CONVOLUTIONAL NEURAL NETWORKS:

DISCRETE CONVOLUTIONS

Convolution operations first published by D’Alembert in 1754

Discrete convolutions are matrix operations that can be used to apply **filters** to a matrix or array

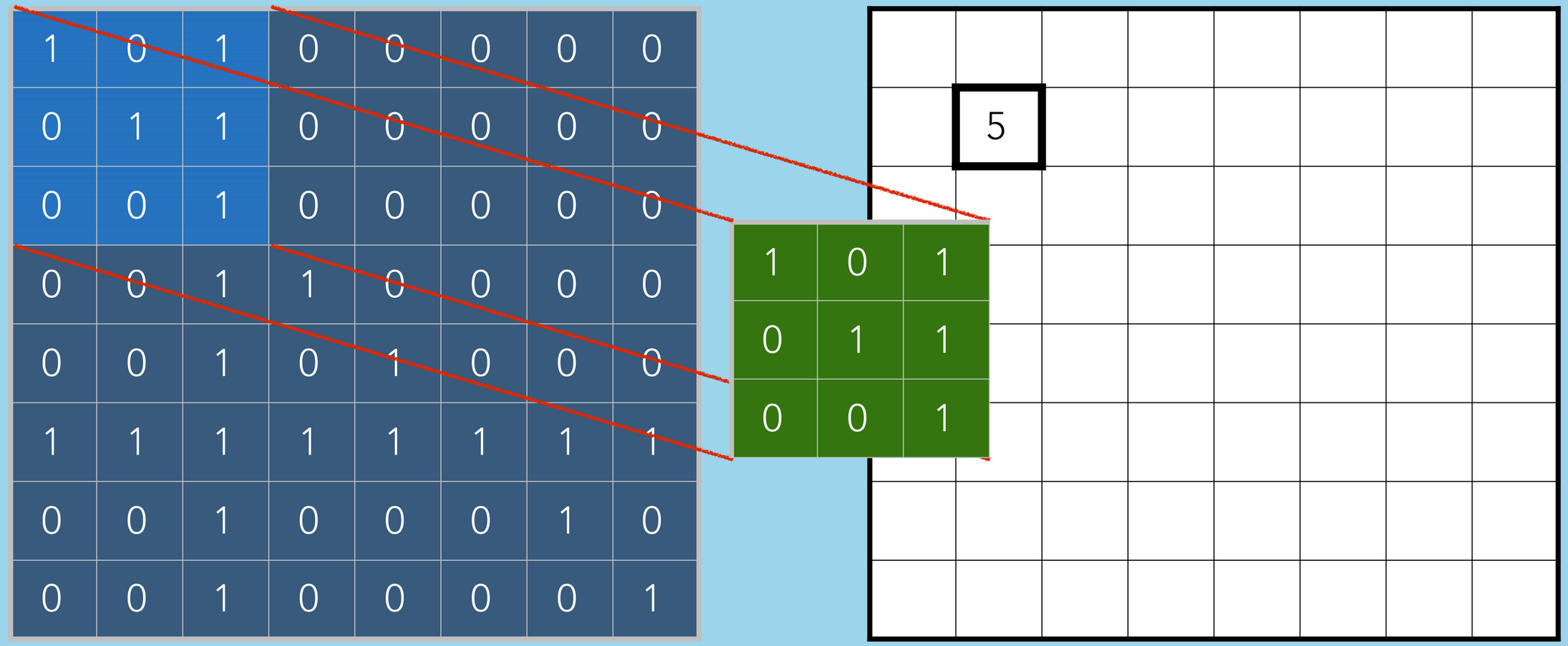
The convolutional neural network architecture was first described by Kunihiko Fukushima in 1980

pre-defined filters

Discrete convolutions are matrix operations that can be used to apply **filters** to a matrix or array

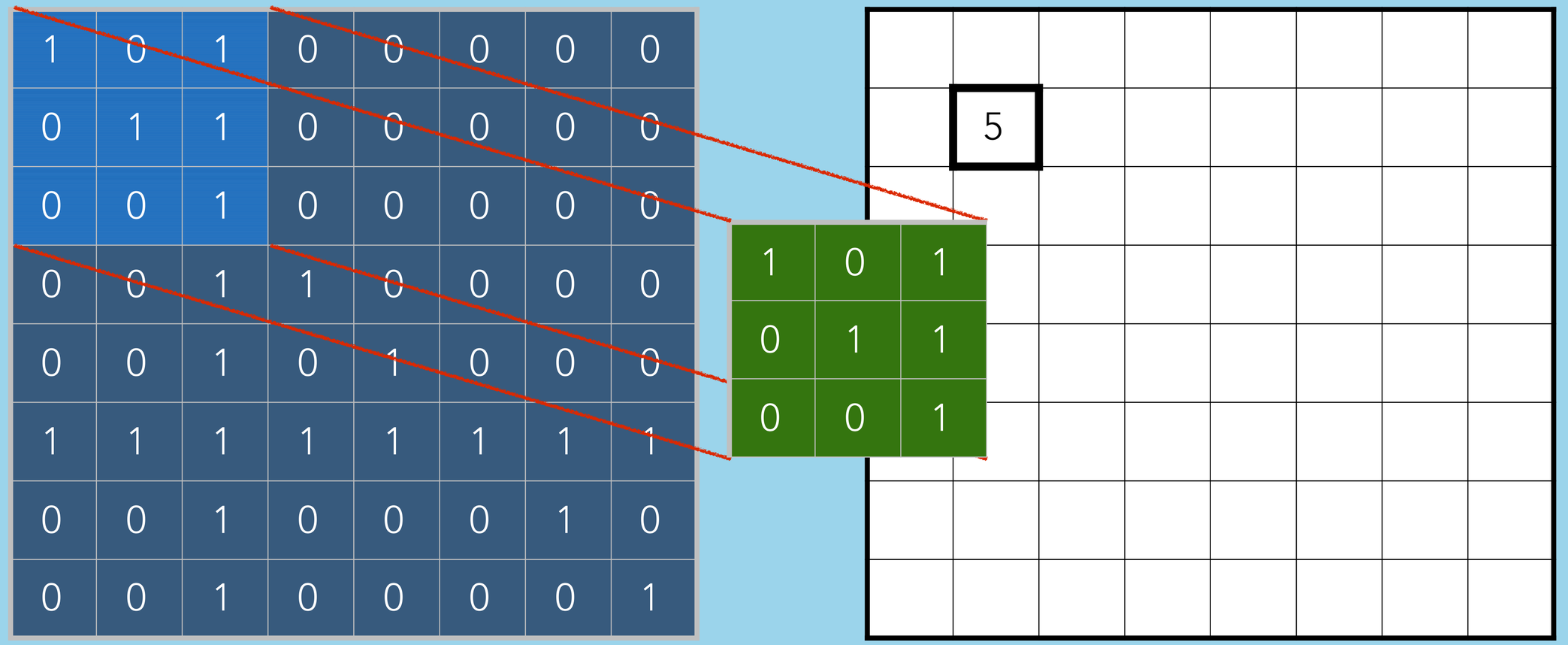
Discrete Convolutions $C = A \circledast h$

$$C[m,n] = \sum_{j=-\omega}^{\omega} \sum_{i=-\omega}^{\omega} h[i+\omega, j+\omega] * A[m+i, n+j]$$



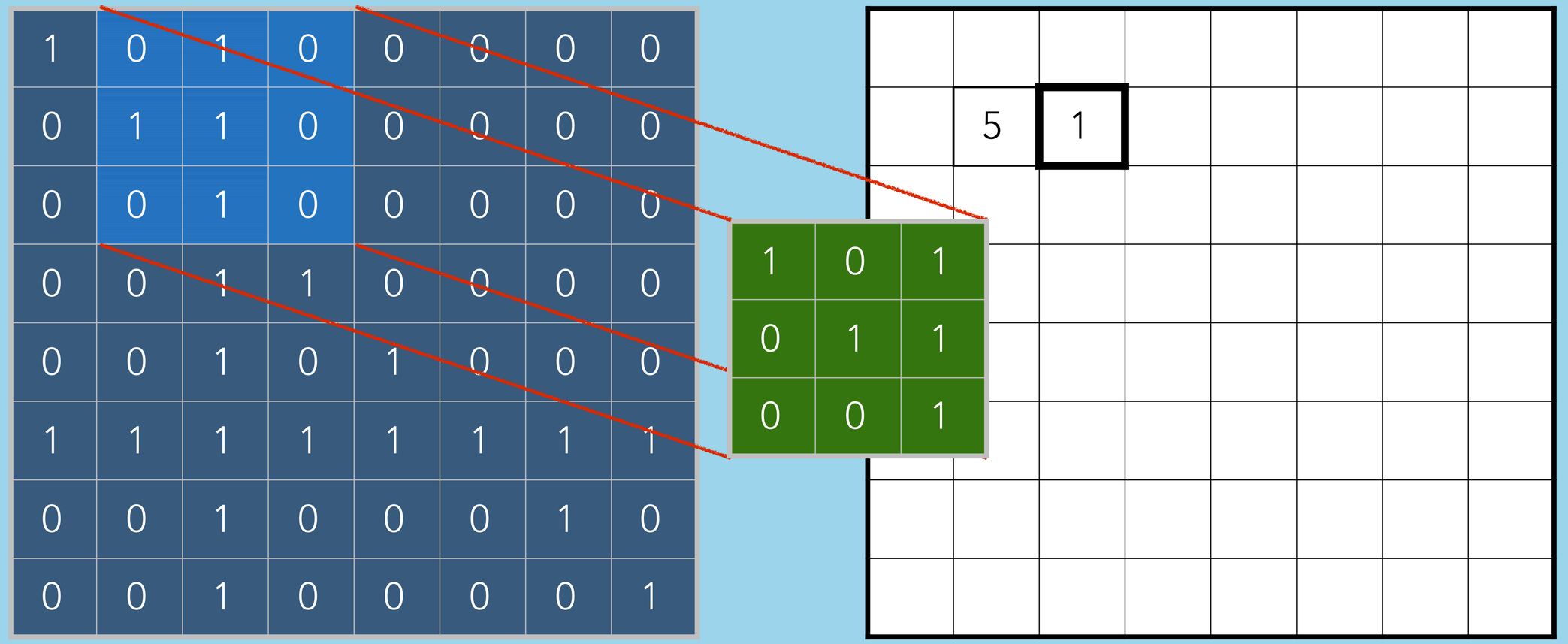
Discrete Convolutions $C = A \circledast h$

Filter Stride Edges



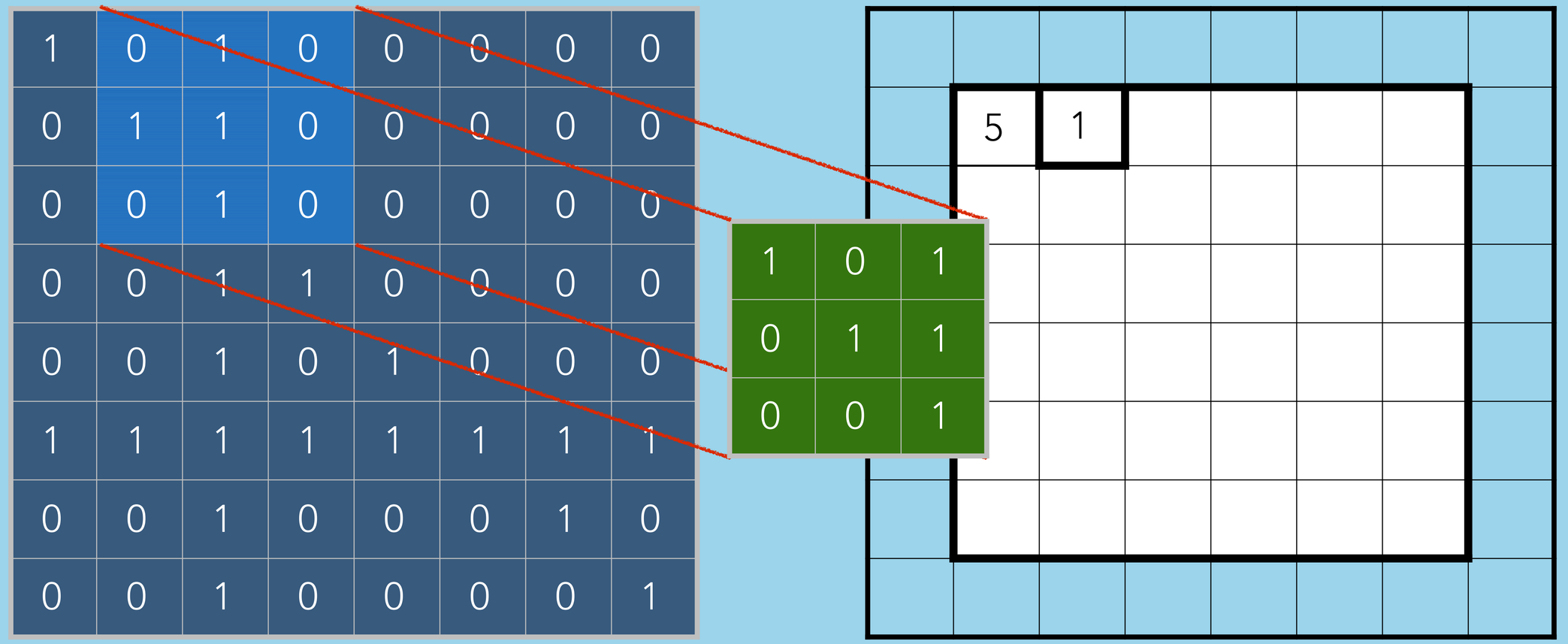
Discrete Convolutions $C = A \circledast h$

Filter Stride Edges



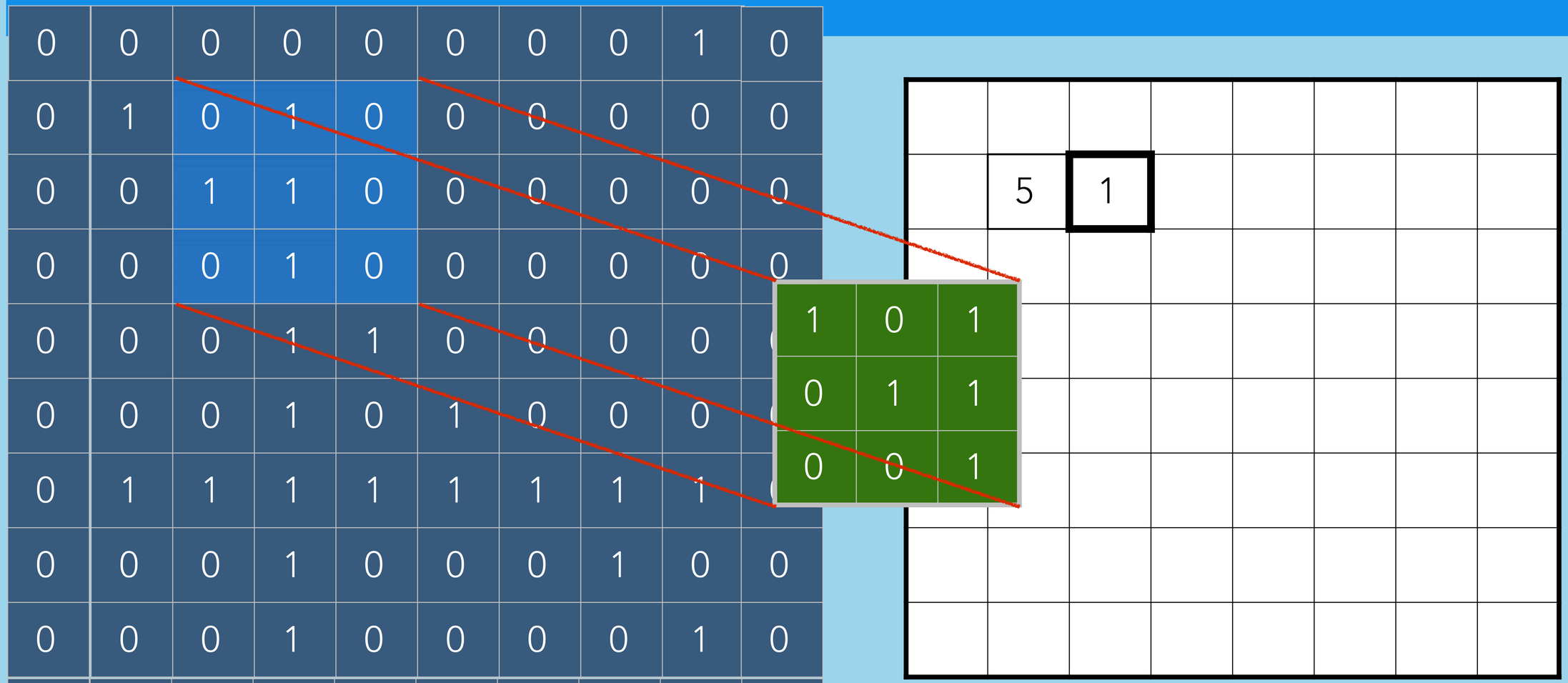
Discrete Convolutions $C = A \circledast h$

Filter Stride Edges



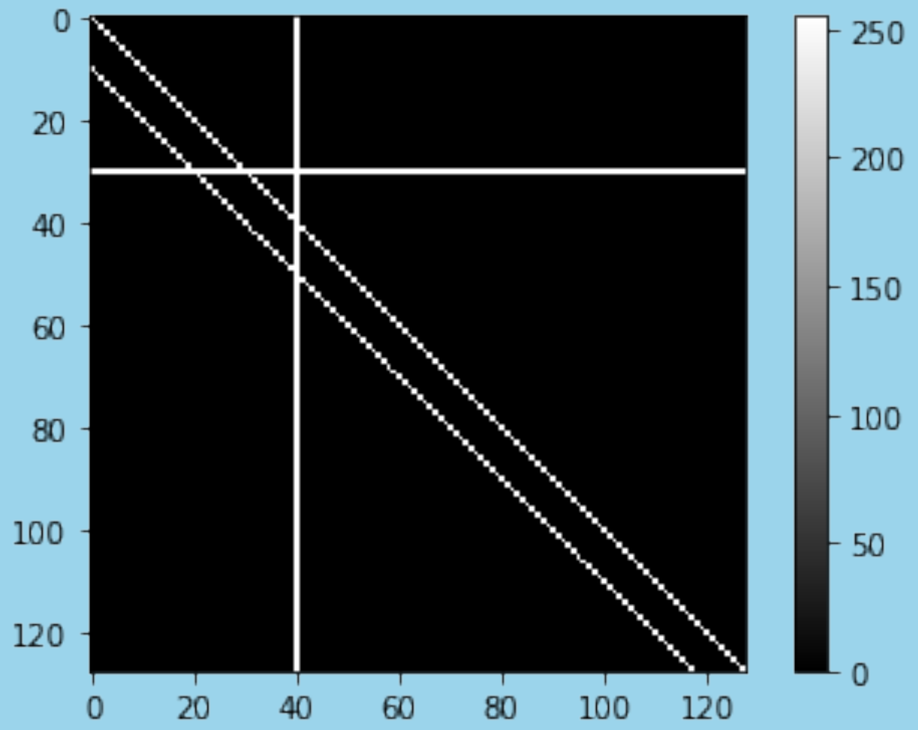
Discrete Convolutions $C = A \circledast h$

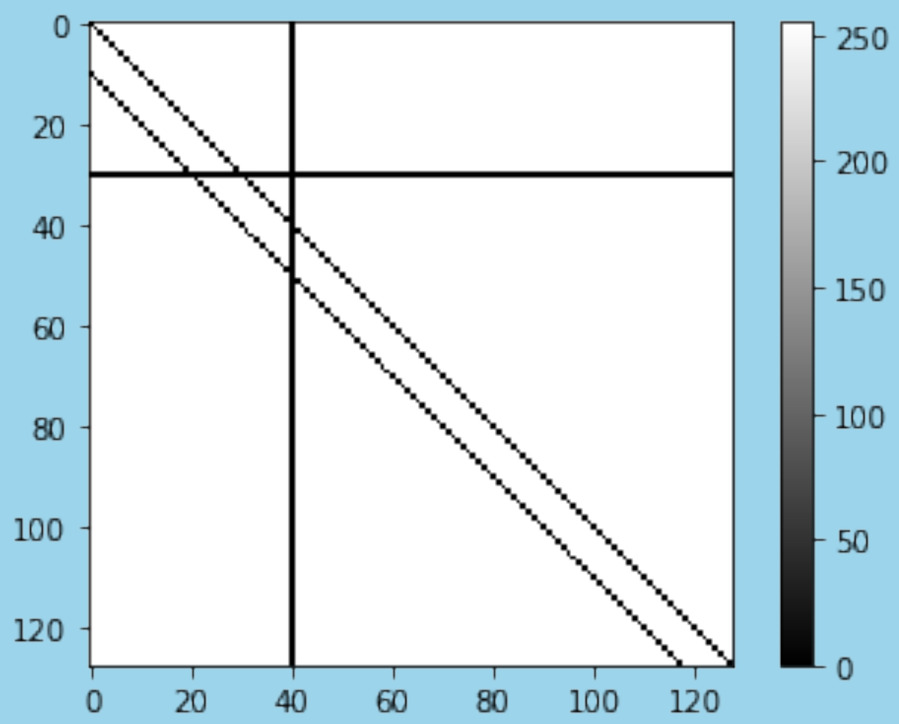
Filter Stride Edges



Discrete Convolutions $C = A \circledast h$

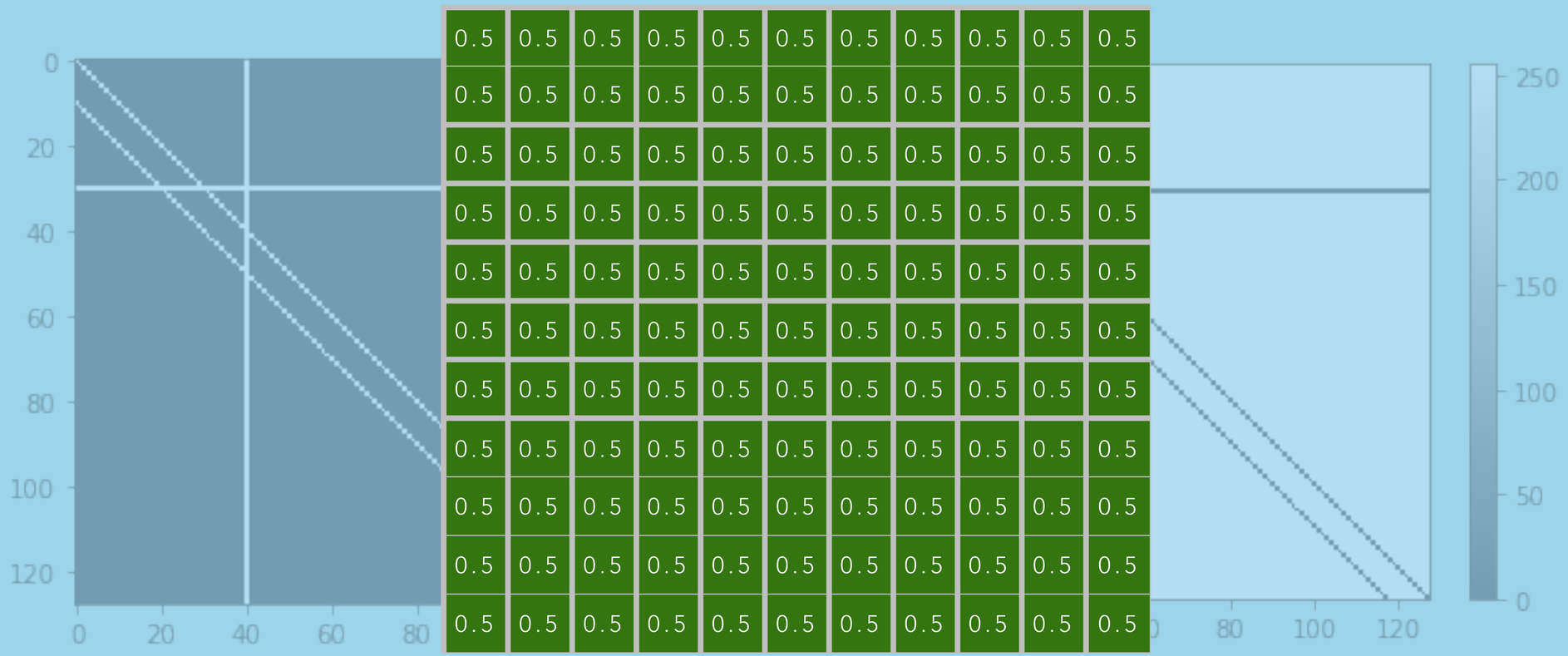
Filter	Stride	Edges
(11x11)	1	None





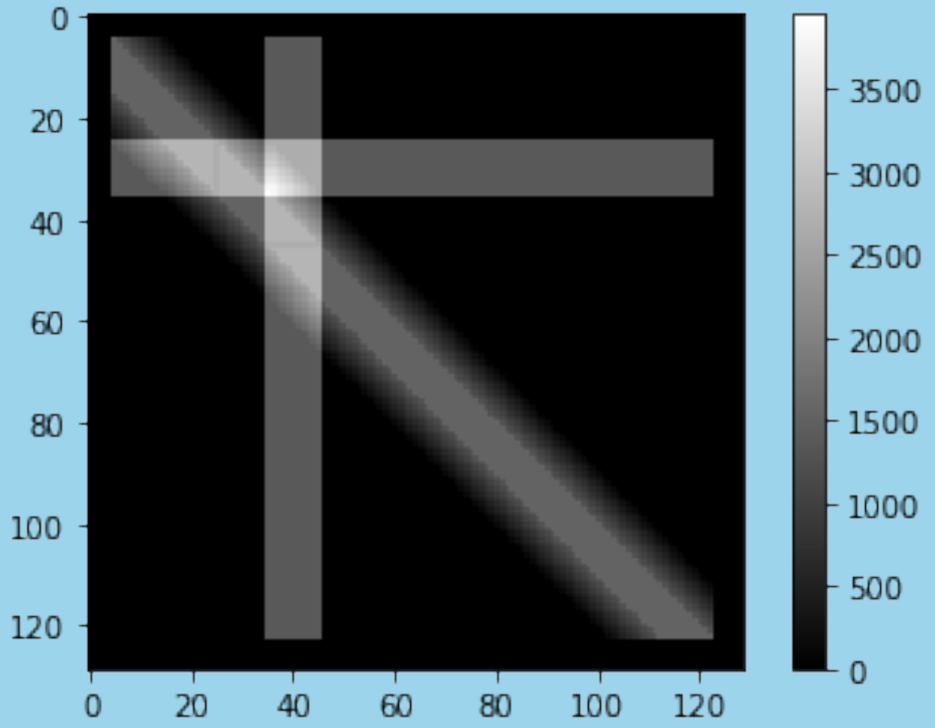
Discrete Convolutions $C = A \circledast h$

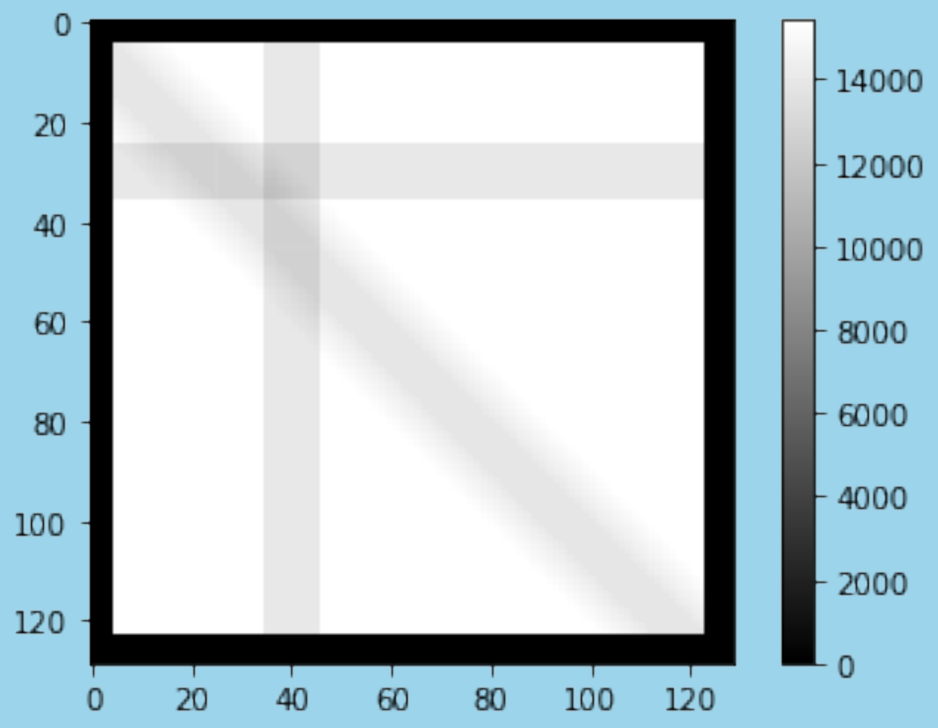
Filter	Stride	Edges
(11x11)	1	None



Discrete Convolutions $C = A \circledast h$

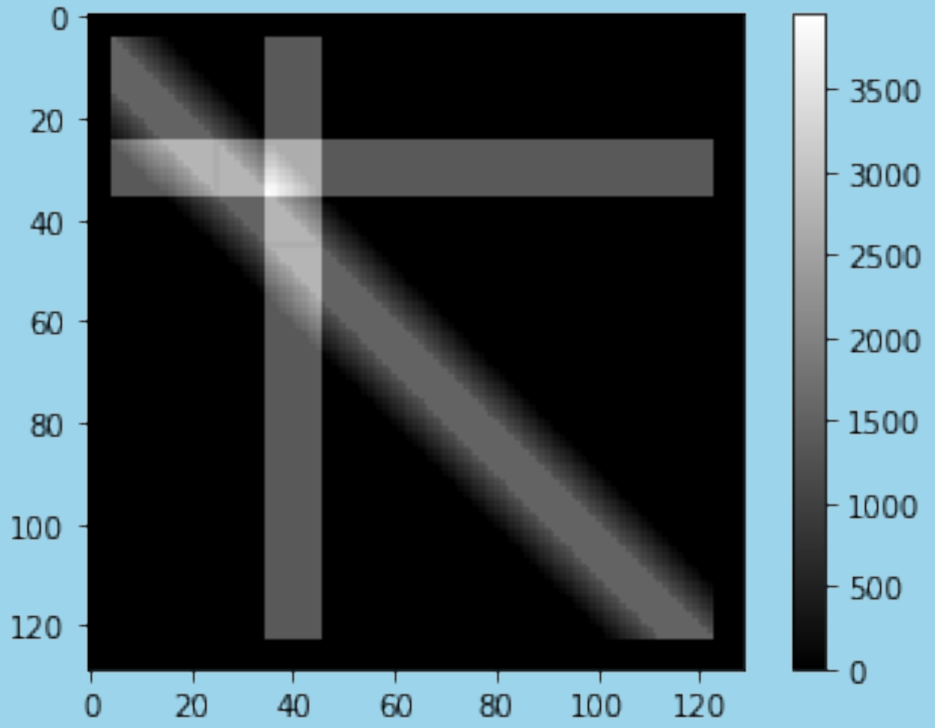
Filter	Stride	Edges
(11x11)	1	None

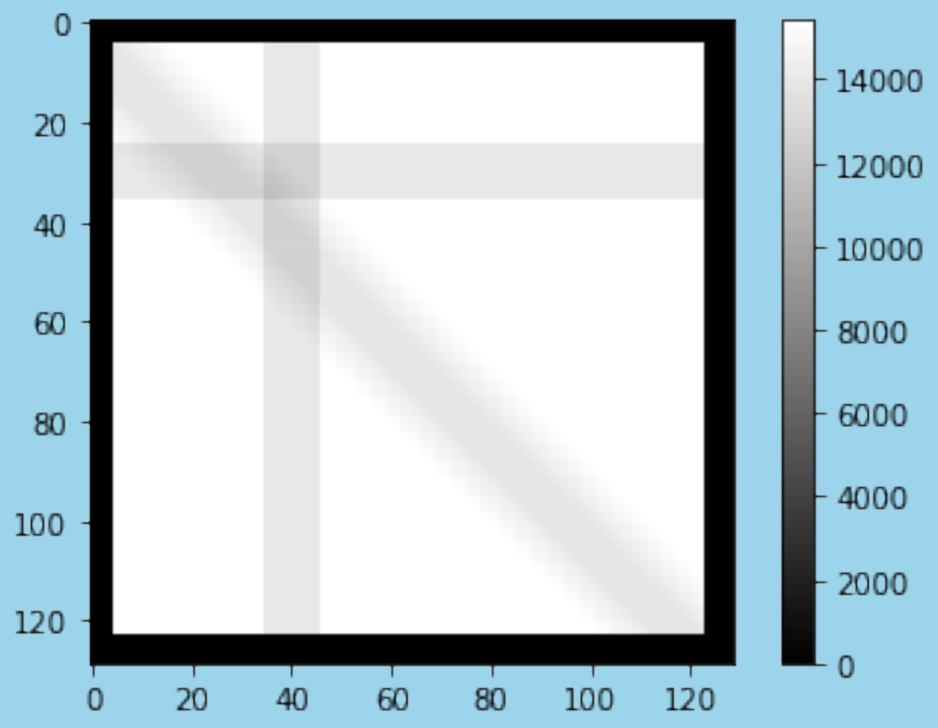




Discrete Convolutions $C = A \circledast h$

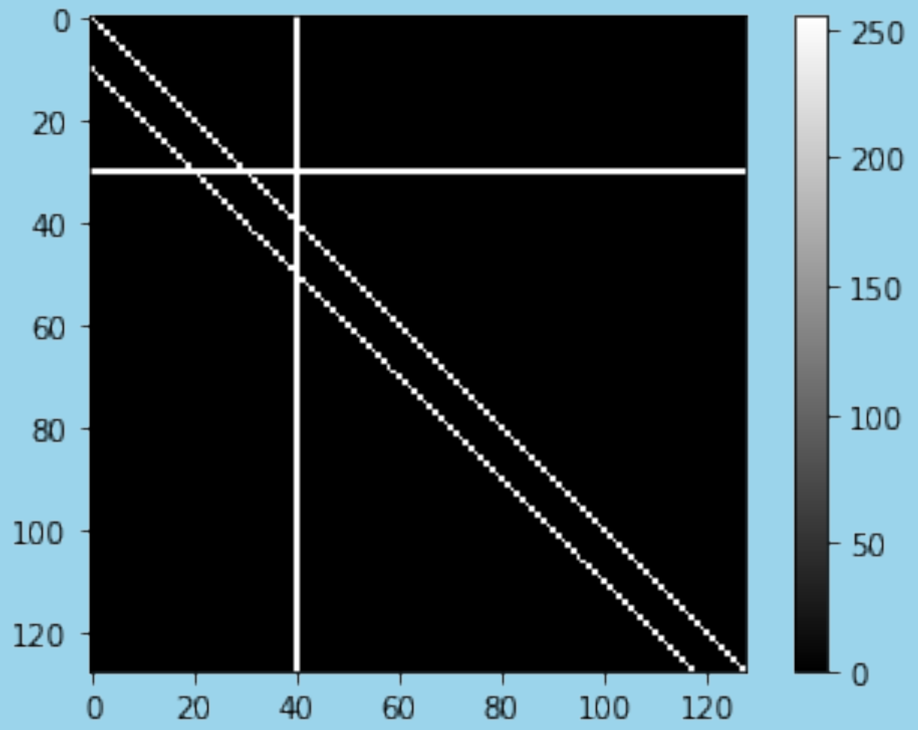
Blurring Filter

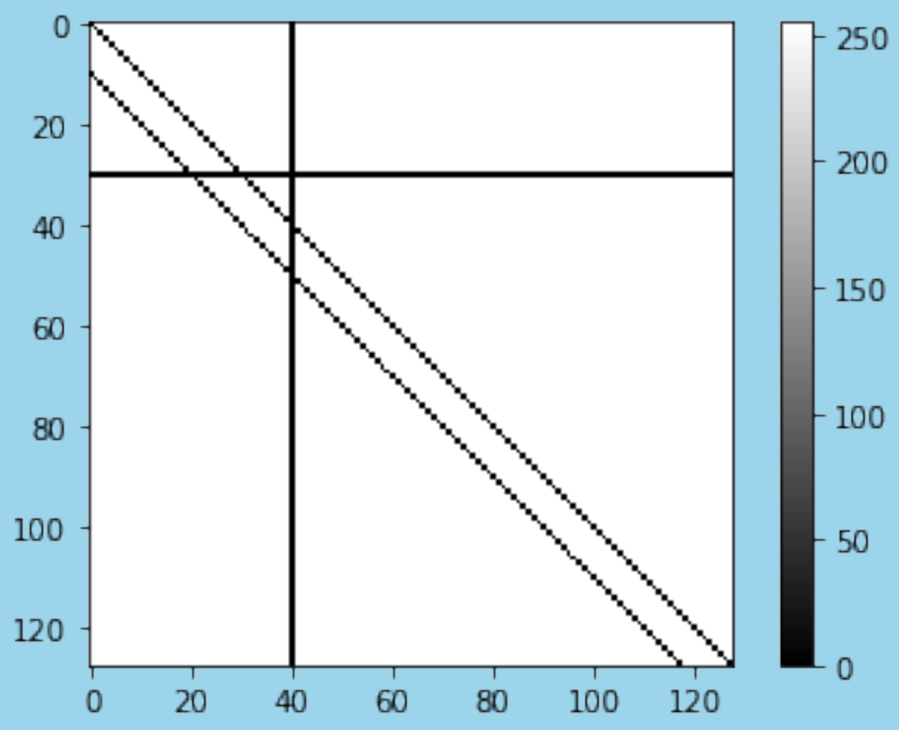




Discrete Convolutions $C = A \circledast h$

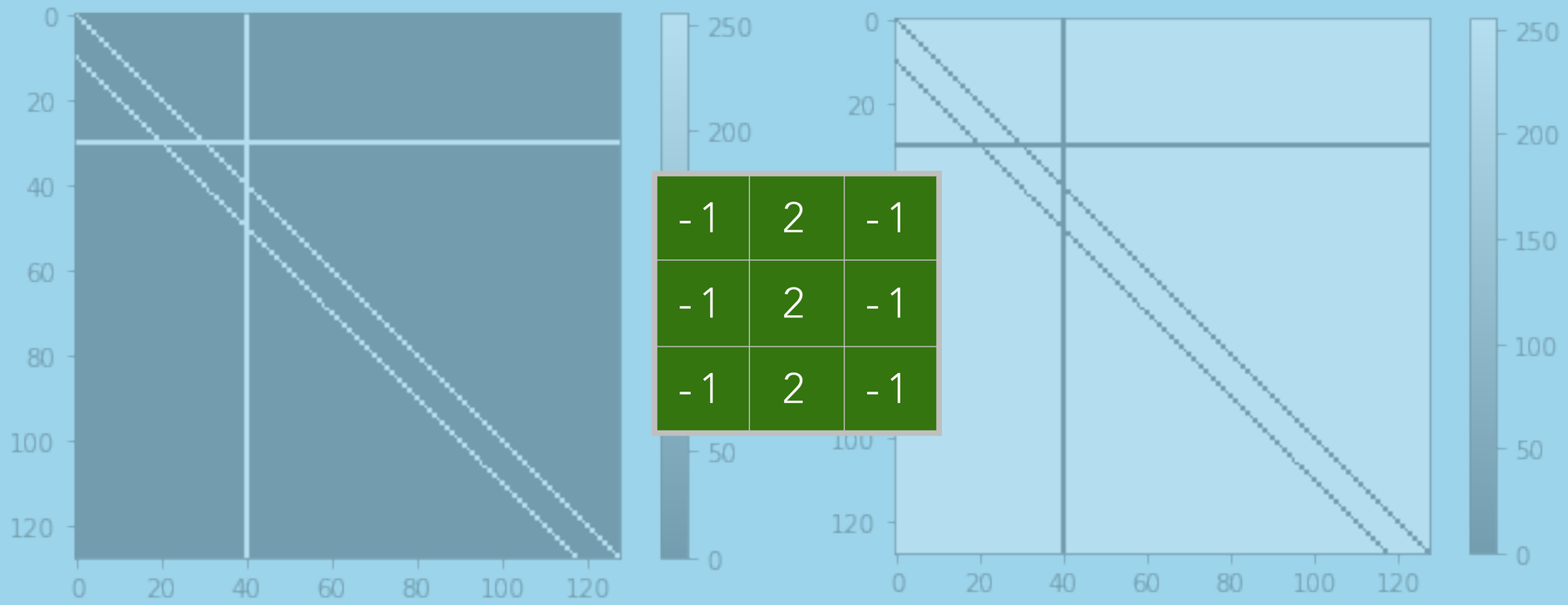
Filter	Stride	Edges
(3x3)	1	None





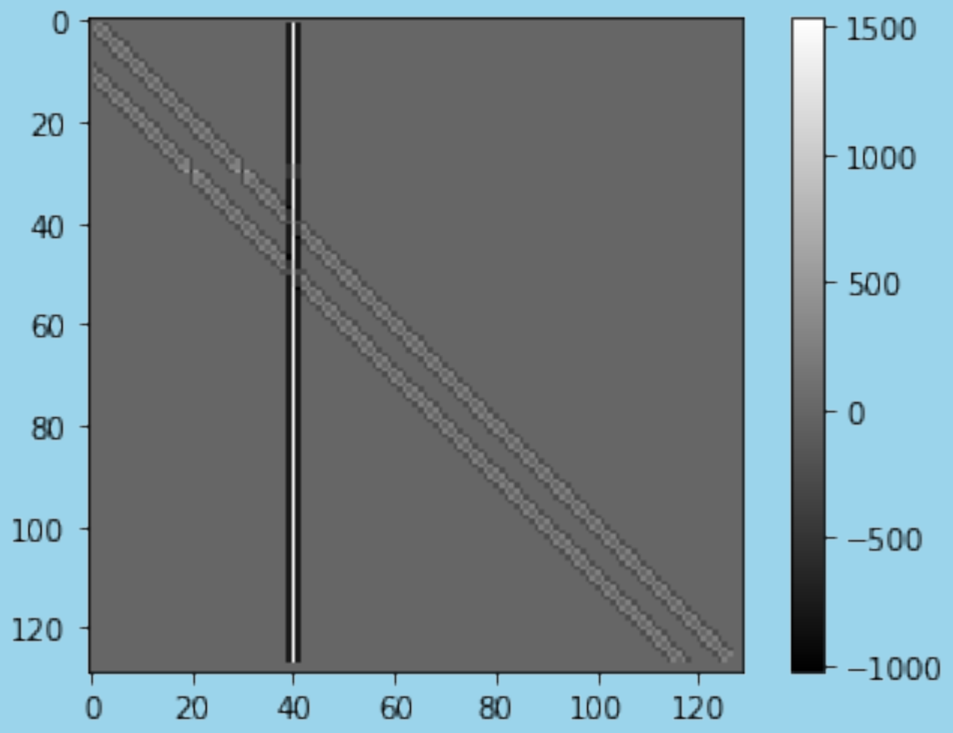
Discrete Convolutions $C = A \circledast h$

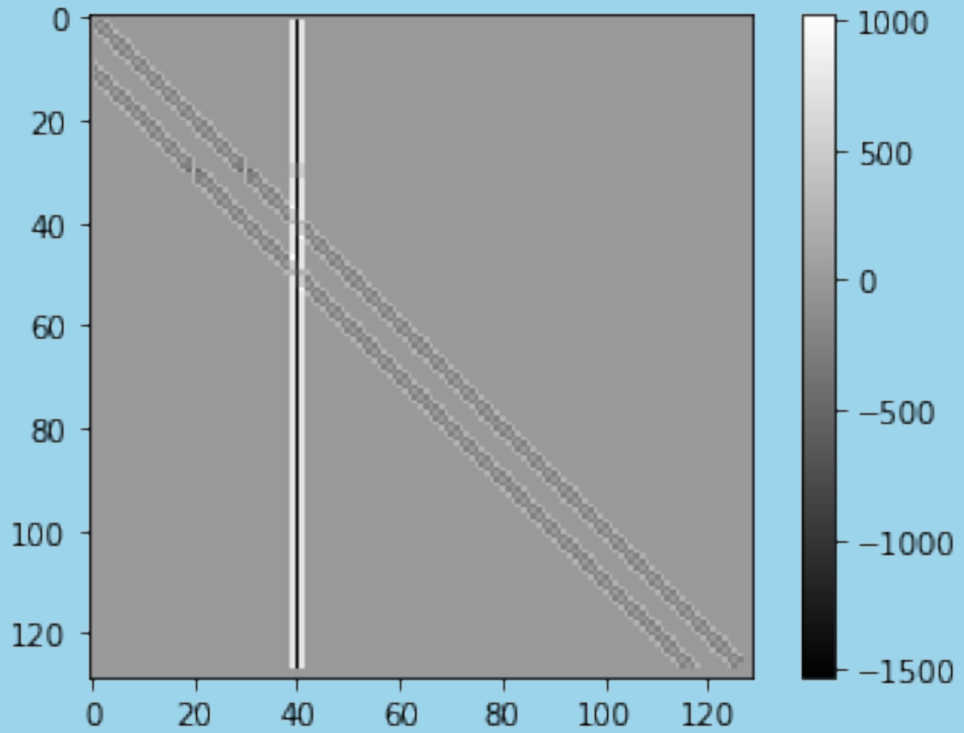
Filter	Stride	Edges
(3x3)	1	None



Discrete Convolutions $C = A \circledast h$

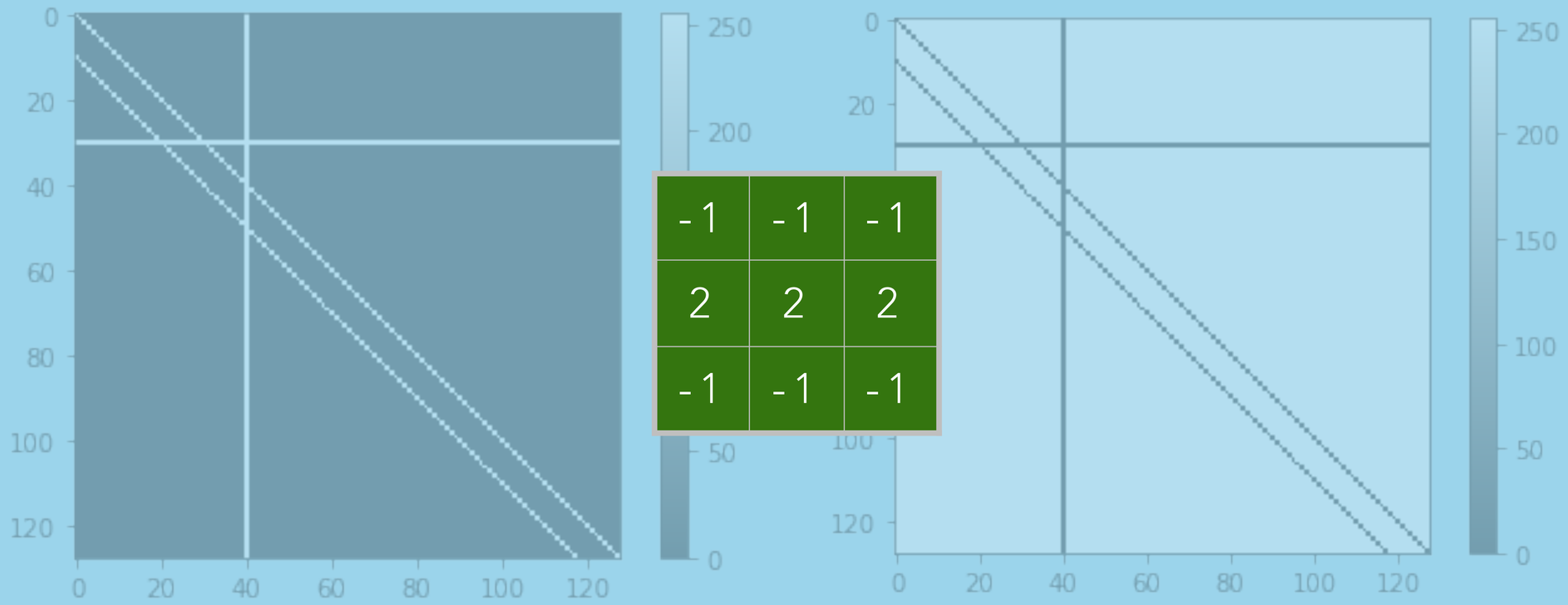
Filter	Stride	Edges
(3x3)	1	None





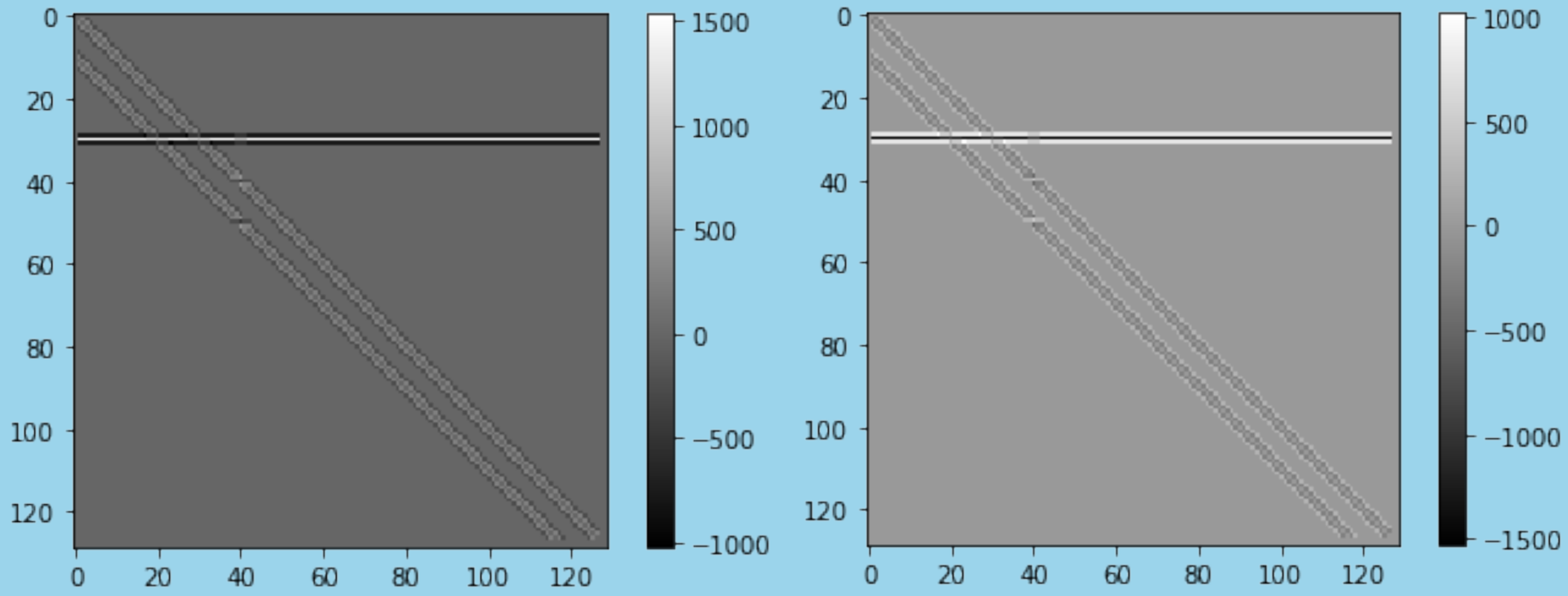
Discrete Convolutions $C = A \circledast h$

Filter	Stride	Edges
(3x3)	1	None



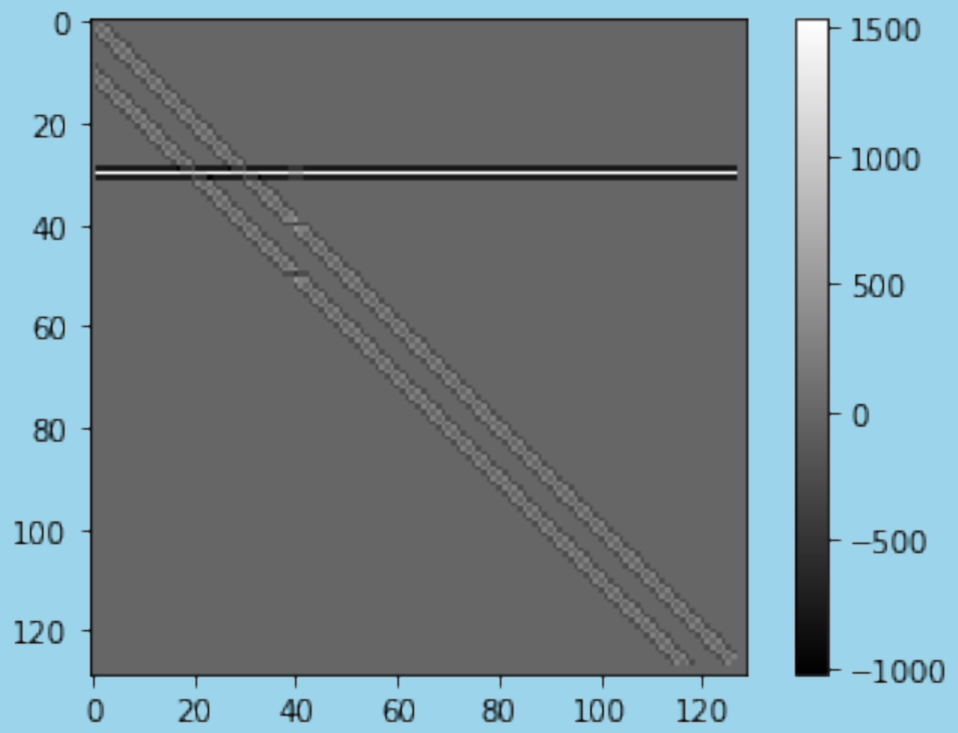
Discrete Convolutions $C = A \circledast h$

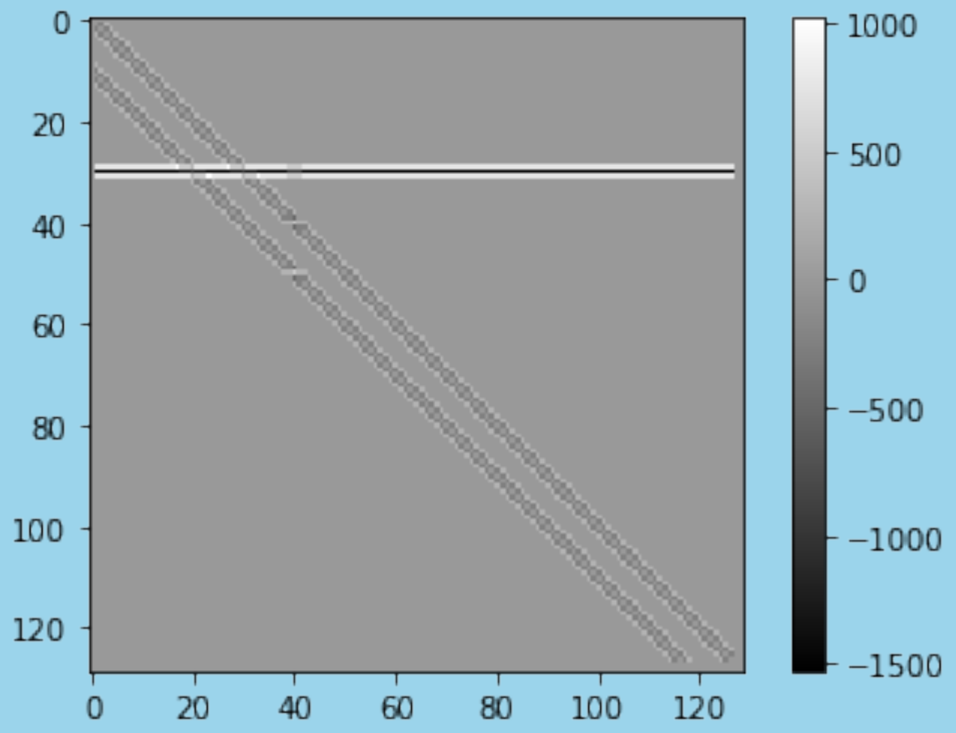
Filter	Stride	Edges
(3x3)	1	None



Discrete Convolutions $C = A \circledast h$

Line Detection / Extraction





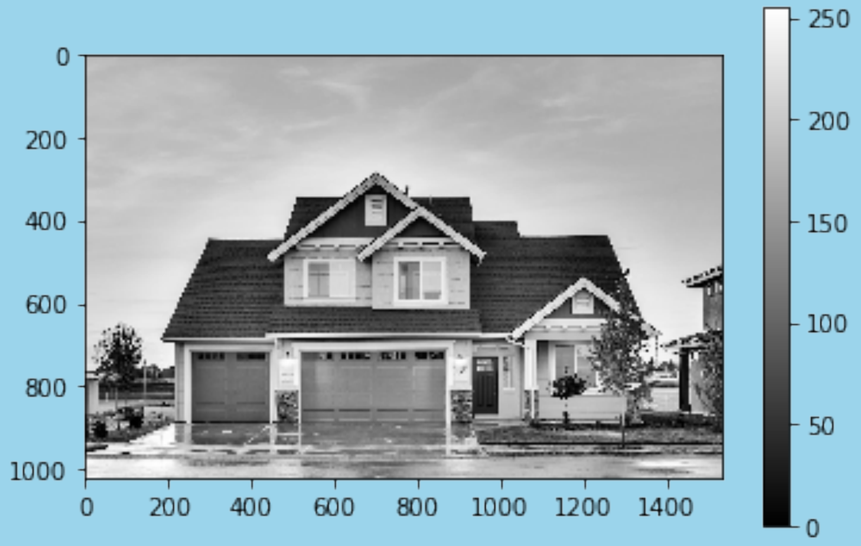
Discrete Convolutions $C = A \circledast h$

Photograph

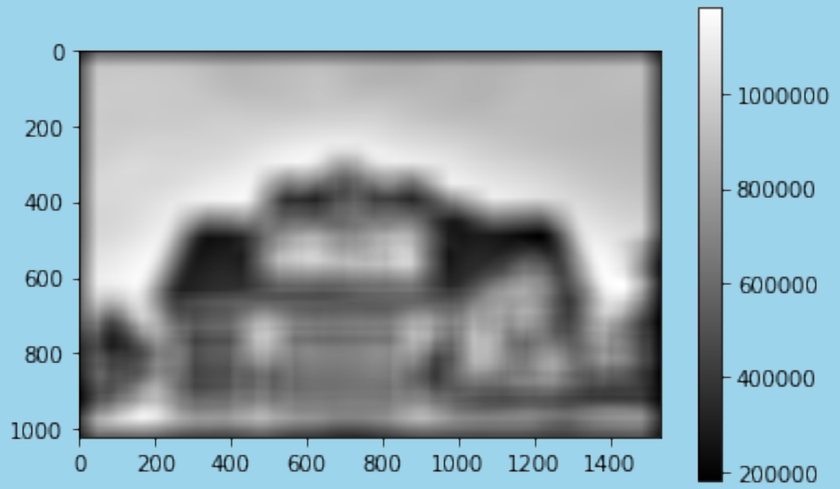
Discrete Convolutions $C = A \circledast h$

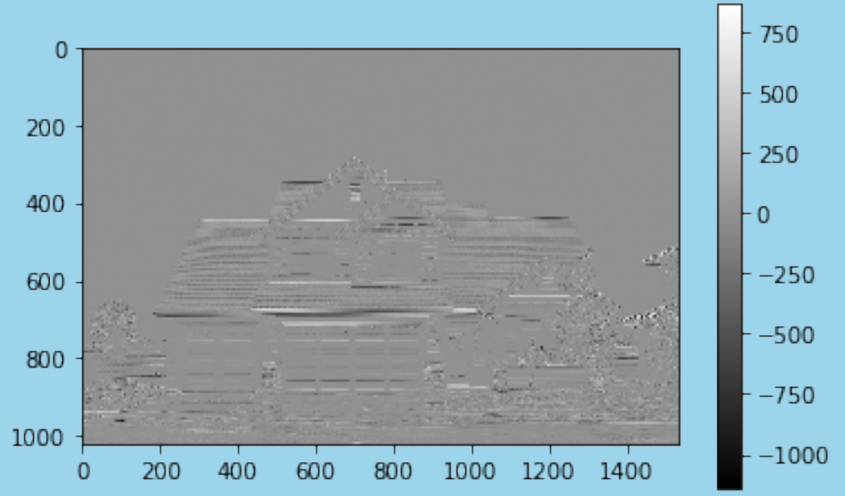
Photograph

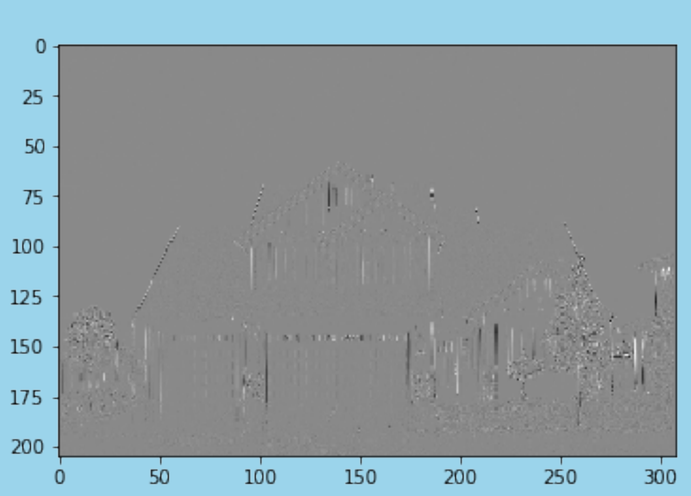
Discrete Convolutions $C = A \circledast h$



Filter (101x101)

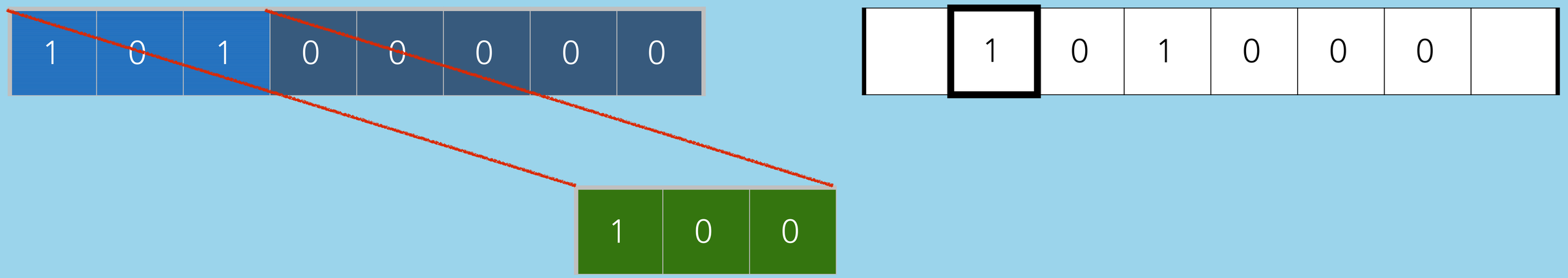






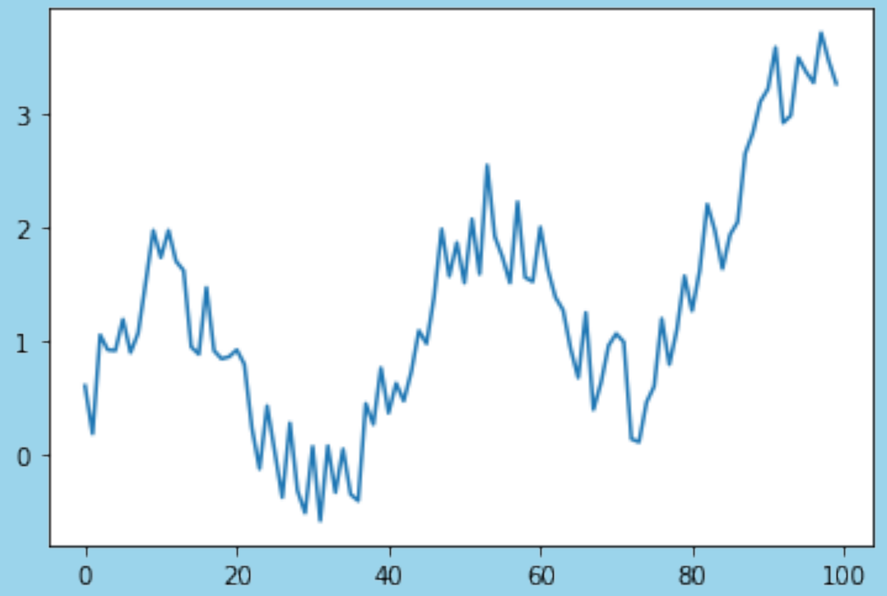
N-D Discrete Convolutions

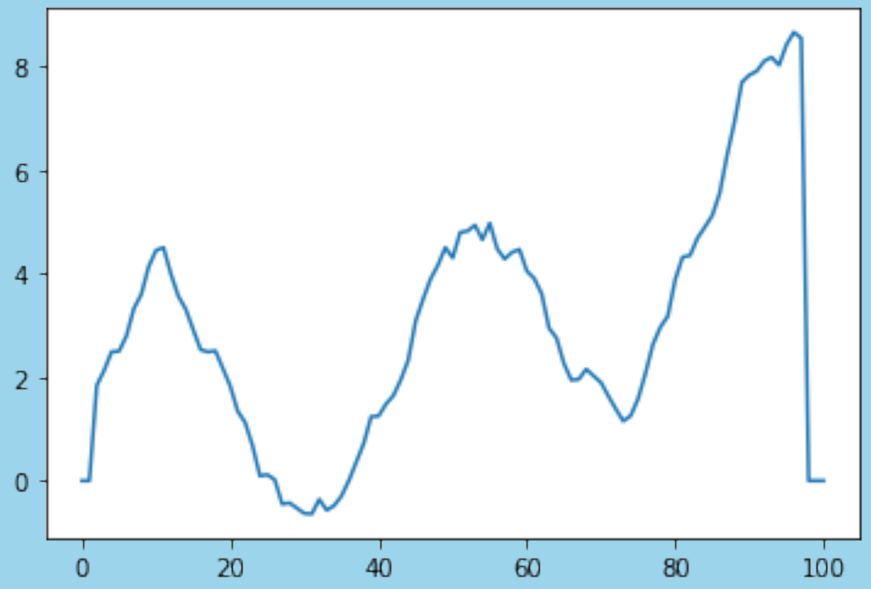
$$C[m,n] = \sum_{i=-\omega}^{\omega} h[i+\omega] * A[m+i]$$



N-D Discrete Convolutions

Filter	Stride	Edges
(1x5)	1	None





0.5	0.5	0.5	0.5	0.5

Discrete Convolutions

Discrete convolutions apply to **array** or **matrix**-like data

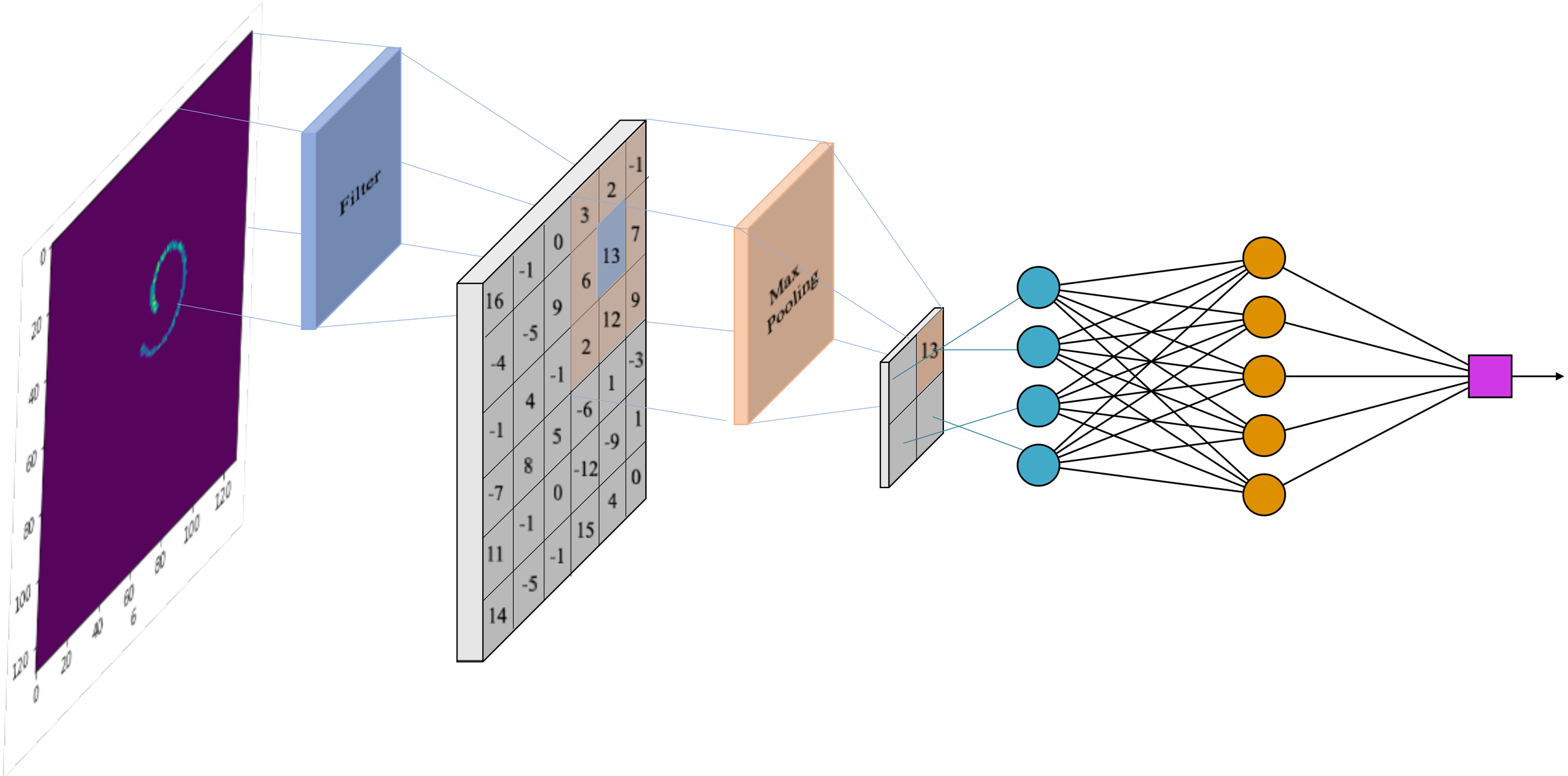
Discrete convolutions are matrix operations that can be used to apply **filters** to a matrix or array

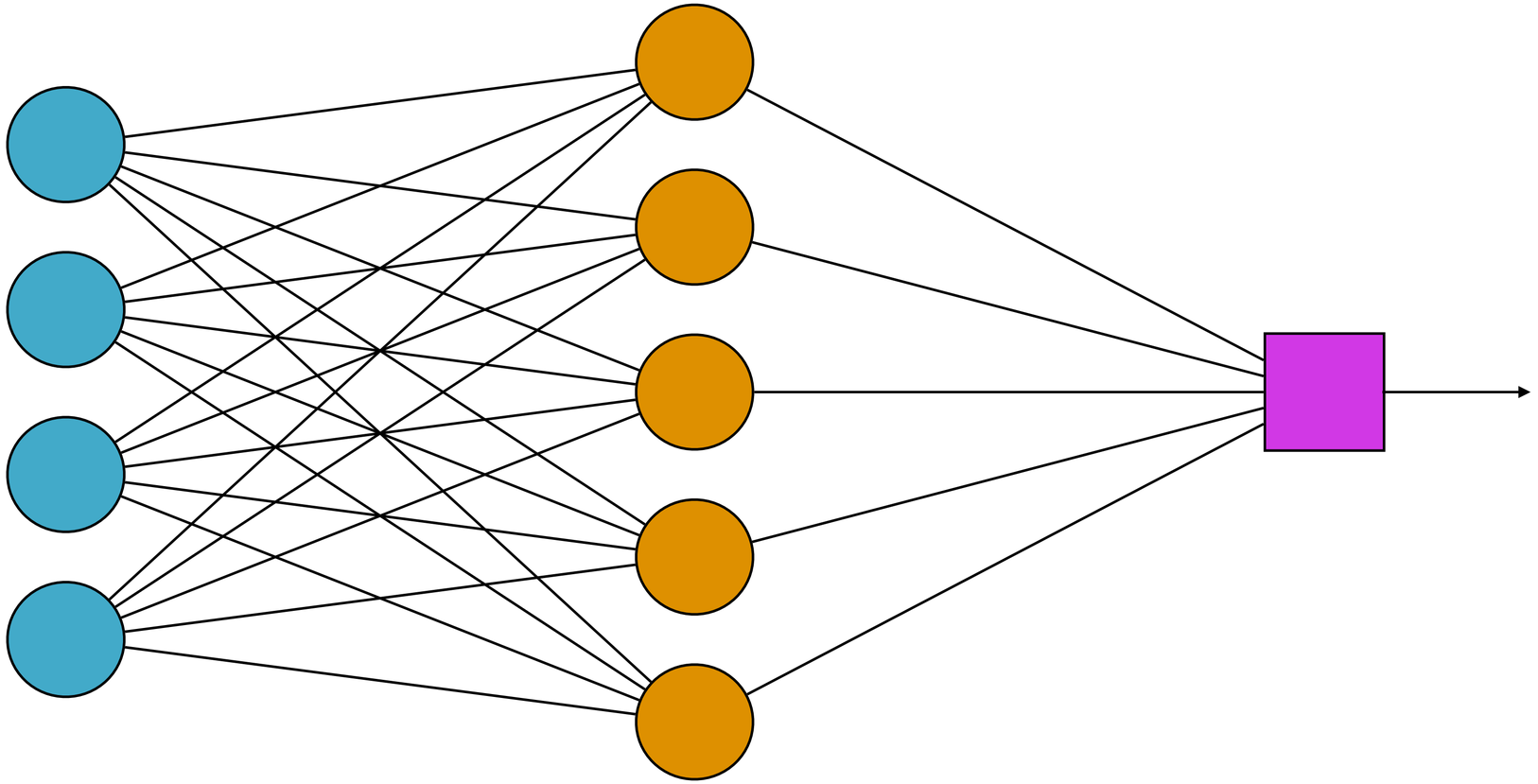
These filters can **extract features** or transform the input

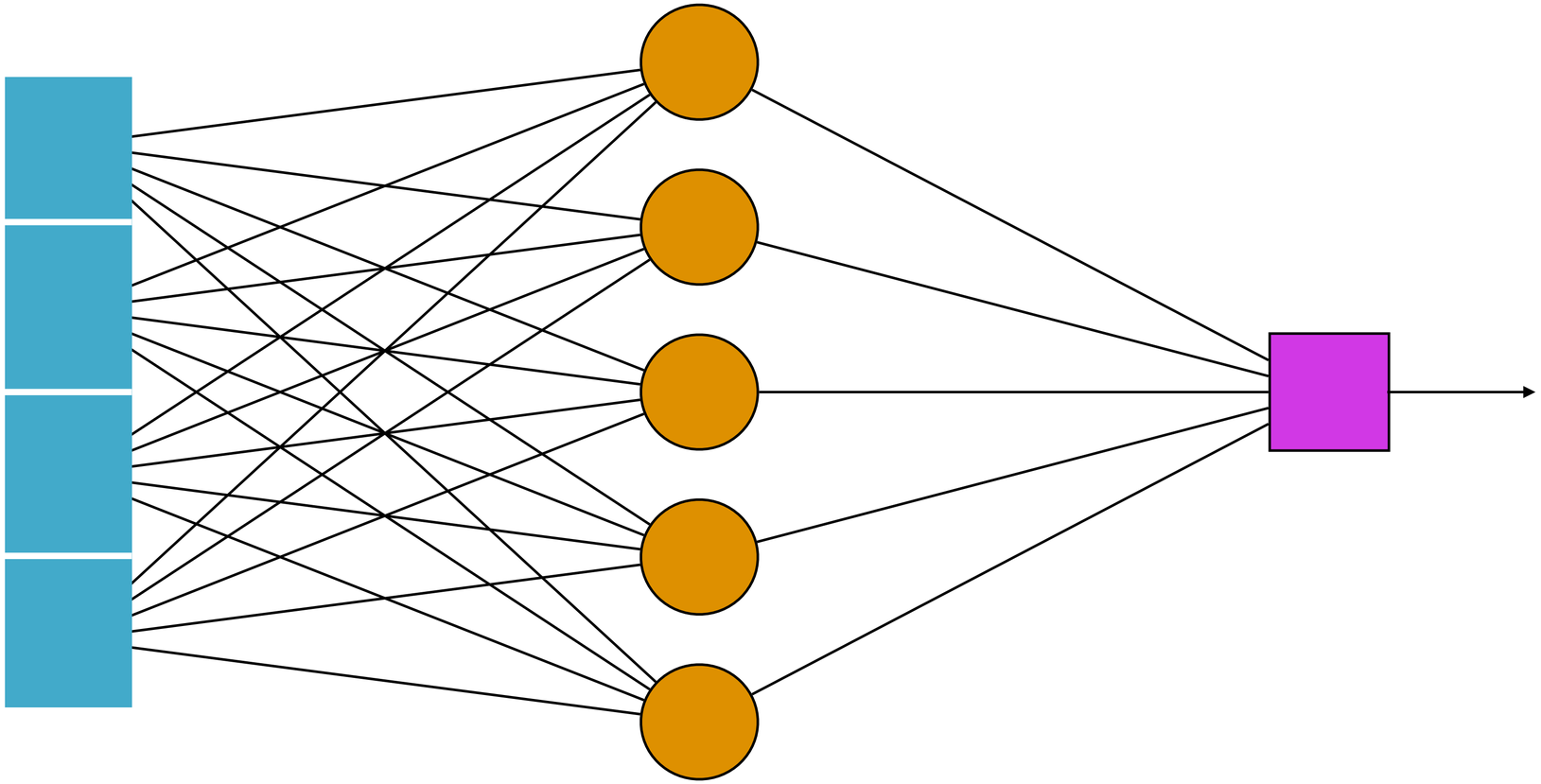
CONVOLUTIONAL NEURAL NETWORKS:

ARCHITECTURE AND TRAINING

CONVOLUTIONAL NEURAL NETWORKS

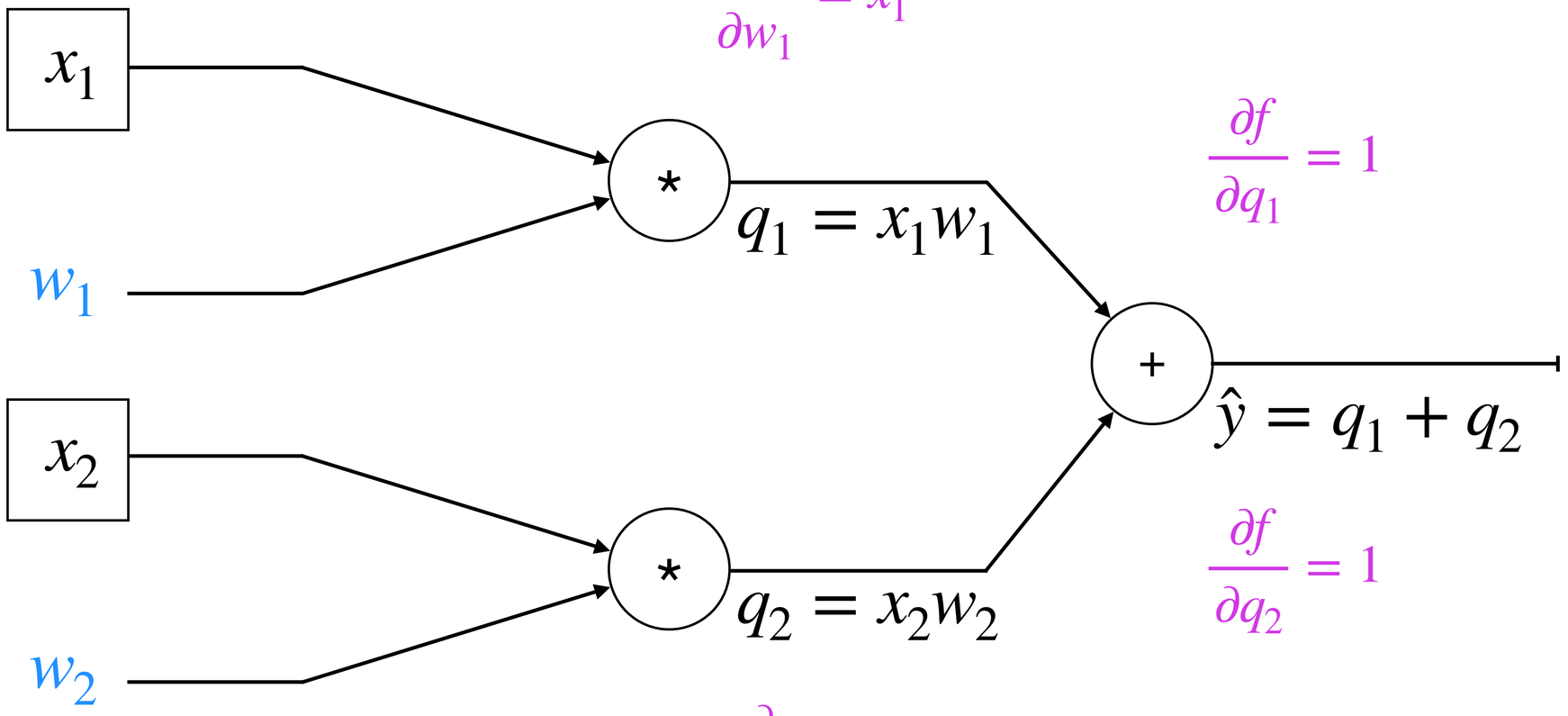






BACKPROPAGATION

$$w_1 = w_1 + \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_1} \frac{\partial q_1}{\partial w_1}$$



$$\frac{\partial q_1}{\partial w_1} = x_1$$

$$\frac{\partial f}{\partial q_1} = 1$$

$$\frac{\partial J}{\partial \hat{y}} = 1$$

$$\frac{\partial f}{\partial q_2} = 1$$

$$\frac{\partial q_2}{\partial w_2} = x_2$$

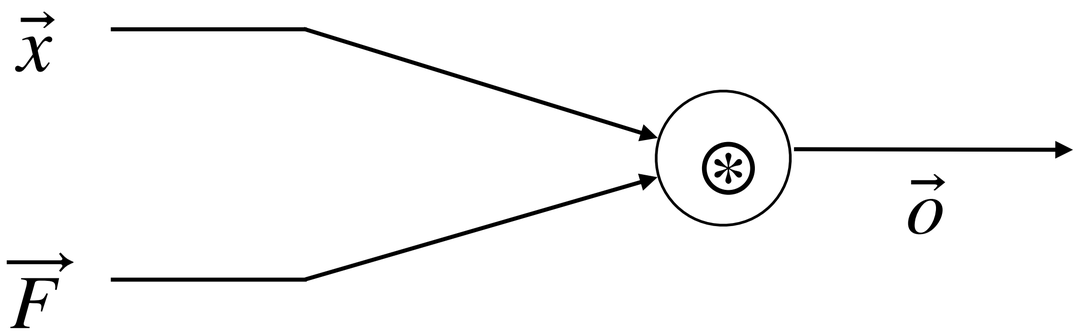
$$w_2 = w_2 + \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_1} \frac{\partial q_1}{\partial w_2}$$

Loss function

$$J(w) = \hat{y} - y$$

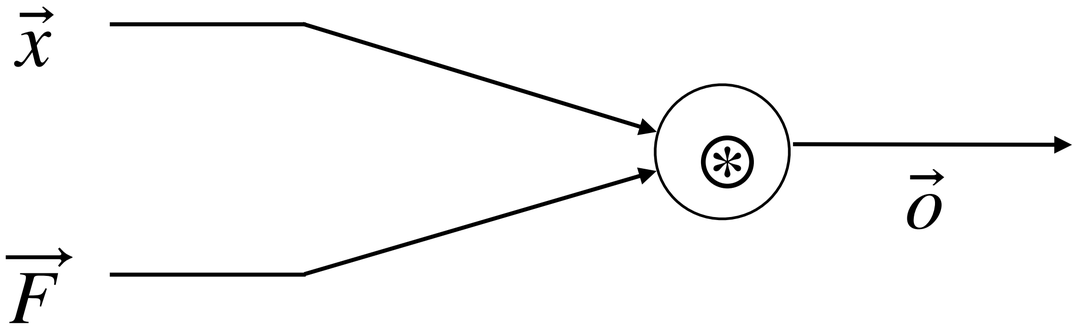
BACKPROPAGATION

$$\vec{o} = \vec{x} \circledast \overrightarrow{F}$$

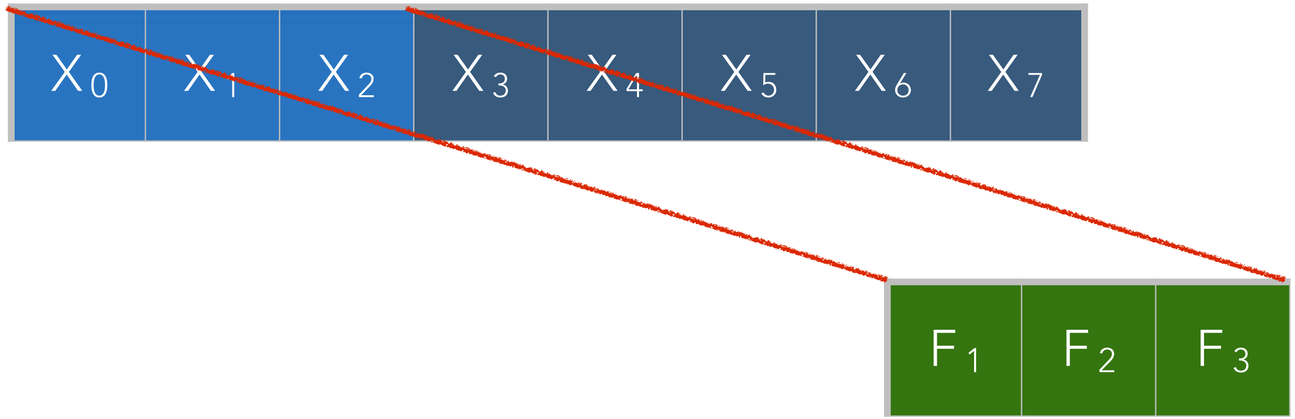


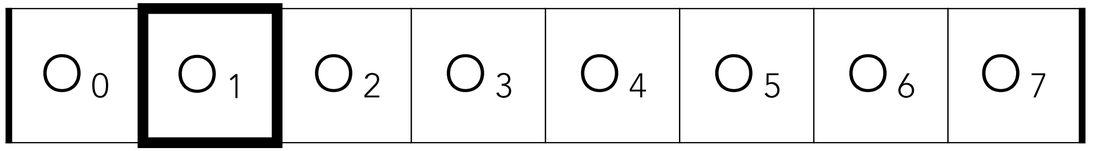
BACKPROPAGATION

$$\vec{o} = \vec{x} \circledast \vec{F}$$



$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$



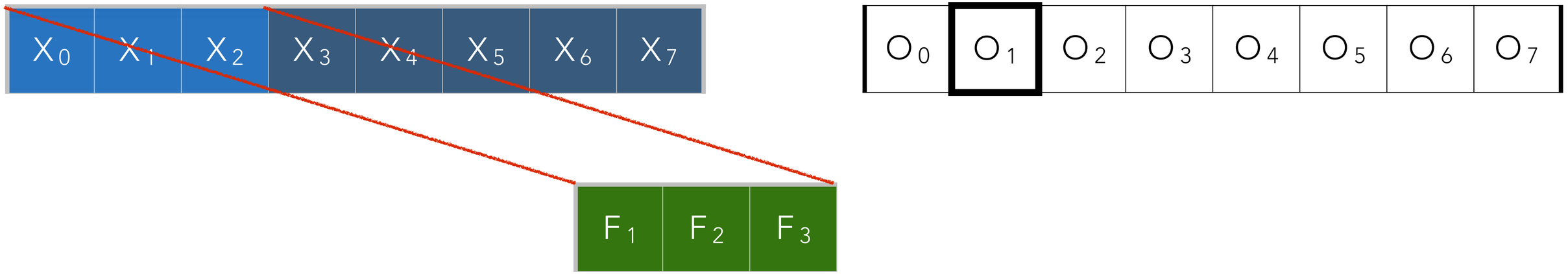


BACKPROPAGATION

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k} \frac{\partial o_k}{\partial F_i}$$

$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$

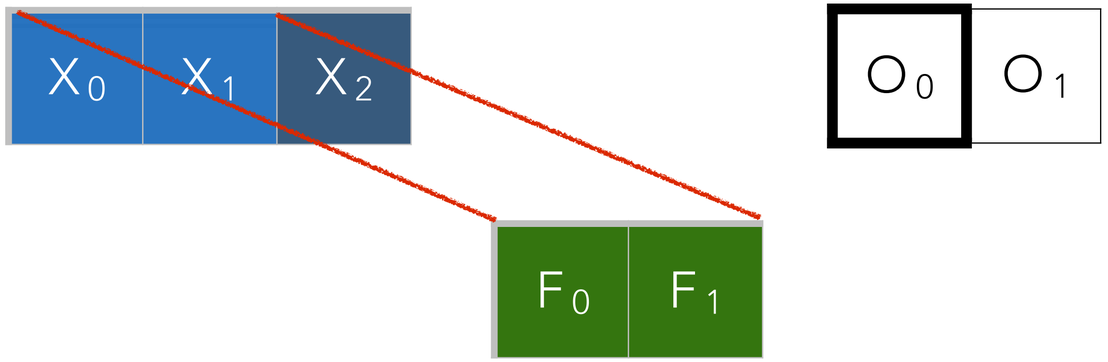


BACKPROPAGATION

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k} \frac{\partial o_k}{\partial F_i}$$

$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$



$$o_0 = F_0 x_0 + F_1 x_1$$

$$o_1 = F_0 x_1 + F_1 x_2$$

BACKPROPAGATION

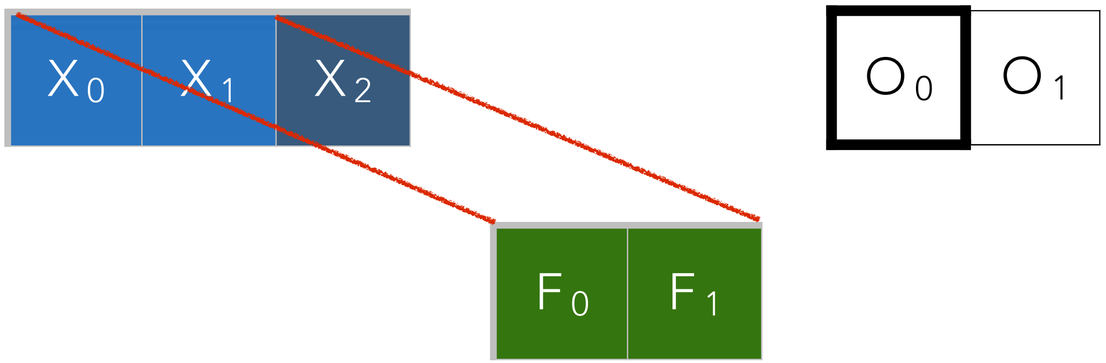
$$\frac{\partial J}{\partial F_1} = \frac{\partial J}{\partial o_0}\frac{\partial o_0}{\partial F_1} + \frac{\partial J}{\partial o_1}\frac{\partial o_1}{\partial F_1}$$

$$o_0 = F_0 x_0 + F_1 x_1$$
$$o_1 = F_0 x_1 + F_1 x_2$$

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k}\frac{\partial o_k}{\partial F_i}$$

$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$



BACKPROPAGATION

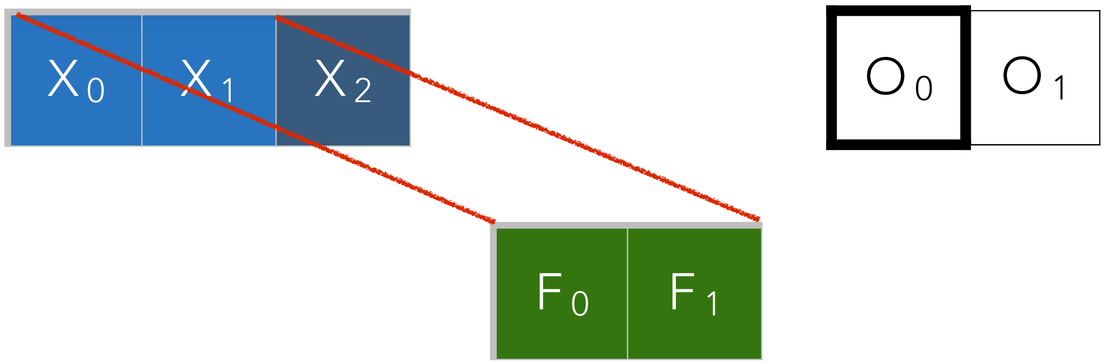
$$\frac{\partial J}{\partial F_1} = \frac{\partial J}{\partial o_0}\frac{\partial o_0}{\partial F_1} + \frac{\partial J}{\partial o_1}\frac{\partial o_1}{\partial F_1}$$

$$o_0 = F_0 x_0 + F_1 x_1$$
$$o_1 = F_0 x_1 + F_1 x_2$$

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k}\frac{\partial o_k}{\partial F_i}$$

$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$



BACKPROPAGATION

$$\frac{\partial J}{\partial F_1} = \frac{\partial J}{\partial o_0}\frac{\partial o_0}{\partial F_1} + \frac{\partial J}{\partial o_1}\frac{\partial o_1}{\partial F_1}$$

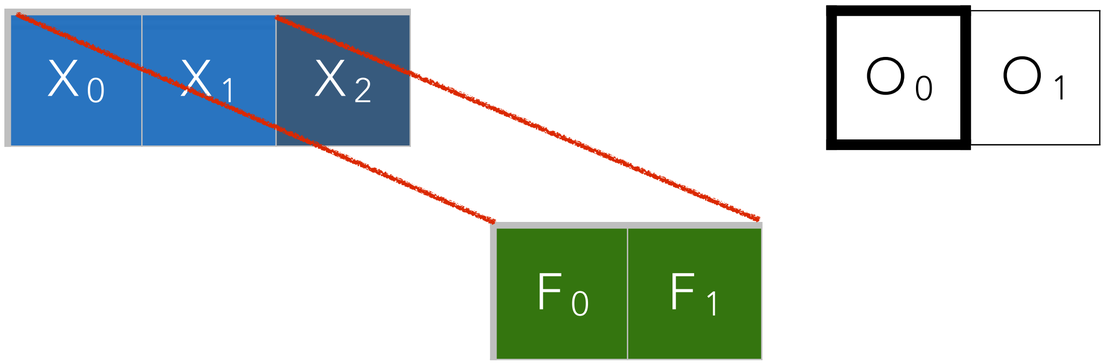
$$o_0 = F_0 x_0 + F_1 x_1$$

$$o_1 = F_0 x_1 + F_1 x_2$$

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k}\frac{\partial o_k}{\partial F_i}$$

$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$



BACKPROPAGATION

$$\frac{\partial J}{\partial F_1} = \frac{\partial J}{\partial o_0}\frac{\partial o_0}{\partial F_1} + \frac{\partial J}{\partial o_1}\frac{\partial o_1}{\partial F_1}$$

$$F_i \leftarrow F_i - \eta \frac{\partial J}{\partial F_i}$$

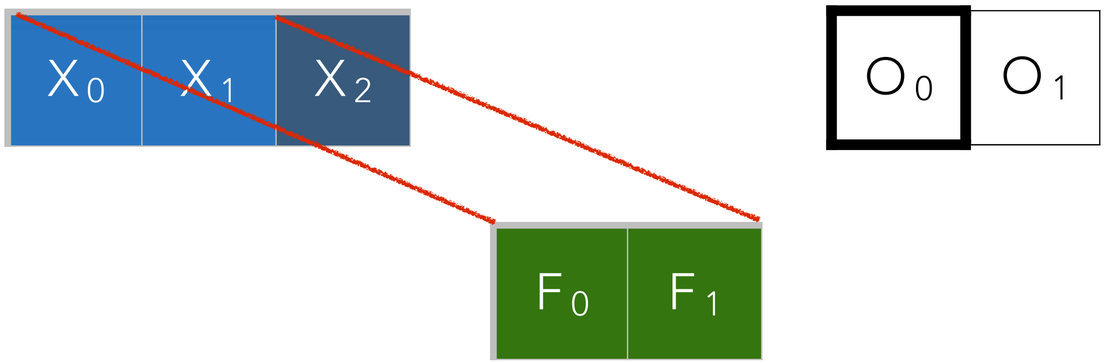
$$\frac{\partial J}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial J}{\partial o_k}\frac{\partial o_k}{\partial F_i}$$

$$o_0 = F_0 x_0 + F_1 x_1$$
$$o_1 = F_0 x_1 + F_1 x_2$$

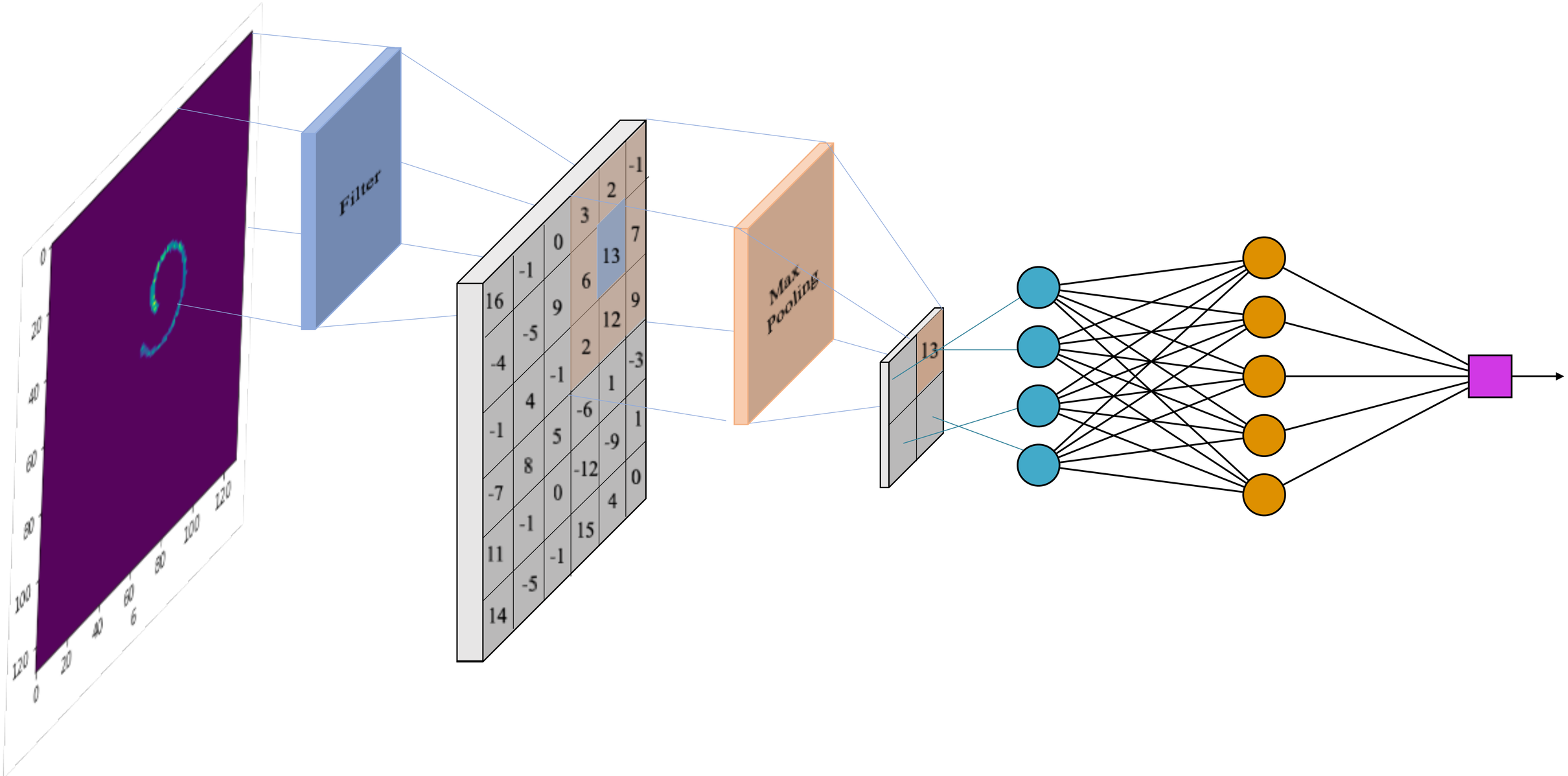
$$o[n] = (x \circledast F)[n] = \sum_{i=-\omega}^{\omega} x[i+n+\omega] * F[i+\omega]$$

$$\frac{\partial o_0}{\partial F_1} = x_1$$

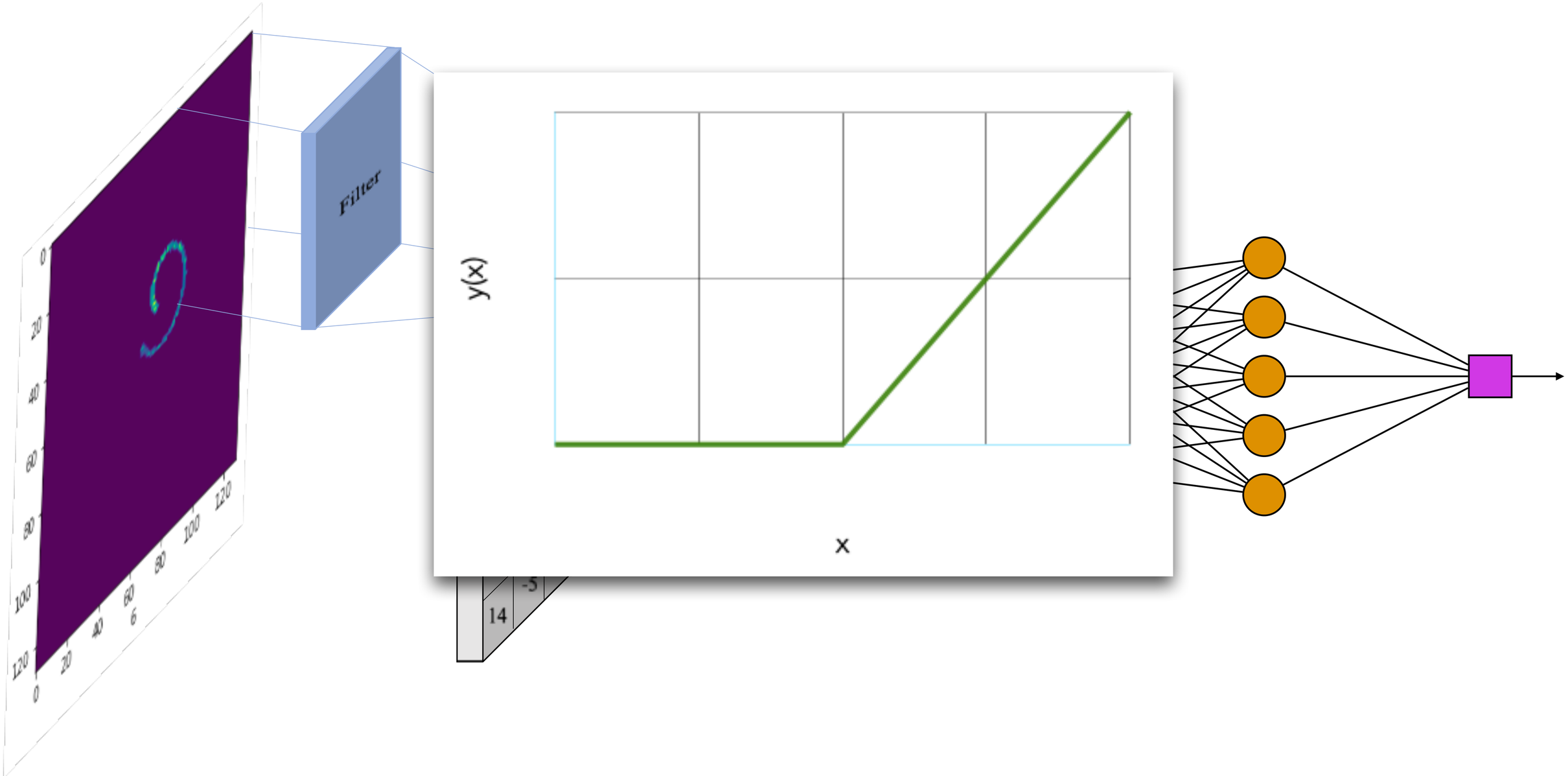
$$\frac{\partial o_1}{\partial F_1} = x_2$$



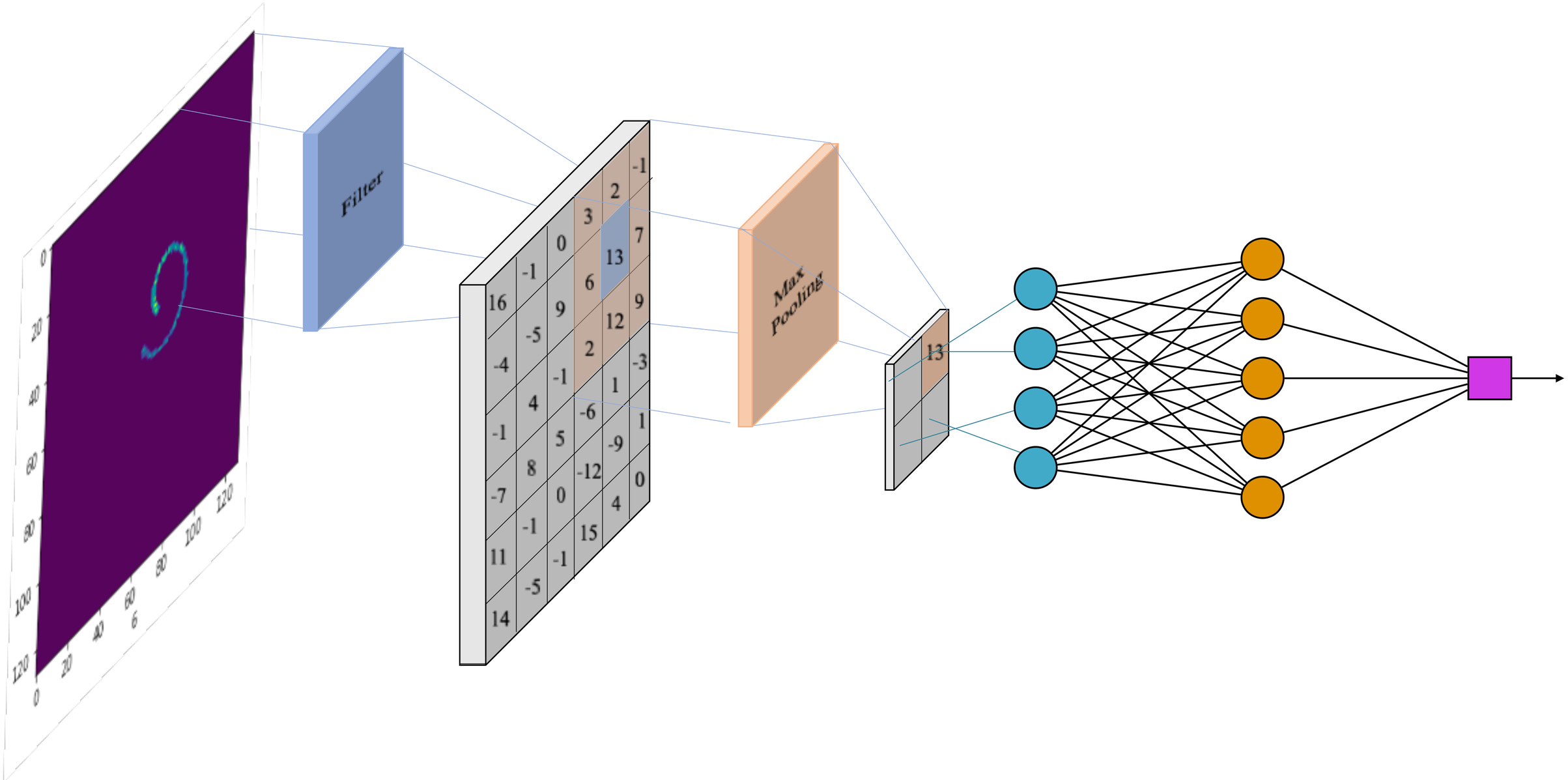
CONVOLUTIONAL NEURAL NETWORKS



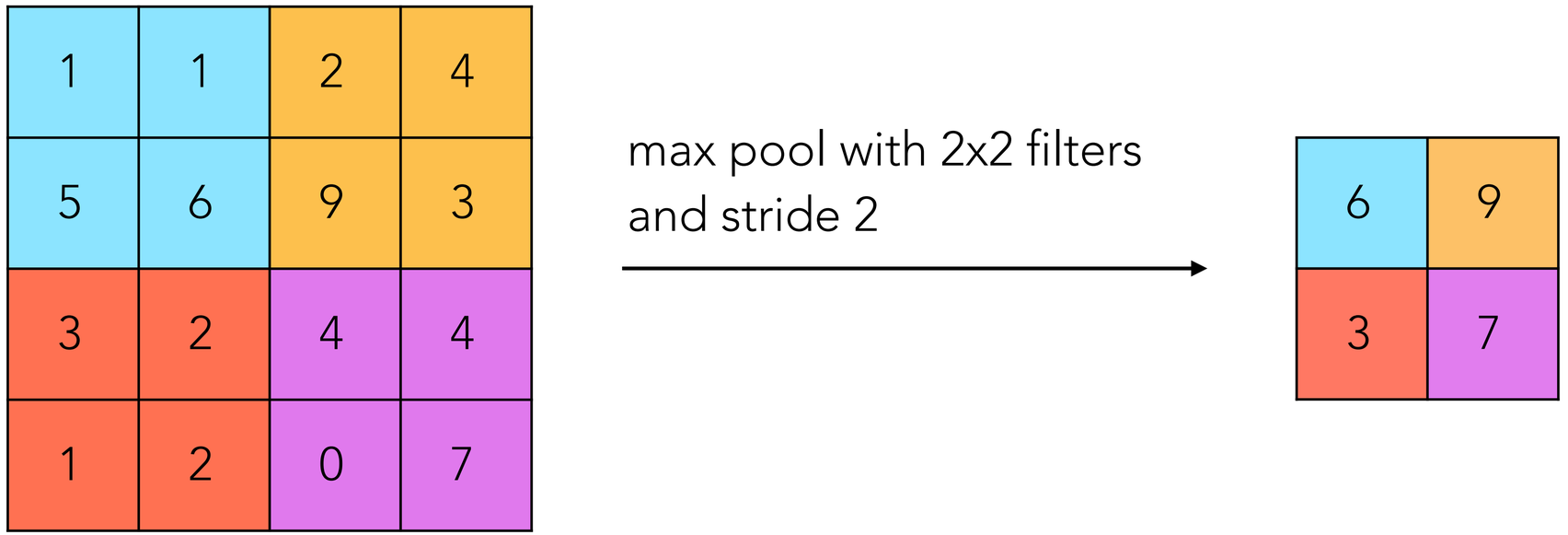
CONVOLUTIONAL NEURAL NETWORKS



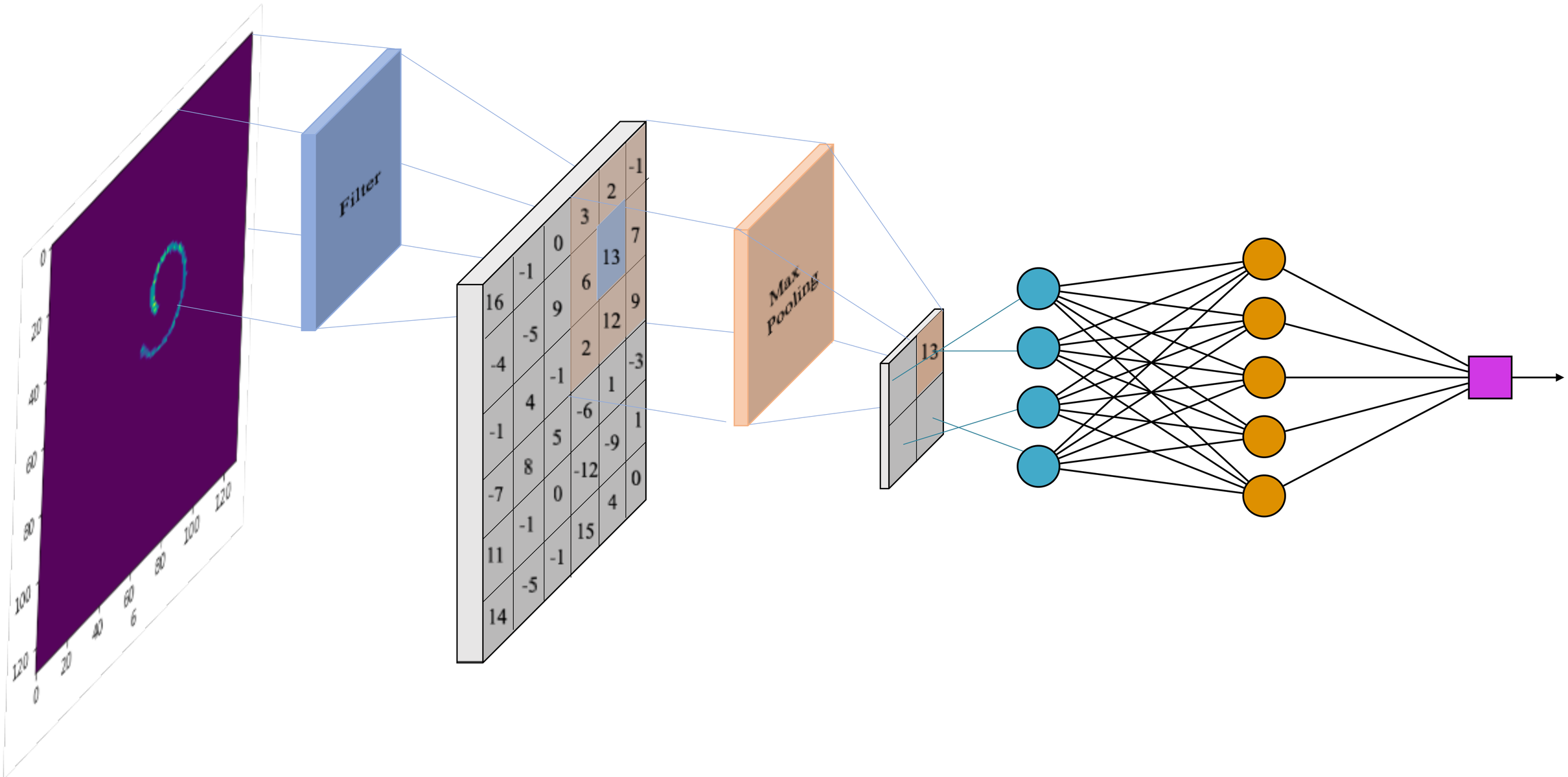
CONVOLUTIONAL NEURAL NETWORKS



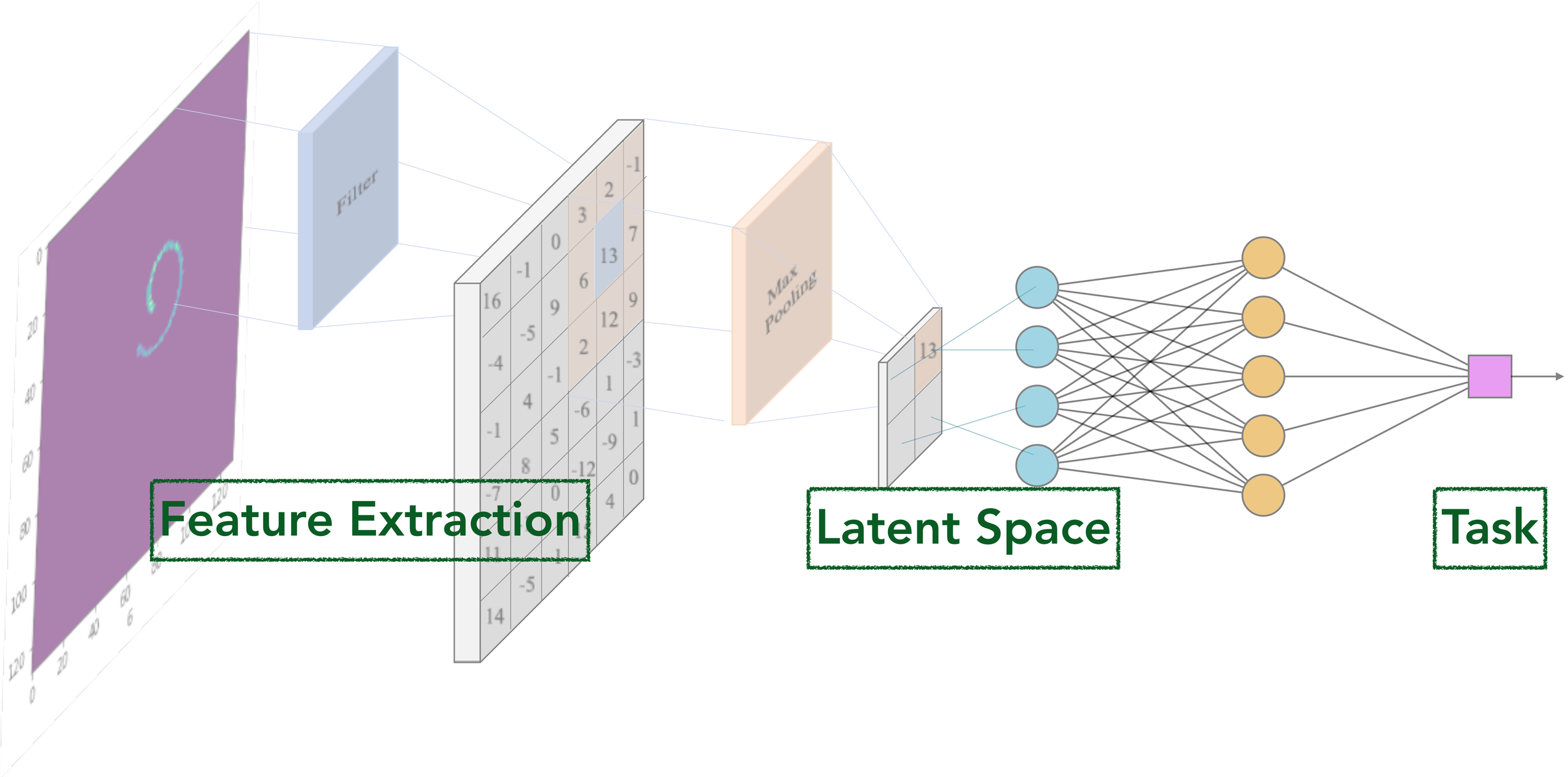
POOLING


1 1 2 4
5 6 9 3
3 2 4 4
1 2 0 7
max pool with 2x2 filters
and stride 2
6 9
3 7

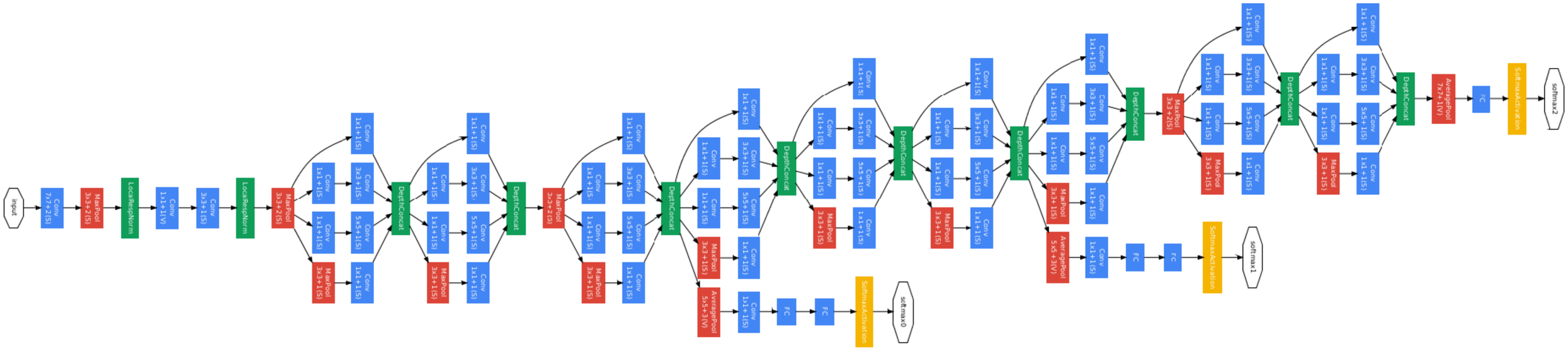
BACKPROPOGATION



CONVOLUTIONAL NEURAL NETWORKS



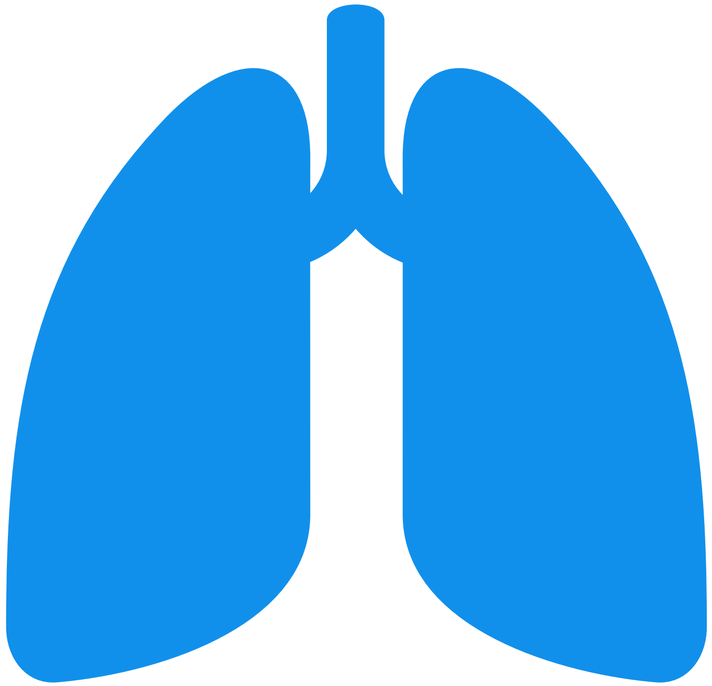
CONVOLUTIONAL NEURAL NETWORKS



CHRISTIAN SZEGEDY ET. AL. GOING DEEPER WITH CONVOLUTIONS.

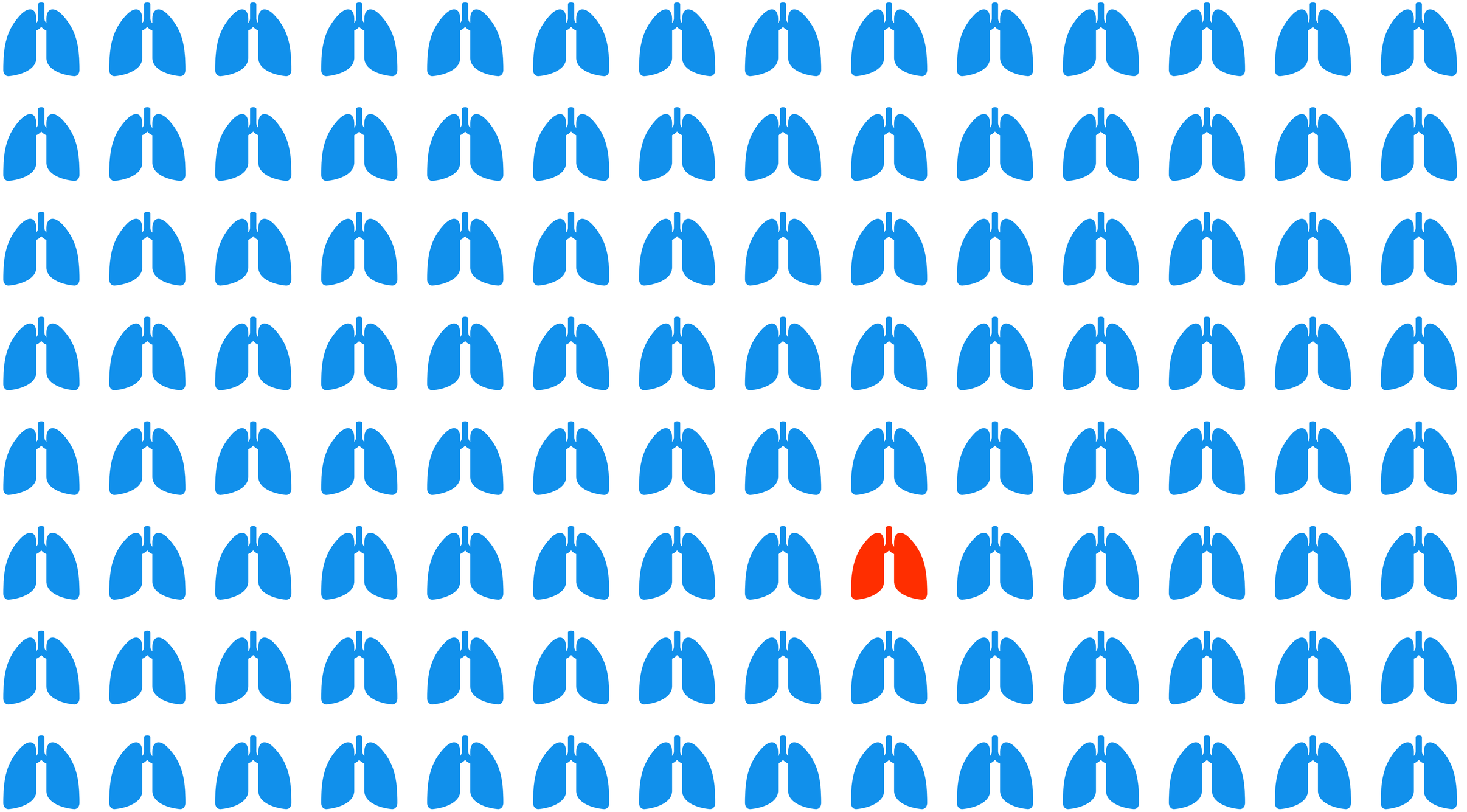
Application 2: Can we use machine learning to **accurately** classify events in detectors?

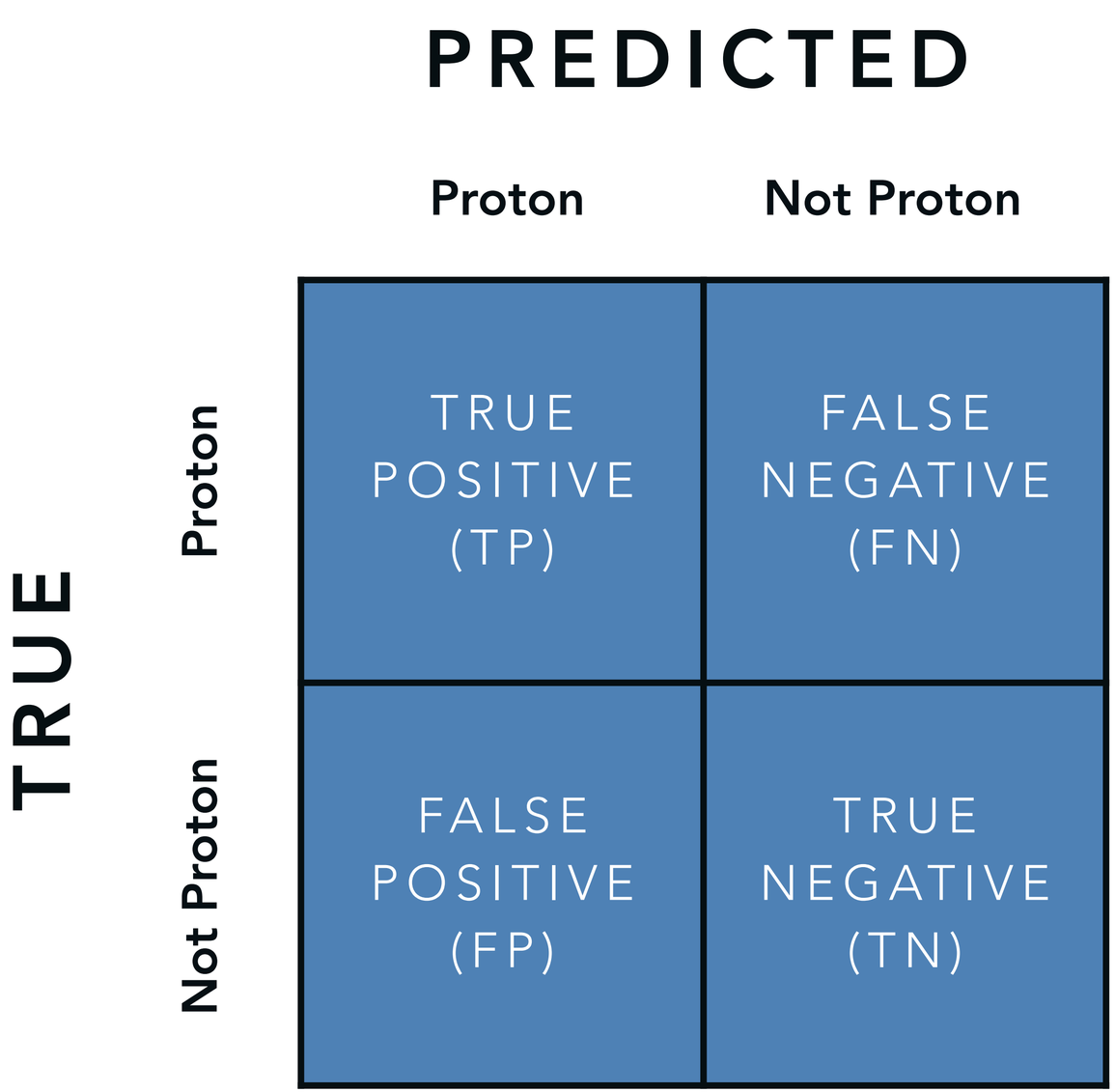
Metrics

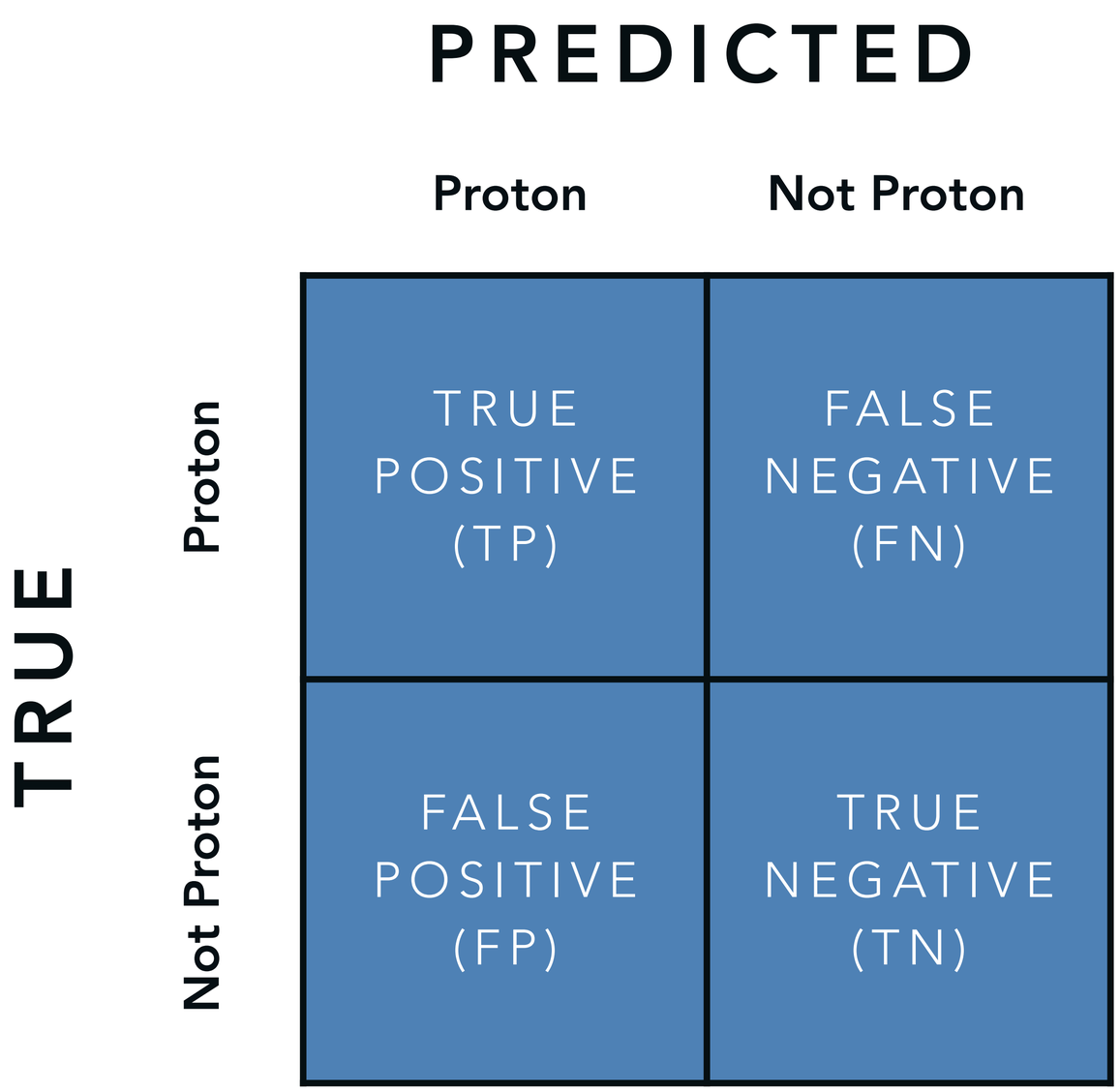


Detect Lung Cancer

99% Accuracy




PREDICTED
Proton
Not Proton
TRUE
Proton
Not Proton
TRUE POSITIVE (TP)
FALSE NEGATIVE (FN)
FALSE POSITIVE (FP)
TRUE NEGATIVE (TN)



$$\text{accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

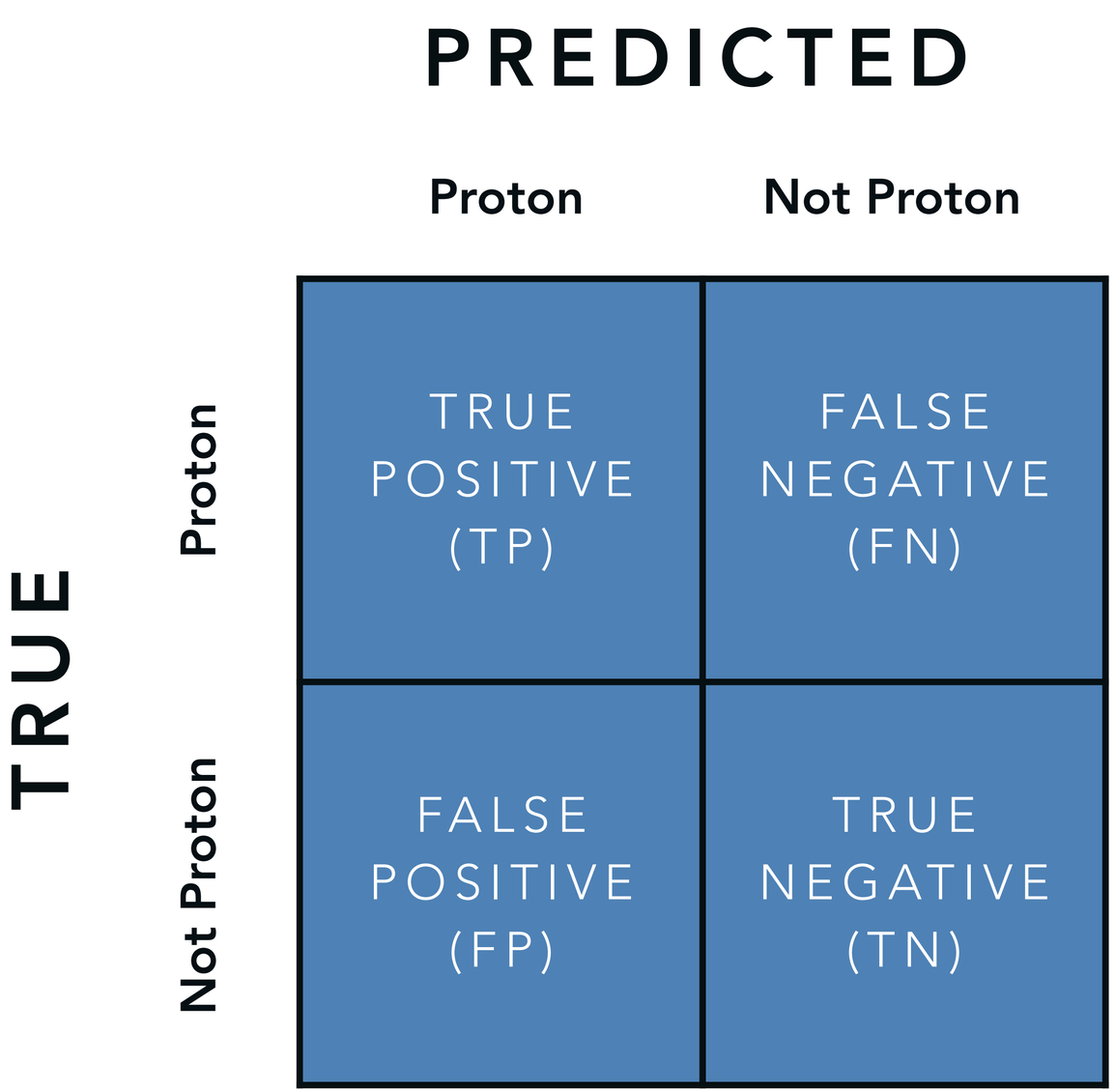
$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

PREDICTED

TRUE

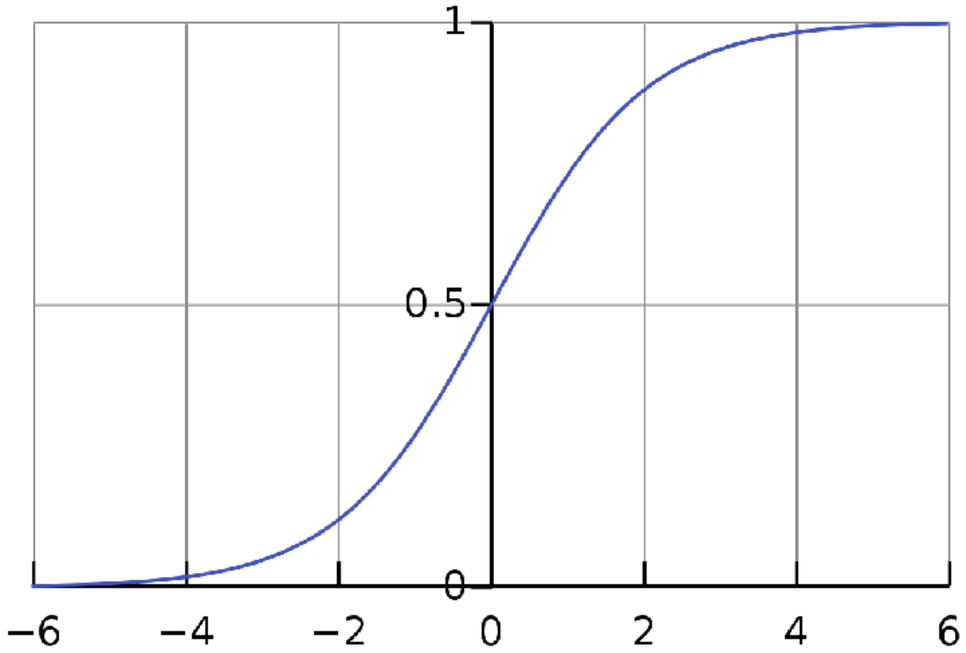
	Proton	Not Proton
Proton	TRUE POSITIVE (TP)	
Not Proton		TRUE NEGATIVE (TN)

PERFECT MODEL



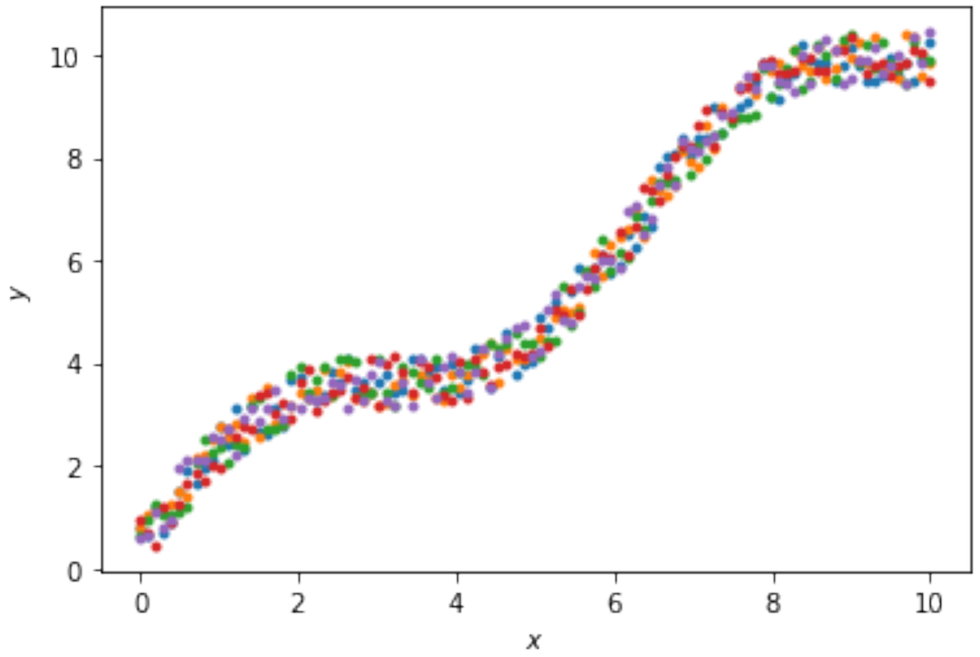
What does this matrix look like for multi-class classification?

MULTI-CLASS CLASSIFICATION



$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}$$

LOSS FUNCTIONS



Loss function

Mean squared error (MSE)

$$J(w) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2$$

Mean absolute error (MAE)

$$J(w) = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i|$$

Cross entropy (CE)

$$\mathscr{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot \log(\hat{y}_{ij})$$

CHOOSING AN ARCHITECTURE

HOW MANY LAYERS?

HOW MANY NODES PER LAYER?

LEARNING RATE

DROPOUT?

WHAT ACTIVATION FUNCTION(S)?

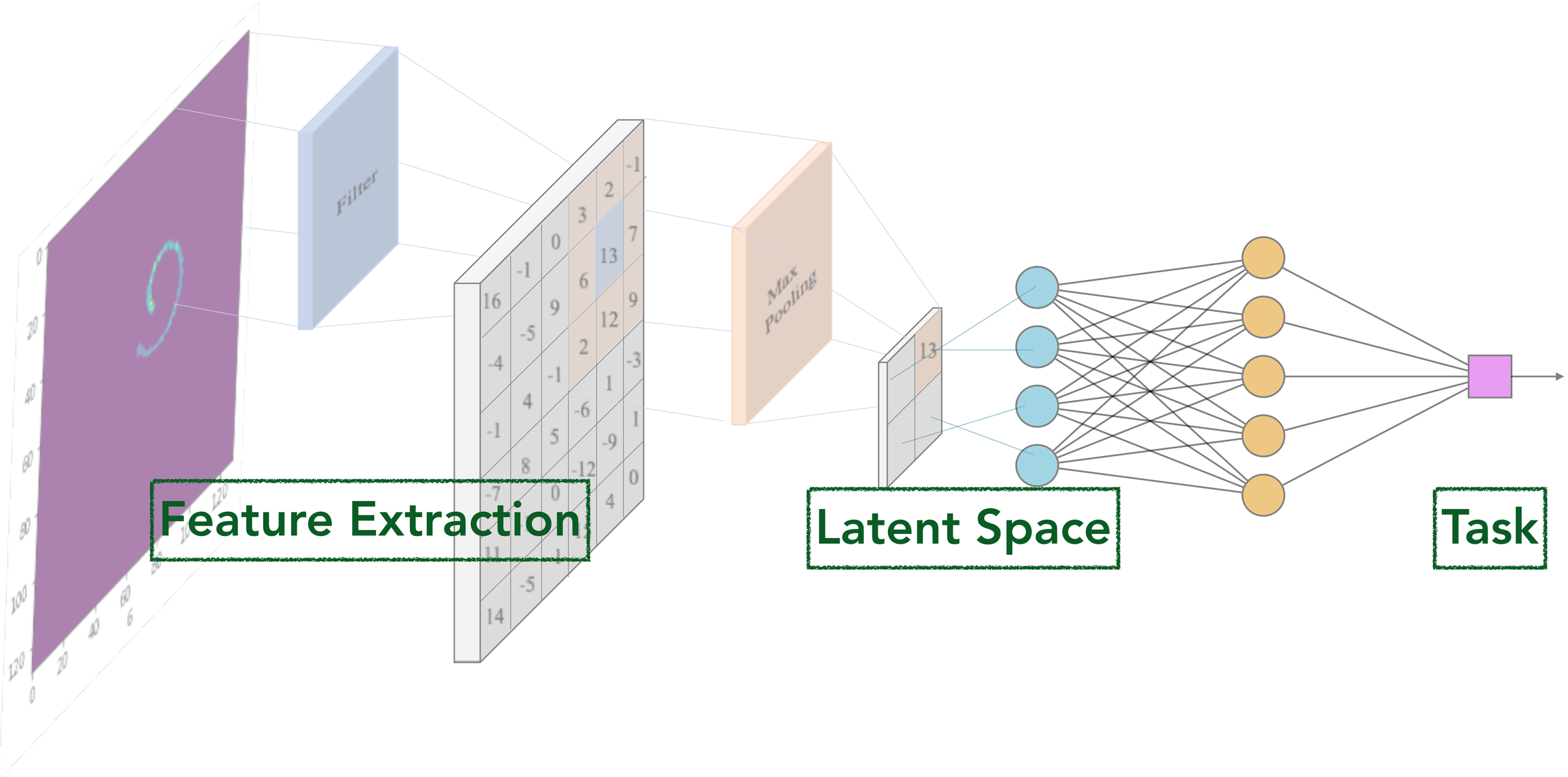
HOW MANY CONVOLUTION LAYERS?

FILTER SIZE?

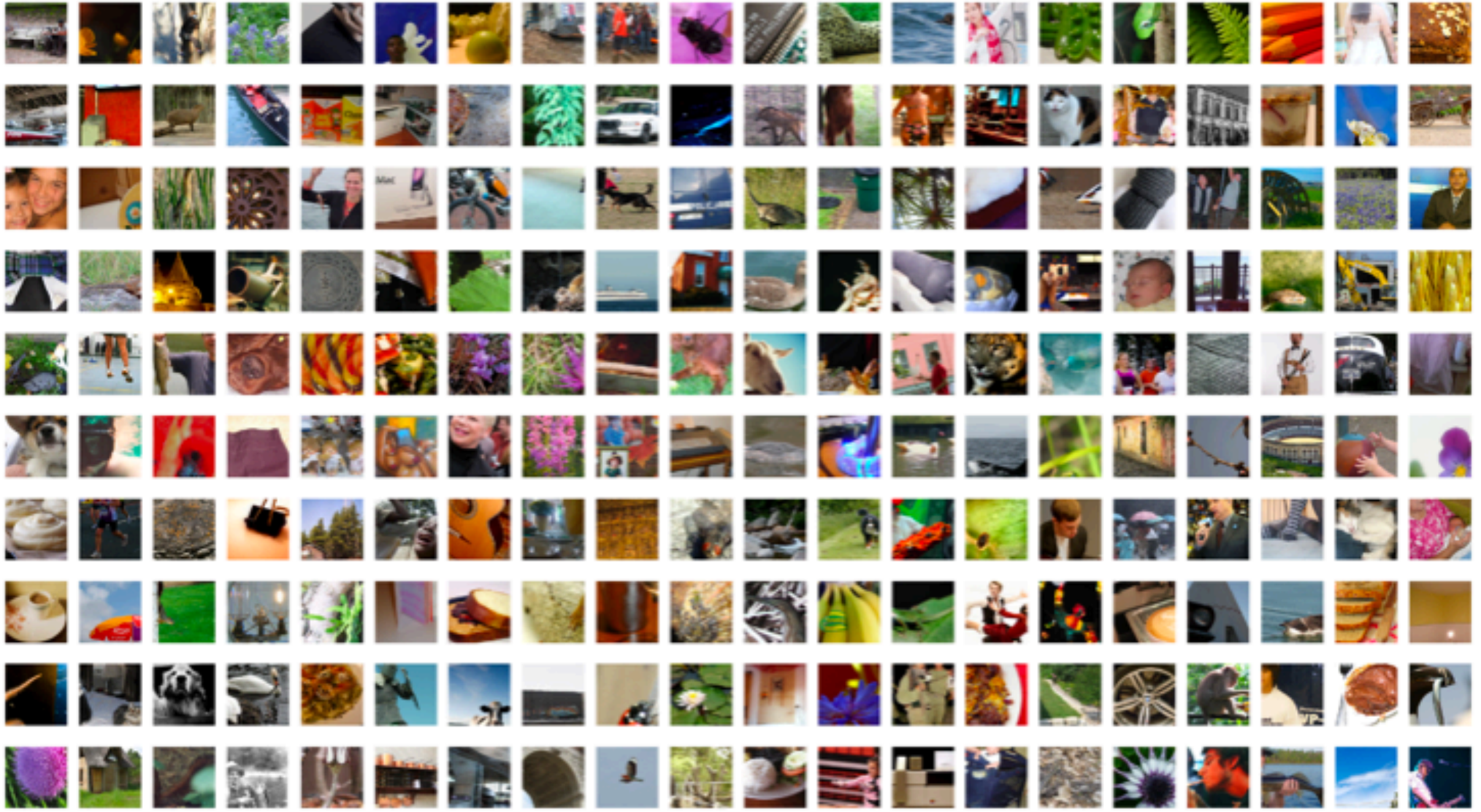
STRIDE?

POOLING?

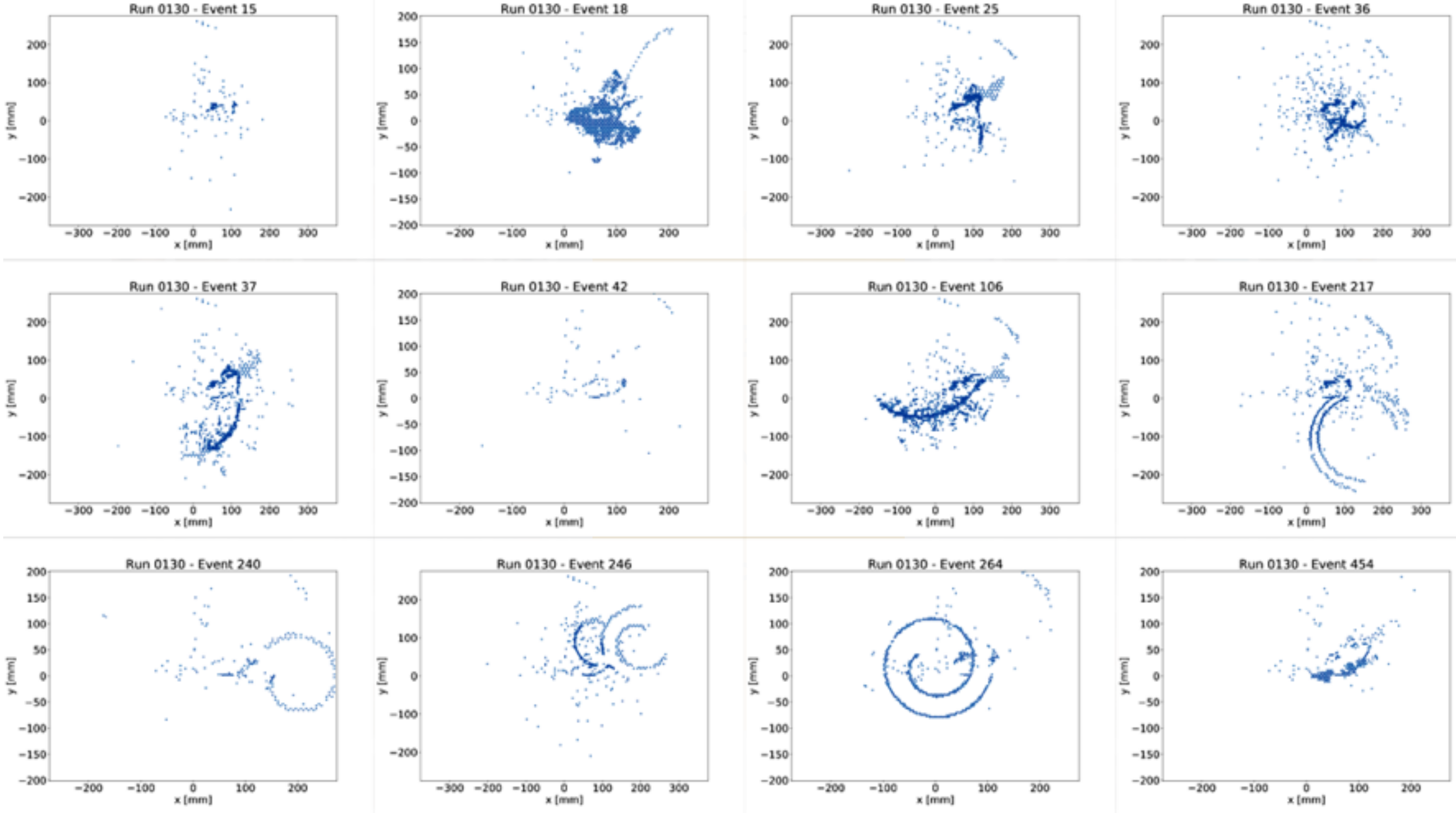
PRE-TRAINED MODELS

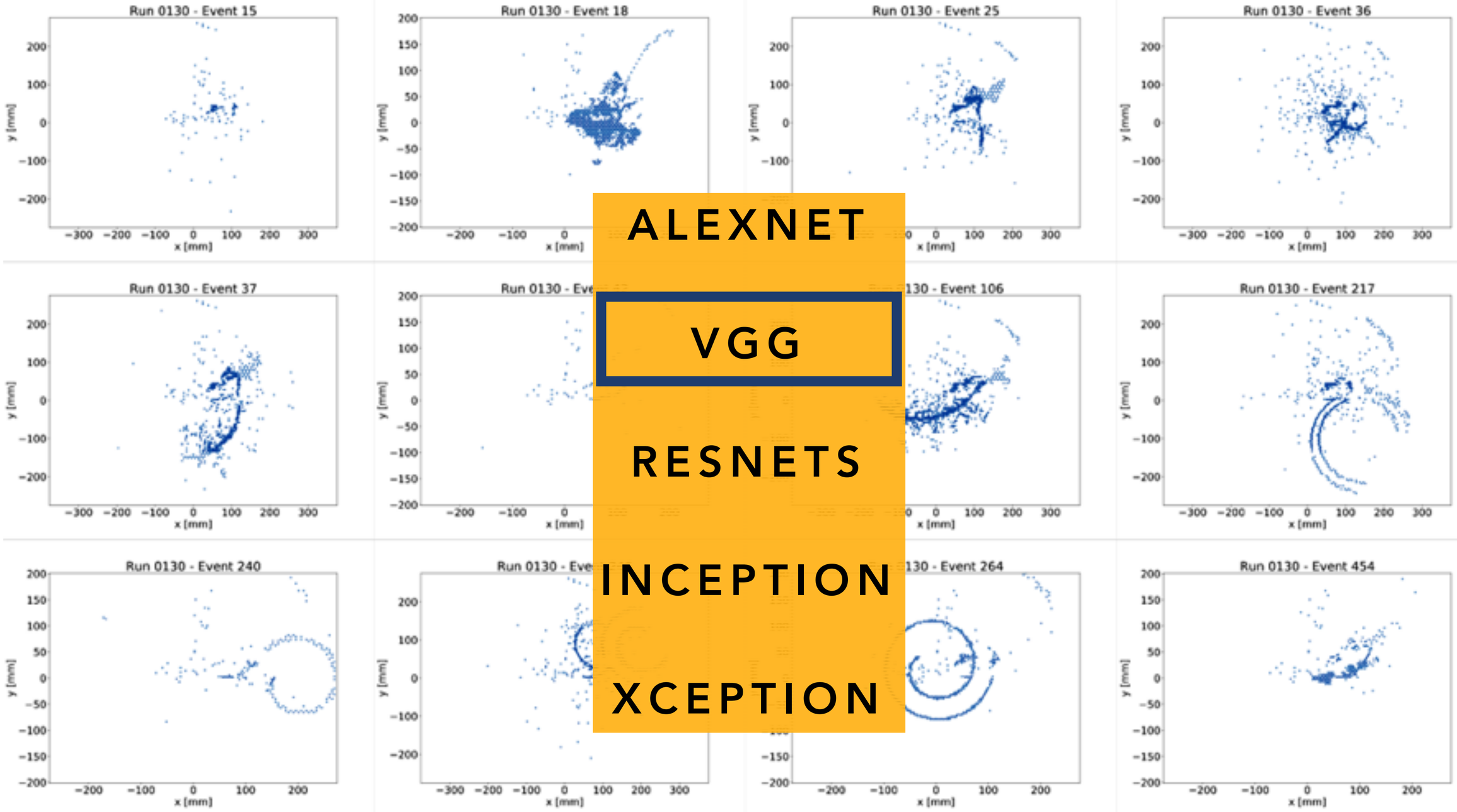


PRE-TRAINED MODELS



PRETRAINED MODELS


Run 0130 - Event 15
Run 0130 - Event 18
Run 0130 - Event 25
Run 0130 - Event 36
Run 0130 - Event 37
Run 0130 - Event 42
Run 0130 - Event 106
Run 0130 - Event 217
Run 0130 - Event 240
Run 0130 - Event 246
Run 0130 - Event 264
Run 0130 - Event 454
x [mm]
y [mm]



ALEXNET

VGG

RESNETS

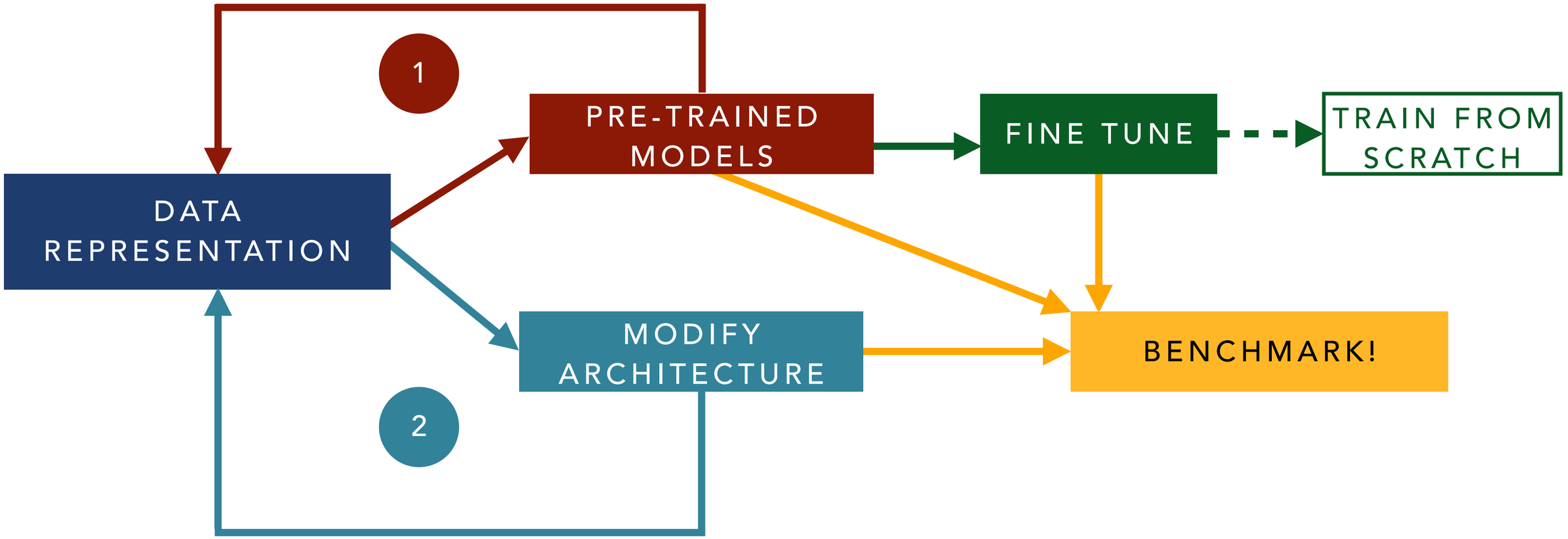
INCEPTION

XCEPTION

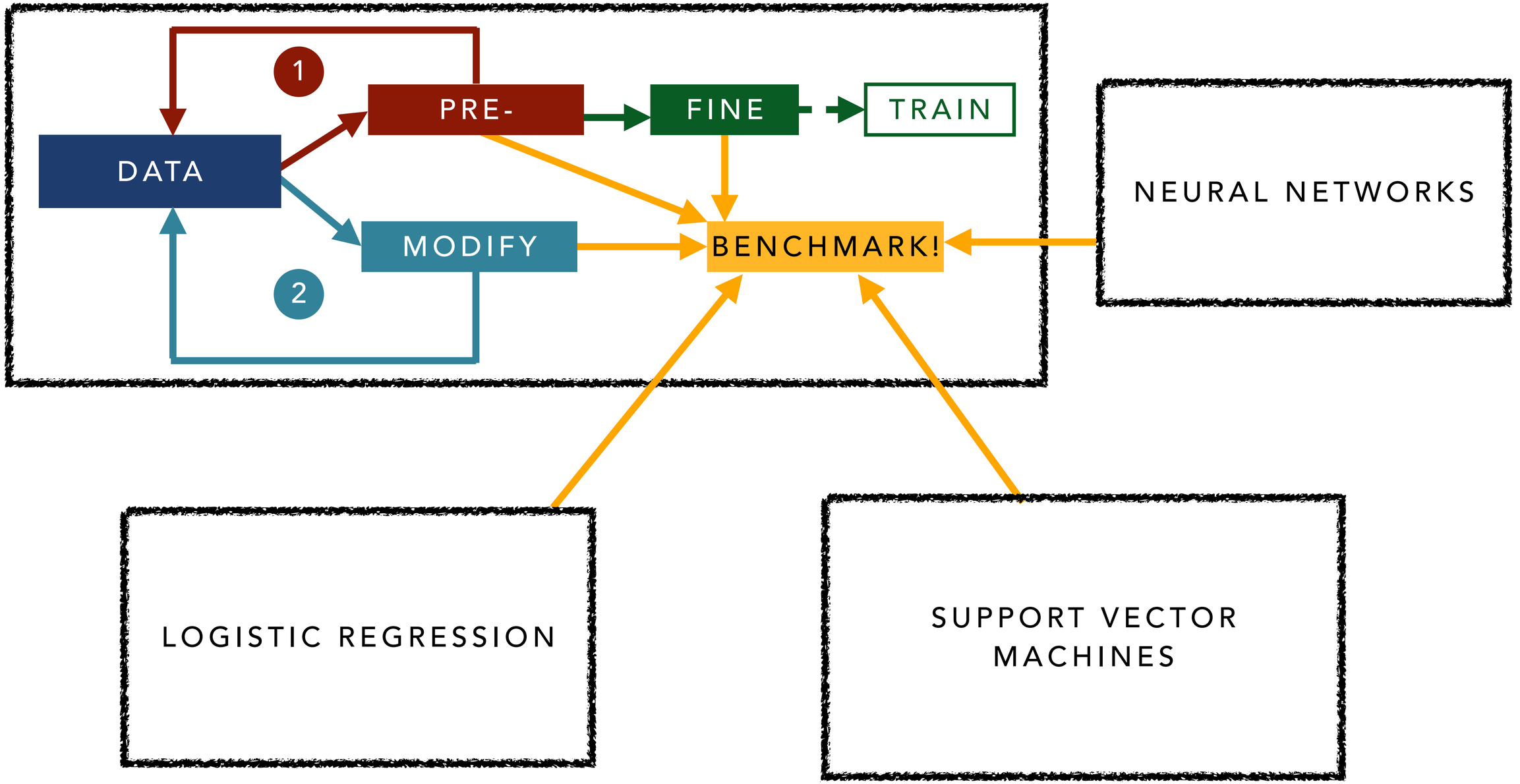
J. Z. TAYLOR, HONOR'S THESIS, DAVIDSON COLLEGE

M.P. KUCHERA, ET. AL. NUCLEAR INSTRUMENTATION AND METHODS A (2019)

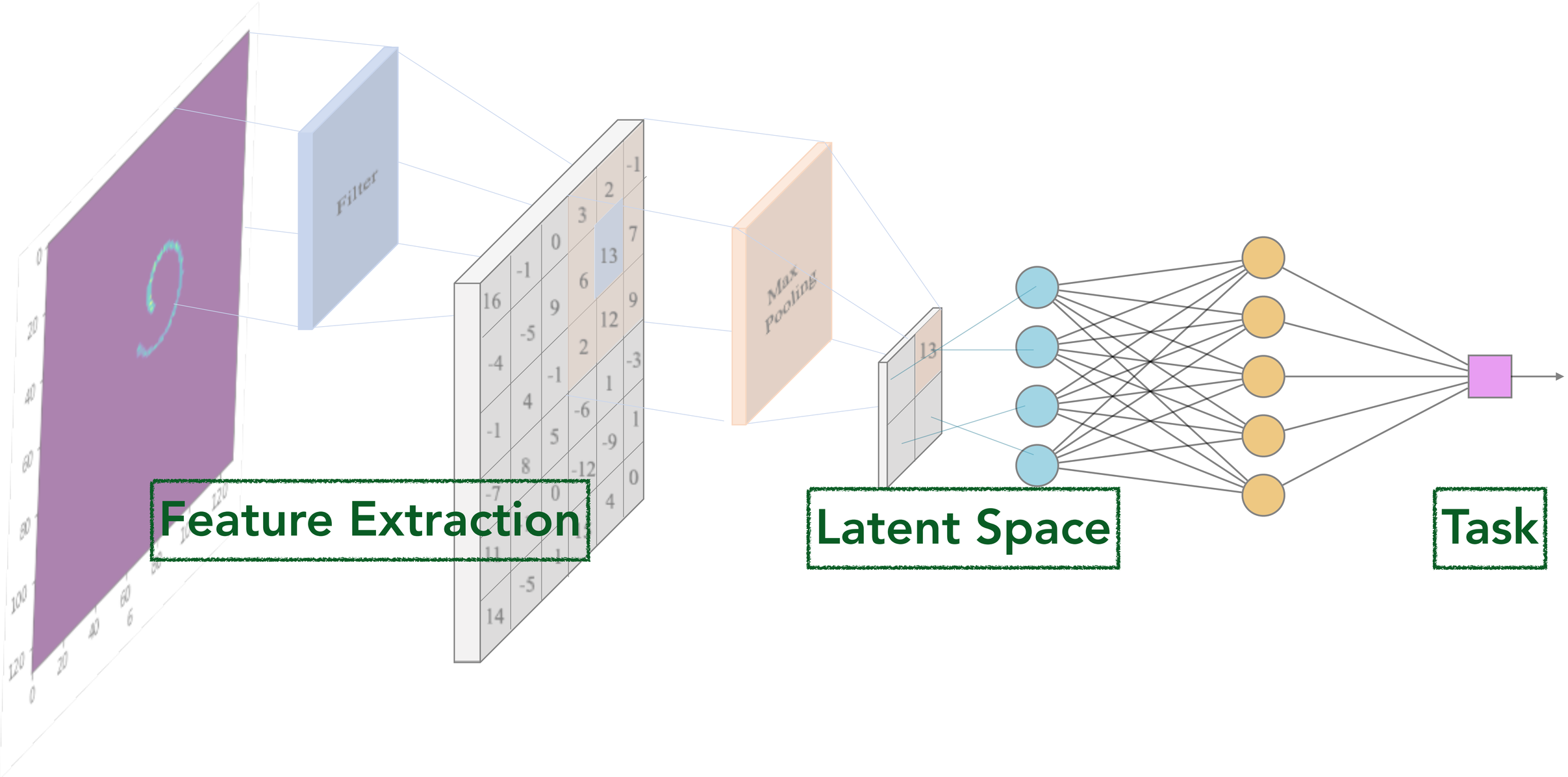
EXAMPLE WORKFLOW



EXAMPLE WORKFLOW



CONVOLUTIONAL NEURAL NETWORKS



OTHER NETWORK MECHANISMS:

-POINT CONVOLUTIONS

-ATTENTION