Convolutional Neural Networks II

Swati Mishra

Applications of Machine Learning (4AL3)

Fall 2024

McMaster University

ENGINEERING

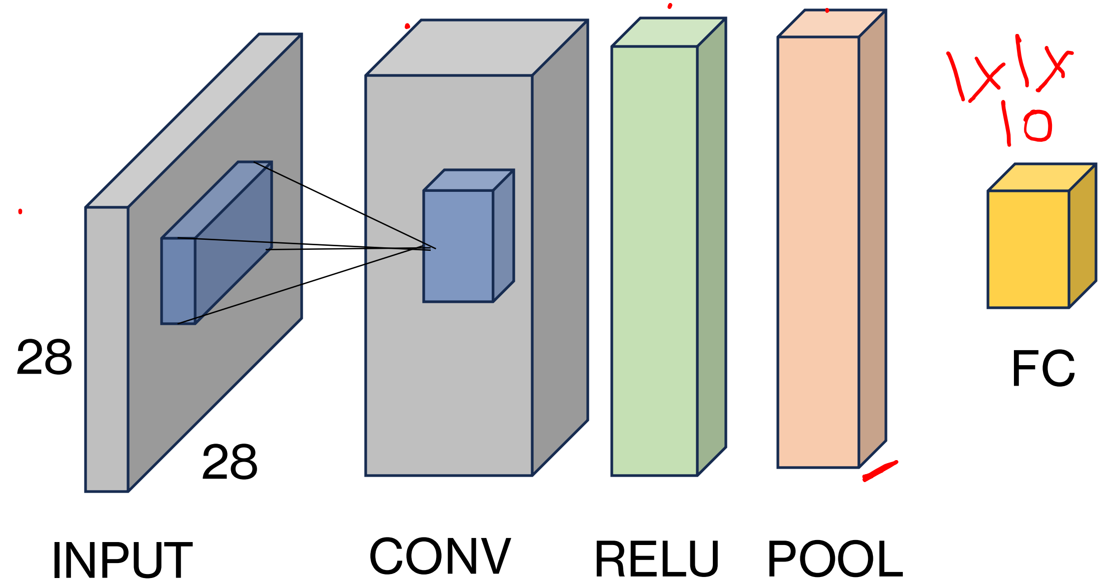
Review

- Convolution Operation
- Convolutional Neural Networks
- Fully Connected, ReLU, Pooling Layers
- CNN Architecture

Review

Typically, neural networks have 4 main types of layers:

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer
- ReLU Layer – element wise



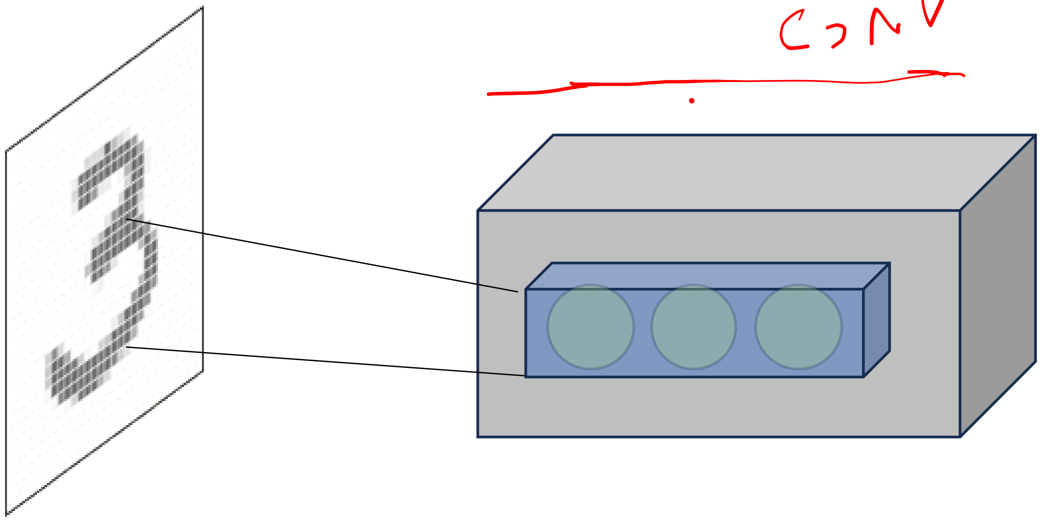
Layer stacking order: CONV – RELU –POOL-FC

INPUT - CONV – RELU –POOL- CONV – RELU –POOL- ------- CONV – RELU –POOL-FC

Convolutional Neural Networks

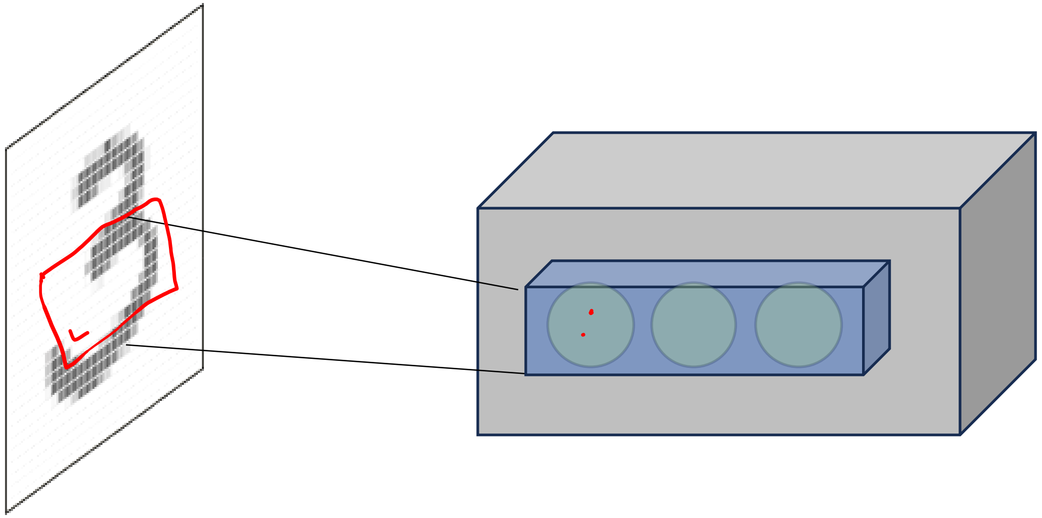
- Convolutional layer architecture:
    - The layer comprises of filters of small width and height than the full image.
    - The number of filters extends along the depth.

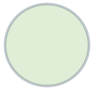
CONV



Convolutional Neural Networks: Intuition

- Convolutional layer architecture :
  - The layer comprises of filters of small width and height than the full image.
  - The number of filters extends along the depth.
  - The filters are stacked behind each other, look at different features of the same **region.**





-1	1	-1
1	0	0
-1	-1	0

This is 1 computational unit! (or 1 neuron)

Convolutional Neural Networks: Intuition

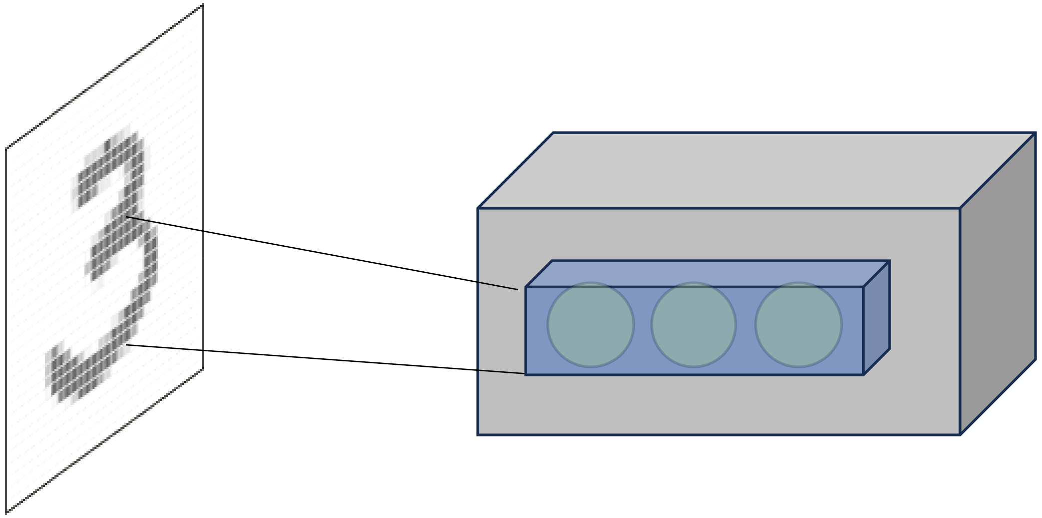
- Convolutional layer architecture :
  - The layer comprises of filters of small width and height than the full image.
  - The number of filters extends along the depth.
  - The filters are stacked behind each other, look at different features of the same **region.**
  - For forward pass, convolve the filter with the input volume (compute dot product)

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} b - a & & \\ & & \end{bmatrix}$$

$$A = aw + bx + ey + fz$$

Convolutional Neural Networks: Hyperparameters

- **Depth (D):** Number of filters used in the convolutional layer .



Convolutional Neural Networks: Hyperparameters

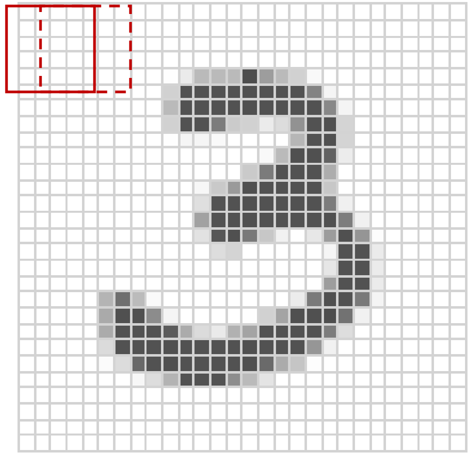
- **Depth (D):** Number of filters used in the convolutional layer .
- **Stride (S):** The number of pixels by which we slide the filter for convolution.

a	b	c	d
e	f	g	h
i	j	k	l

*

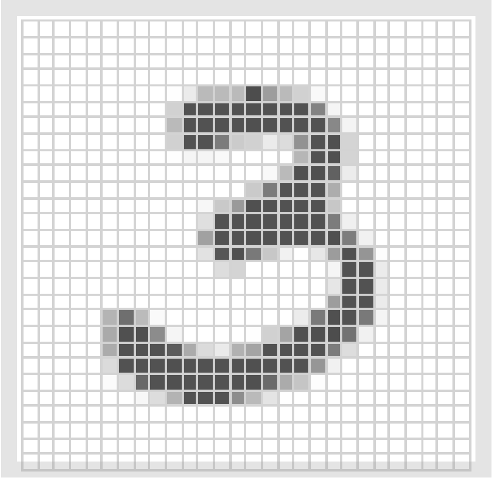
-1	1
0	0

Stride =1



Convolutional Neural Networks: Hyperparameters

- **Depth (D):** Number of filters used in the convolutional layer .
- **Stride (S):** The number of pixels by which we slide the filter for convolution.
- **Zero Padding (P)**: Size of the zero-padding added on all 4 sides, popularly used to keep the input and output width and height are same.

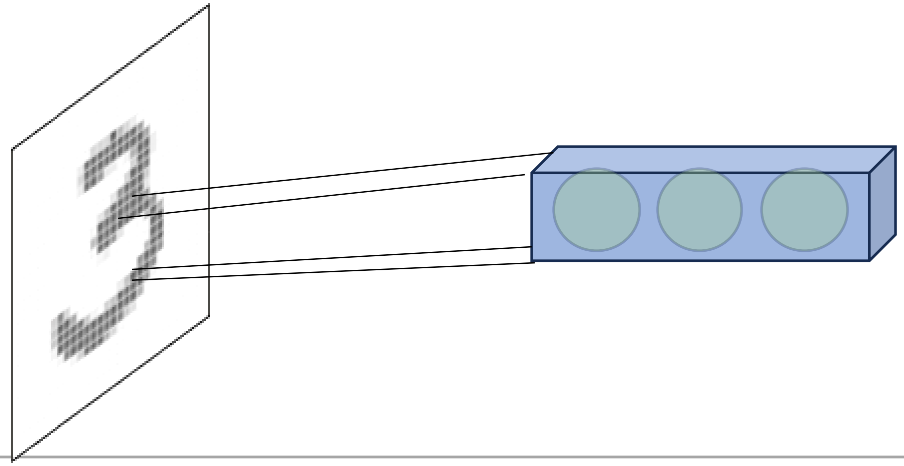


0
0
0
0
0
0

Convolutional Neural Networks: Hyperparameters

- **Depth (D):** Number of filters used in the convolutional layer .
- **Stride (S):** The number of pixels by which we slide the filter for convolution.
- **Zero Padding (P):** Size of the zero-padding added on all 4 sides, popularly used to keep the input and output width and height are same.
- **Filter Size(F):** The spatial extent of each filter looking at a specific region (also called Receptive Field)

$\frac{128}{12.8}$



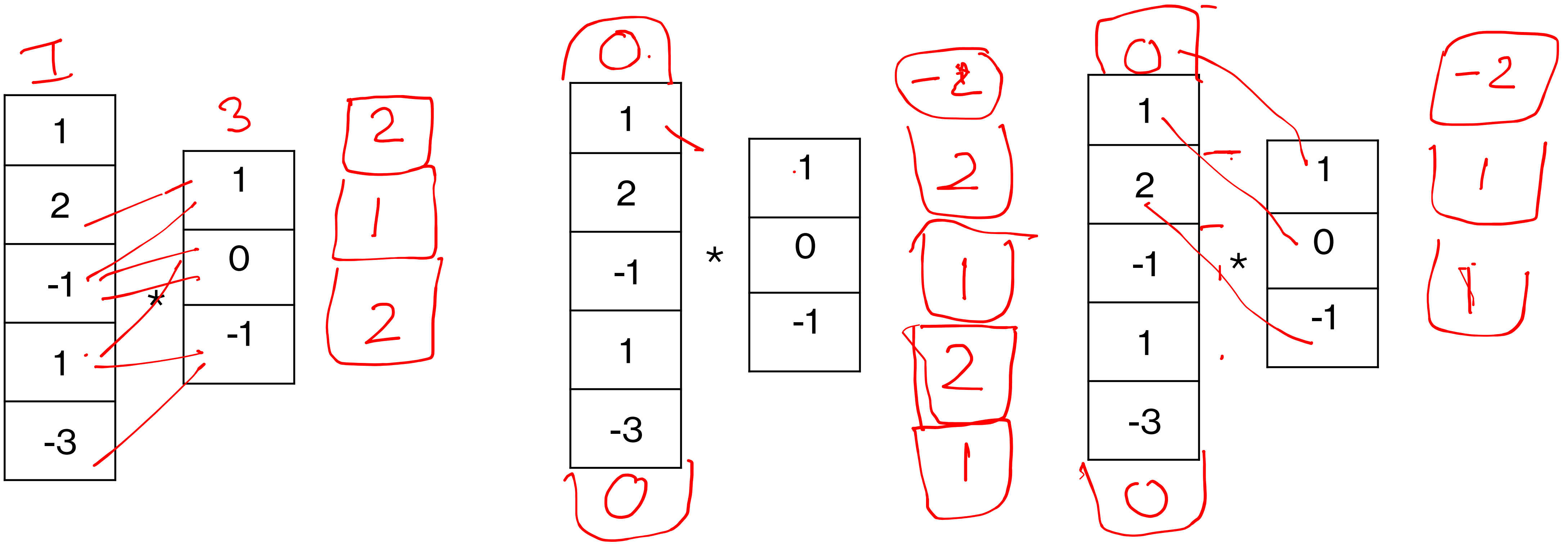
Convolutional Neural Networks: Hyperparameters

- **Depth (D)**: Number of filters used in the convolutional layer .
- **Stride (S)**: The number of pixels by which we slide the filter for convolution.
- **Zero Padding (P)**: Size of the zero-padding added on all 4 sides, popularly used to keep the input and output width and height are same.
- **Filter Size(F)**: The spatial extent of each filter looking at a specific region.

Number of computational units that fit into a layer = (I – F + 2P)/S+1 where I is the input volume

I 28 28 5

Convolutional Neural Networks: Hyperparameters



I = 5, F = 3, S = 1, P = 0

I = 5, F = 3, S = 1, P = 1,

I = 5, F = 3, S = 2, P = 1,

Convolutional Neural Networks: Hyperparameters

- **Depth (D)**: Number of filters used in the convolutional layer .
- **Stride (S)**: The number of pixels by which we slide the filter for convolution.
- **Zero Padding (P)**: Size of the zero-padding added on all 4 sides, popularly used to keep the input and output width and height are same.
- **Filter Size(F)**: The spatial extent of each filter looking at a specific region.

Number of computational units that fit into a layer = (I – F + 2P)/S+1 where I is the input volume

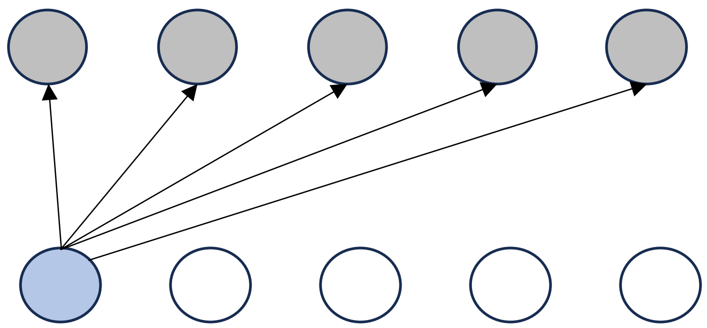
4.5

Some configurations are impossible! For I=10, P=0,F=3, number of computational units not feasible.

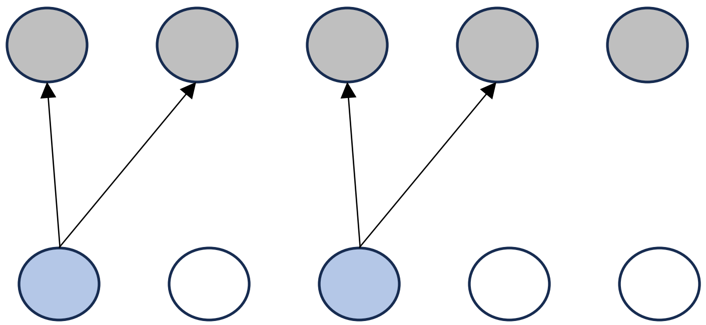
Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Sparse Interactions:
    - This means every output unit does not interact with every input unit.
    - Focus on small, meaningful features such as edges with filters smaller than full image.

NN



CNN



Convolutional Neural Networks

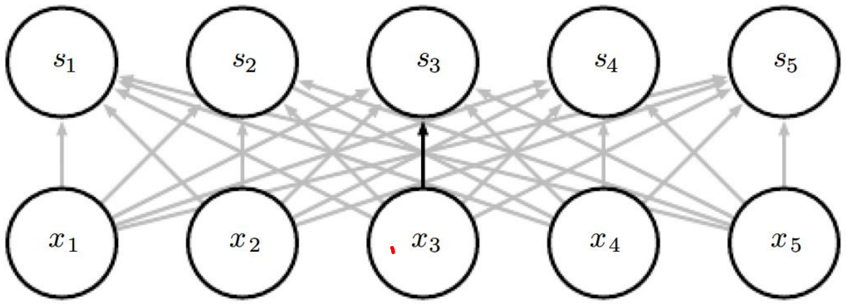
- Three important properties of Convolutional Neural Network.:

$$s(t) = (x * w)(t)$$

  - Sparse Interactions:
    - This means every output unit does not interact with every input unit.
    - Focus on small, meaningful features such as edges with filters smaller than full image.
    - Reduce memory requirements.
    - Require fewer computations.

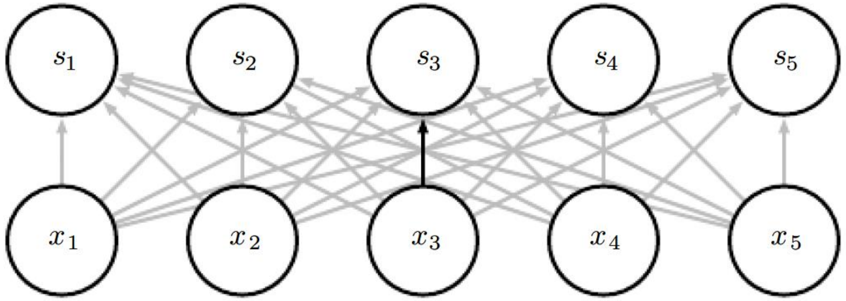
Convolutional Neural Networks

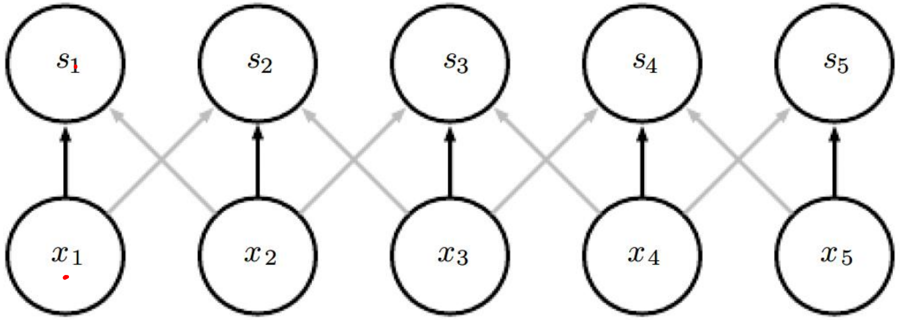
- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.



Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.

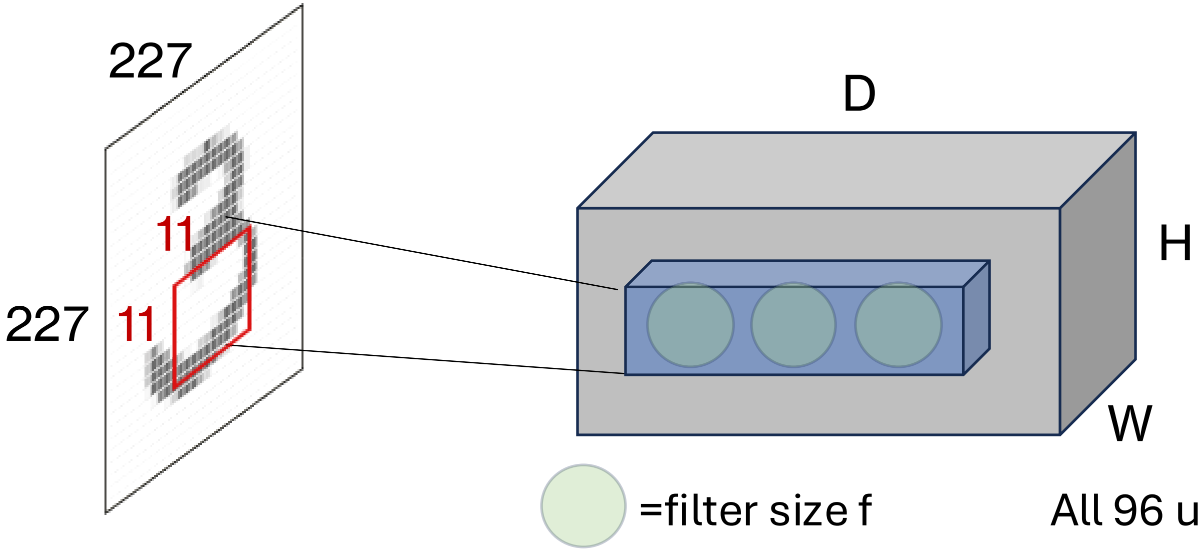




In other words, network has tied weights!

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.



F=11
S=4
P=0
D=96

Output of this layer =

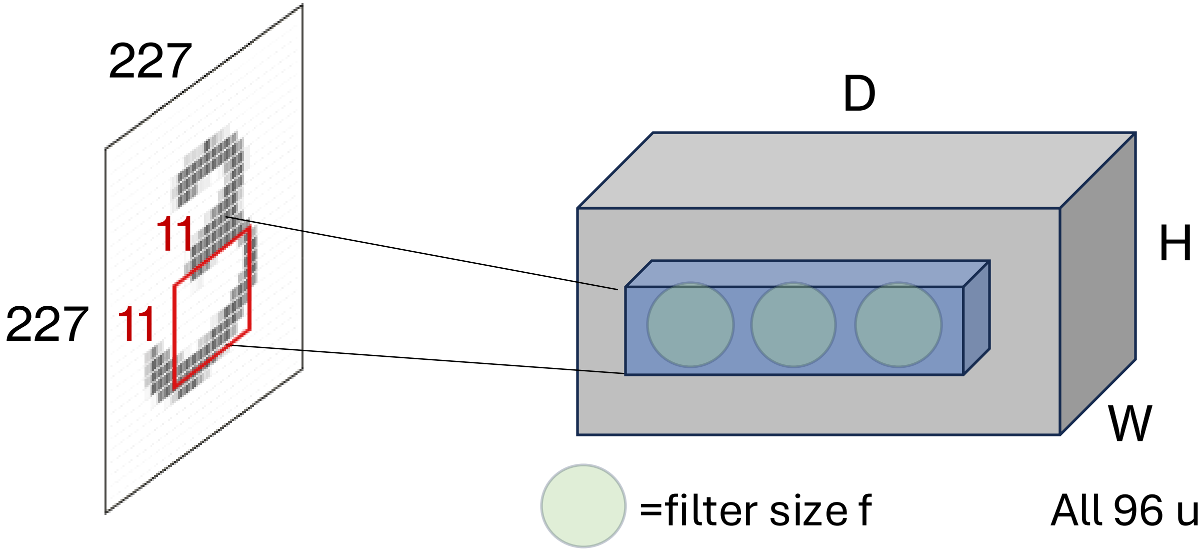
(I – F + 2P)/S+1=

55x55x96

All 96 units are connected to a region of 11x11x3

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.



F=11
S=4
P=0
D=96

Output of this layer =

(I – F + 2P)/S+1=

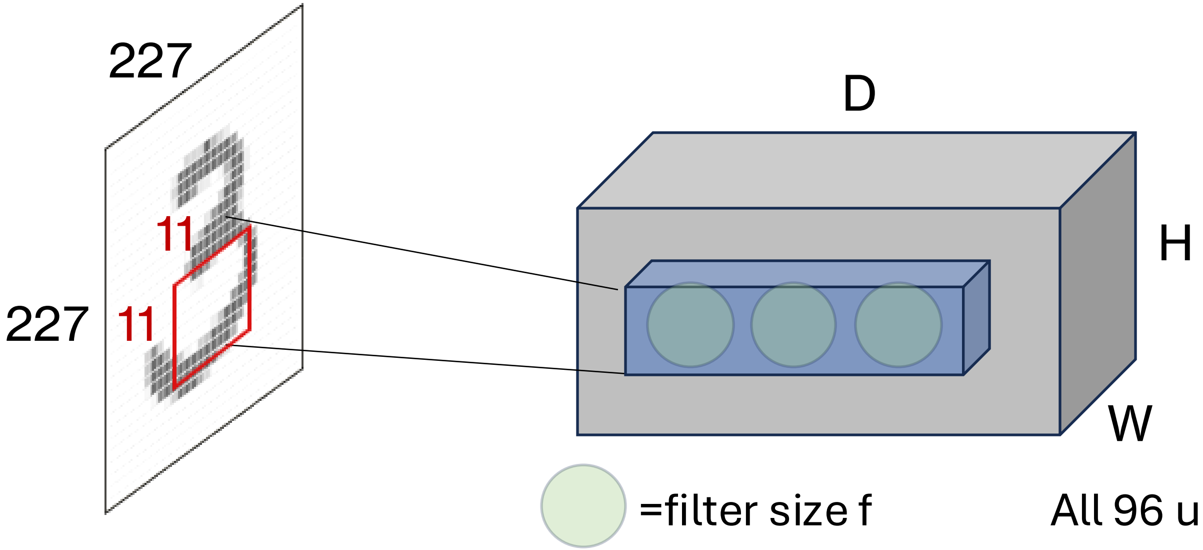
55x55x96

Each computational unit has 11x11x3 weights and 1 bias

All 96 units are connected to a region of 11x11x3

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.



F=11
S=4
P=0
D=96

Output of this layer =

(I – F + 2P)/S+1=

55x55x96

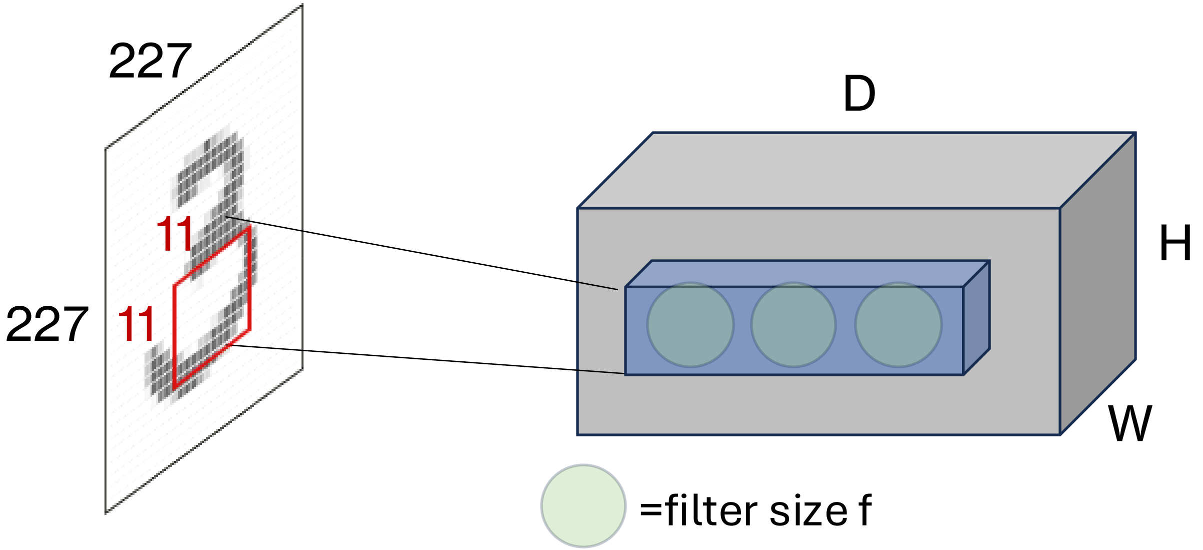
Total parameters in this layer?

All 96 units are connected to a region of 11x11x3

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.:
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.

$$s(t) = (x * w)(t)$$

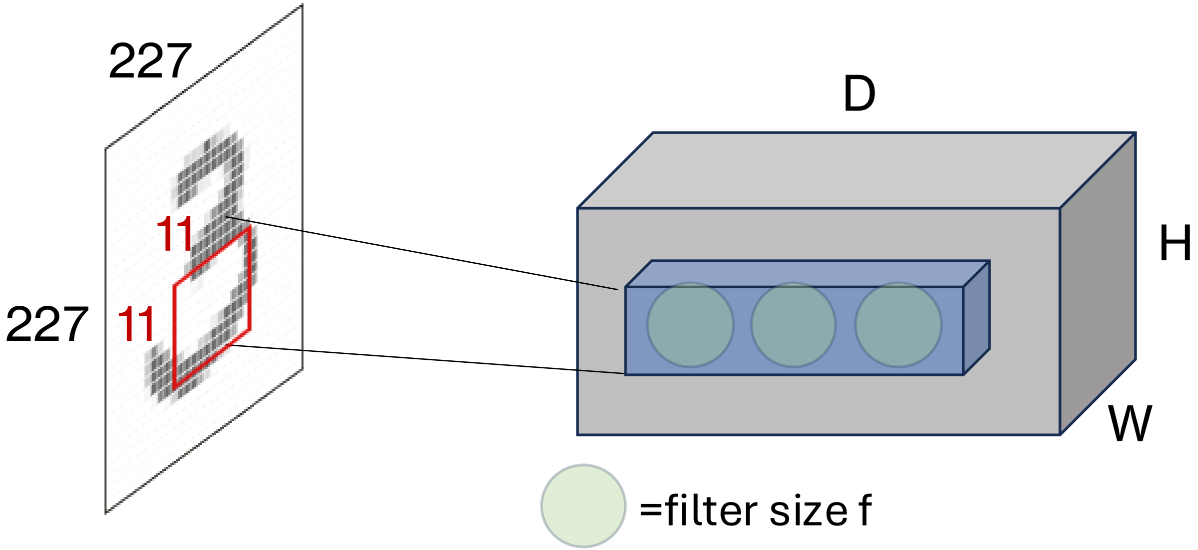


We assume:
If one feature is useful to compute at some spatial position, then it should also be useful to compute at a different but related position as well.

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.:
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.

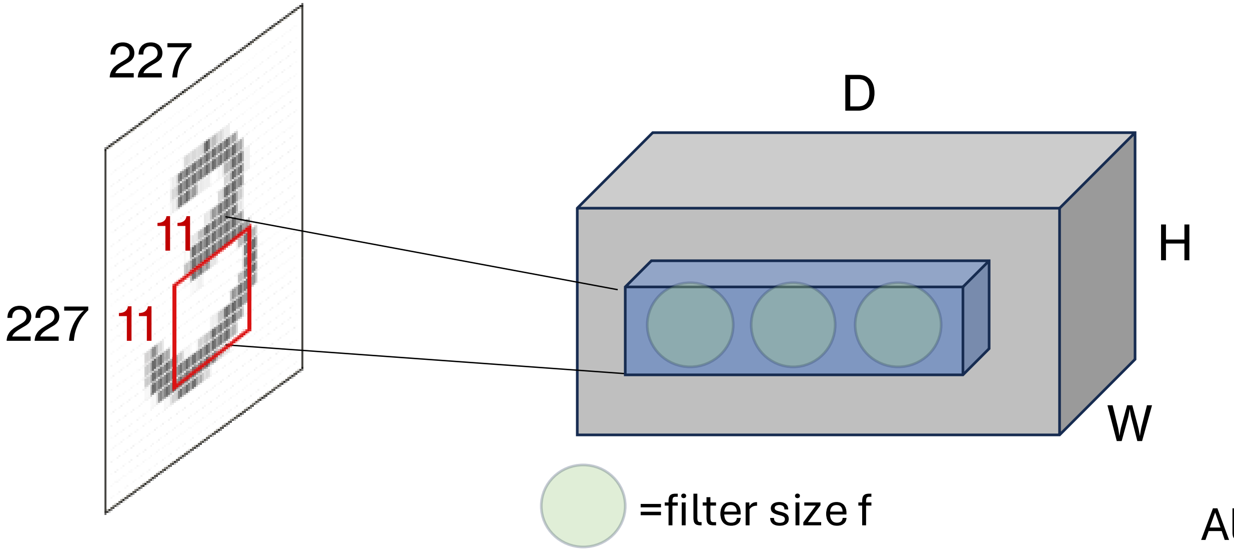
$$s(t) = (x * w)(t)$$



We constrain the computational units in each depth slice to use the same weights and bias.

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.



F=11, S=4, P=0, D=96

Output of this layer = 55x55x96

Without parameter sharing: 55x55x96x(11x11x3 +1)

With parameter sharing: 96 x (11x11x3 +1)

All 96 units are connected to a region of 11x11x3

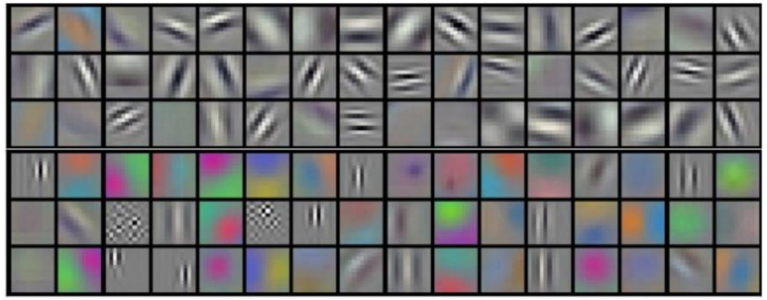
Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.:
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.
    - In practice during backpropagation, every unit computes the gradient for its weights,
    - However, these gradients will be added up across each depth slice
    - The gradients only update a single set of weights per slice.
    - Only because of this technique, can we apply convolution during forward pass.

$$s(t) = (x * w)(t)$$

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
  - Parameter Sharing:
    - In traditional NN, each element of the weight matrix is used exactly once, multiplied by 1 element and never visited.
    - In CNNs, we can use the same parameter for more than one function in a model.
    - In practice during backpropagation, every unit computes the gradient for its weights,
    - However, these gradients will be added up across each depth slice
    - The gradients only update a single set of weights per slice.
    - Only because of this technique, can we apply convolution during forward pass.



Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.: $s(t) = (x * w)(t)$
    - Equivariant Representations:
        - The parameter sharing causes the property of equivariance to translation.
        - Equivariant means that if the input changes, the output changes in the same way.
        - If g translates the input (or shifts it), then convolution function is equivariant to g

$$f(g(x)) = g(f(x))$$

Convolutional Neural Networks

- Three important properties of Convolutional Neural Network.:
  - Equivariant Representations:
    - The parameter sharing causes the property of equivariance to translation.
    - Equivariant means that if the input changes, the output changes in the same way.
    - If g translates the input (or shifts it), then convolution function is equivariant to g

$$s(t) = (x * w)(t)$$

$$f(g(x)) = g(f(x))$$

$Let's\ assume\ an\ image\ I\ (x, y)$

- $shifitng\ tranformation\ \ I'(x, y) = I(x-1, y)$

Transformation to I + convolution

=

Convolution to I’ + Transformation

Convolutional Neural Networks: Example

0	0	0	0	0	0	0
0	1	1	0	0	2	0
0	1	1	0	1	0	0
0	1	1	1	2	1	0
0	1	0	2	0	2	0
0	0	0	2	1	0	0
0	0	0	0	0	0	0

```
x[0,:,:]
```

0	0	0	0	0	0	0
0	0	0	2	0	1	0
0	1	1	0	1	1	0
0	2	0	2	2	0	0
0	1	1	1	2	0	0
0	0	1	2	2	0	0
0	0	0	0	0	0	0

```
x[1,:,:]
```
✓ 0.0s

0	0	0	0	0	0	0
0	0	0	2	0	1	0
0	1	1	0	1	1	0
0	2	0	2	2	0	0
0	1	1	1	2	0	0
0	0	1	2	2	0	0
0	0	0	0	0	0	0

```
x[2,:,:]
```
✓ 0.0s

```
x = torch.rand(3,5,5)
```

Convolutional Neural Networks: Example

0	0	0	0	0	0	0
0	1	1	0	0	2	0
0	1	1	0	1	0	0
0	1	1	1	2	1	0
0	1	0	2	0	2	0
0	0	0	2	1	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	0	2	0	1	0
0	1	1	0	1	1	0
0	2	0	2	2	0	0
0	1	1	1	2	0	0
0	0	1	2	2	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	0	2	0	1	0
0	1	1	0	1	1	0
0	2	0	2	2	0	0
0	1	1	1	2	0	0
0	0	1	2	2	0	0
0	0	0	0	0	0	0

Filter F0

1	1	1
1	0	-1
-1	0	1

```
f[0,:,:]
```
✓ 0.0s

-1	0	-1
0	1	0
0	0	0

```
f[1,:,:]
```
0.0s

1	1	-1
0	1	0
-1	0	0

```
f[2,:,:]
```
✓ 0.0s

0	7	3
4	4	6
8	8	5

Output

Readings

Required Readings:

Introduction to Statistical Learning

- Chapter 10 – Section 10.3 page 406 - 412

***Supplemental Readings (Not required but recommended)*:**

Deep Learning

- Chapter 9 – page 330 - 340

McMaster University

Thank You

McMaster University