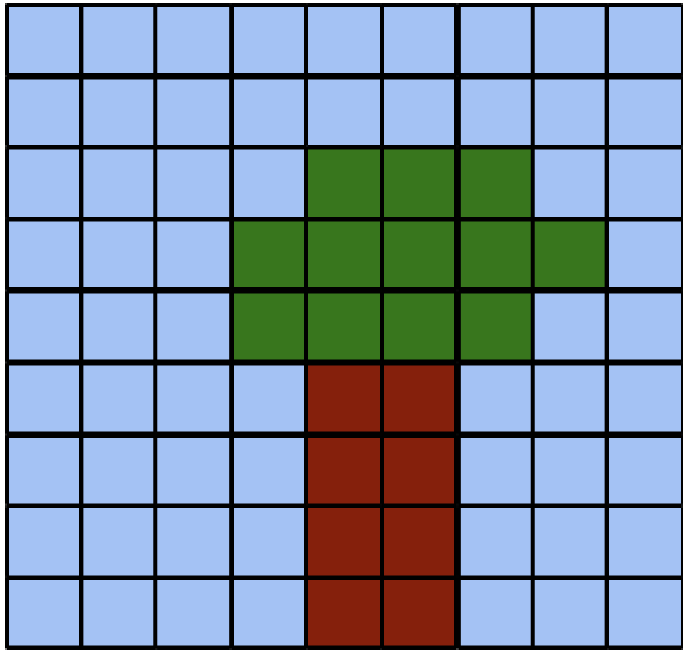
Yes, but…

How to put that in practice?

Feeding Images to a Neural Network

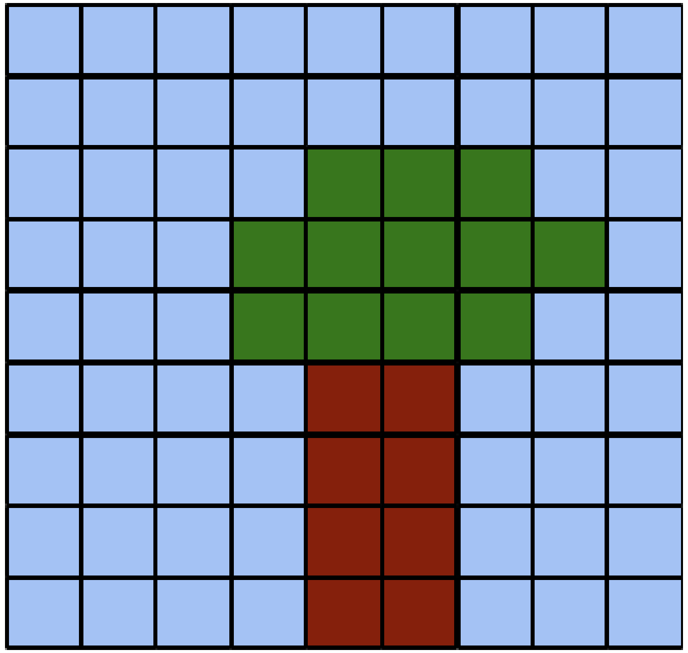
- Digital image → **2D pixel matrix**

Feeding Images to a Neural Network



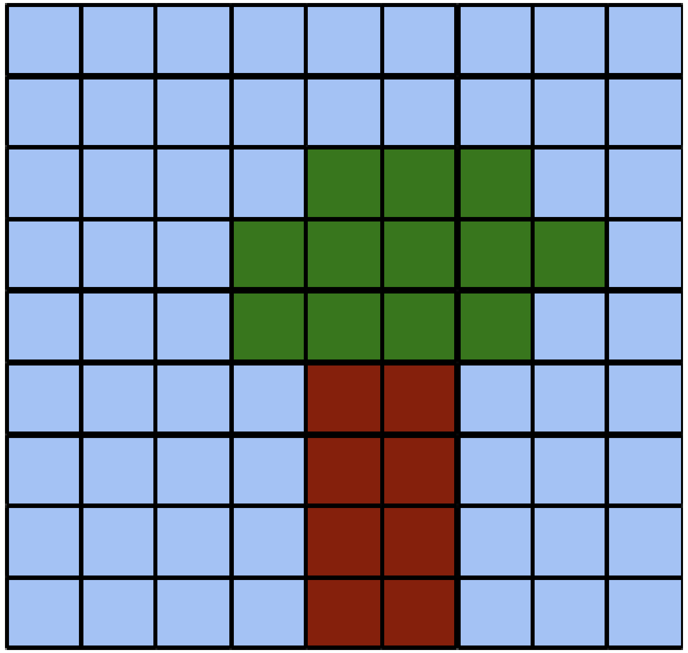
- Digital image → **2D pixel matrix**

Feeding Images to a Neural Network



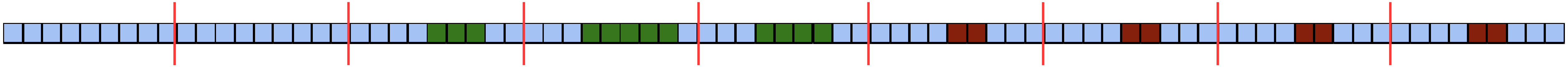
- Digital image → **2D pixel matrix**
- Our previous network expects a **vector of numbers as input**

Feeding Images to a Neural Network

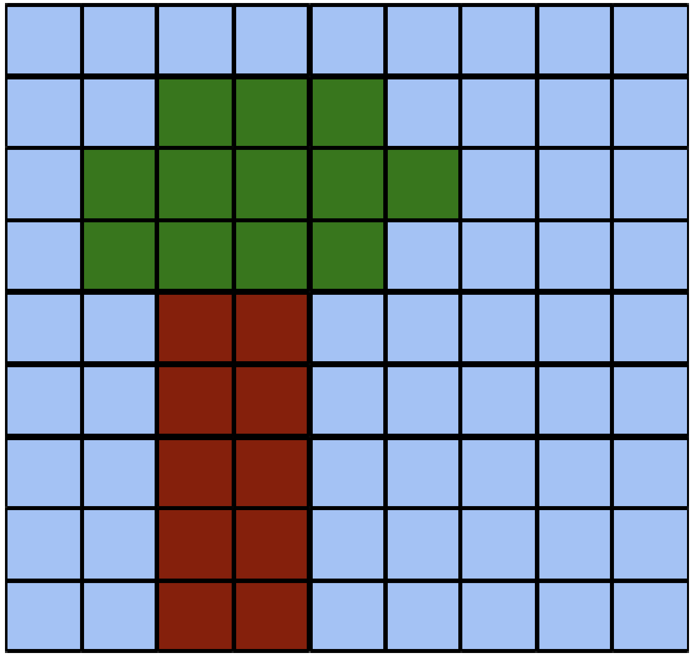


- Digital image → **2D pixel matrix**
- Our previous network expects a **vector of numbers as input**

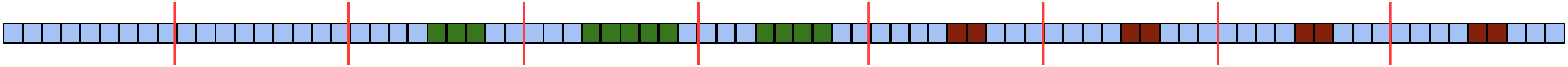
Original



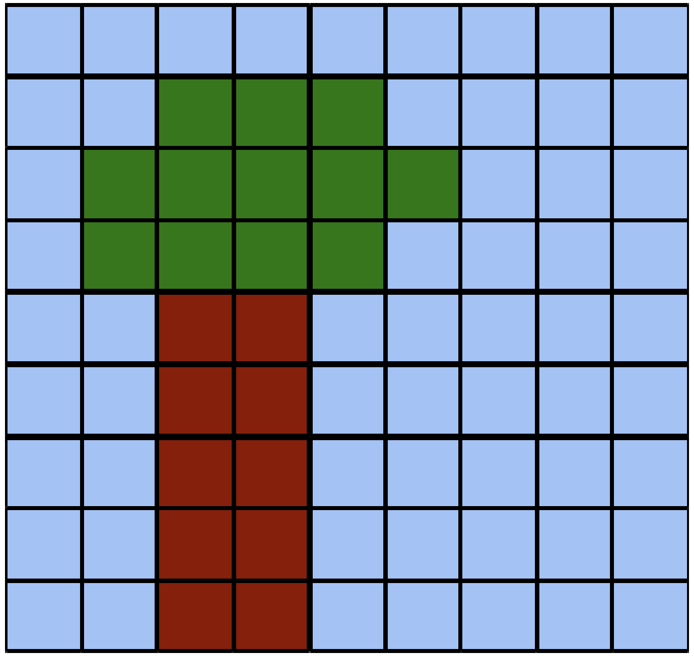
Feeding Images to a Neural Network



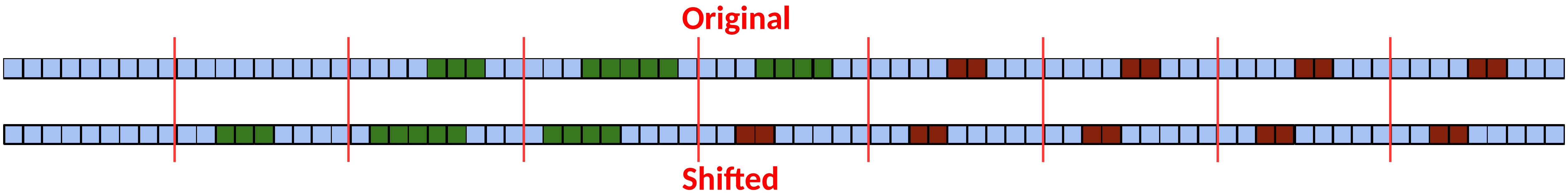
- Digital image → **2D pixel matrix**
- Our previous network expects a **vector of numbers as input**



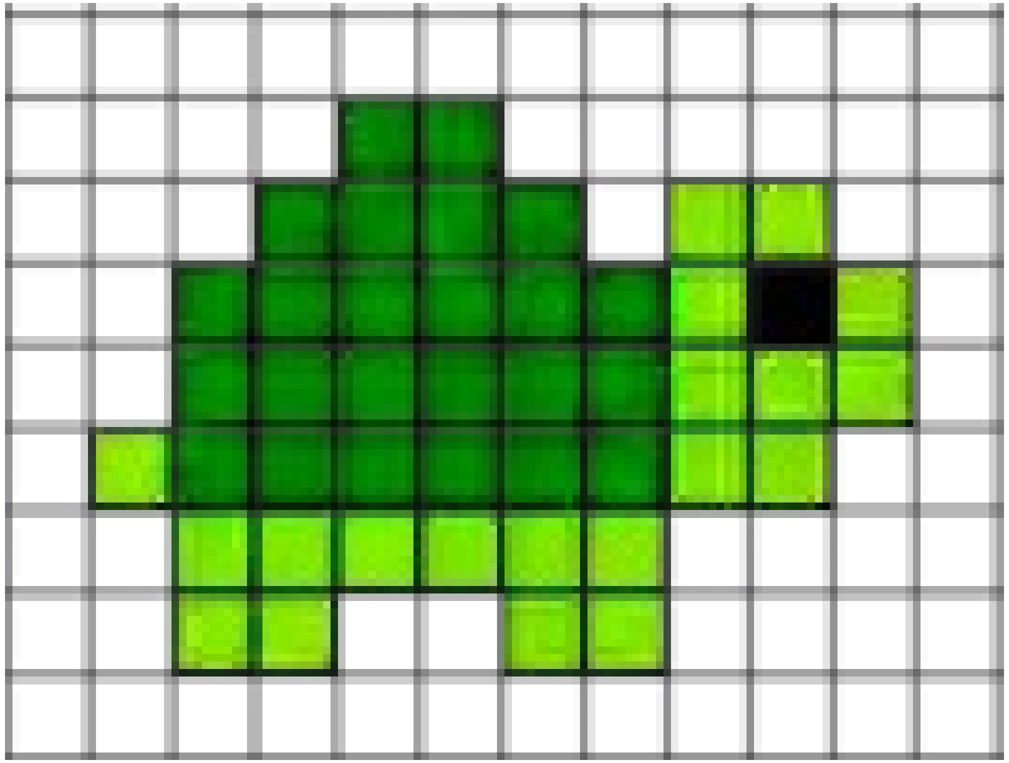
Feeding Images to a Neural Network



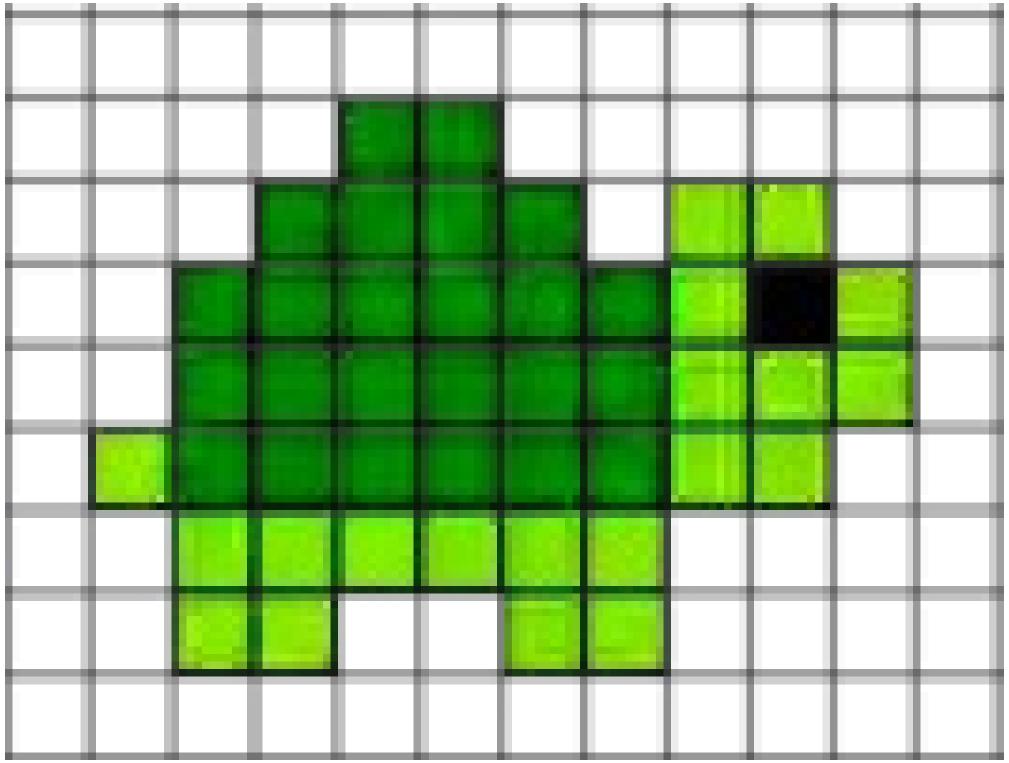
- Digital image → **2D pixel matrix**
- Our previous network expects a **vector of numbers as input**

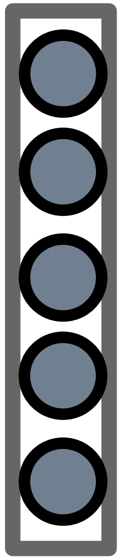


Earlier Setting – looking at the complete image

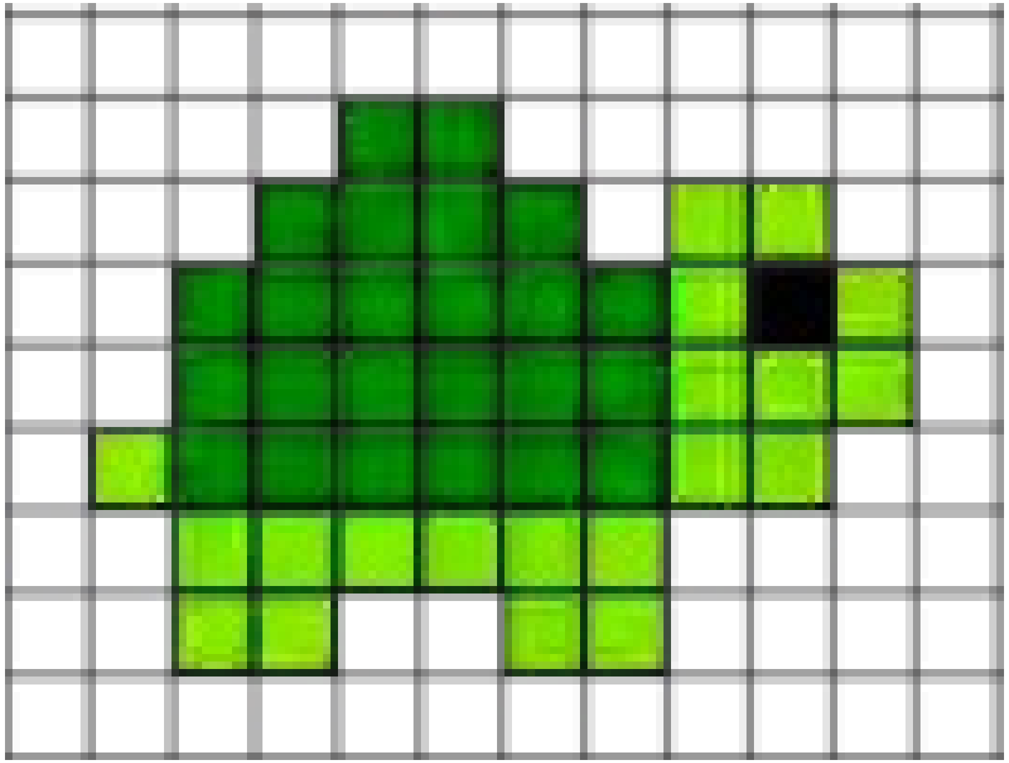


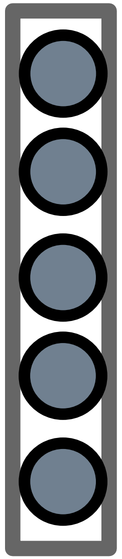
Earlier Setting – looking at the complete image





Earlier Setting – looking at the complete image



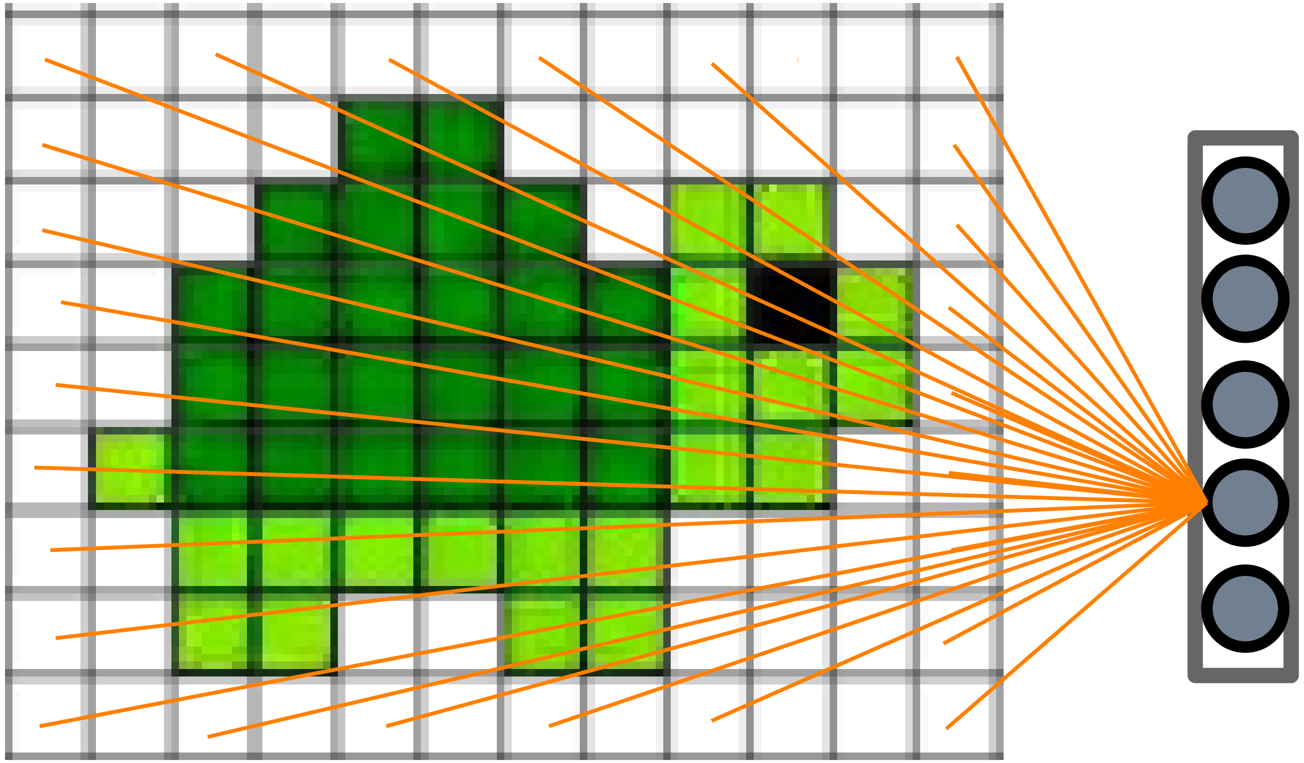


- Dense fully-connected layer

$$\sum_{i=1}^{d} w_i x_i + b$$

$$\sum_{i=0}^{d} w_i x_i, \quad x_0 := 1$$

Earlier Setting – looking at the complete image

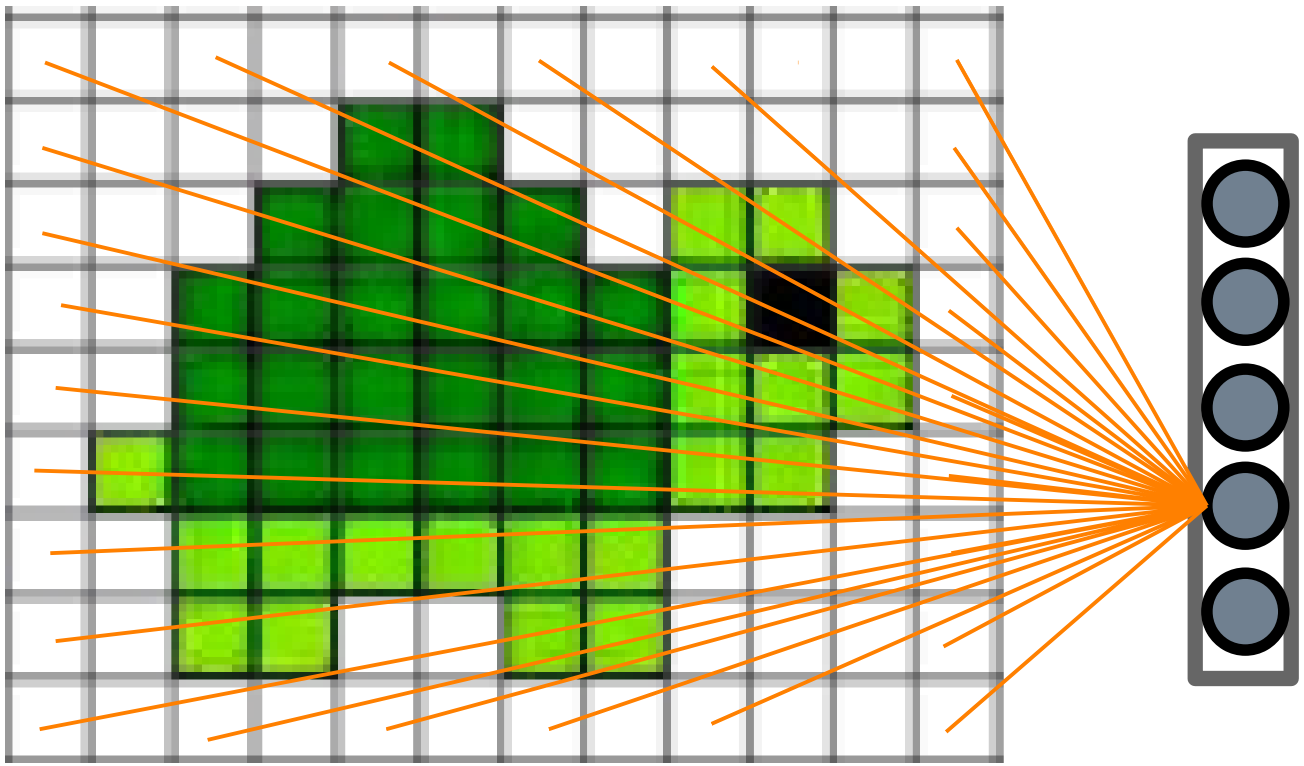


- Dense fully-connected layer

$$\sum_{i=1}^{d} w_i x_i + b$$

$$\sum_{i=0}^{d} w_i x_i, \quad x_0 := 1$$

Earlier Setting – looking at the complete image



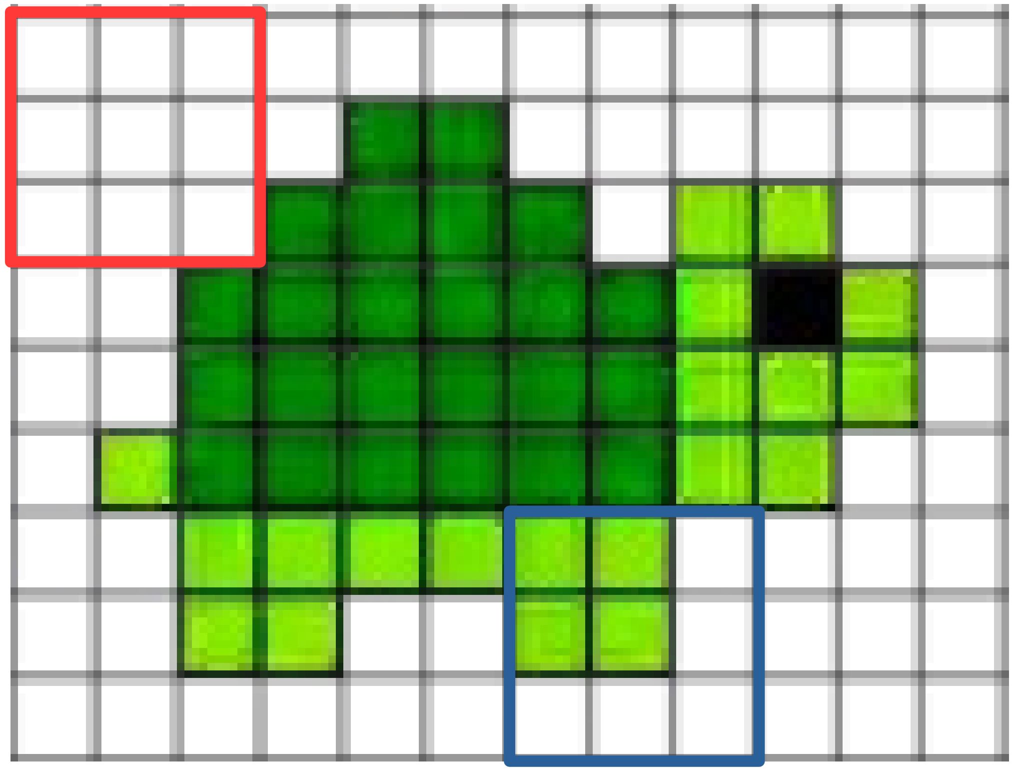
- Dense fully-connected layer

$$\sum_{i=1}^{d} w_i x_i + b$$

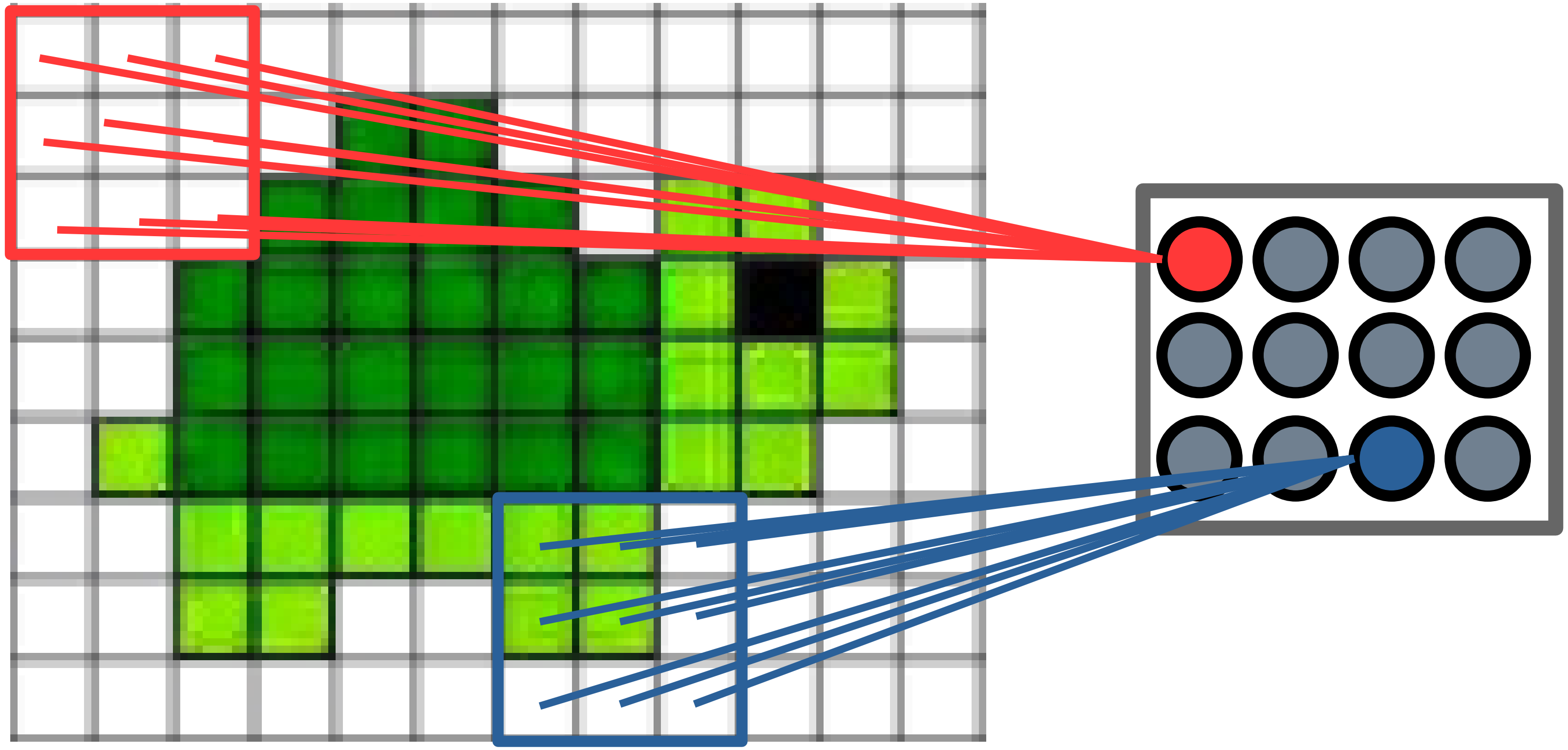
$$\sum_{i=0}^{d} w_i x_i, \quad x_0 := 1$$

i covers the entire input [image] space

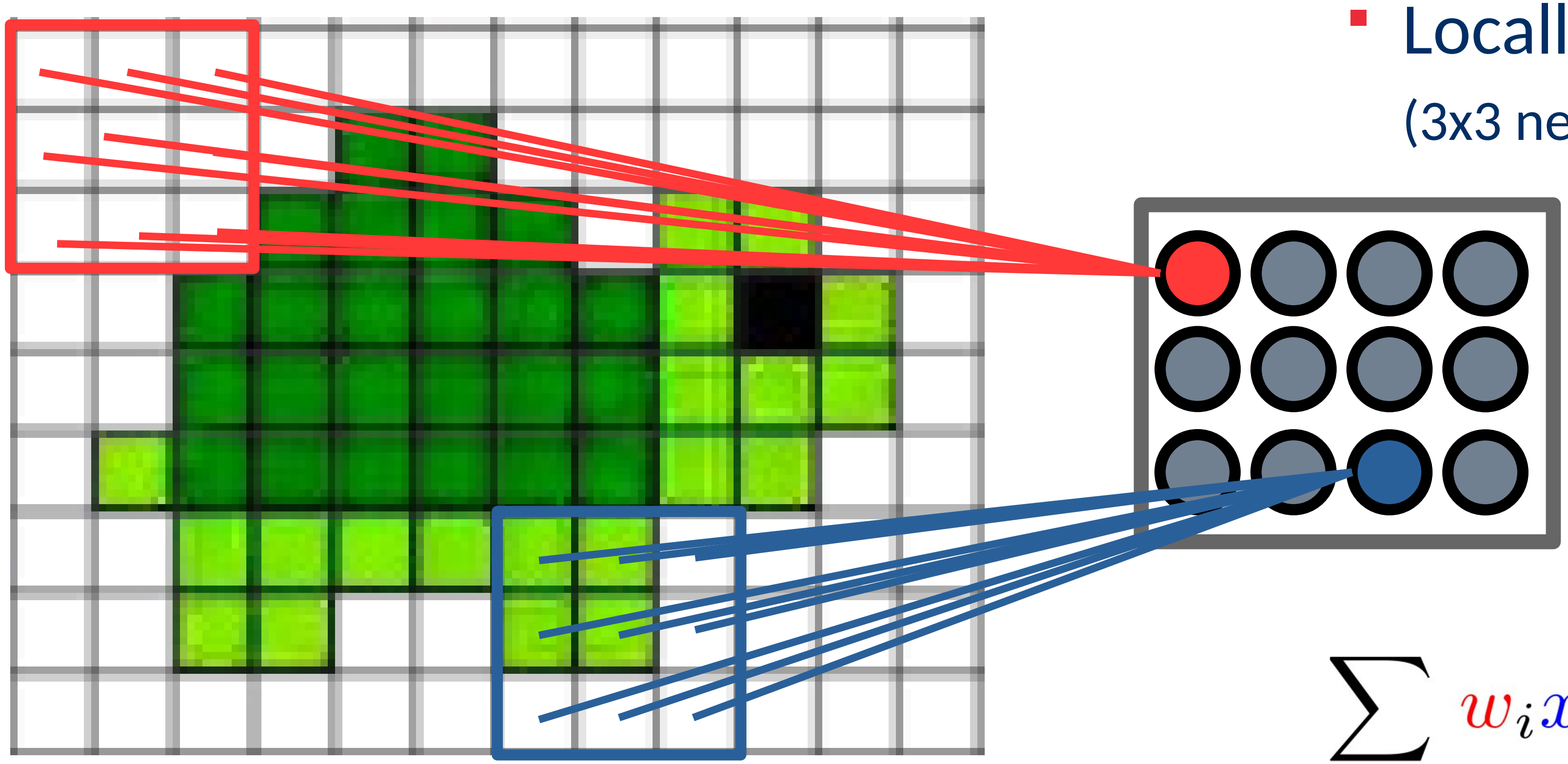
Connecting Neighboring Regions



Connecting Neighboring Regions



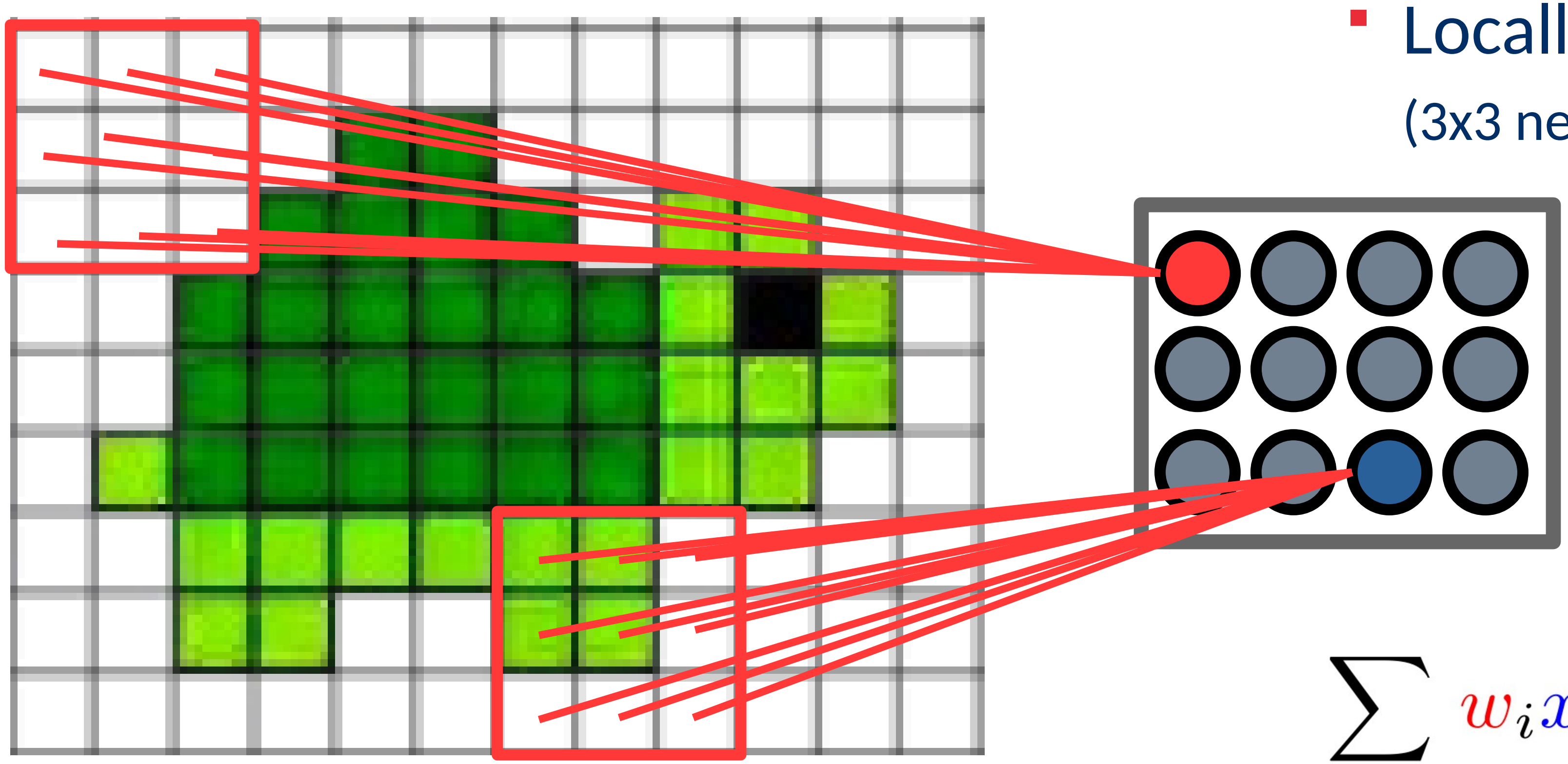
Connecting Neighboring Regions



- Locally-connected Neurons
  (3x3 neighborhood)

$$\sum_{i \in N_i} w_i x_i + b$$

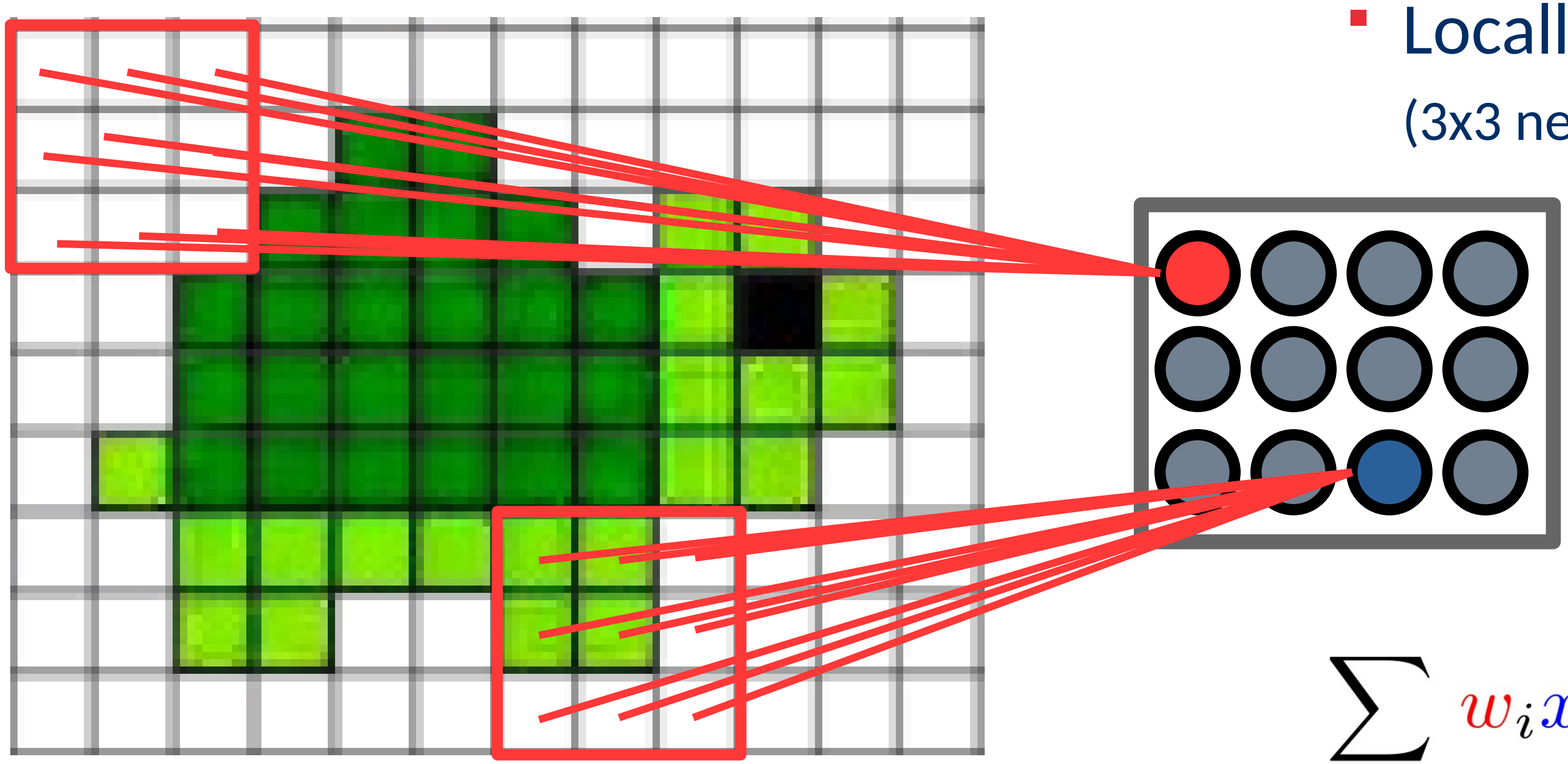
From Locally Connected to Convolutions



- Locally-connected Neurons
  (3x3 neighborhood)
- Weights are shared
  (over the space)

$$\sum_{i \in N_i} w_i x_i + b$$

From Locally Connected to Convolutions



- Locally-connected Neurons

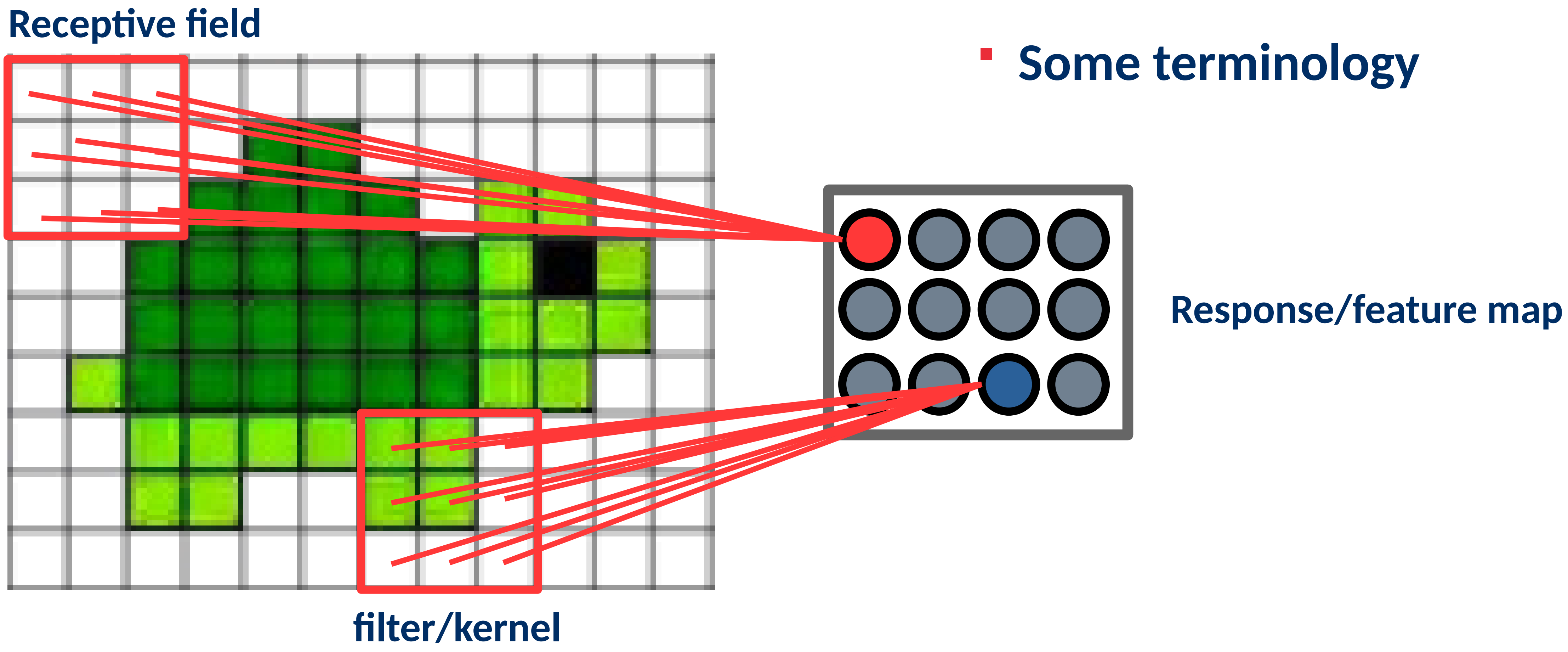
  (3x3 neighborhood)

- Weights are shared

  (over the space)

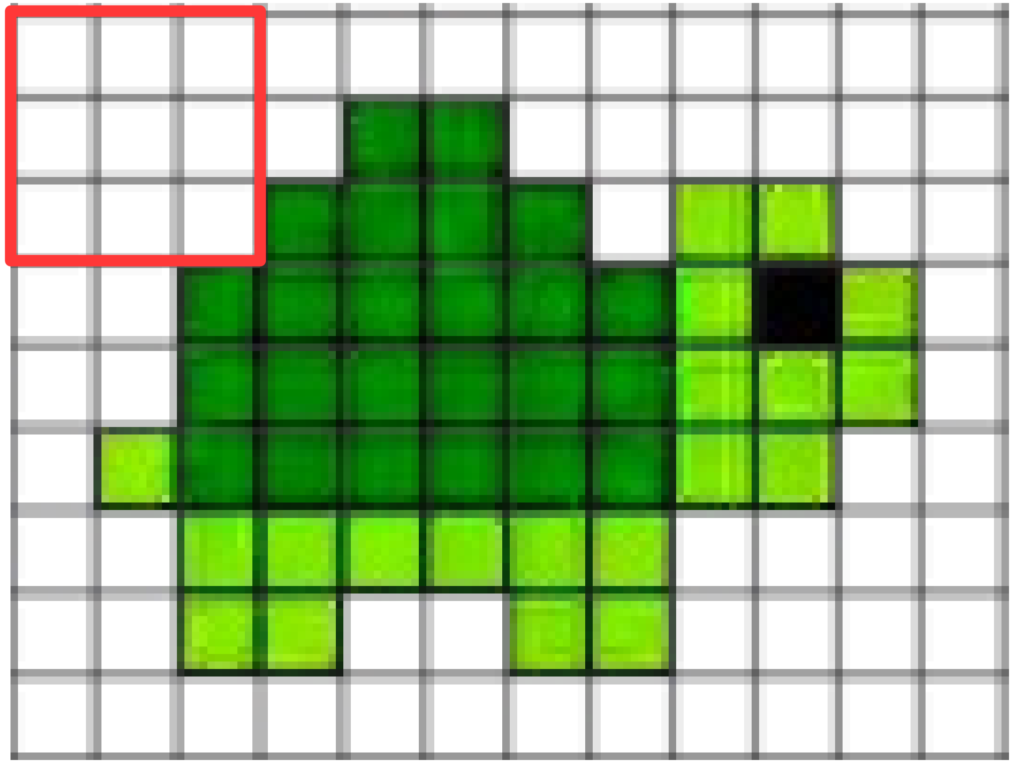
$$\sum_{i \in N_i} w_i x_i + b \rightarrow w * x + b$$

From Locally Connected to Convolutions

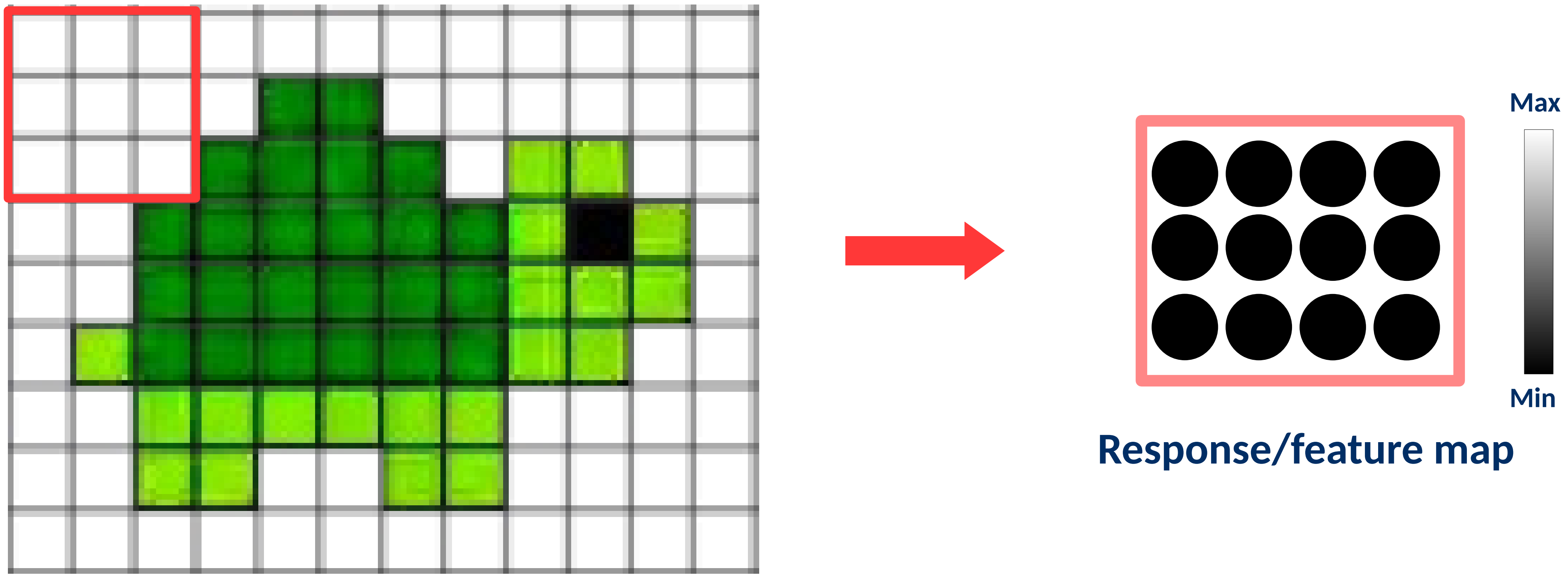


- Some terminology

In Practice: From Locally Connected to Convolutions

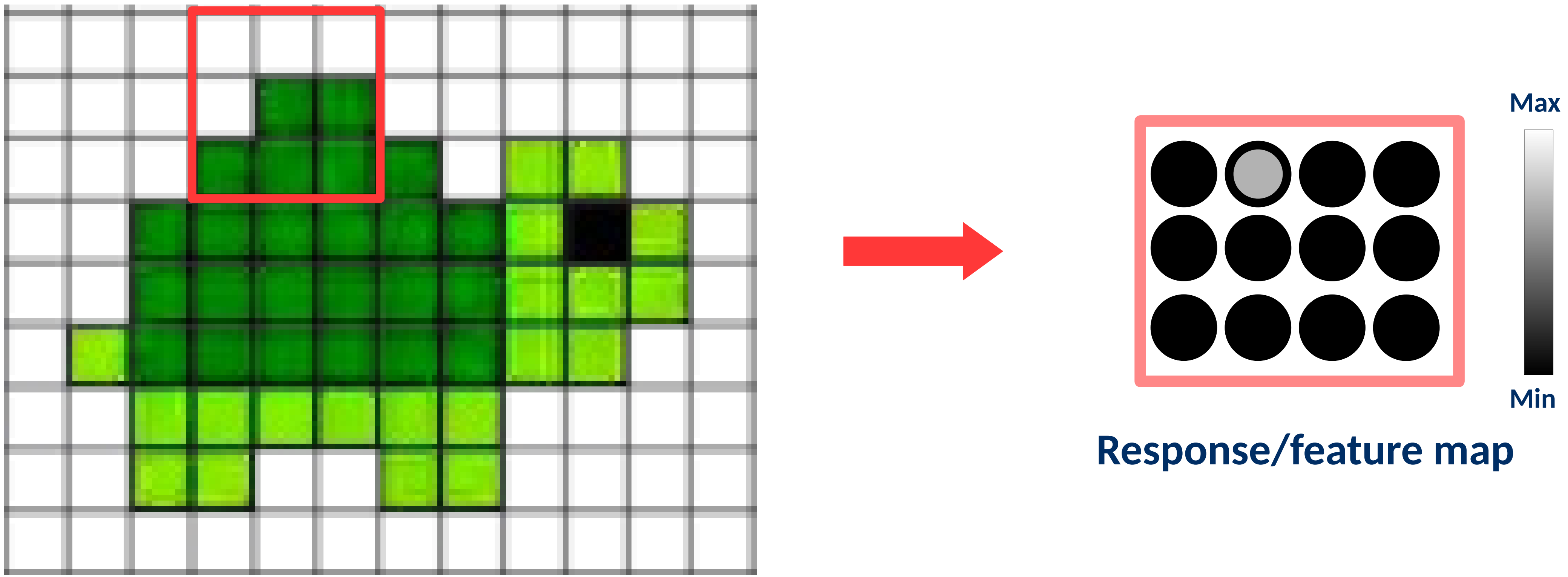


In Practice: From Locally Connected to Convolutions

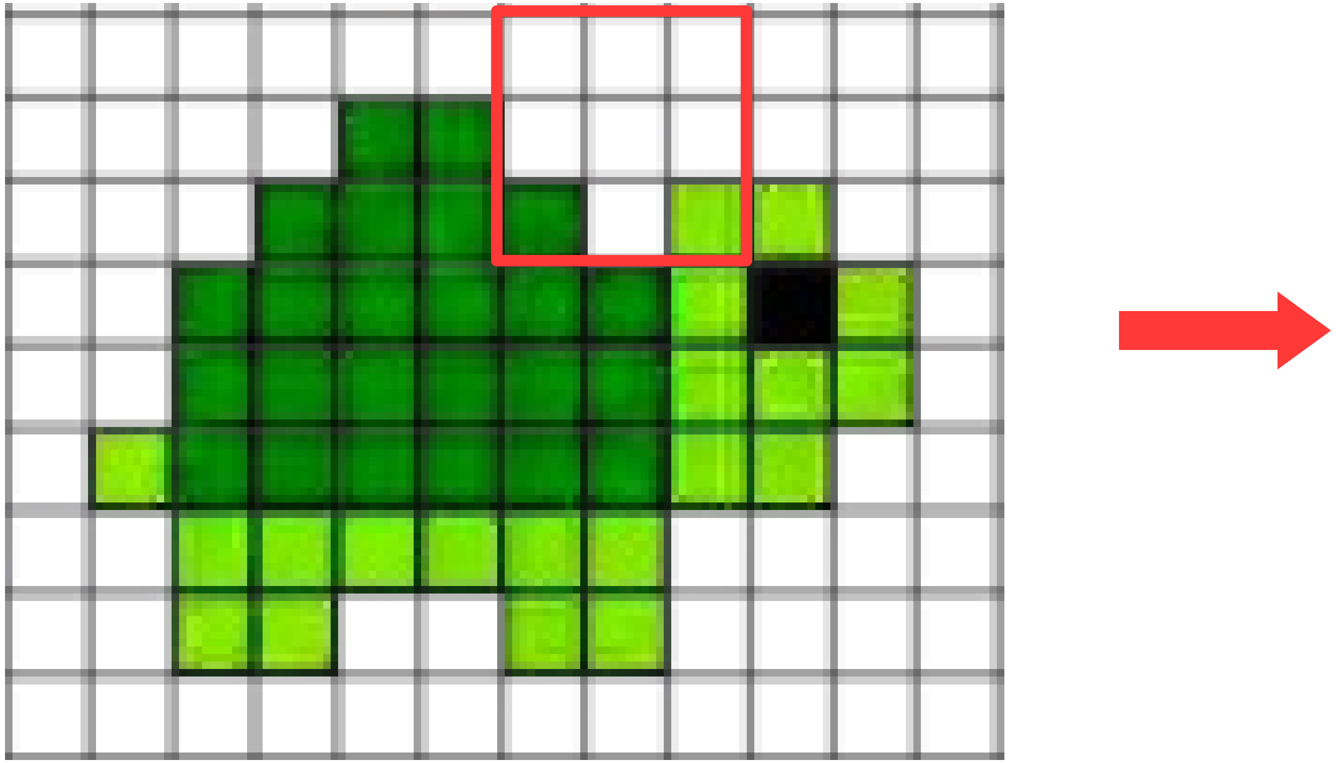


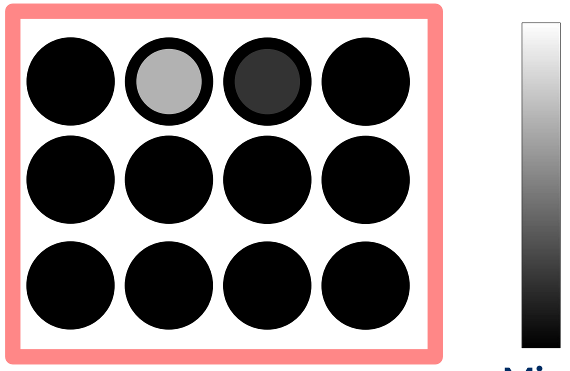
Response/feature map

In Practice: From Locally Connected to Convolutions



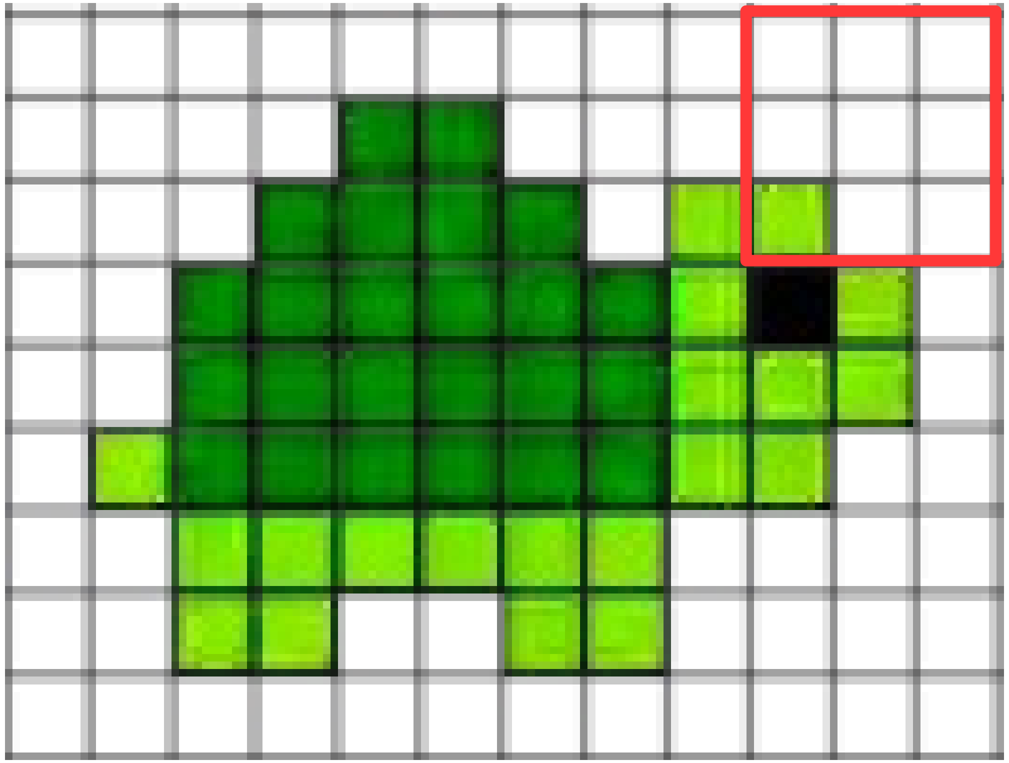
In Practice: From Locally Connected to Convolutions

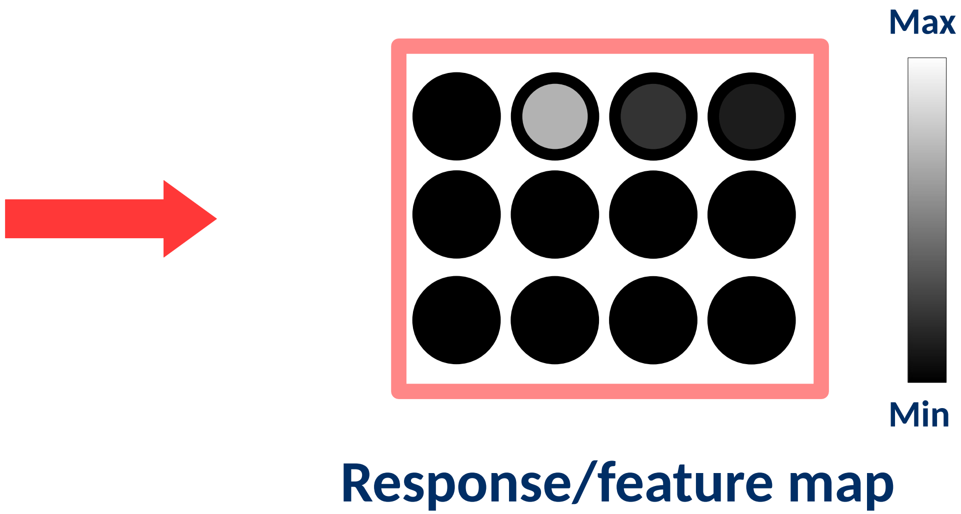




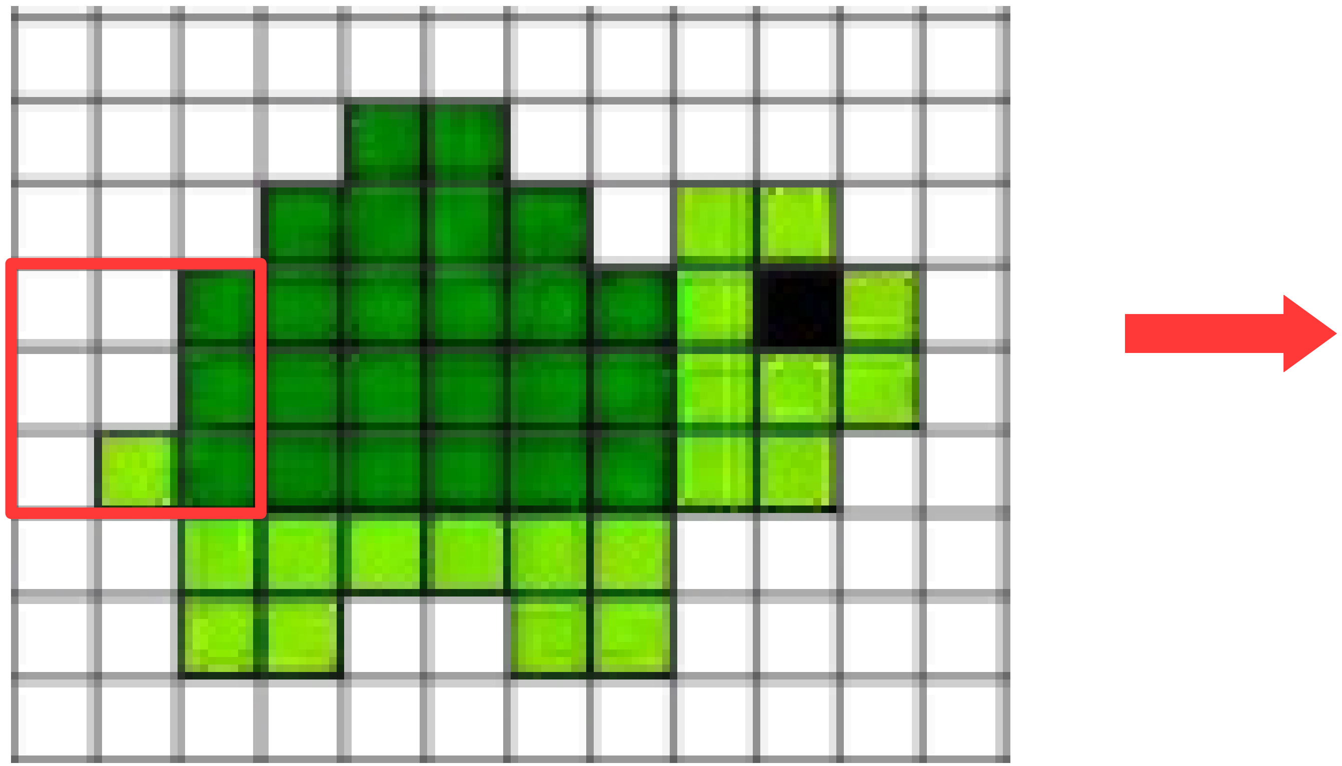
Response/feature map

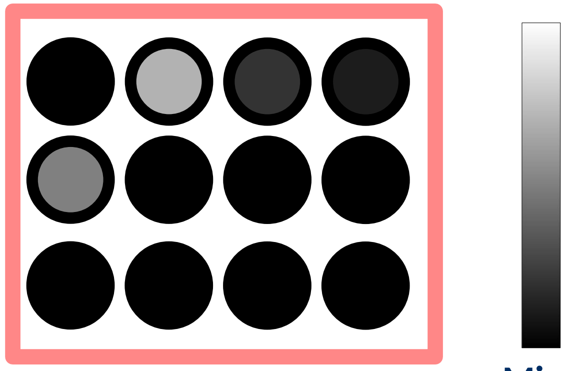
In Practice: From Locally Connected to Convolutions





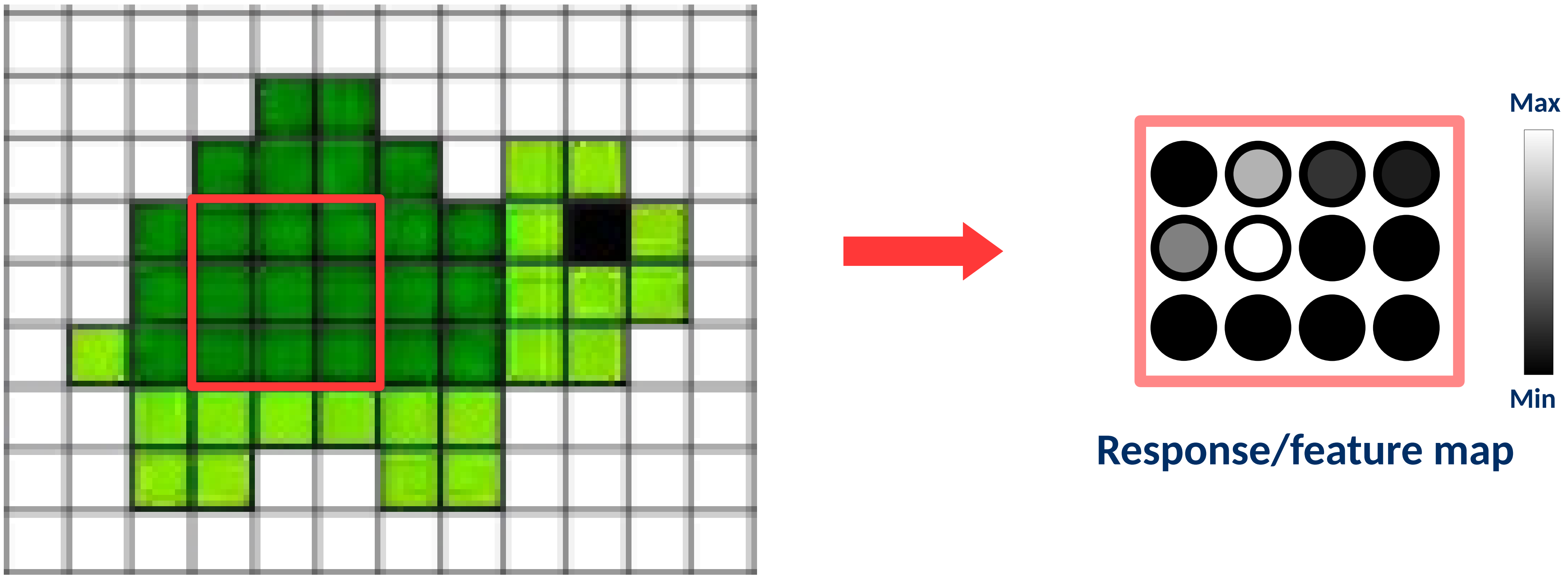
In Practice: From Locally Connected to Convolutions



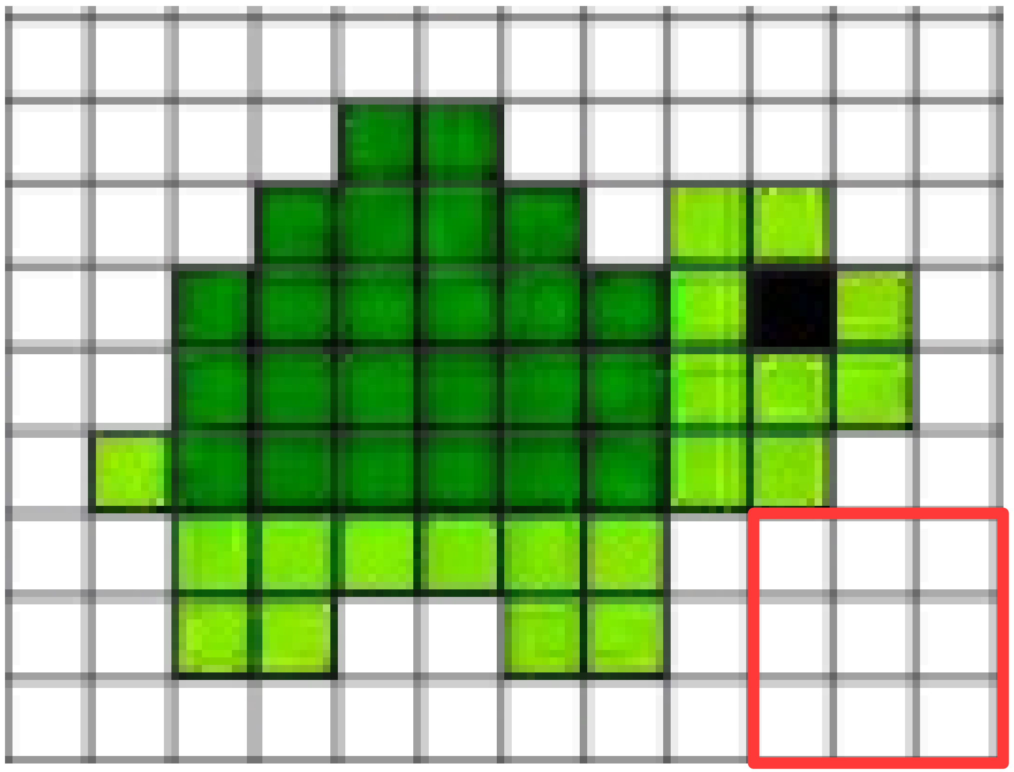


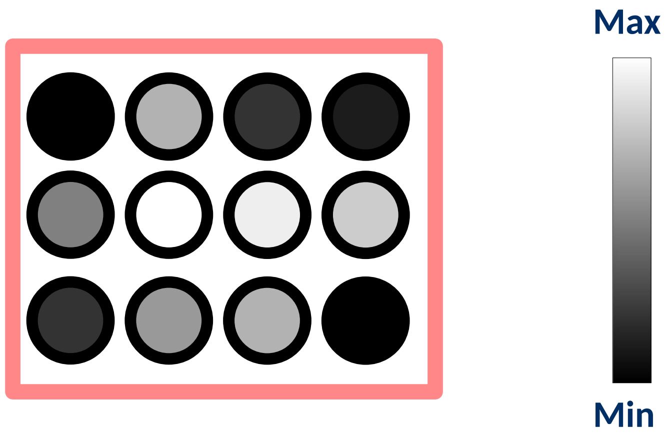
Response/feature map

In Practice: From Locally Connected to Convolutions



In Practice: From Locally Connected to Convolutions

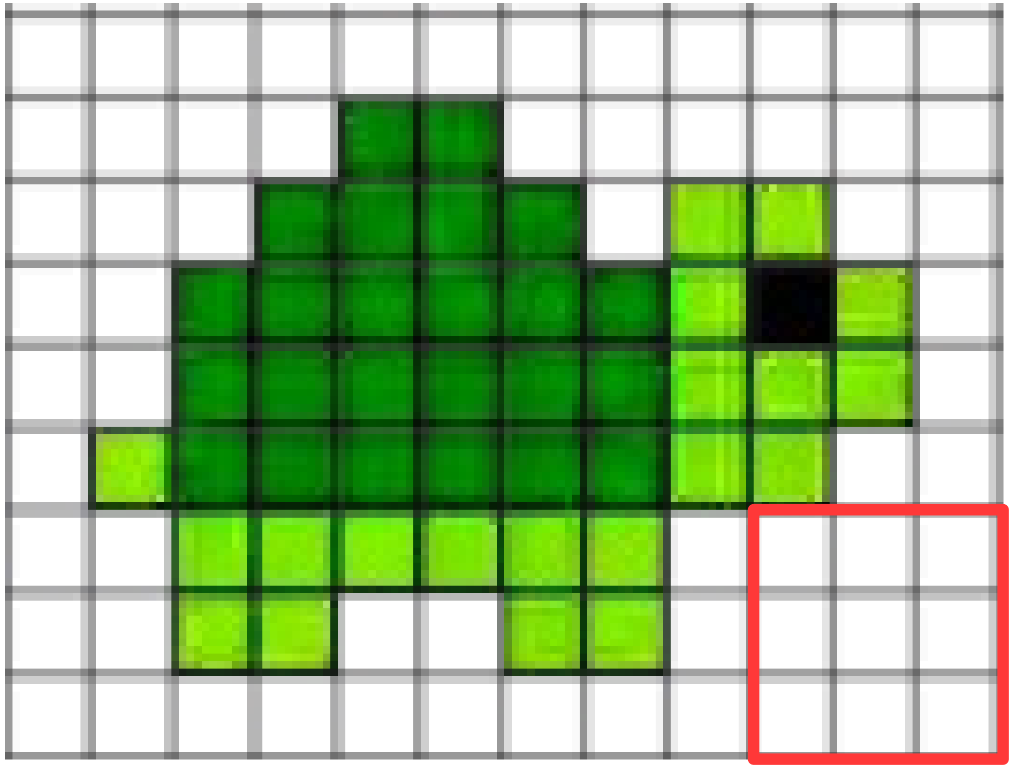


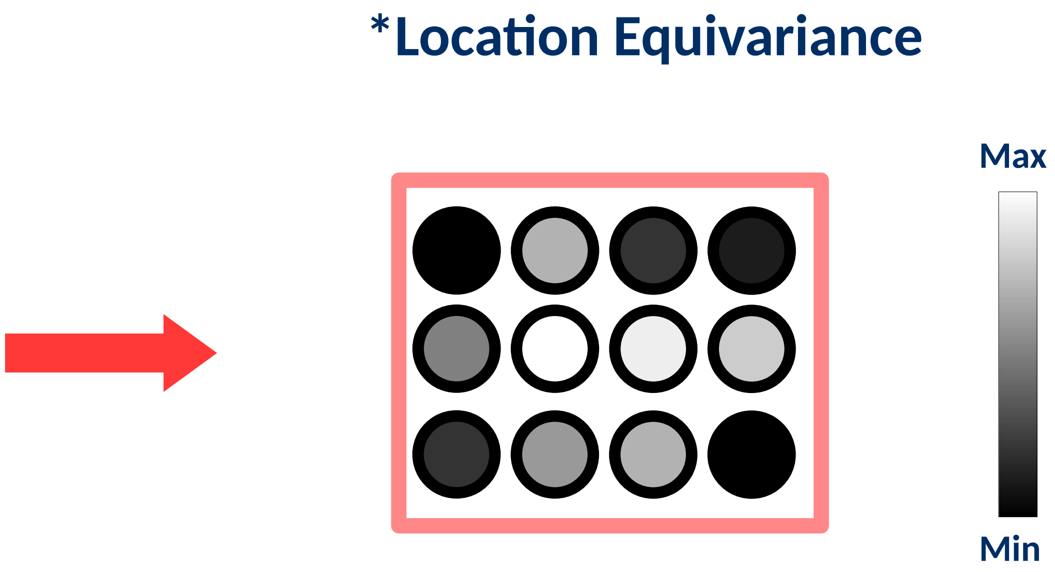


Response/feature map

- The kernel slides across the input
- Produces an output (or response) for every location where it is evaluated

In Practice: From Locally Connected to Convolutions

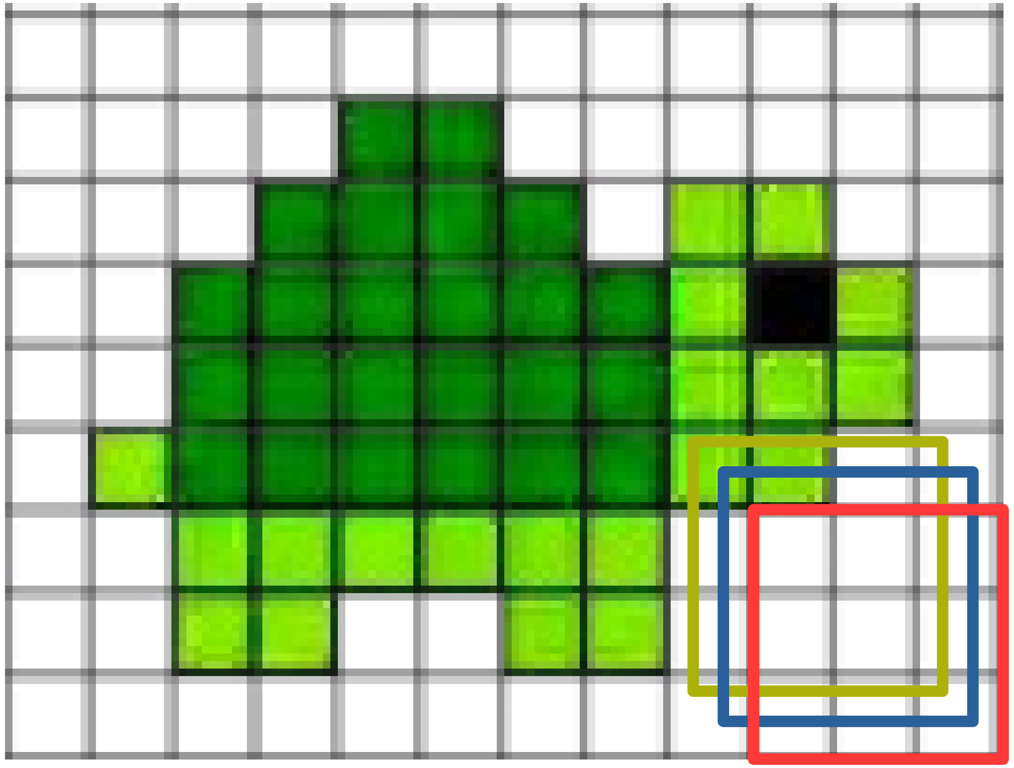


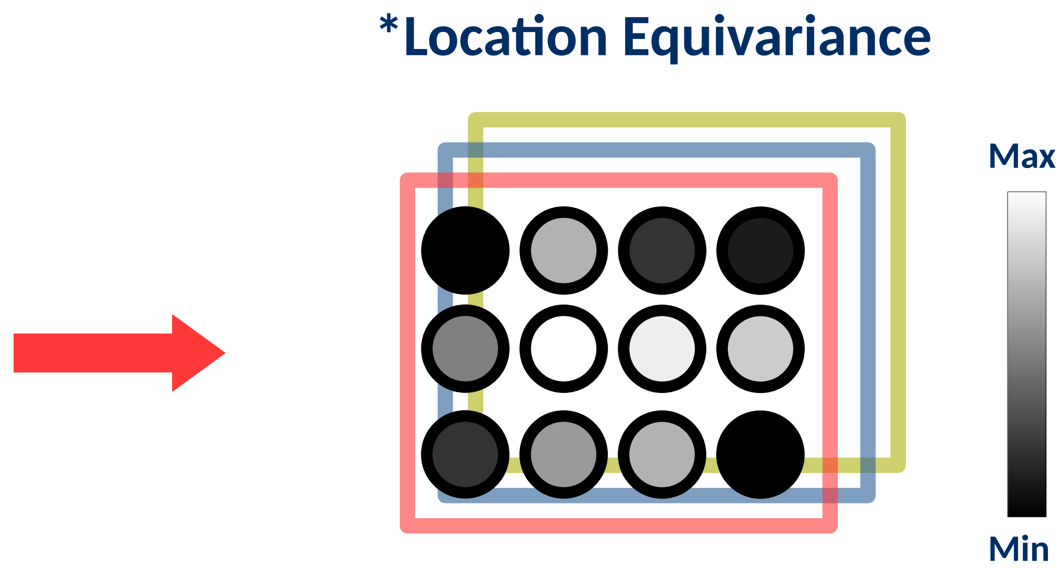


Response/feature map

- The kernel slides across the input
- Produces an output (or response) for every location where it is evaluated

In Practice: From Locally Connected to Convolutions

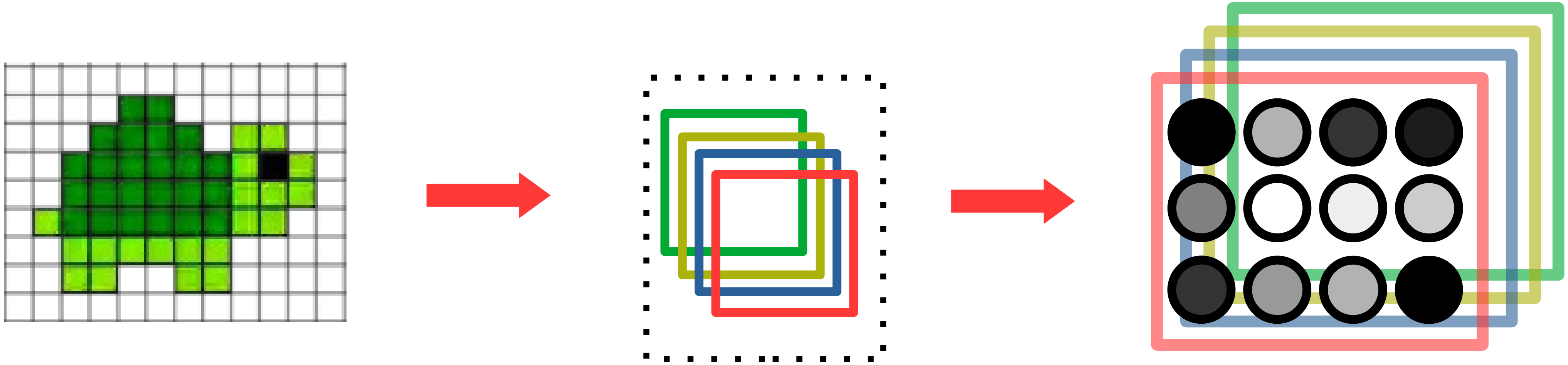




Response/feature map

- The kernel slides across the input
- Produces an output (or response) for every location where it is evaluated
- Repeating the process with *k* multiple kernels produces multiple features maps (channels)

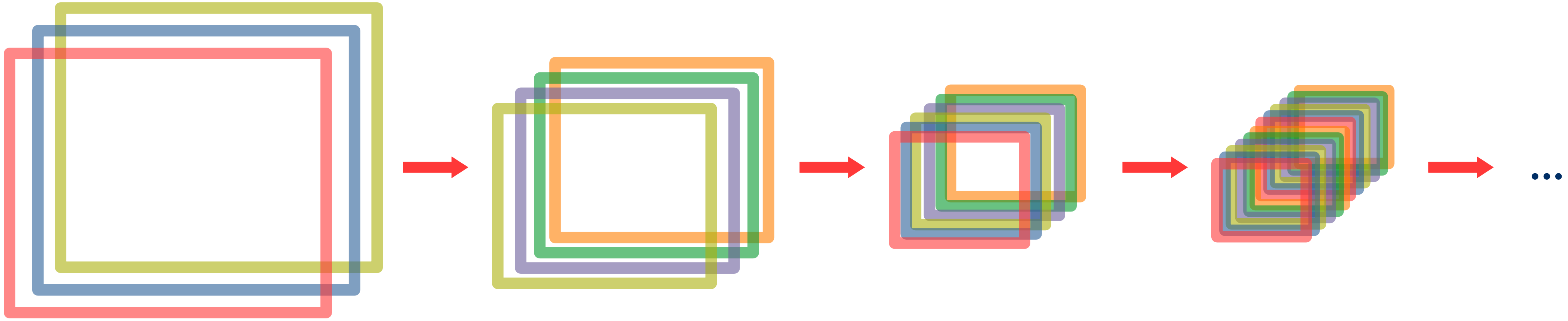
In Practice: From Locally Connected to Convolutions



Input [h x w x 3] **Convolutional Layer** **Output** [h’ x w’ x k]

- Inputs an ouputs are usually “data cubes” [**Tensors**]
- Filter reponses across inputs are aggregated

Putting everything together



Input [h x w x 3]

Convolutional Neural Network

***Promotes Compositionality**

Convolution Operations

Variants

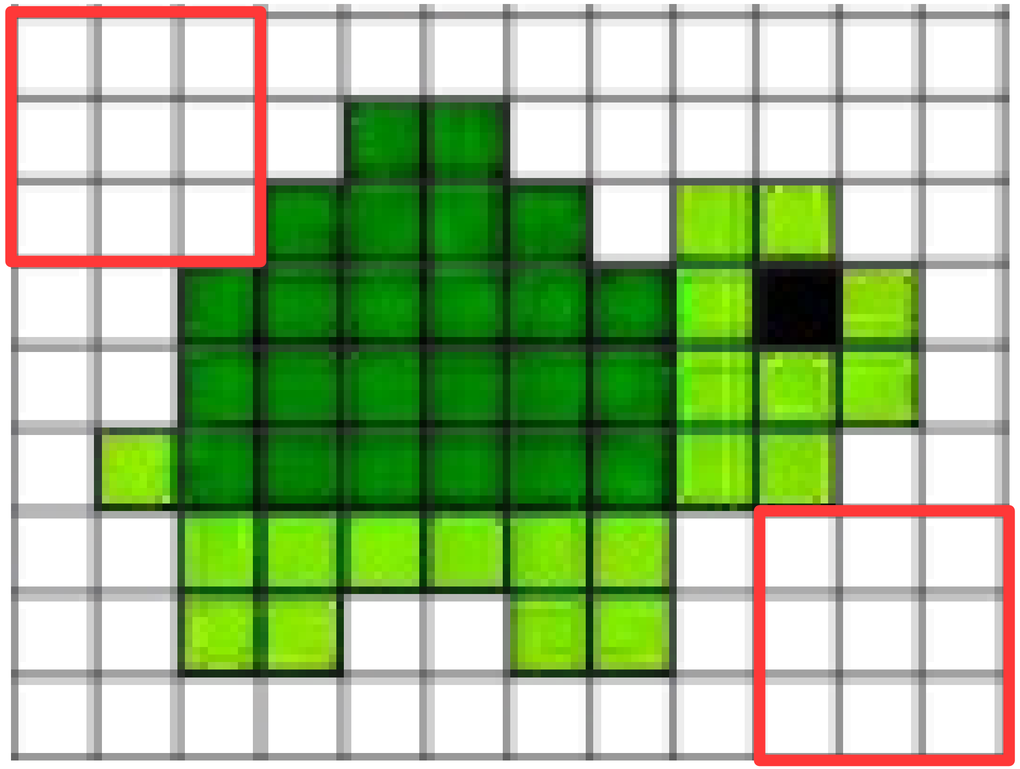
Break

See you in 15 mins.

Convolution Operations

Variants

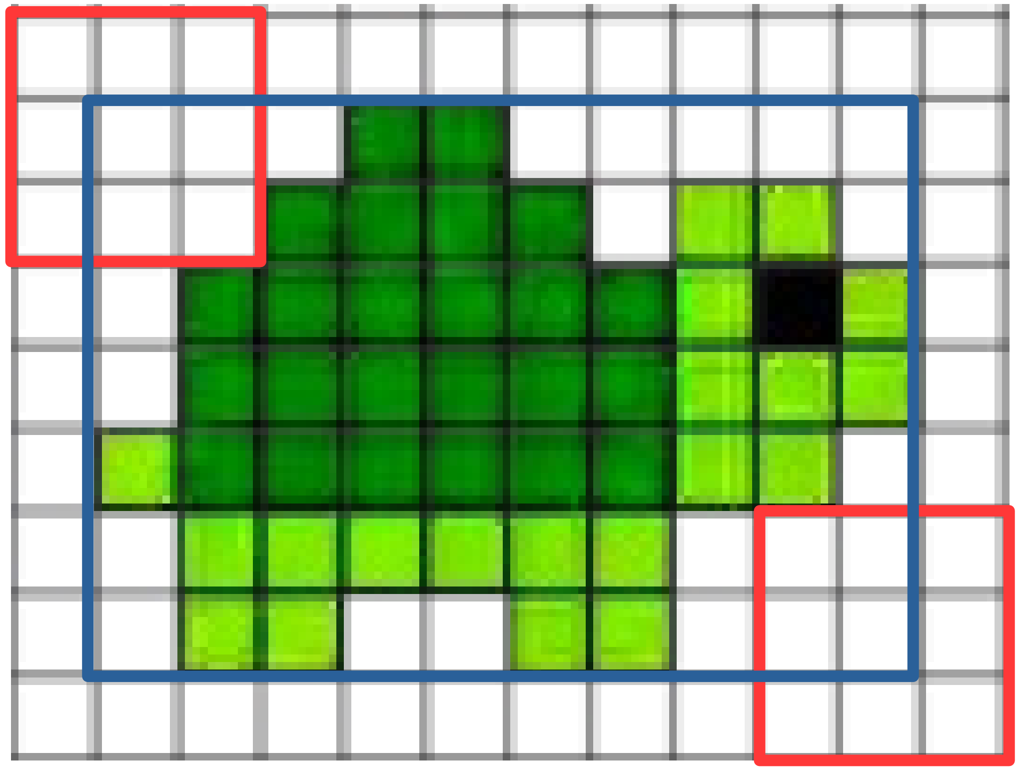
Variants: Valid Convolution



- **Valid Convolution**

  [Every considered point lies within the input]

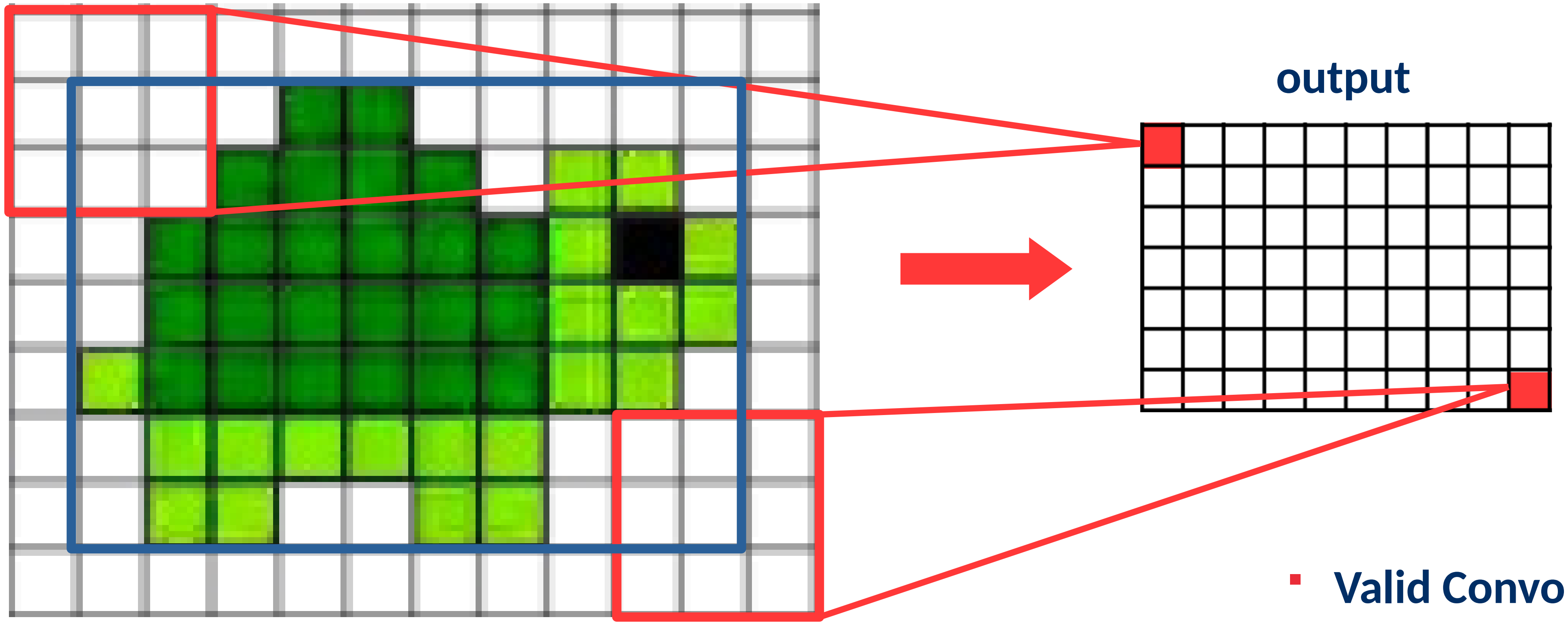
Variants: Valid Convolution



- **Valid Convolution**

  [Every considered point lies within the input]

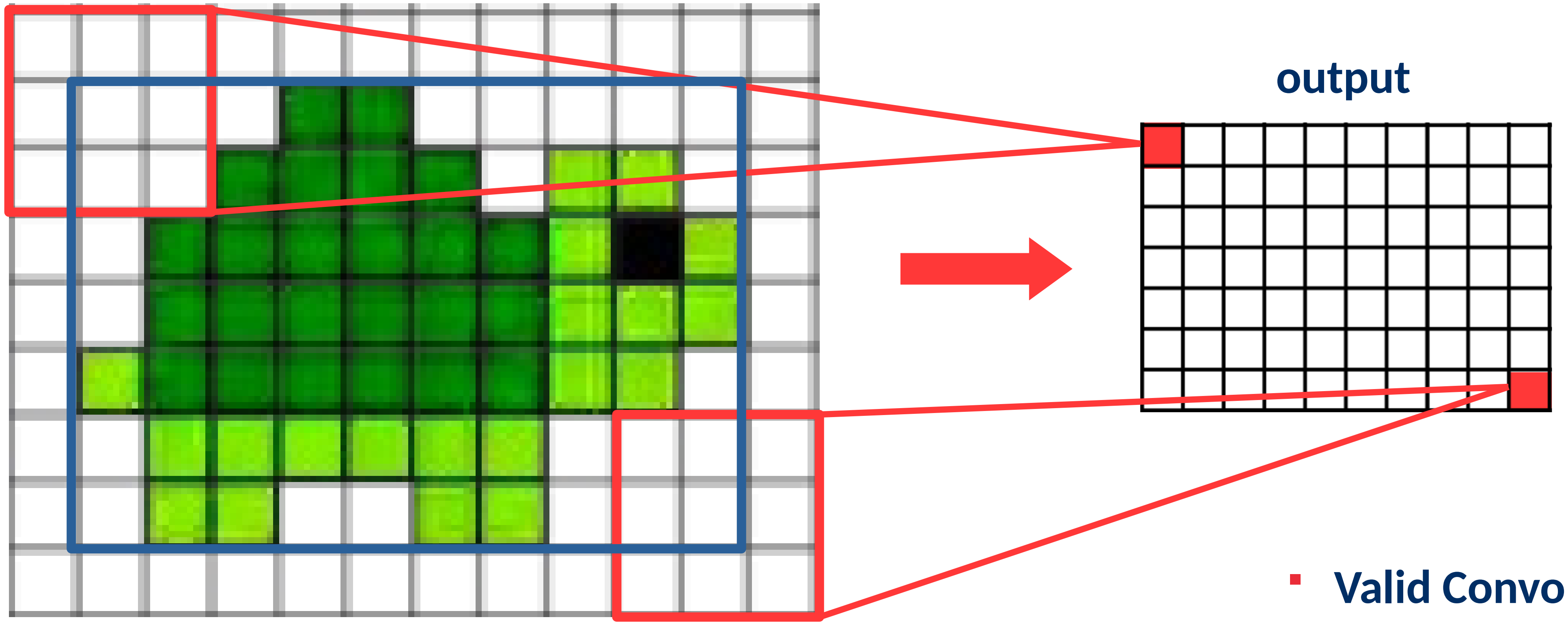
Variants: Valid Convolution



- **Valid Convolution**

  [Every considered point lies within the input]

Variants: Valid Convolution

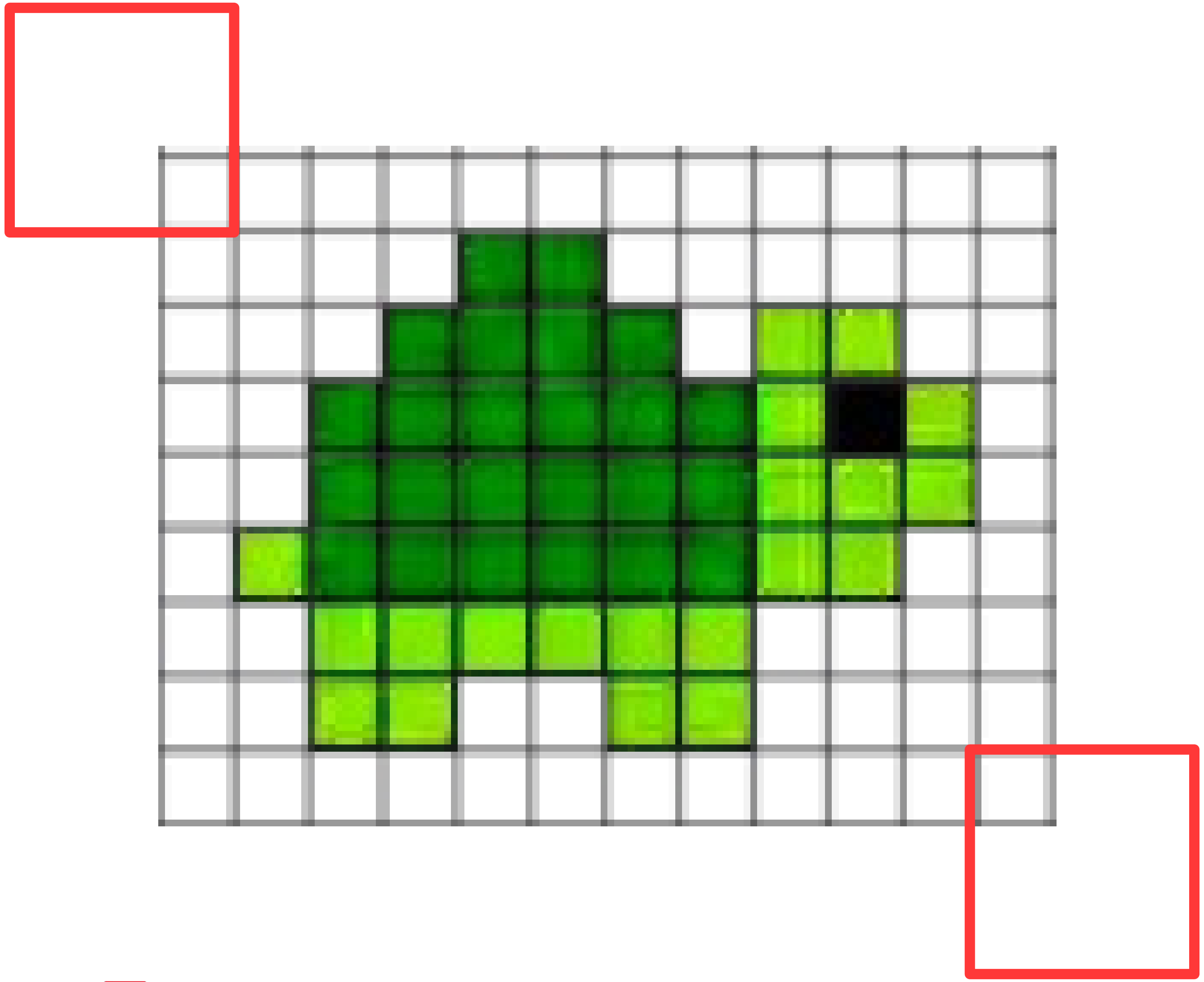


- **Valid Convolution**

  [Every considered point lies within the input]

$\text{Size}_{input} > \text{Size}_{output}$

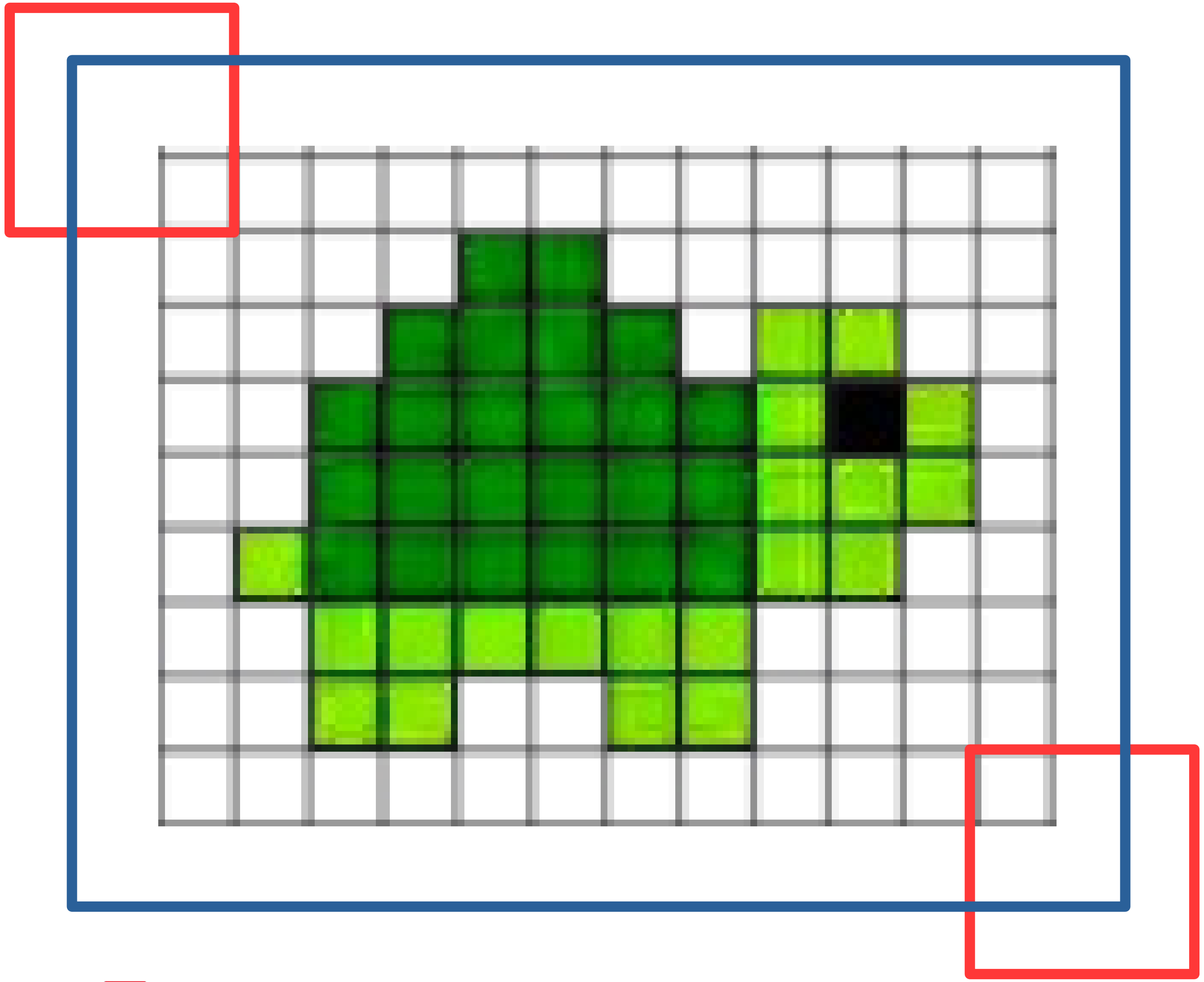
Variants: Full Convolution



- **Full Convolution**

  [At least one value of the kernel covers the input]

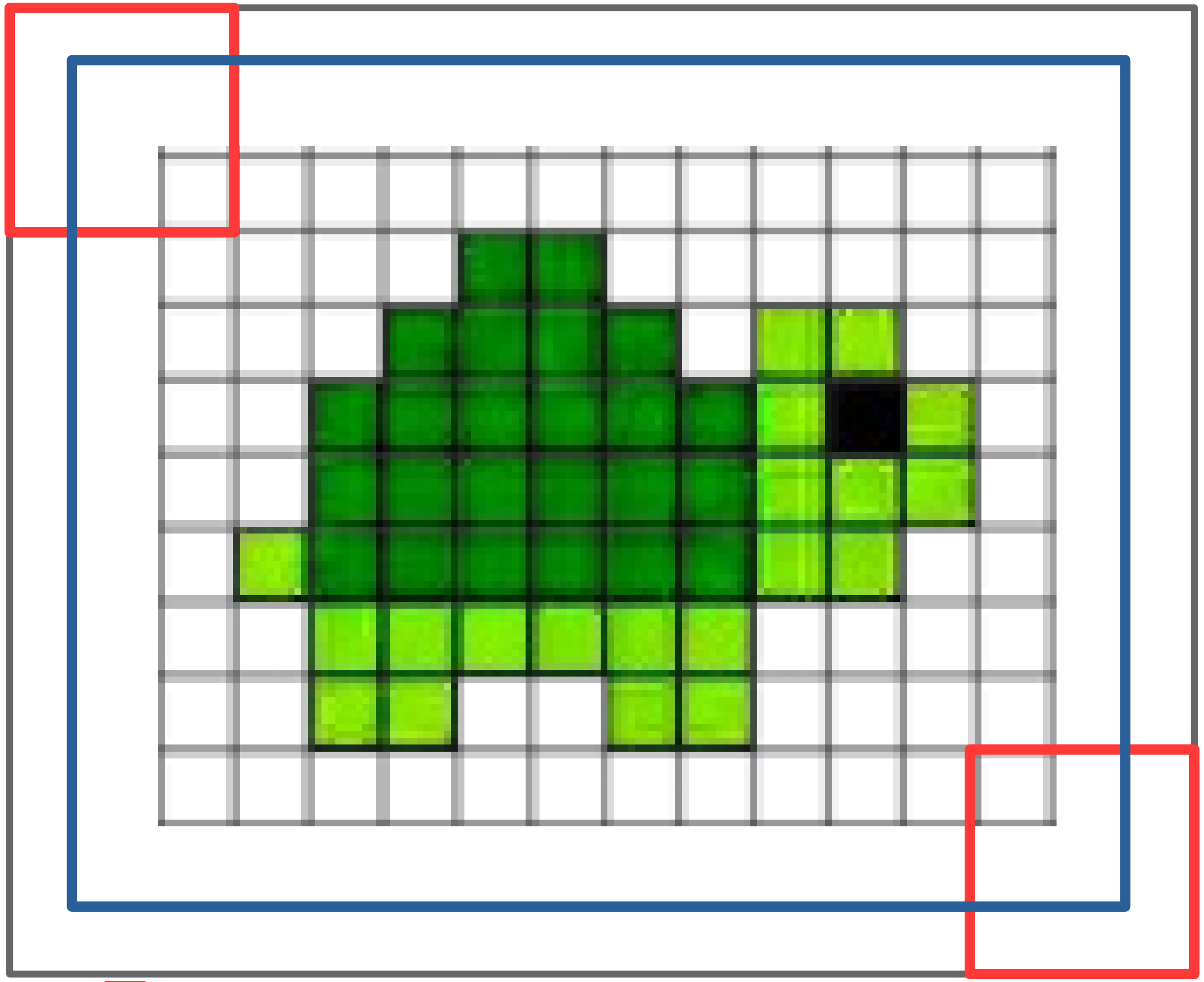
Variants: Full Convolution



- **Full Convolution**

  [At least one value of the kernel covers the input]

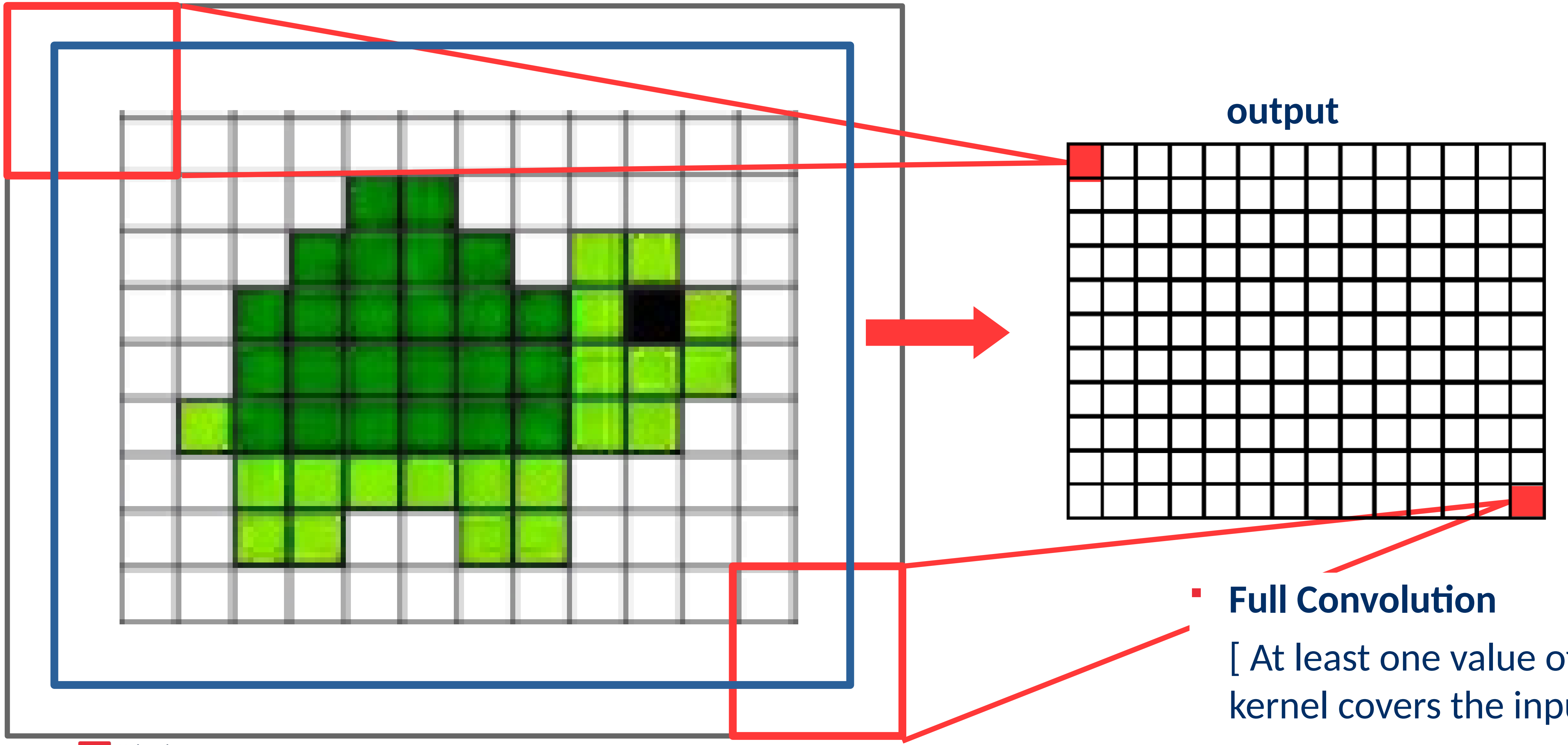
Variants: Full Convolution



- **Full Convolution**

  [At least one value of the kernel covers the input]

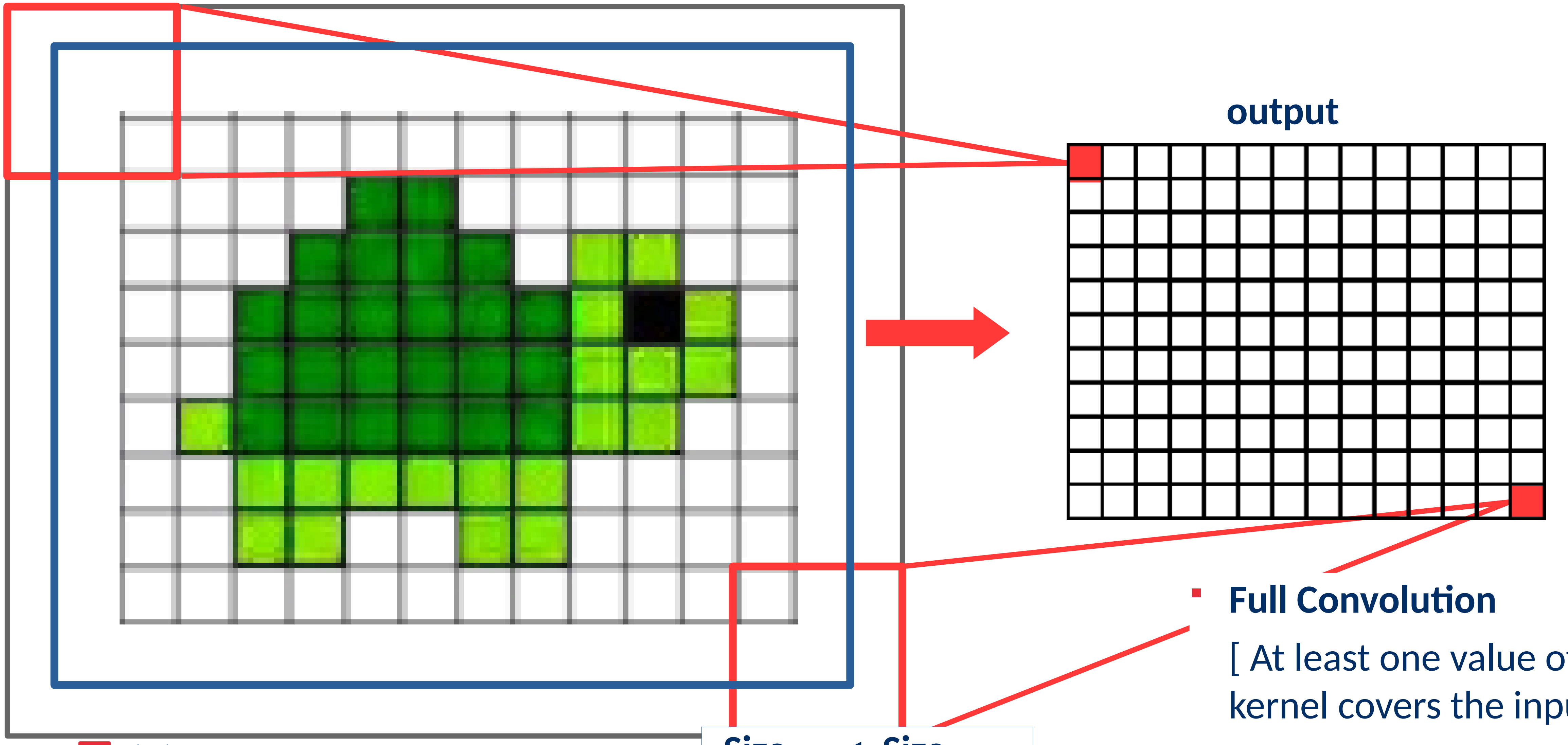
Variants: Full Convolution



- **Full Convolution**

  [At least one value of the kernel covers the input]

Variants: Full Convolution

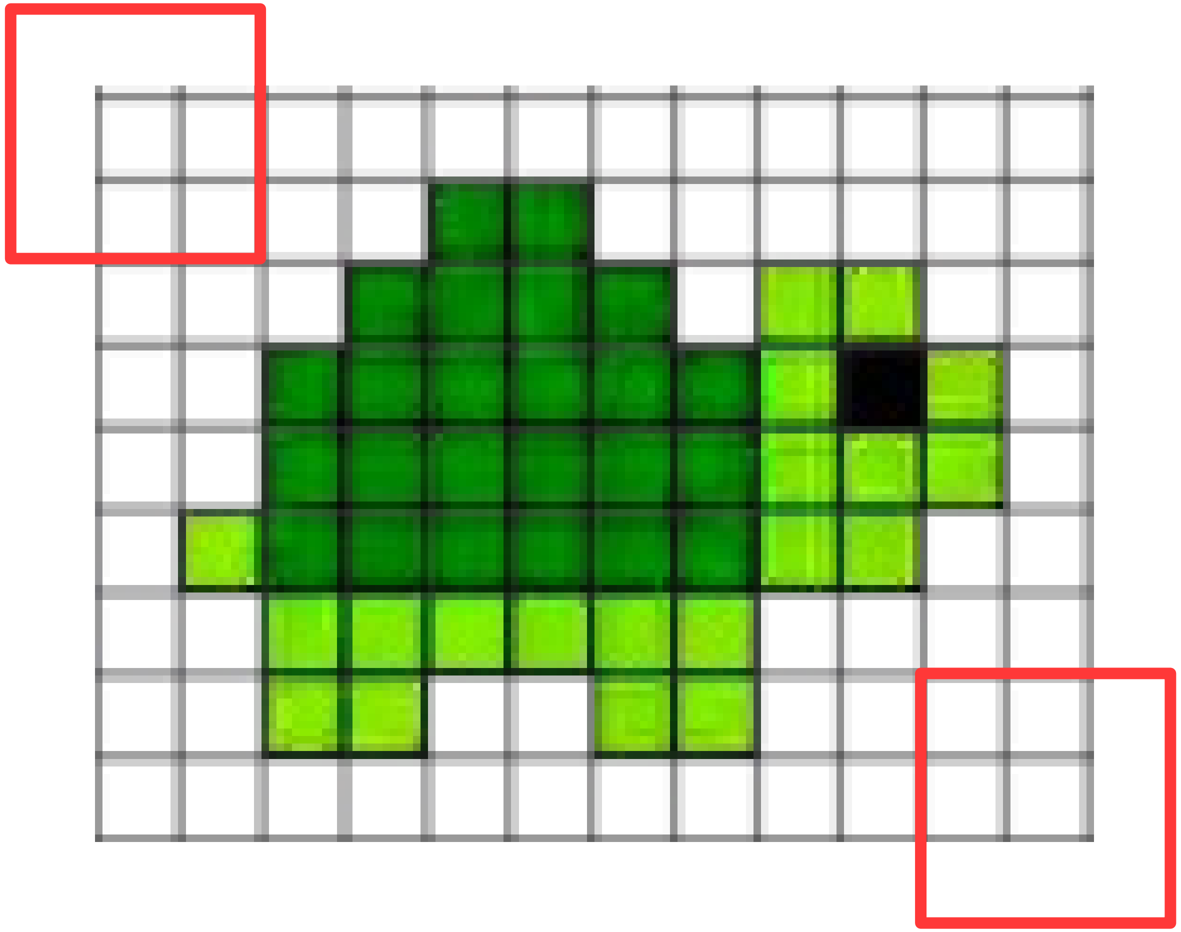


- **Full Convolution**

  [At least one value of the kernel covers the input]

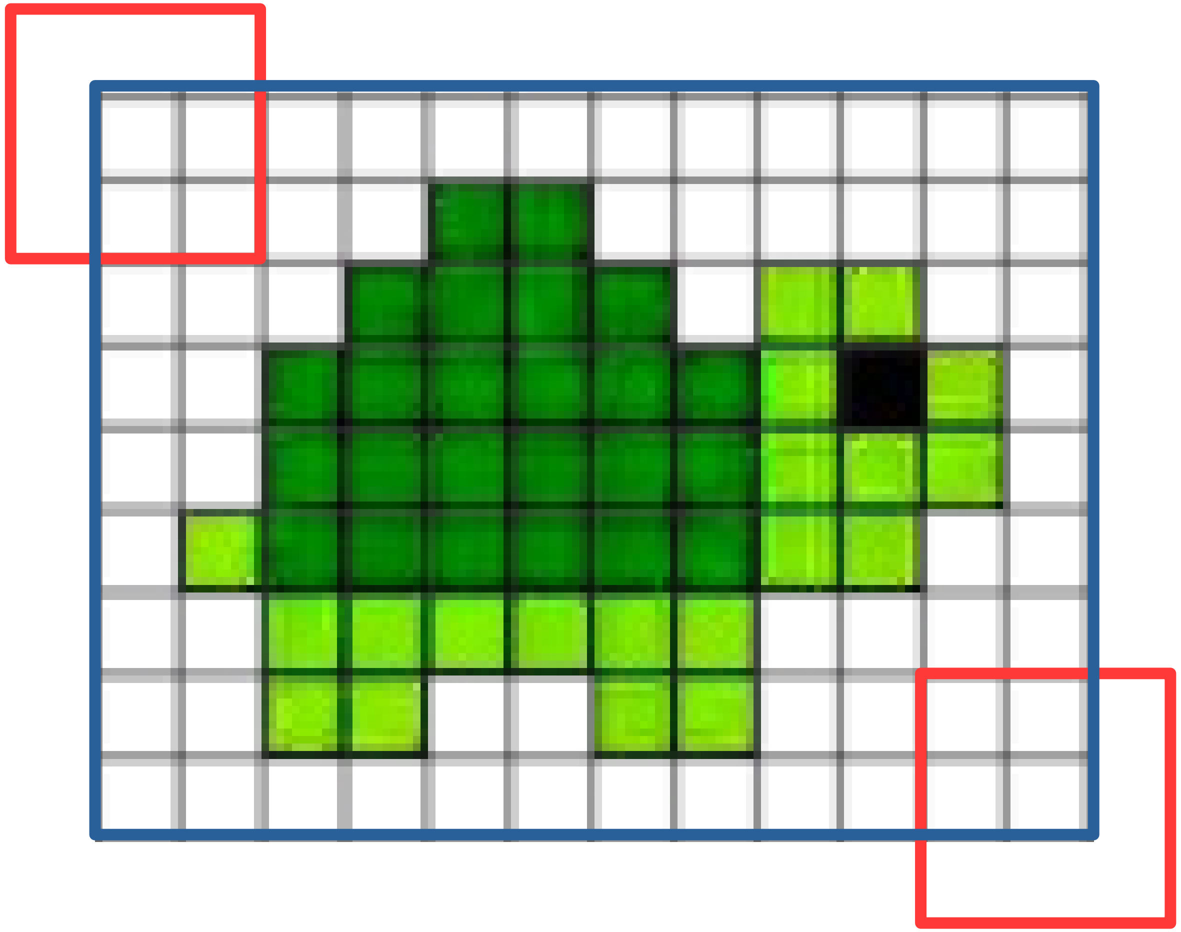
$\text{Size}_{input} < \text{Size}_{output}$

Variants: Same Convolution



- **Same Convolution**
  [Kernel evaluated (centered) at every location of the input]

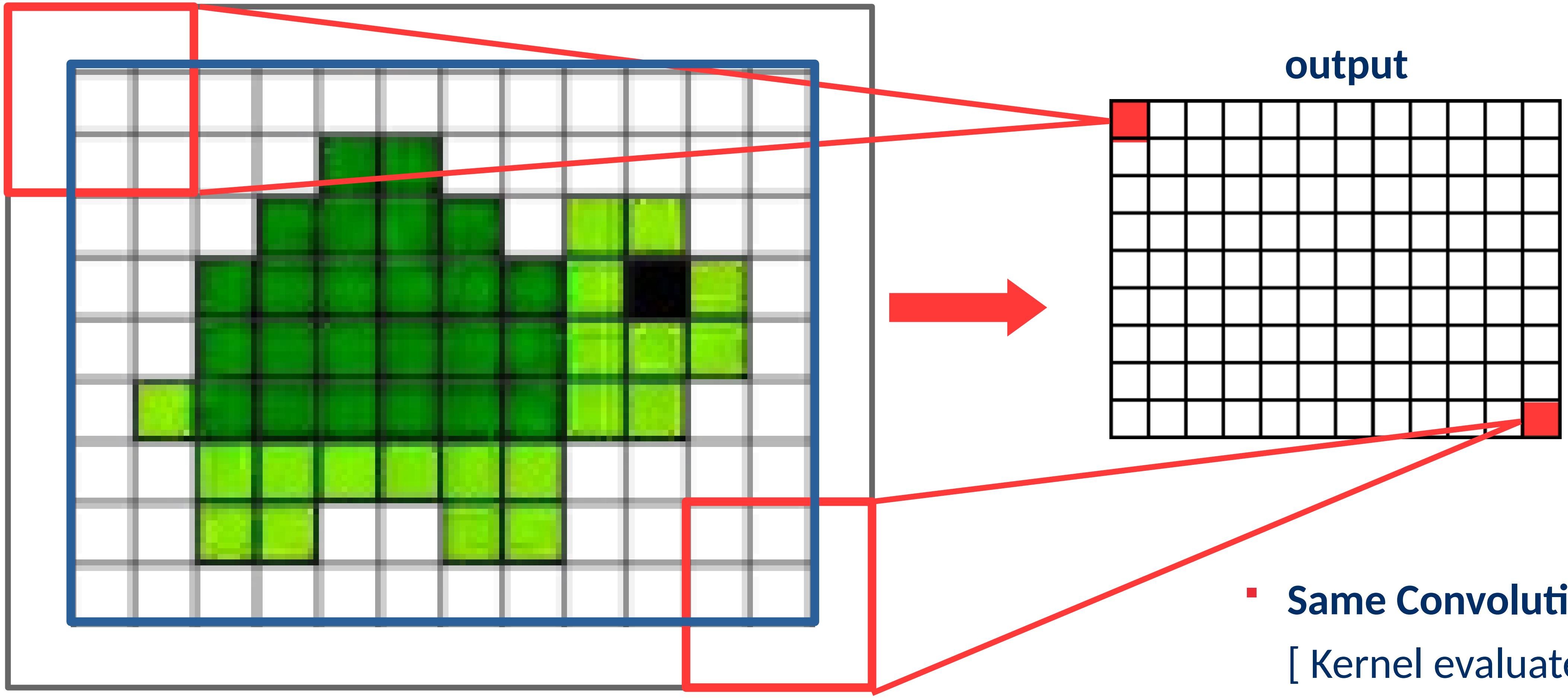
Variants: Same Convolution



- **Same Convolution**

  [Kernel evaluated (centered) at every location of the input]

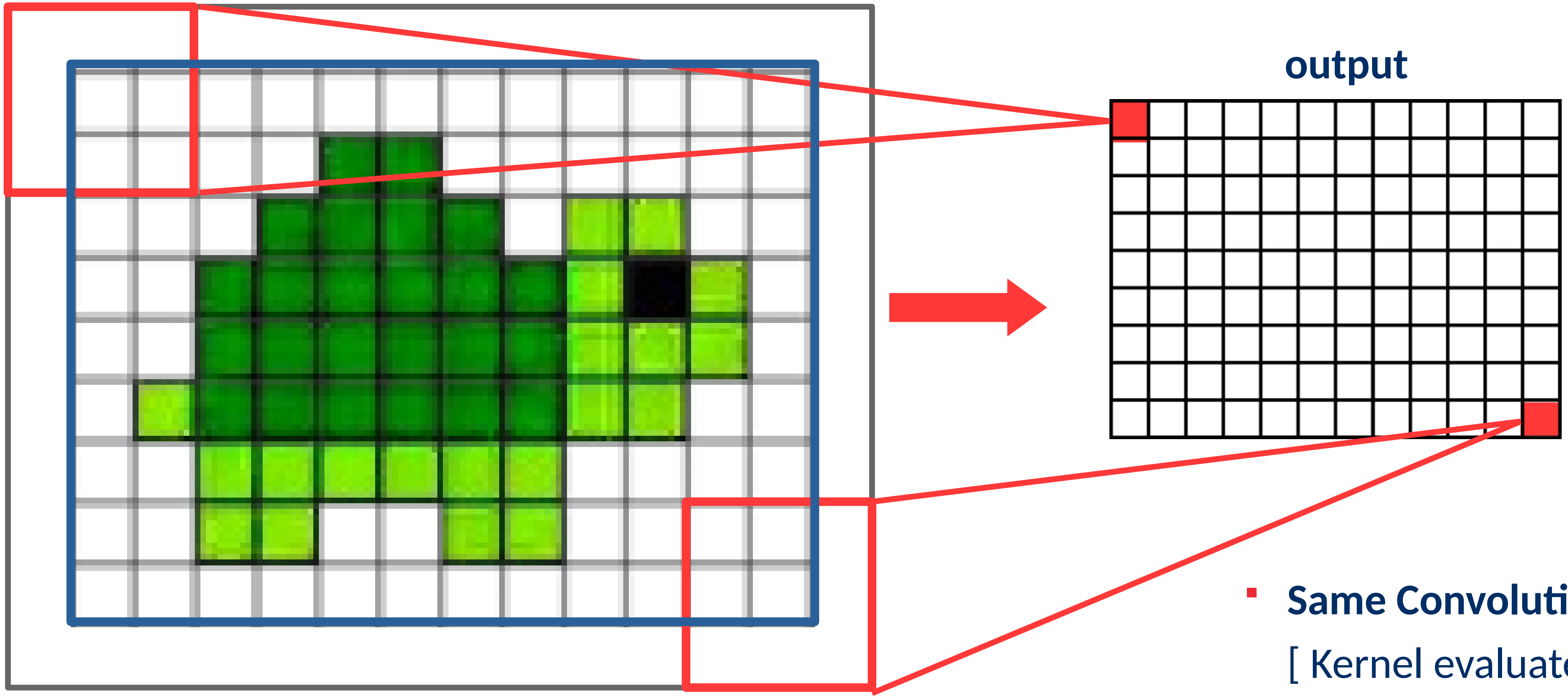
Variants: Same Convolution



- **Same Convolution**

  [Kernel evaluated (centered) at every location of the input]

Variants: Same Convolution

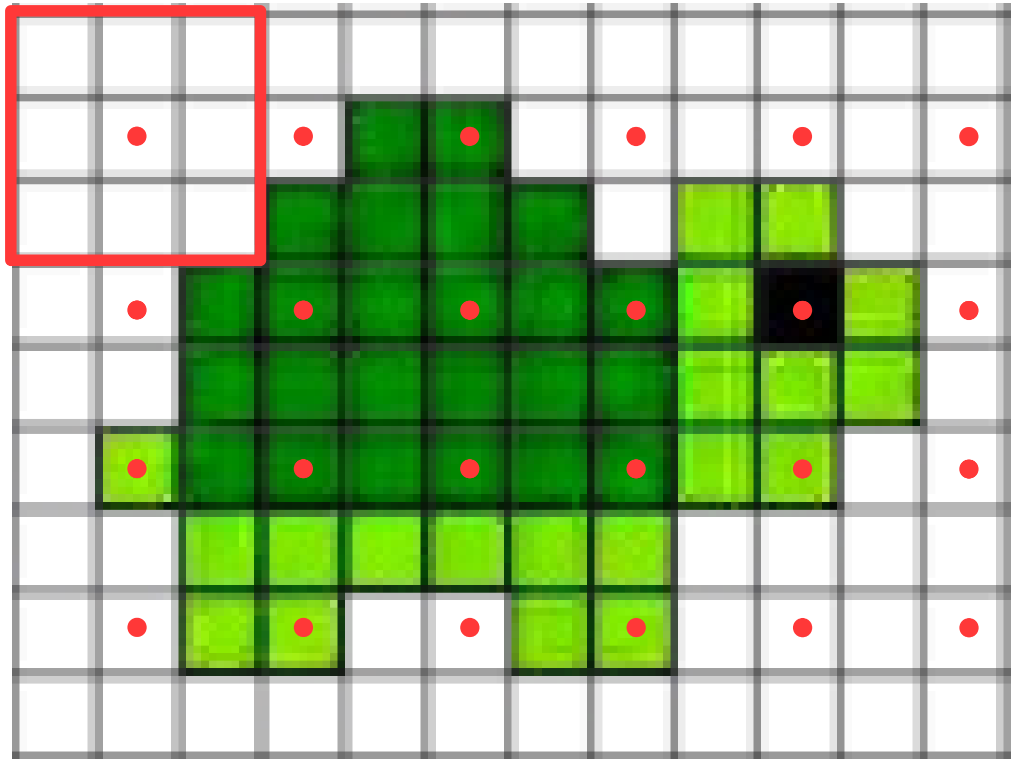


- **Same Convolution**

  [Kernel evaluated (centered) at every location of the input]

$\mathbf{Size}_{input}$ == $\mathbf{Size}_{output}$

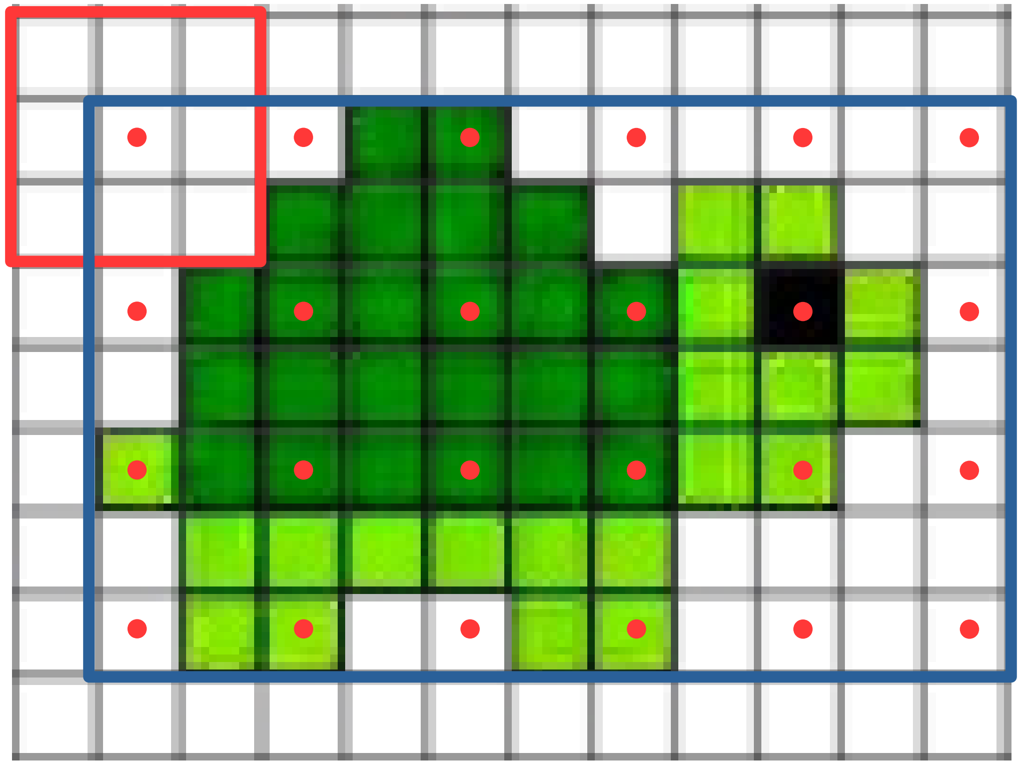
Variants: Strided Convolution



- **Strided Convolution**

  [Sparser kernel evaluations]

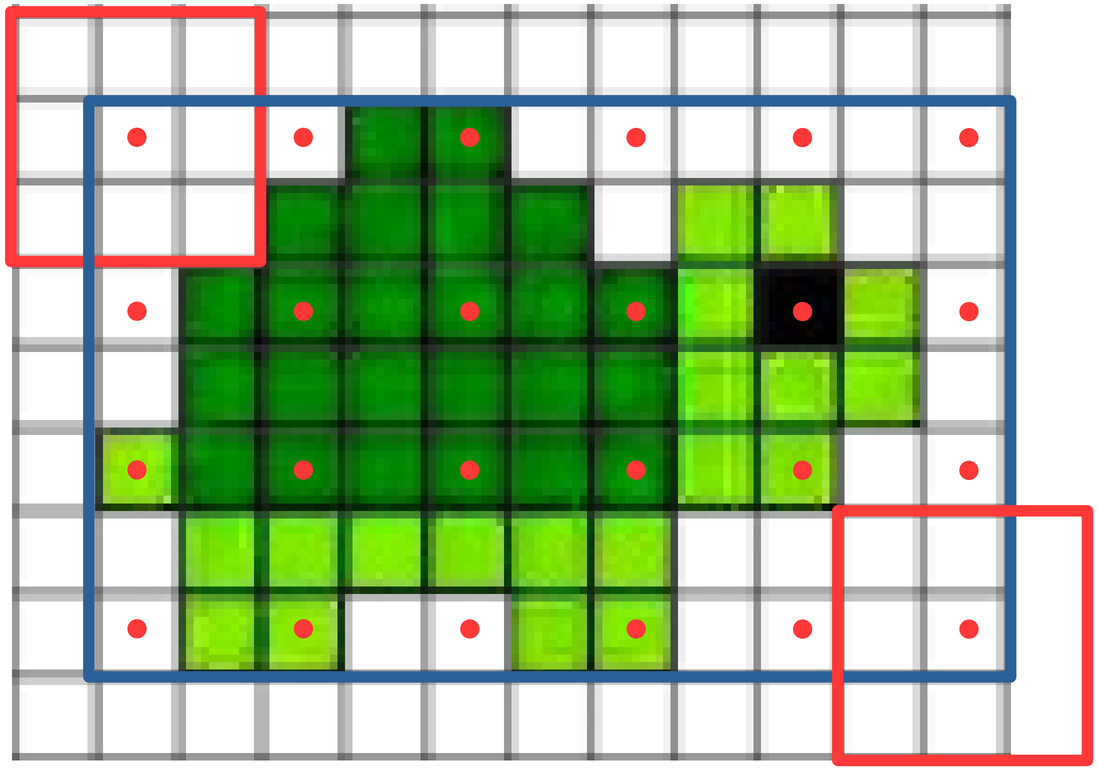
Variants: Strided Convolution



- **Strided Convolution**

  [Sparser kernel evaluations]

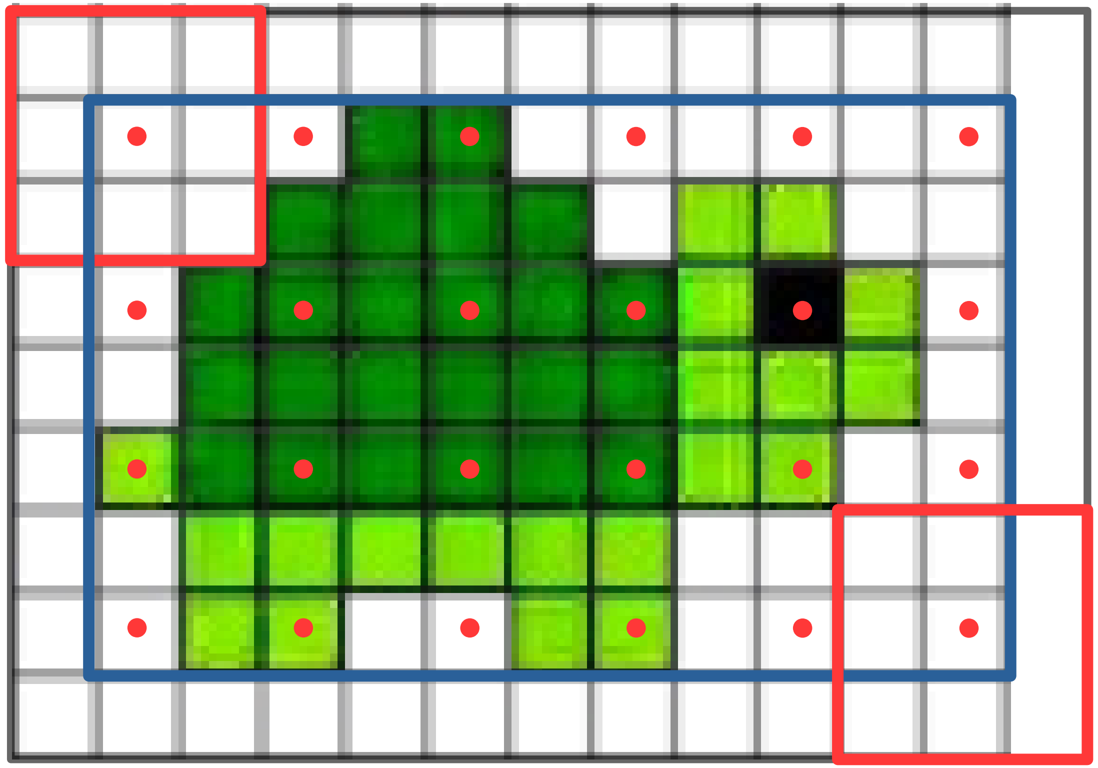
Variants: Strided Convolution



- **Strided Convolution**

  [Sparser kernel evaluations]

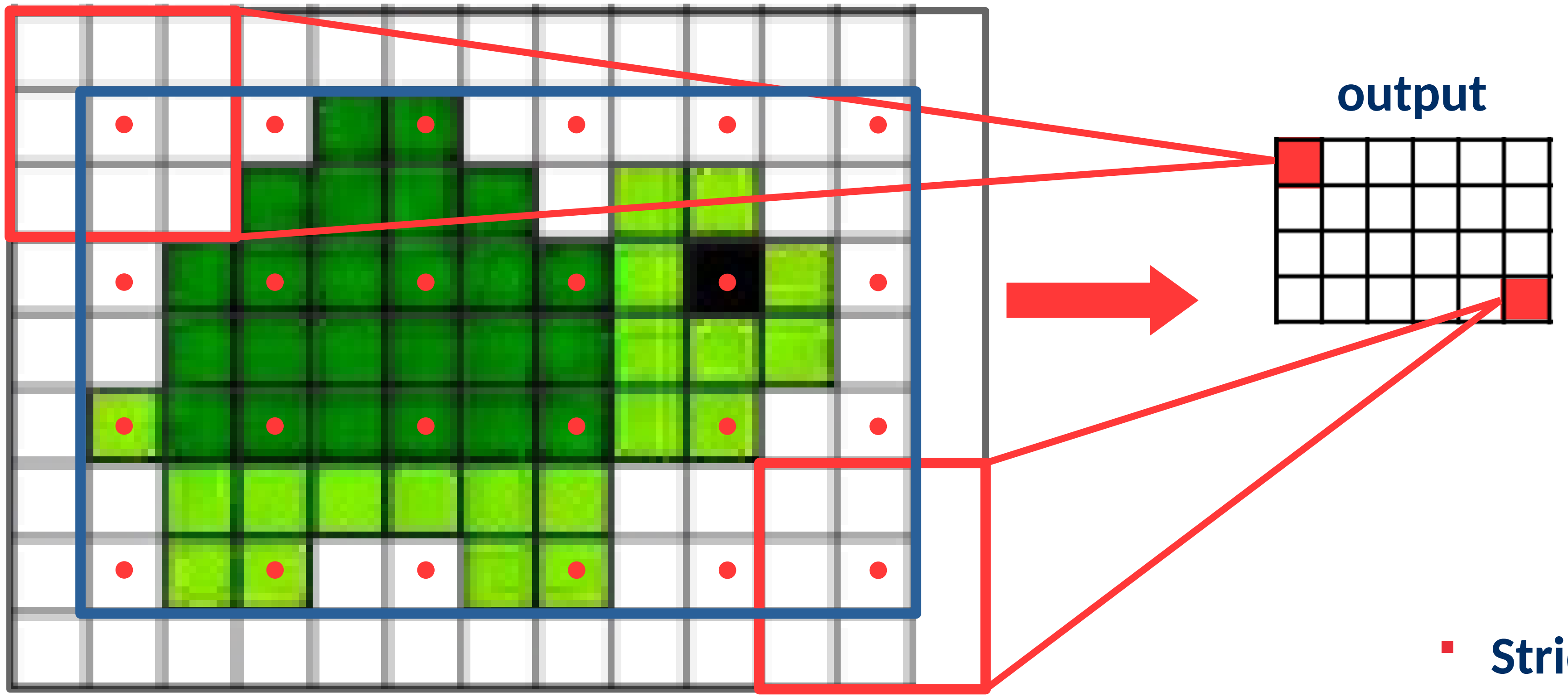
Variants: Strided Convolution



- **Strided Convolution**

  [Sparser kernel evaluations]

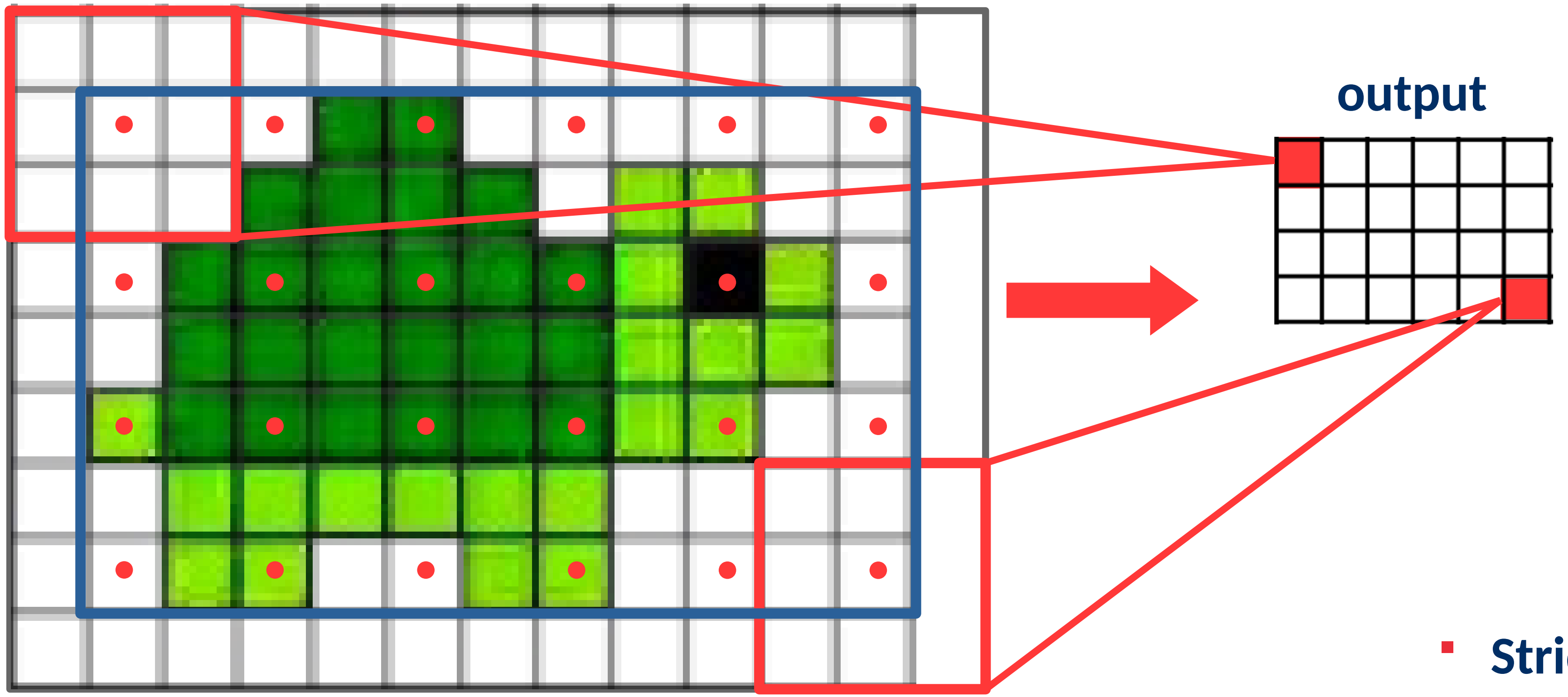
Variants: Strided Convolution



- **Strided Convolution**

  [Sparser kernel evaluations]

Variants: Strided Convolution

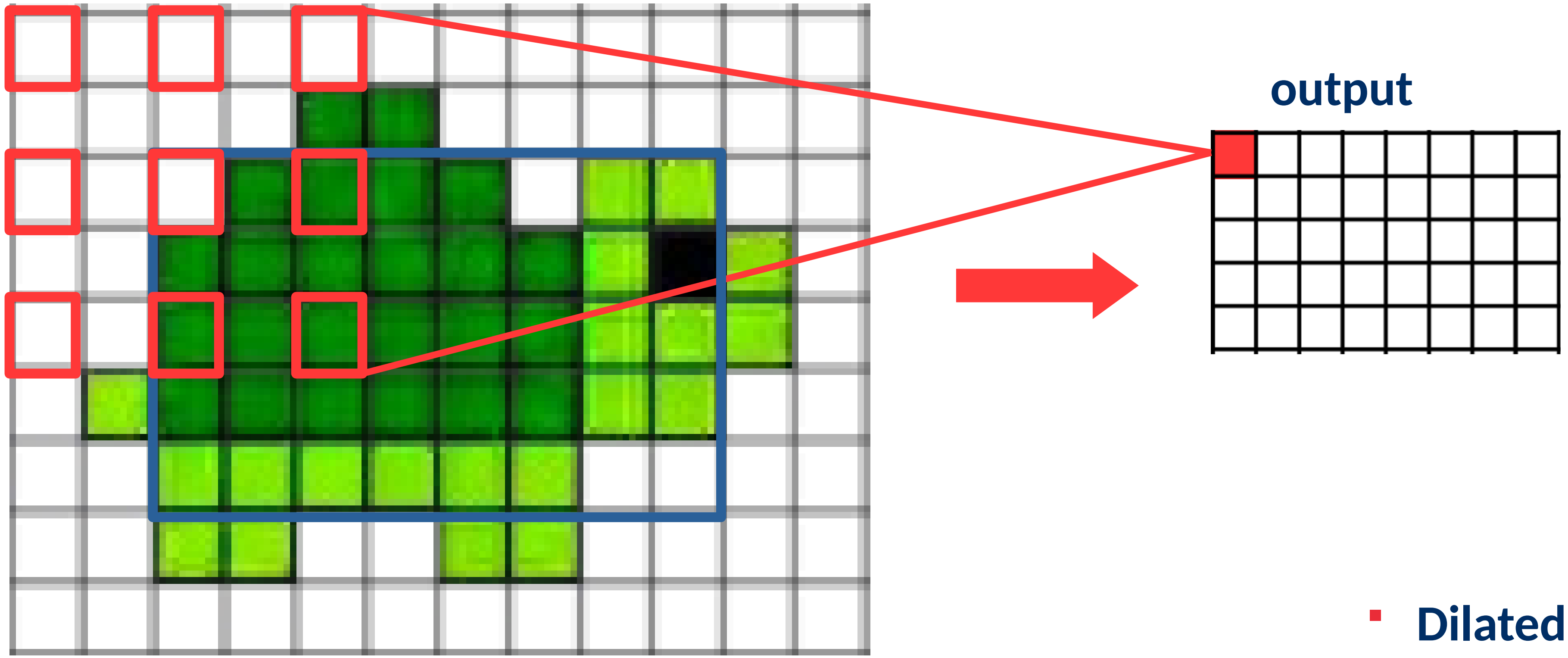


- **Strided Convolution**

  [Sparser kernel evaluations]

***Effective for decreasing the scale**

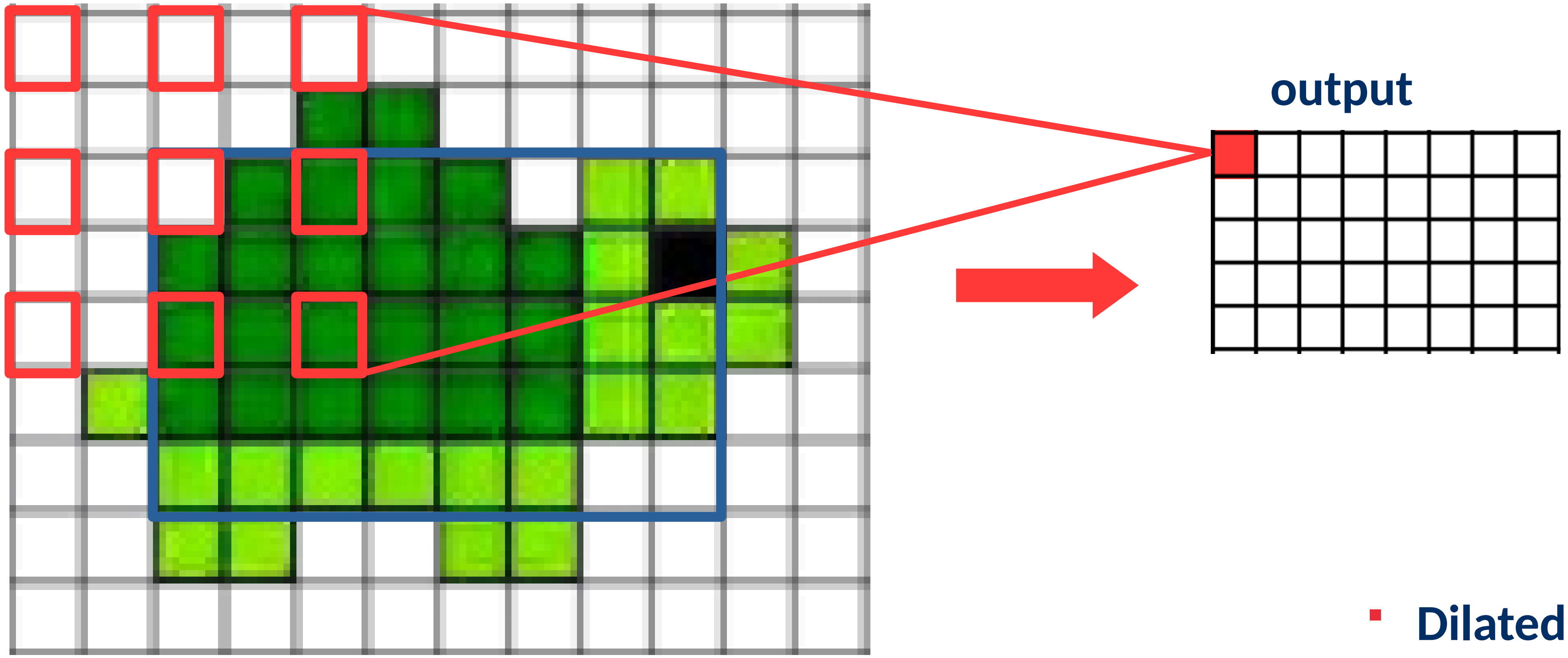
Variants: Dilated Convolution (aka. Atrous Convolution)



- **Dilated Convolution**

  [Points considered in the kernel are spread]

Variants: Dilated Convolution (aka. Atrous Convolution)



- **Dilated Convolution**

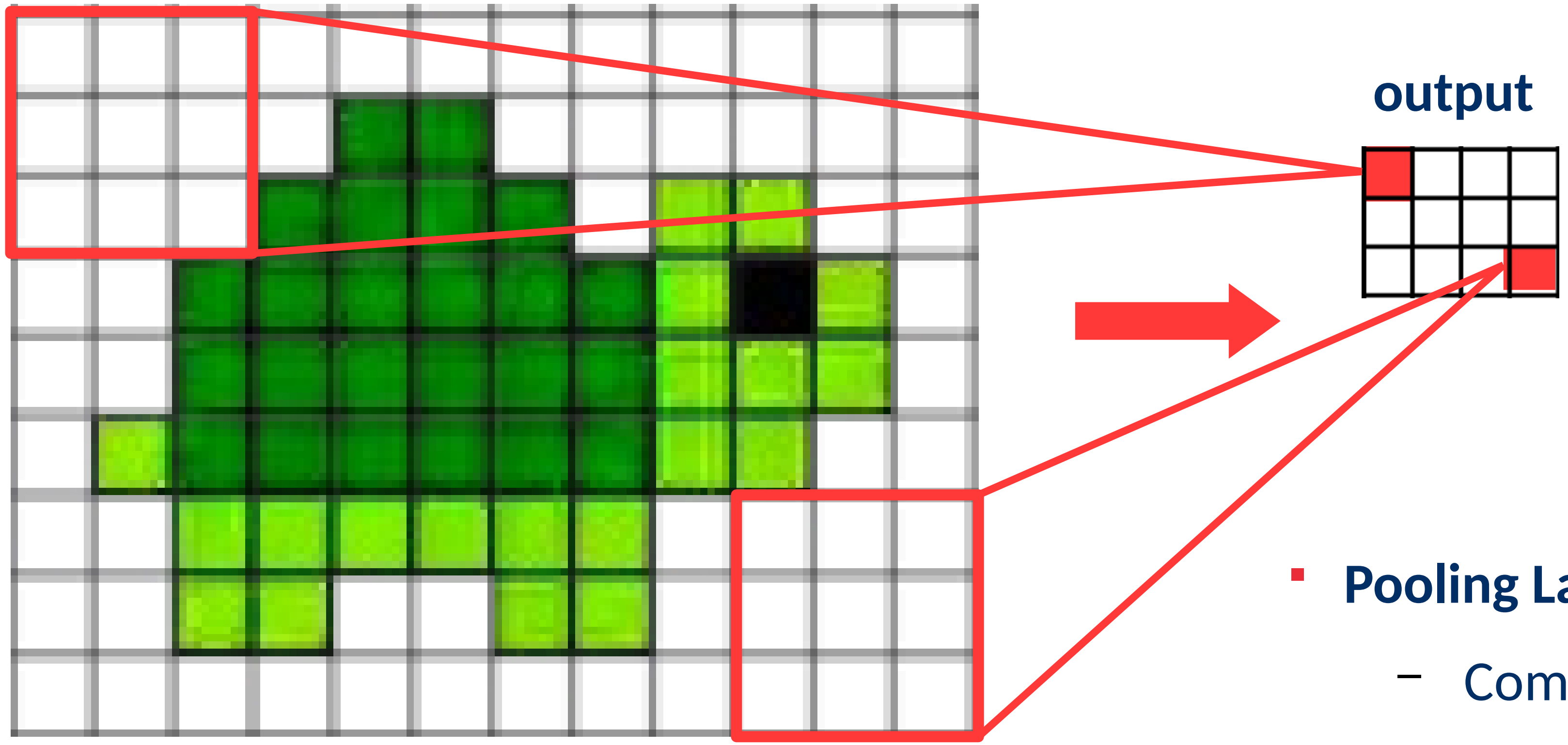
  [Points considered in the kernel are spread]

***Effective for increasing the receptive field**

University of Antwerp

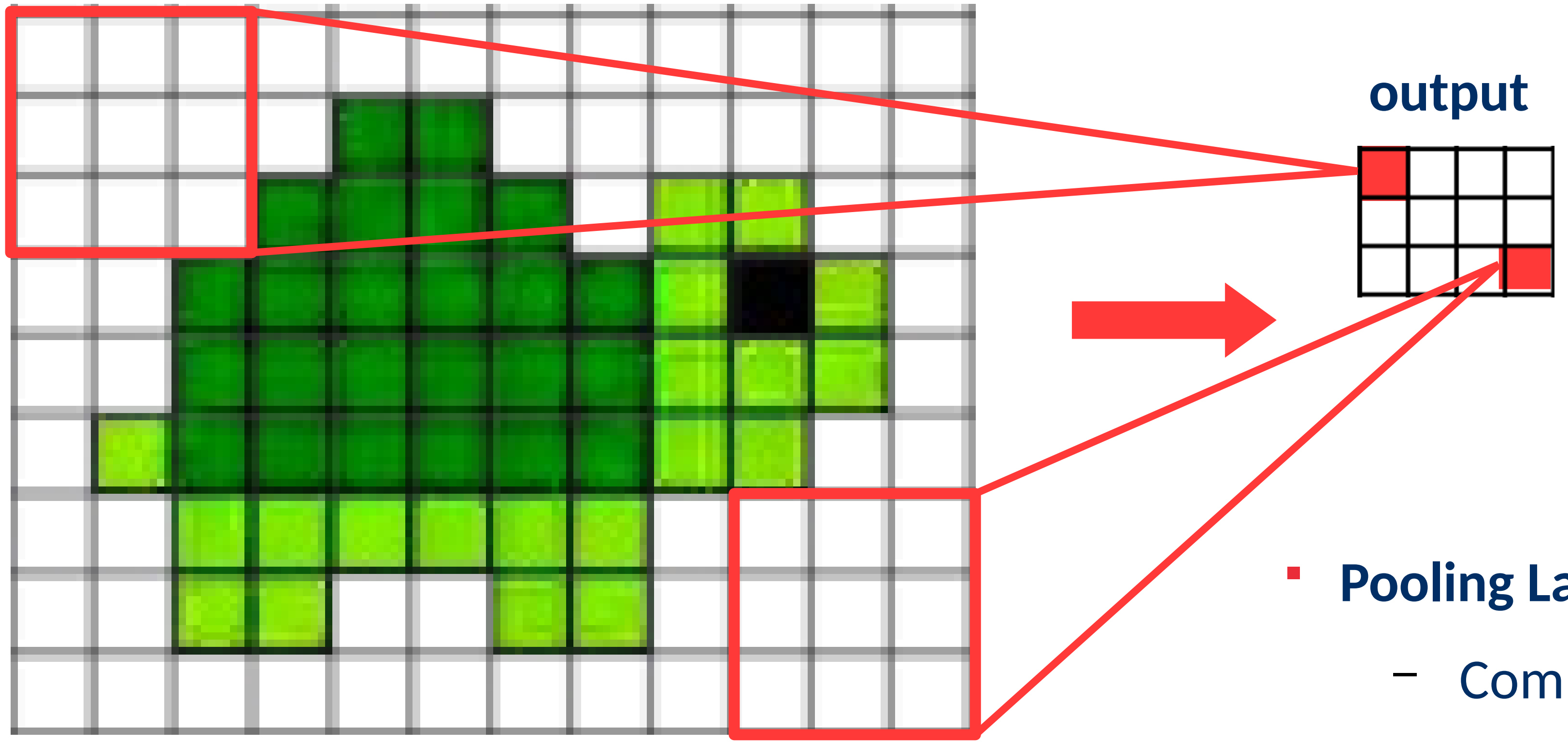
Other

Other Variants: Pooling



- **Pooling Layer**
  - Compute a single value per region
  - Region size is variable

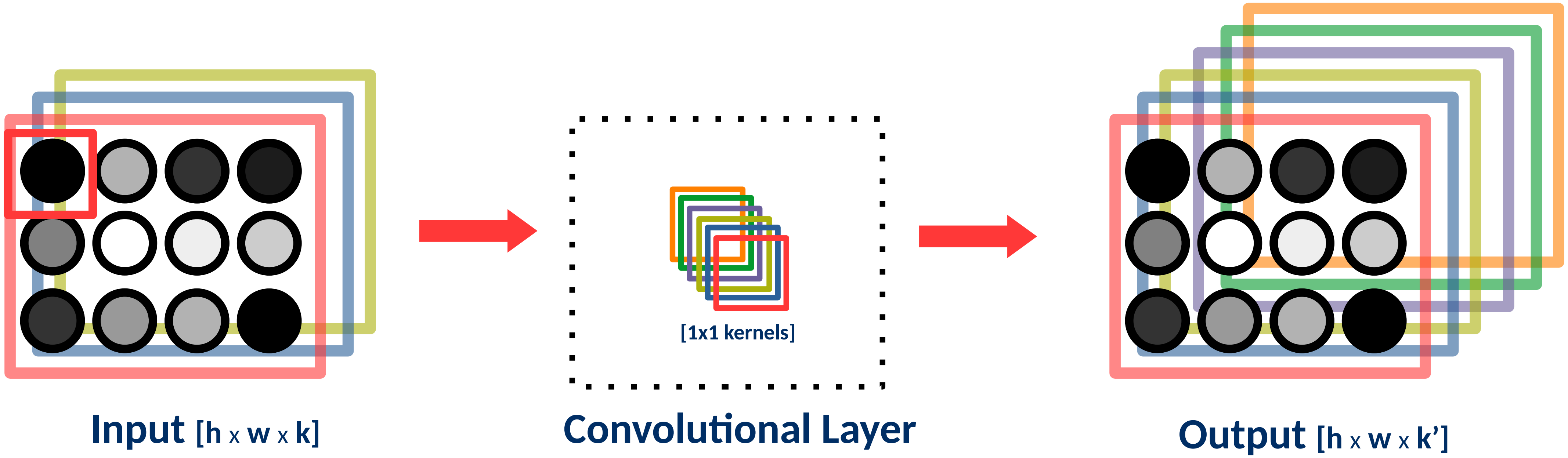
Other Variants: Pooling



- **Pooling Layer**
  - Compute a single value per region
  - Region size is variable

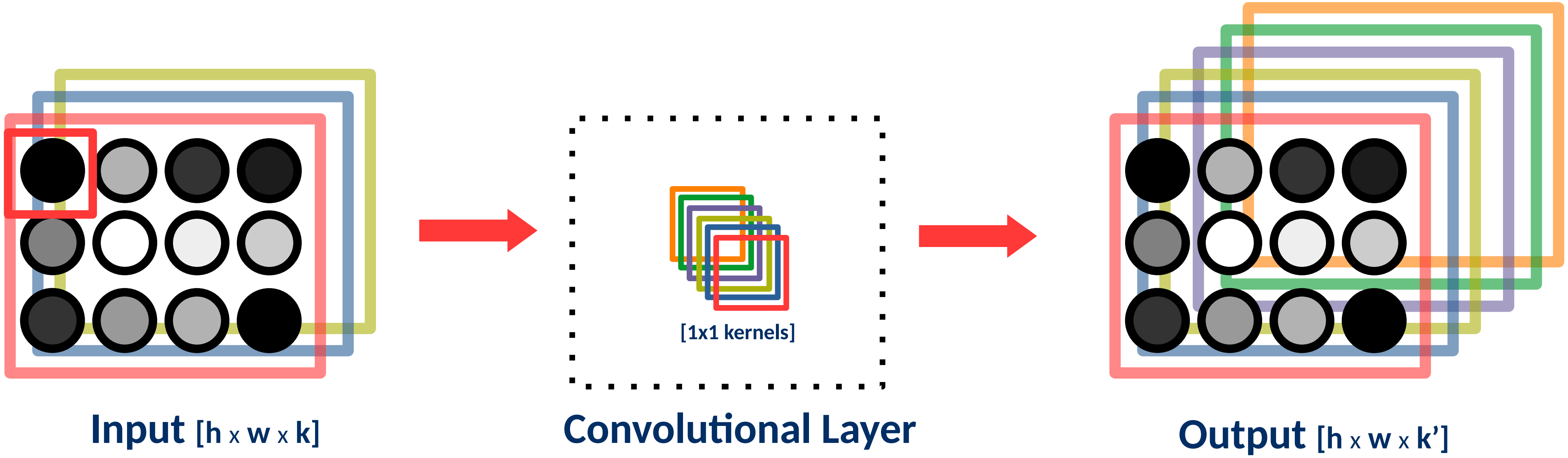
***Effective for decreasing the scale**

Other Variants: 1x1 Convolutions



- Perform a neuron-level operation
- Integration over the channels

Other Variants: 1x1 Convolutions



- Perform a neuron-level operation
- Integration over the channels

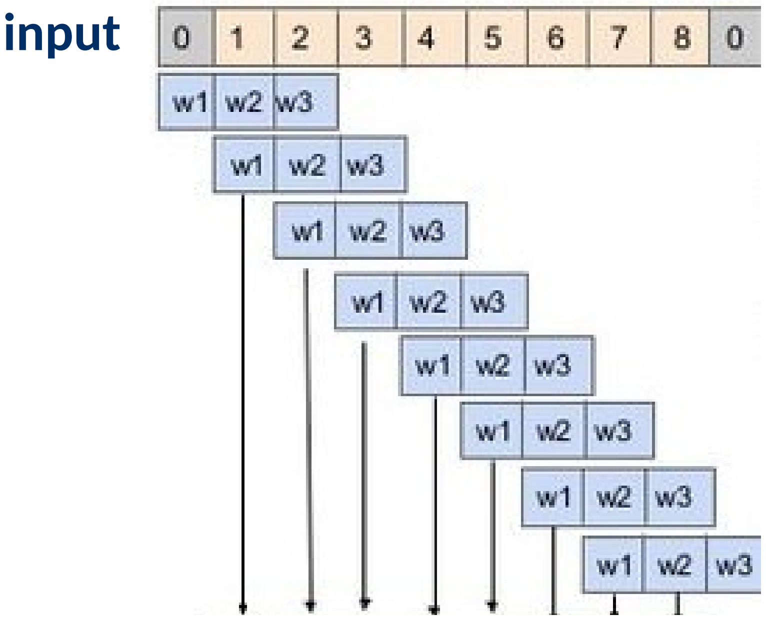
***Effective for modifying number of channels**

Other Variants: 1x1 Convolutions

input

1D Convolutions

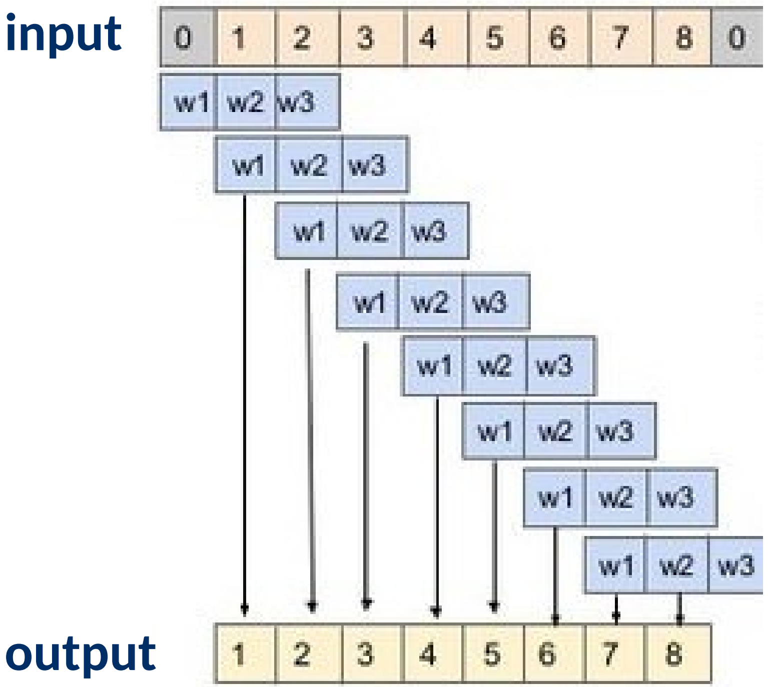
Other Variants: 1x1 Convolutions



1D Convolutions

S. Bansal, 3D Convolutions: Understanding + Use Case, Kaggle, 2019

Other Variants: 1x1 Convolutions

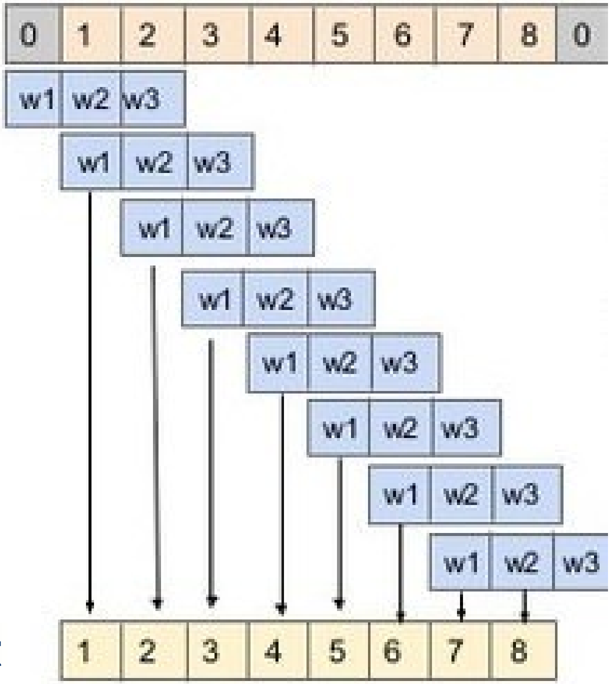


1D Convolutions

S. Bansal, 3D Convolutions: Understanding + Use Case, Kaggle, 2019

Other Variants: 1x1 Convolutions

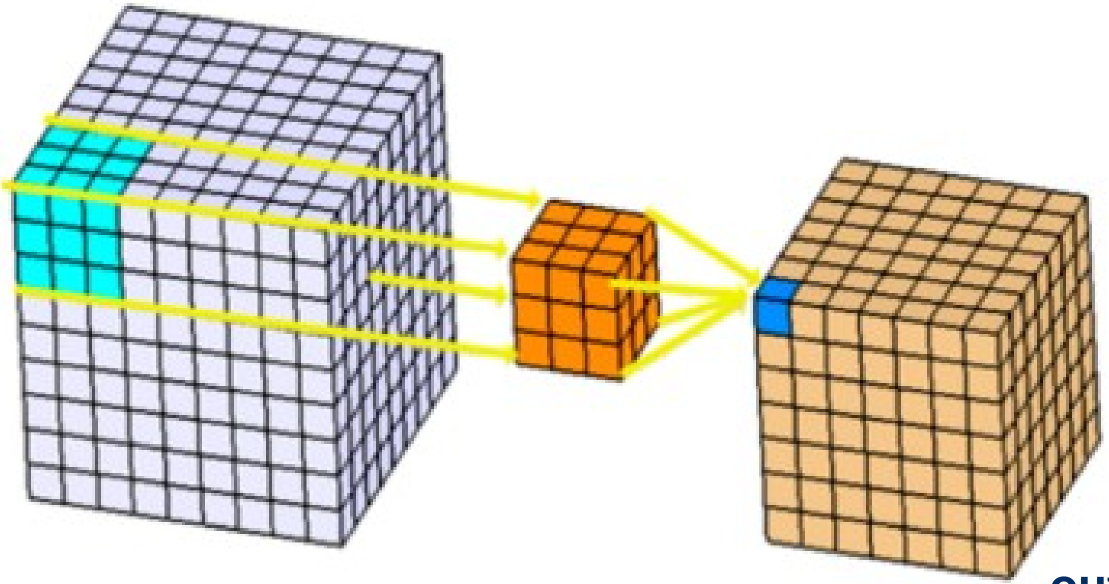
input



output

1D Convolutions

input



output

3D Convolutions

S. Bansal, 3D Convolutions: Understanding + Use Case, Kaggle, 2019

Summarizing

[Finally :D]

Summarizing

- **ConvNets enable introducing data characteristics**
  - (locality, position invariance, compositionality, etc.)

Summarizing

- **ConvNets enable introducing data characteristics**
  - **(locality, position invariance, compositionality, etc.)**

- **There are serveral ways to define a convolution**
  - **Full | same | valid | dilated | strided**

Summarizing

- ConvNets enable introducing data characteristics
  - (locality, position invariance, compositionality, etc.)

- There are serveral ways to define a convolution
  - Full | same | valid | dilated | strided

- Some types of convolutions may have “special” effects
  - Strided → decrease spatial resolution
  - Dilated→ increase receptive field
  - 1x1 → modify the number of channels

Summarizing

- ConvNets enable introducing data characteristics
  - (locality, position invariance, compositionality, etc.)

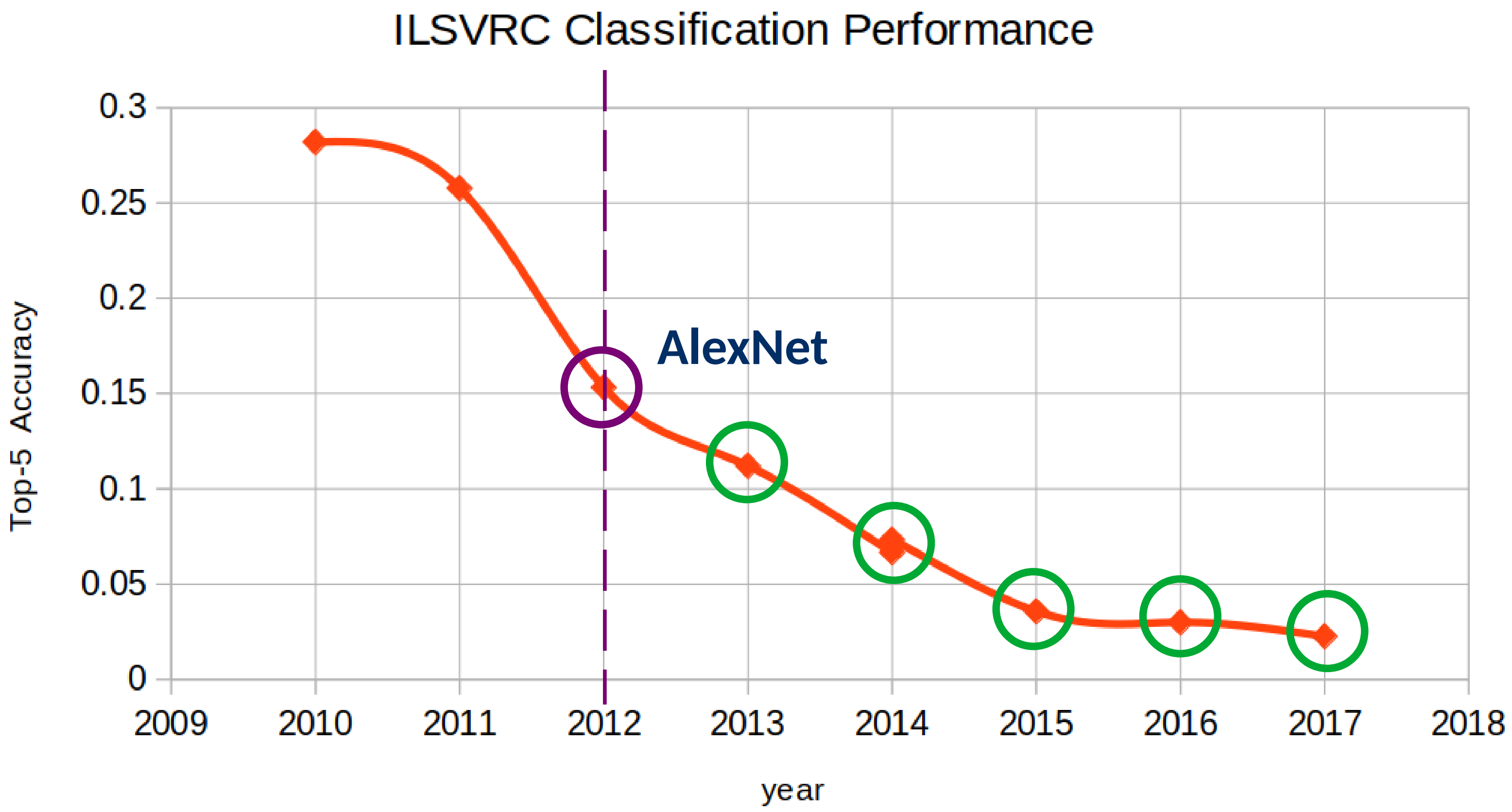
- There are serveral ways to define a convolution
  - Full | same | valid | dilated | strided

- Some types of convolutions may have “special” effects
  - Strided → decrease spatial resolution
  - Dilated→ increase receptive field
  - 1x1 → modify the number of channels

Next Lecture

Revelant Architectures:

[AlexNet, VGG-Net, GoogLeNet, ResNet, *-Net]



References

- Kunihiko Fukushima, Sei Miyake, ***Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position***, Pattern Recognition, Volume 15, Issue 6. 1982.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, ***Handwritten digit recognition with a back-propagation network***. NeurIPS 1989
- Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. ***Gradient-based Learning Applied to Document Recognition***. Proceedings of IEEE, 1998
- A. Krizhevsky, I. Sutskever, G. E. Hinton. ***ImageNet Classification with Deep Convolutional Neural Networks***. NeurIPS 2012
- Y. LeCun, K. Kavukcuoglu and C. Farabel. ***Convolutional Networks and Applications in Vision***
- D. E. Rumelhart, G. E. Hinton & R. J. Williams. ***Learning representations by back-propagating errors.*** 1986
- L. Antanas, M. van Otterlo, J. Oramas, T. Tuytelaars and L. De Raedt. ***There are Plenty of Places like Home: Using Hierarchies and Relational Representations for Distance-based Image Understanding***. Neurocomputing 2014.

Convolutional Neural Networks

[ConvNets, CNNs]