

Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 12, 2024



1 Motivation

2 CNN vs. FCN

3 Convolution

4 Backpropagation

5 References

1 Motivation

2 CNN vs. FCN

3 Convolution

4 Backpropagation

5 References

How Do Humans See?

How can we enable computers to see?



Figure adapted from [4]

What Computers See: Images As Numbers

An image is just a matrix of numbers .i.e., $1080 \times 1080 \times 3$ for an RGB image.

Question: is this Lincoln? Washington? Jefferson? Obama?

How can the computer answer this question?

	157	153	174	168	160	152	129	151	172	161	165	166
155	182	163	74	75	62	33	17	110	210	180	154	
180	180	50	14	94	6	10	33	48	106	159	181	
206	109	5	124	131	111	120	204	166	15	56	180	
194	68	137	251	237	239	239	228	227	87	71	201	
172	105	207	233	233	214	220	239	228	98	74	206	
188	88	179	209	185	215	211	158	139	75	20	169	
189	97	165	84	10	168	134	11	31	62	22	148	
199	168	191	193	158	227	178	143	182	106	36	190	
205	174	155	252	236	231	149	178	228	43	95	234	
190	216	116	149	236	187	85	150	79	38	218	241	
190	224	147	108	227	210	127	102	36	101	255	224	
190	214	173	66	103	143	96	80	2	109	249	215	
187	196	235	75	1	81	47	0	6	217	255	211	
183	202	237	145	0	0	12	108	200	138	243	236	
195	206	123	297	377	121	123	200	175	13	96	218	

Learn Image Processing & Computer Vision

Figure adapted from [4]

What Is Computer Vision?

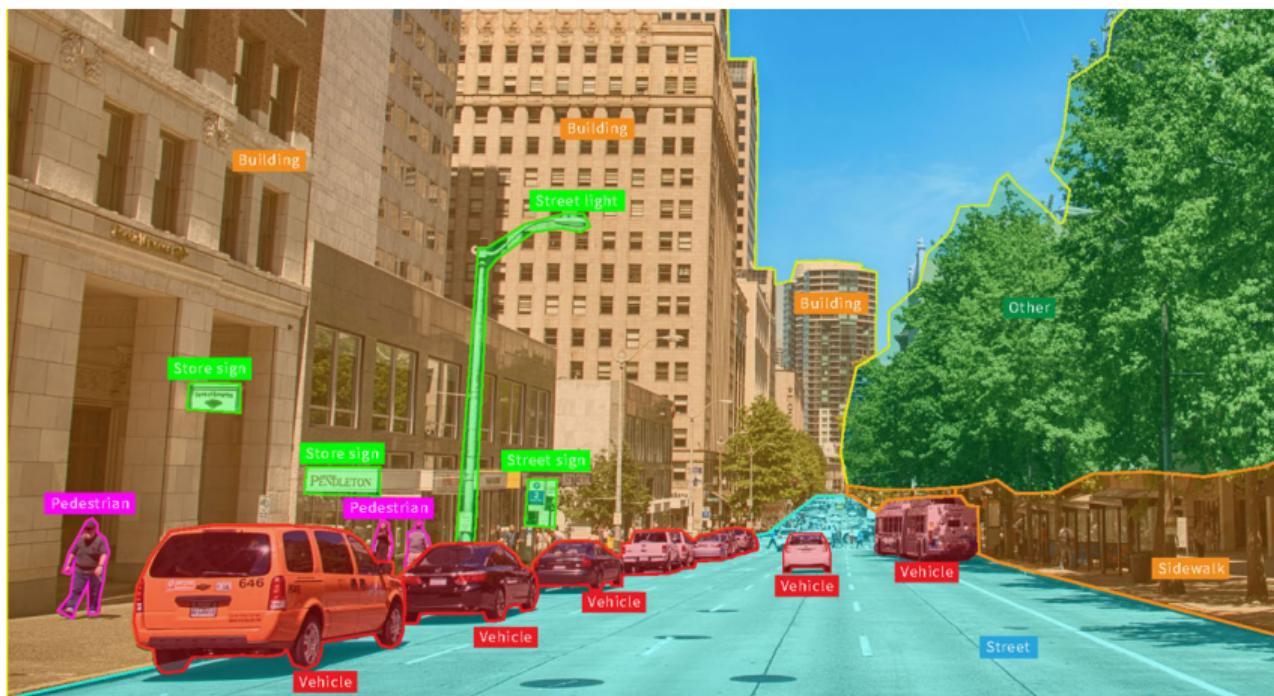
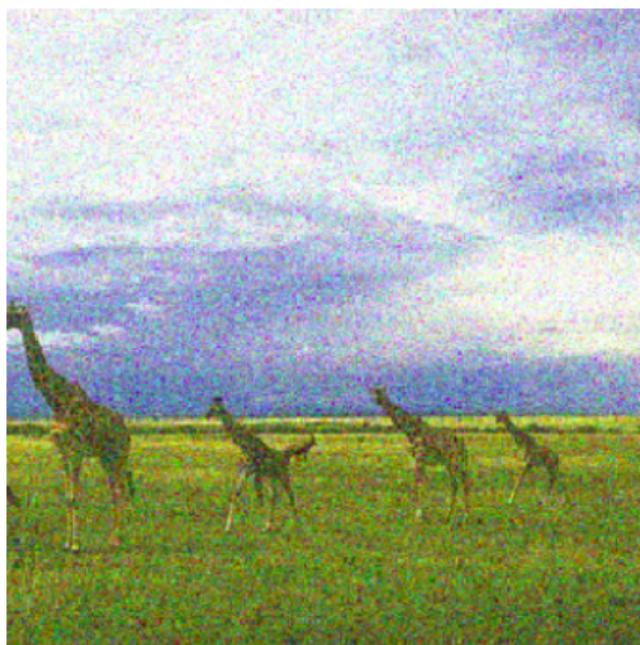


Figure adapted from source

What Is Computer Vision?



Noisy image



Denoised image

Figure adapted from source

What Is Computer Vision?



Hand-crafted Features: Before Deep Learning Features

Two types of features used for action recognition.

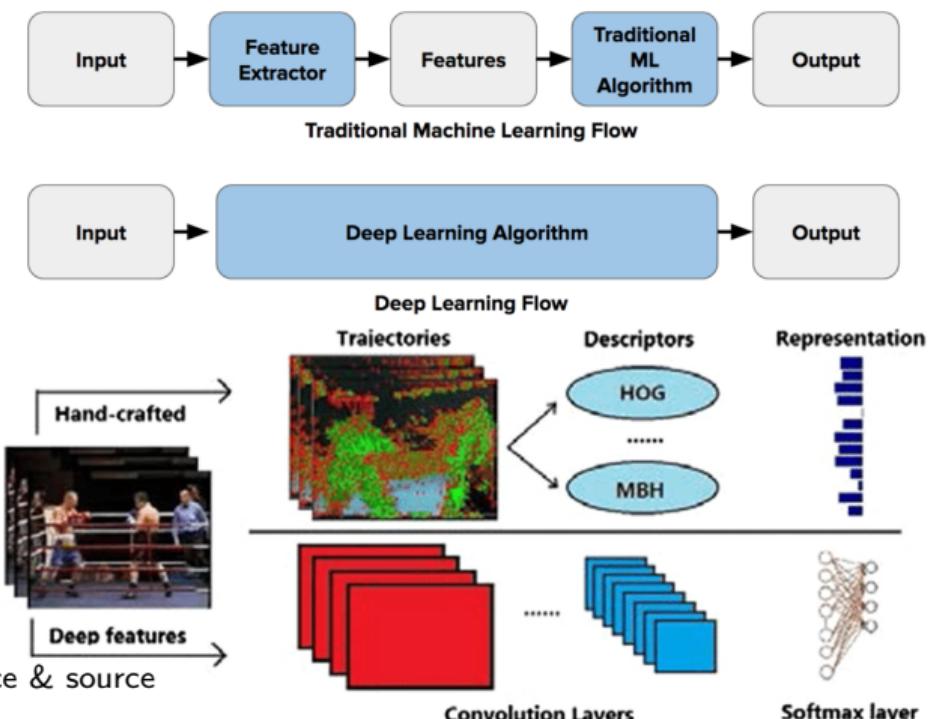
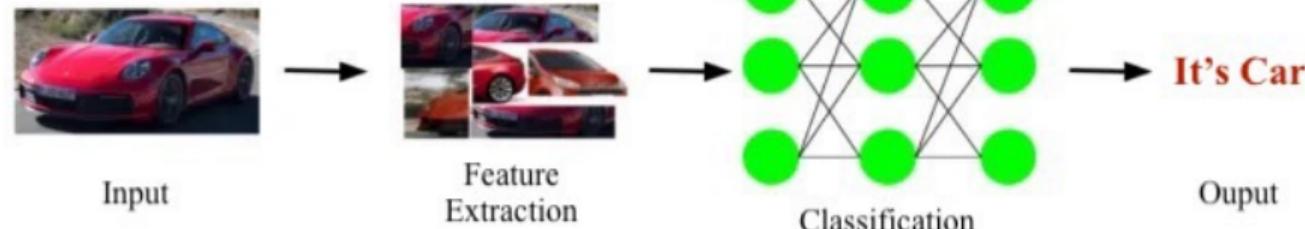


Figure adapted from source & source

Hand-crafted Features: Before Deep Learning Features

Hand-crafted machine learning



End-to-end learning with neural networks

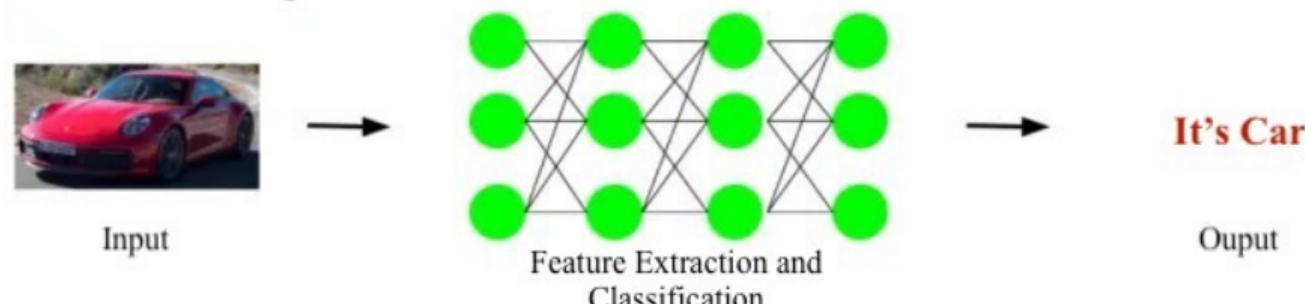


Figure adapted from source

Image Features

Fusion of multi-scale bag of deep visual words features of chest X-ray images to detect COVID-19 infection

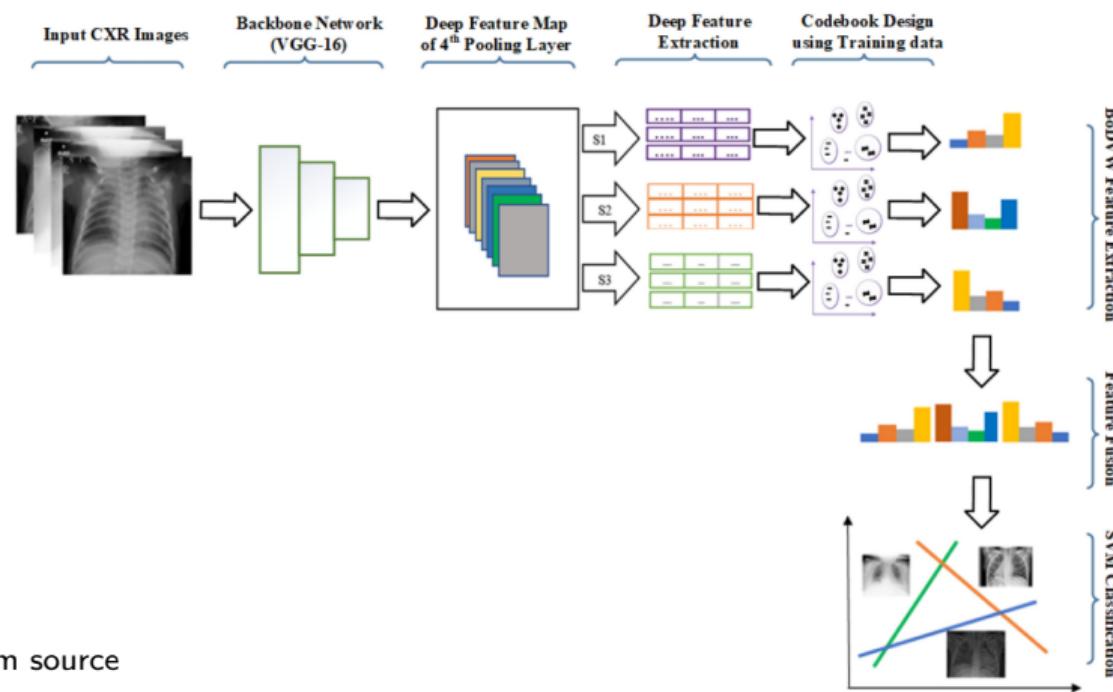


Figure adapted from source

Effect Of DL

Comparison between Deep Learning and Traditional Models

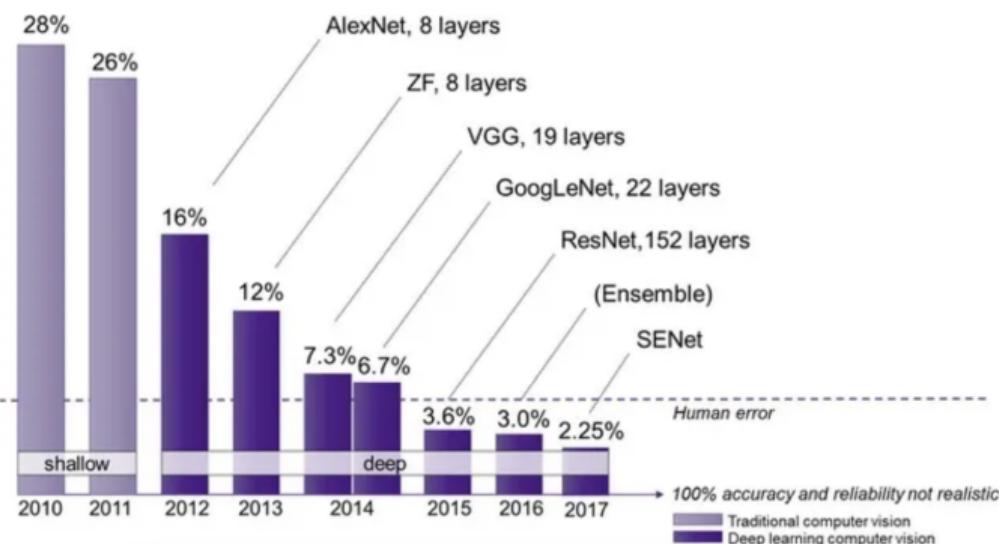


Figure adapted from source

1 Motivation

2 CNN vs. FCN

3 Convolution

4 Backpropagation

5 References

FCNs On Images

- What we've been using?
 - Dense Vector Multiplication

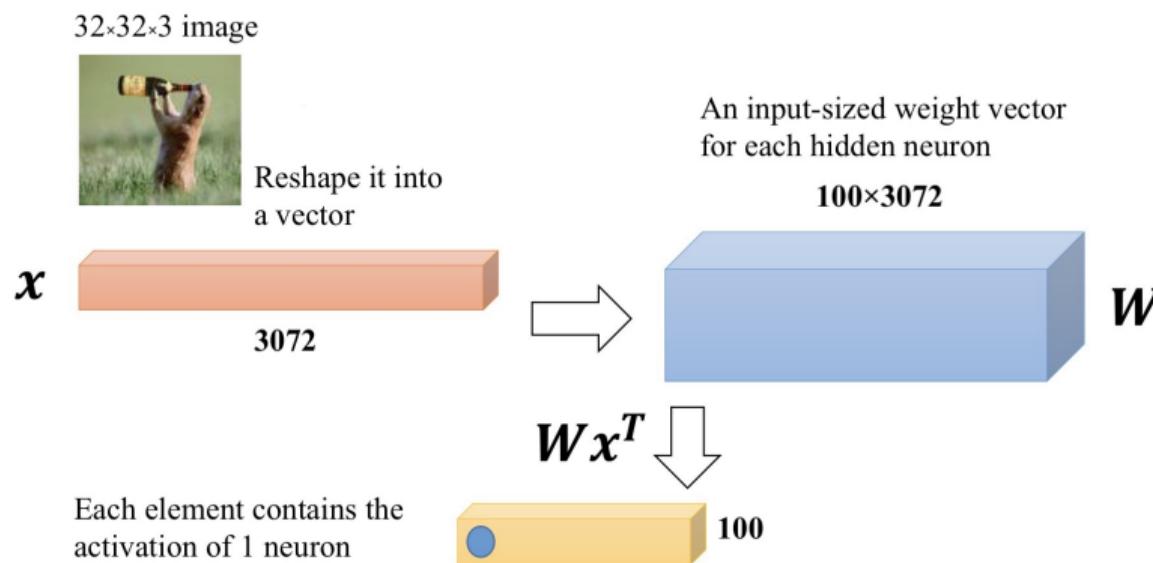


Figure adapted from [3]

Fully Connected Layers

- Neurons in a single layer operate independently and **do not share connections**, even for inputs.
- To be shift-invariant, **many samples** (in various time or locations) must be shown to them.
- **Regular Neural Nets don't scale well to full images.**
 - Parameters would **add up** quickly!
 - Full connectivity is wasteful and the huge number of parameters would quickly **lead to overfitting**.

Solution?

What can we do?

Review: Visualizing Inductive Bias

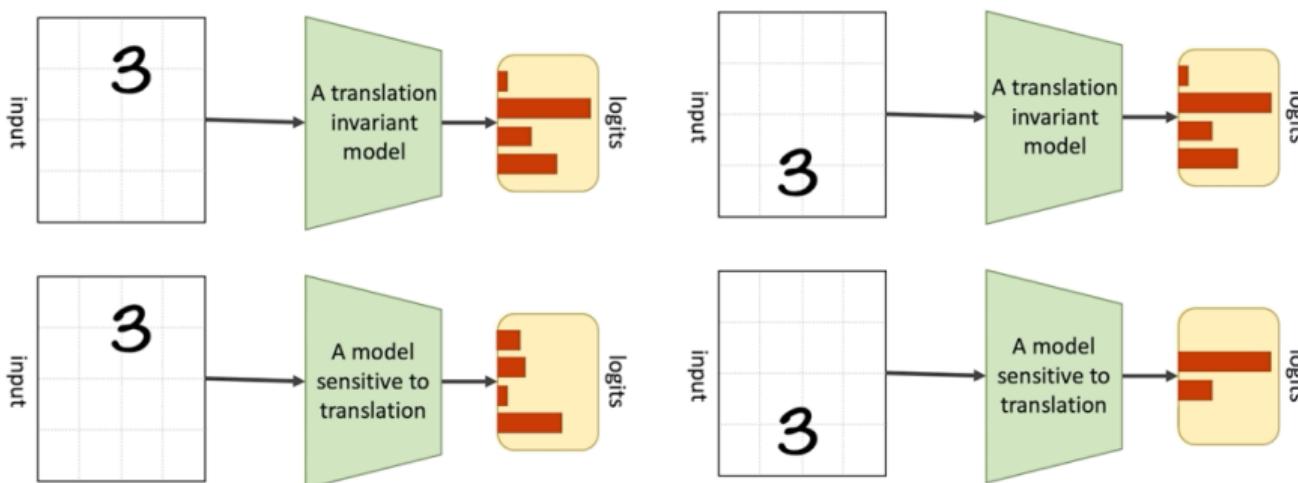
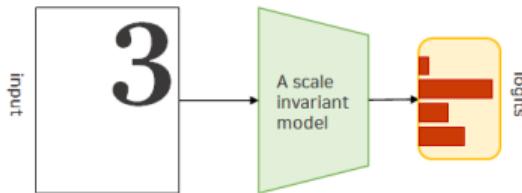
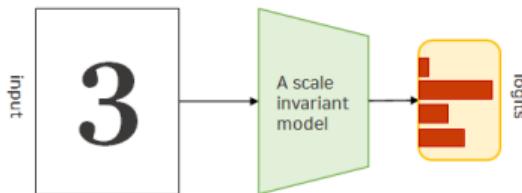


Figure 1: Adapted from Distilling Inductive Biases

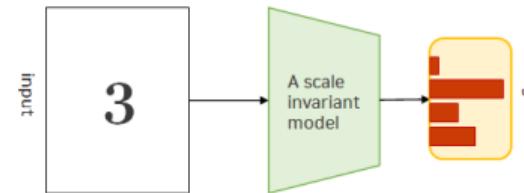
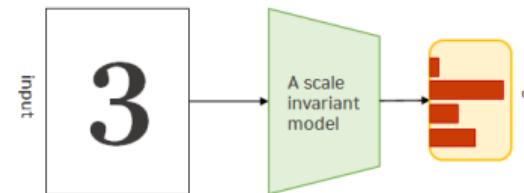
Review: Visualizing Inductive Bias

Inductive Biases for CNN : Translation Invariance



[Translation invariance]

Inductive Biases for CNN : Scale Invariance



[Scale invariance]

Figure 2: Adapted from Transferring Inductive Bias Through Knowledge Distillation

Review: Visualizing Inductive Bias

A drawing of how inductive biases can affect models' preferences to converge to different local minima. The inductive biases are shown by colored regions (green and yellow) which indicates regions that models prefer to explore.

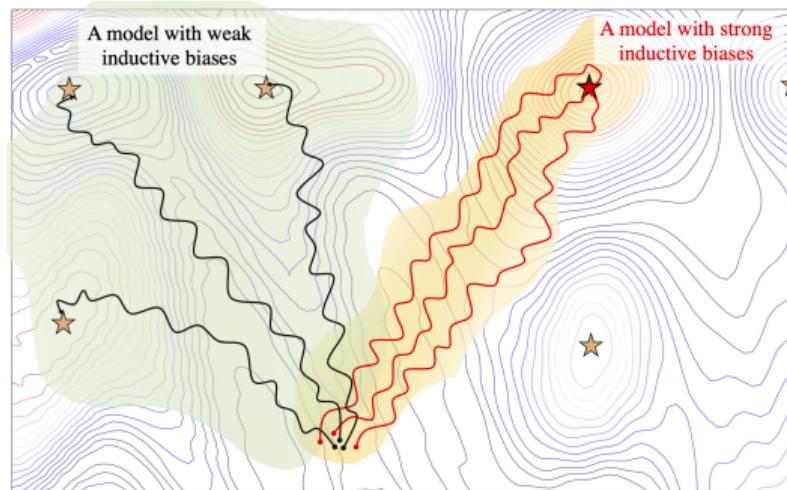


Figure 3: Adapted from Distilling Inductive Biases

Invariance VS. Equivariance

- **Invariance:** The output **remains the same** when the input undergoes a transformation.
- **Equivariance:** The output **varies predictably** when the input undergoes a transformation.

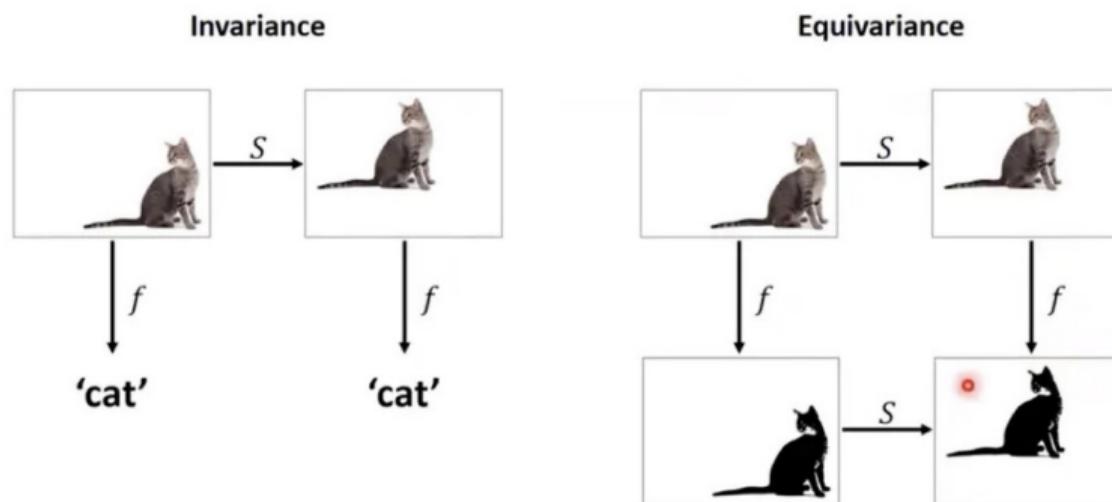
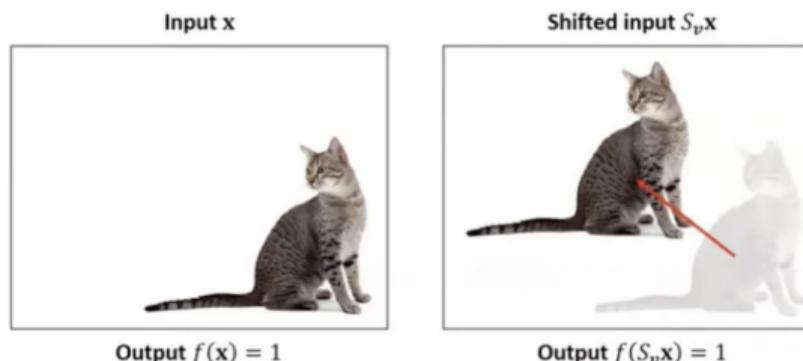


Figure adapted from source

Shift-invariance



- Cat detector $f : \mathbb{R}^d \rightarrow \mathbb{R}$
- Shift operator $S_v : \mathbb{R}^d \rightarrow \mathbb{R}^d$ shifting the image by vector v

Shift invariance: $f(x) = f(S_vx)$

Question: Will an MLP that recognizes the left image as a cat also recognize the shifted image on the right as a cat?

A Problem

- In many problems the **location** of a pattern is not important.
 - Only the **presence** of the pattern matters.
- Traditional MLPs are **sensitive** to pattern location.
 - Moving it by one component results in an entirely different input that the MLP won't recognize.
- **Requirement:** Network must be **shift invariant**.

Convolution On Images

Convolution (Refresher)

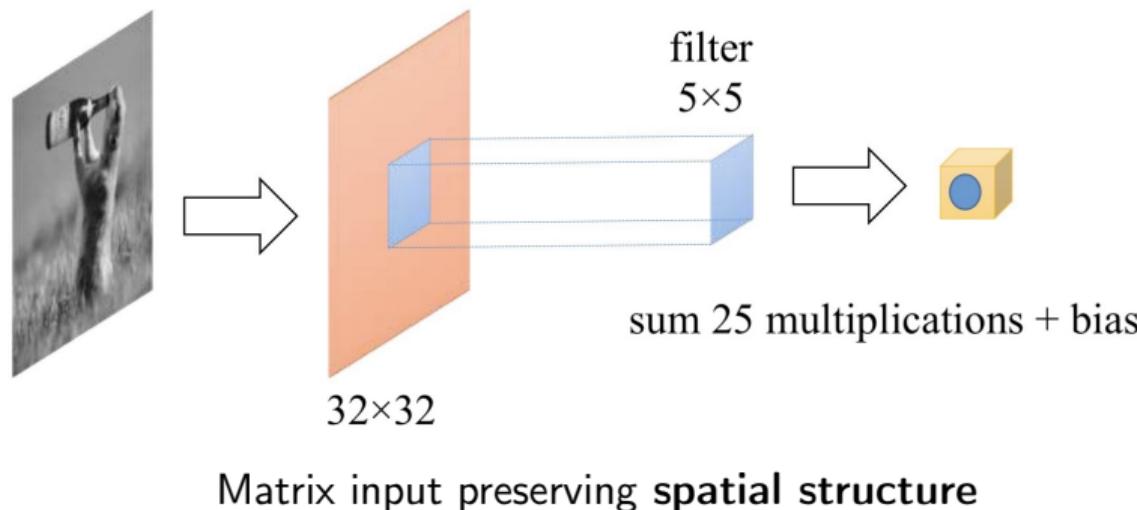


Figure adapted from [3]

Fully Connected Or Convolution

Question: Can I just do classification on an flatten image (e.g., a 1080x1080x3 image) with fully connected layers?

Fully Connected Or Convolution

Question: Can I just do classification on an flatten image (e.g., a 1080x1080x3 image) with fully connected layers?

Answer: No, using fully connected layers on such a large image is inefficient due to:

- Loss of spatial structure
- High parameter count

Fully Connected Or Convolution

Why Use Convolution:

- **Exploit spatial structure:** Convolution sees local patterns by using filters over small patches of the image.
- **Reduce parameters:** By sharing weights across the image, convolution reduce the number of parameters.
- **Build hierarchical features:** Convolution starts with small patterns, then combines them to recognize more complex patters.

Model Architecture	# Params	Test Accuracy
CNN	2M	67.8%
FCN	9M	50.6%

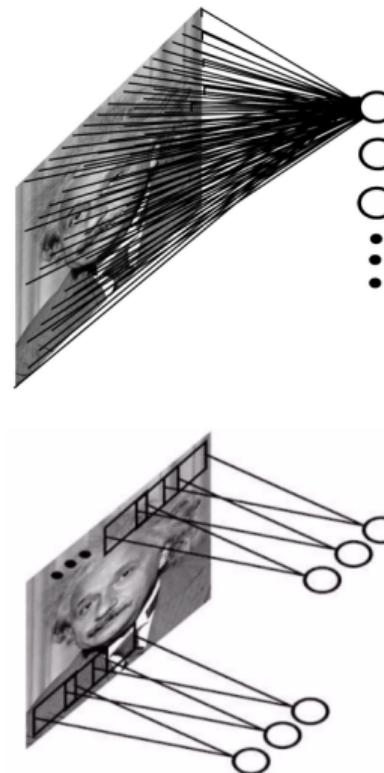
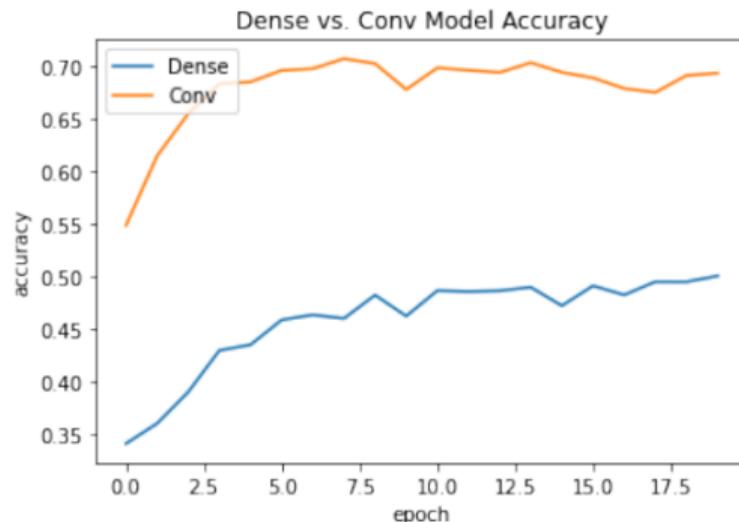


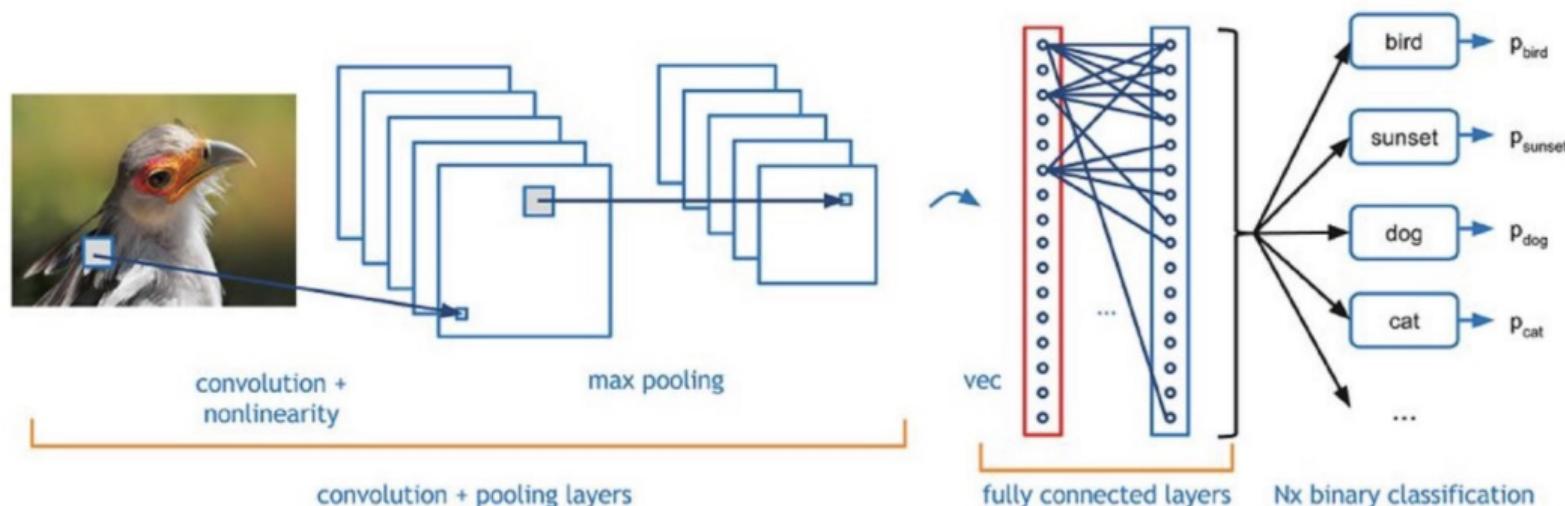
Figure adapted from source

What Is CNN?

- It is a class of deep learning.
- Convolutional neural network (ConvNet's or CNNs) is one of the main categories to do image recognition, image classifications, object detection, recognition of faces, etc.
- It is **similar to the basic neural network**. CNNs also have learnable parameters like weights and biases, similar to neural networks.
- CNN is heavily used in **computer vision**.
- There are **3 basic components** to define CNN:
 - The Convolution Layer
 - The Pooling Layer
 - The Output Layer or Fully Connected Layer

Architecture Of CNN

The basic idea of Convolutional Neural Networks (CNNs) is similar to Backpropagation Neural Networks (BPNNs) but differs in implementation.



After vectorized (vec), the 2D arranged inputs become ID vectors. Then the network is just like a BPNN (Back propagation neural networks)

Figure adapted from source



The Basic Structure

- Alternating Convolution (C) and subsampling layer (S)
- Subsampling allows flexible positioning of features.

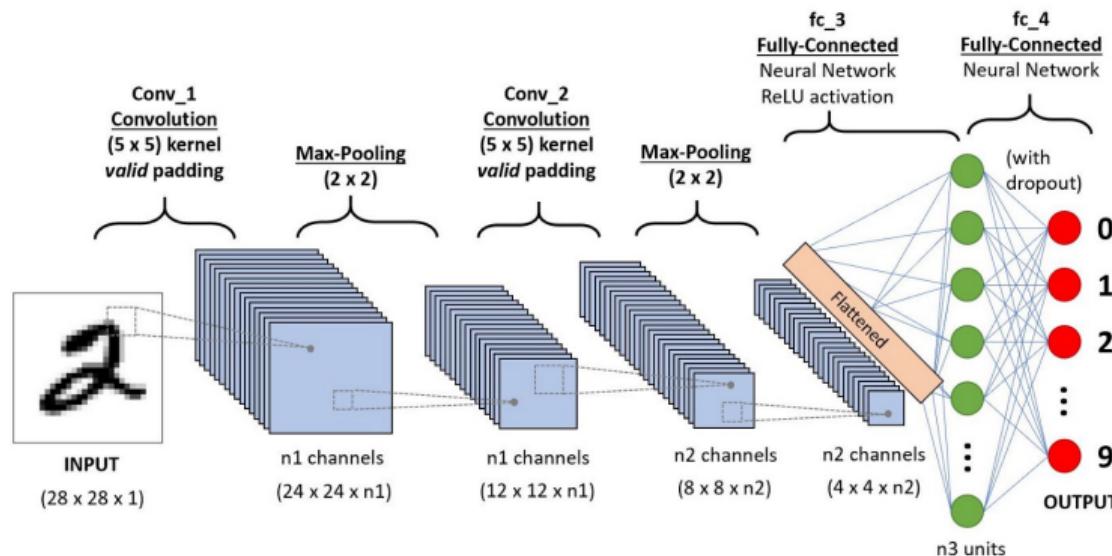


Figure adapted from source

The Basic Structure

Three Main Types of Layers

- **Convolutional Layer**
 - Output of neurons are connected to **local regions** in the input.
 - Applying the **same filter** across the entire image.
 - CONV layer's parameters consist of a set of **learnable filters**.
- **Pooling Layer**
 - Performs a **downsampling** operation along the spatial dimensions.
- **Fully-Connected Layer**
 - Typically used in the final stages of the network to combine high-level features and **make predictions**.

CNN VS. FCN

- **Fully Connected Networks (FCNs):**

- High number of parameters, leading to **overfitting on large inputs** (e.g., images).
- Lack of spatial awareness, making it **sensitive to the exact positioning** of patterns.
- Suitable for structured data but **inefficient for image processing**.

- **Convolutional Neural Networks (CNN):**

- Uses filters to process only small parts of the input at a time (**locality**).
- Weight sharing and local connectivity **reduce the number of parameters**.
- Can recognize patterns independent of their location within the image (**shift invariance**).
- **Efficient** for image, video, and speech data.

1 Motivation

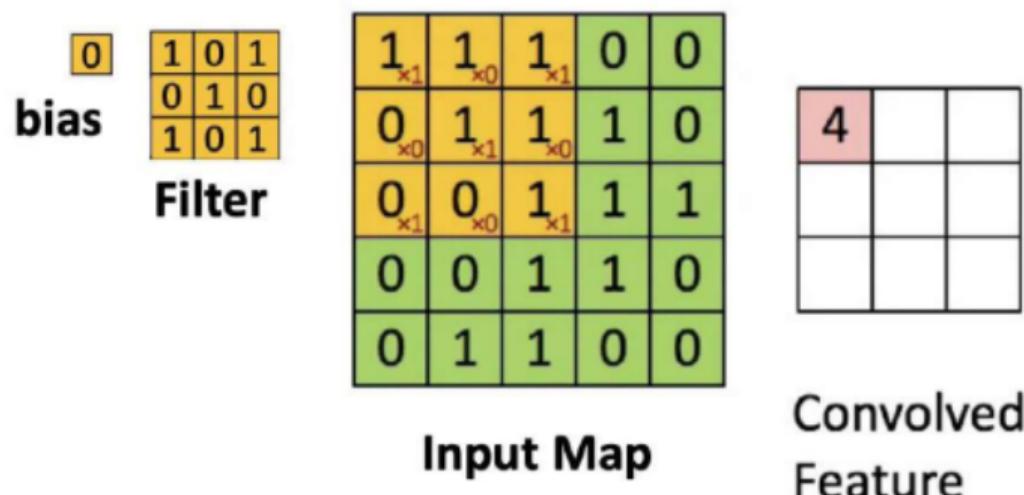
2 CNN vs. FCN

3 Convolution

4 Backpropagation

5 References

What Is A Convolve

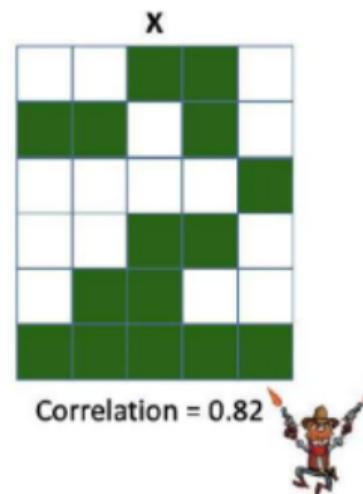
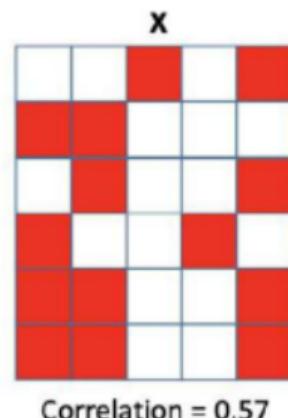
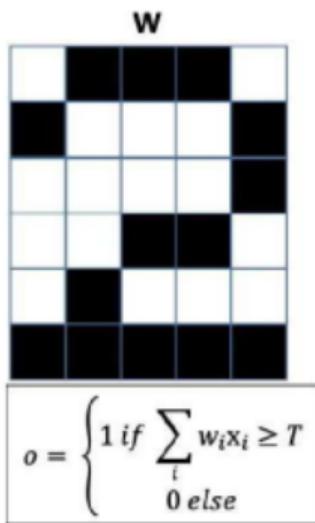


Scanning an image with a "filter"

- A filter is really just a perceptron, with weights and a bias.
- At each location, the filter is multiplied component-wise with the underlying map values, and the products are summed along with the bias.

Figure adapted from [3]

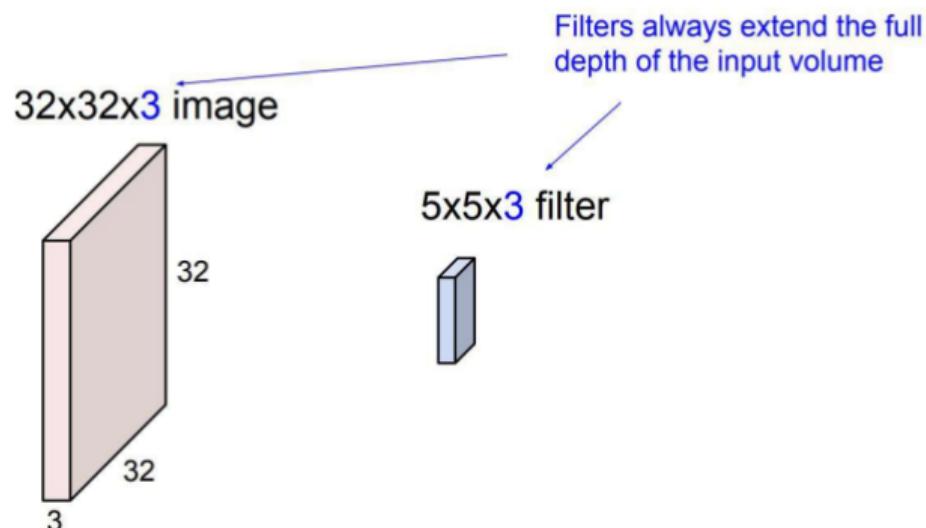
Weights Showing Correlation



- The **weights** of the filter **represent the appearance** of the number "2".
- The **green** has a higher correlation with the filter compared to the **red**.
- The **green pattern** is more likely to represent the number "2".

Figure adapted from [1]

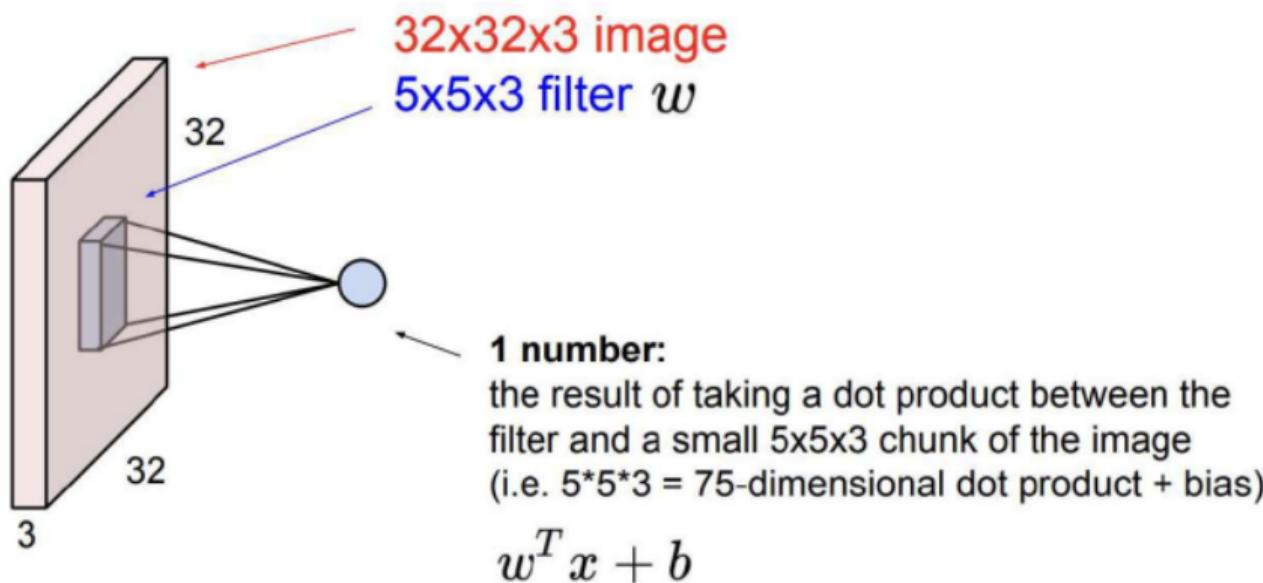
What Is Convolution



- **Convolve:** Slide over the image spatially, computing dot products.
- This allows us to **preserve the spatial structure** of the input.

Figure adapted from [2]

What Is The Output



- It's simply a neuron with local connectivity!

Figure adapted from [2]

Convolution Process

- Consider this as the filter we are going to use:

1	0	1
0	1	0
1	0	1

Convolution Process

- This is how we calculate the convolutional layer's output:

$$\text{Convolved Feature}(i,j) = (I * K)(i,j) = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} I(i+a, j+b)K(a,b)$$

I: Input Image

K: Our Kernel

k_h and k_w : The height and width of the Kernel

Convolution Process

- Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

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

Image

4	3	4

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

1	1	1	0	0
0 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	1	0
0 <small>x0</small>	0 <small>x1</small>	1 <small>x0</small>	1	1
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	0
0	1	1	0	0

Image

4	3	4
2		

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convolving a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

1	1	1	0	0
0	1 _{x1}	1 _{x0}	1 _{x1}	0
0	0 _{x0}	1 _{x1}	1 _{x0}	1
0	0 _{x1}	1 _{x0}	1 _{x1}	0
0	1	1	0	0

Image

4	3	4
2	4	

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convolving a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

1	1	1	0	0
0	1	1 _{x1}	1 _{x0}	0 _{x1}
0	0	1 _{x0}	1 _{x1}	1 _{x0}
0	0	1 _{x1}	1 _{x0}	0 _{x1}
0	1	1	0	0

Image

4	3	4
2	4	3

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convolving a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

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

Image

4	3	4
2	4	3
2		

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convolving a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

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

Image

4	3	4
2	4	3
2	3	

Convolved Feature

Figure adapted from [3]

Convolution Process

- Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

1	0	1
0	1	0
1	0	1

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

Image

4	3	4
2	4	3
2	3	4

Convolved Feature

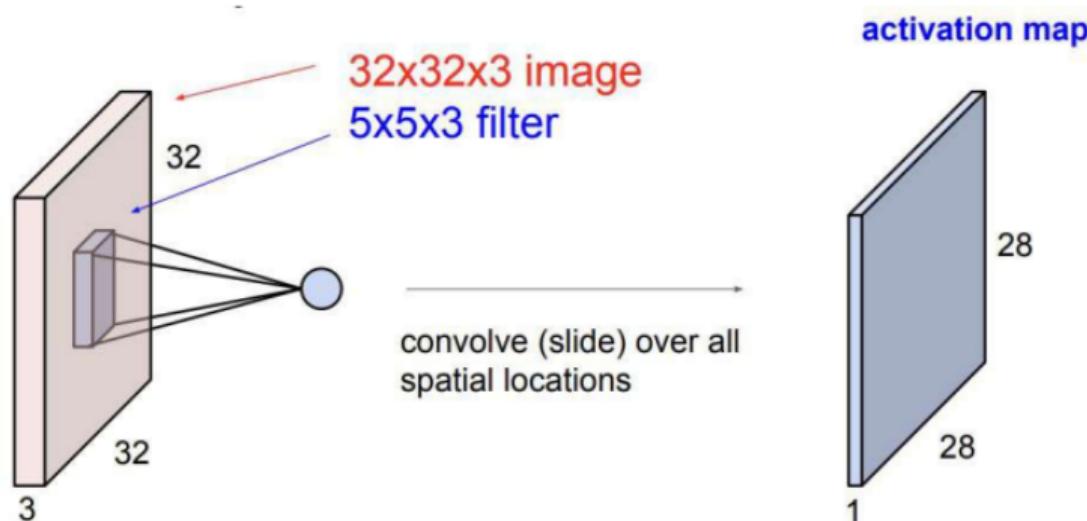
Figure adapted from [3]

To Summarize

- We apply the same filter on different regions of the input.
- Convolutional filters learn to make **decisions based on local spatial input**, which is an important attribute when working with images.
- Uses a lot **fewer parameters** compared to Fully Connected Layers.



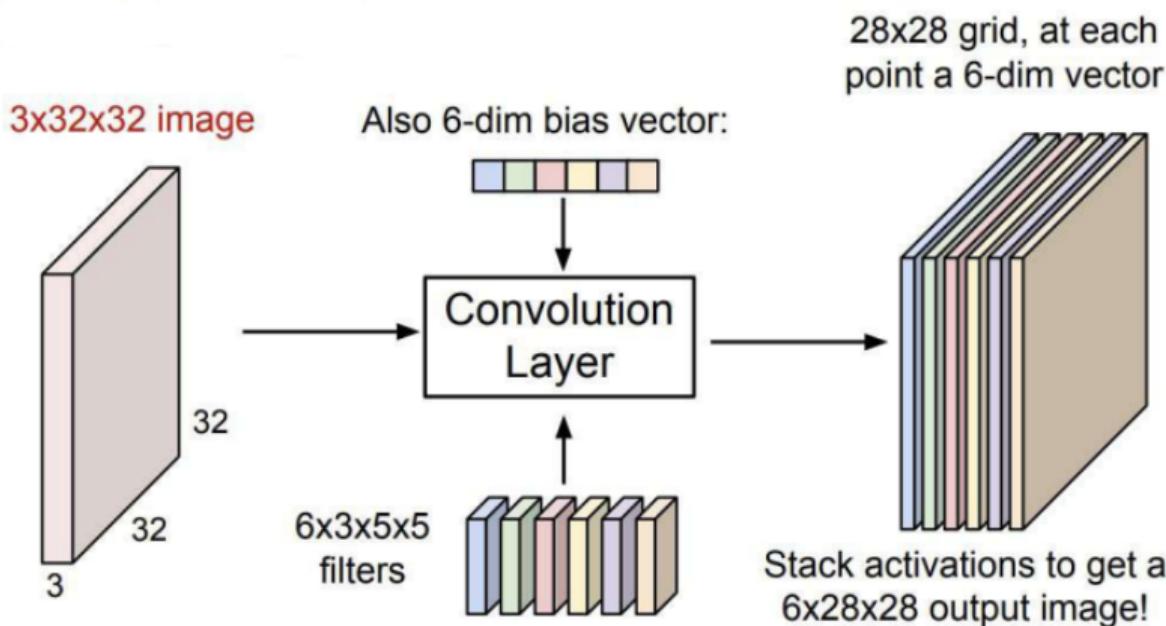
Convolution Outlook



- If we consider the **image** to be of size $n \times m \times b$ and the **filter** to be of size $n' \times m' \times b$, we will have an **activation map** of size $(n - n' + 1) \times (m - m' + 1) \times 1$ for each filter.

Figure adapted from [2]

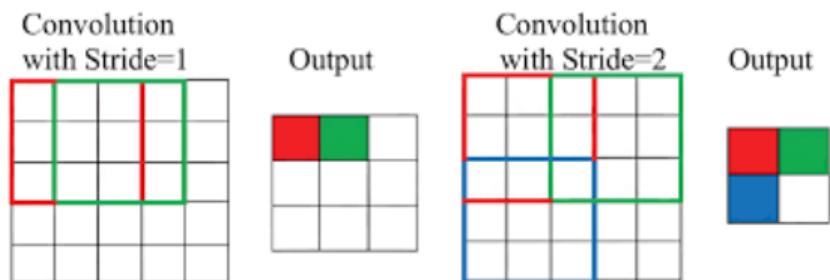
Convolution Outlook



- Multiple filters can be applied, each with its own bias, to extract different features from the input.

Figure adapted from [2]

What Is Stride



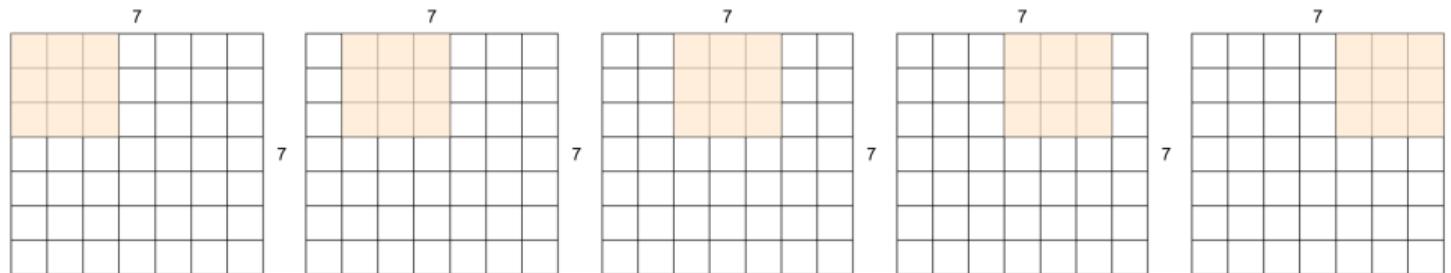
- The scans of the individual filters may **advance by more than one pixel at a time**.
 - The **stride** can be greater than 1.
 - Effectively increasing the granularity of the scan.
 - **Saves computation**, sometimes at the risk of **losing information**.
- This results in a **reduction** in the size of the **resulting maps**.
 - They will shrink by a factor equal to the stride.
- This can happen at **any layer**.

Figure adapted from [3]

Stride-1

Closer look

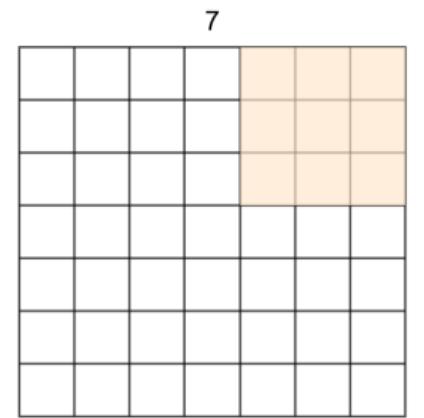
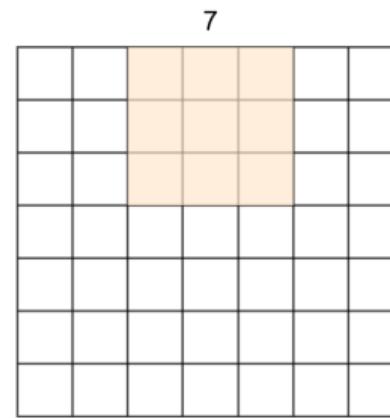
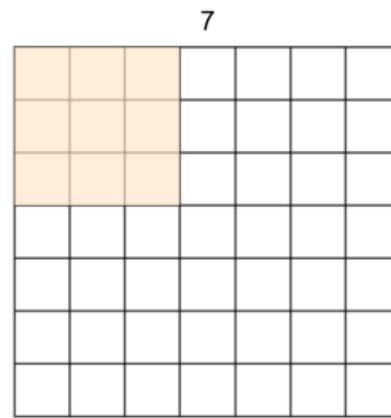
- 7x7 input with 3x3 filter
- This was a Stride 1 filter
- => Outputs 5x5



Strides-2

Now let's use Stride 2

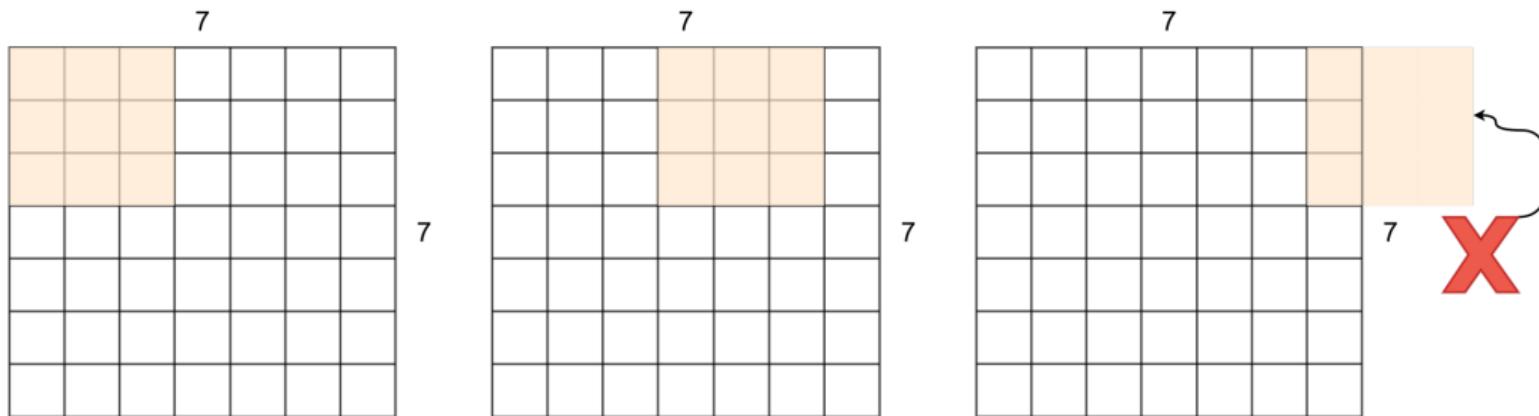
- 7x7 input with 3x3 filter
- This was a Stride 2 filter
- => Outputs 3x3



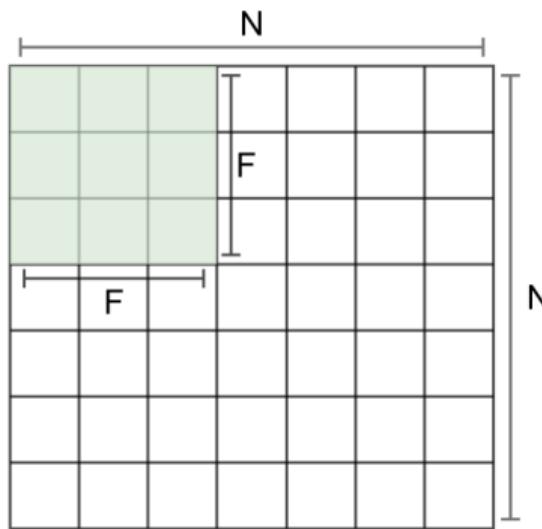
Strides-3

What about Stride 3?

- 7x7 input with 3x3 filter



Strides Formula



$$\text{OutputSize} = \frac{(N-F)}{\text{Stride}} + 1$$

$$N = 7, F = 3 \Rightarrow$$

- Stride 1: $(7 - 3)/1 + 1 = 5$
- Stride 2: $(7 - 3)/2 + 1 = 3$
- **Stride 3:** $(7 - 3)/3 + 1 = 2.33 !!!$

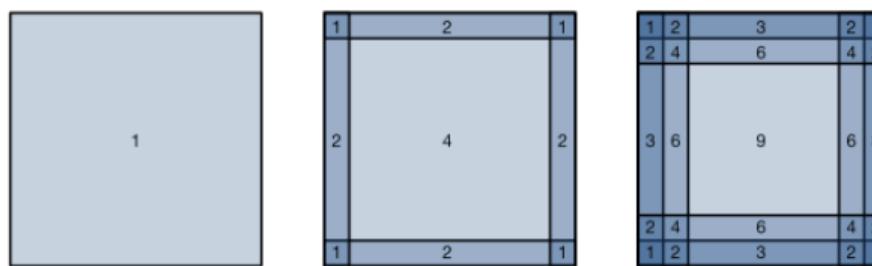
Figure adapted from [2]

A problem?

What could cause problems?

Problem-1

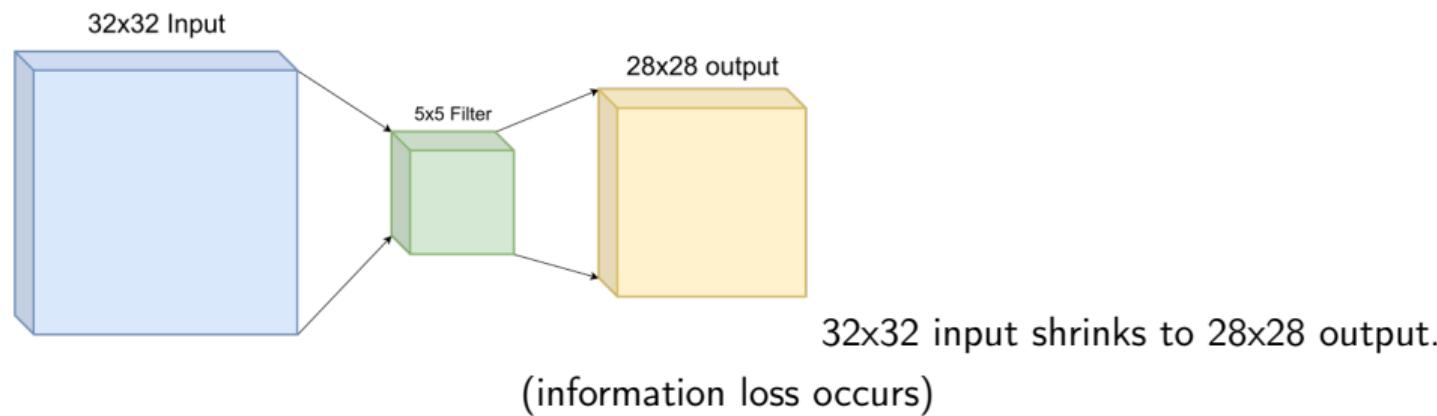
- The borders don't get enough attention.
- Outputs shrink!



Pixel utilization for convolutions of size 1×1 , 2×2 , and 3×3 respectively.

Problem-2

- Borders don't get enough attention.
- **Outputs shrink!**



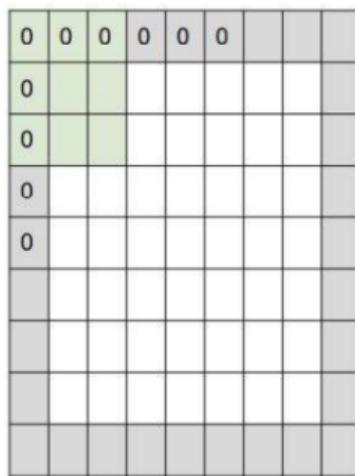
Solution?

What is the solution?

Padding

Padding: the secret ingredient to keep strides in line!

Padding



In practice, it is common to zero-pad the border.

Recall: The original formula for convolution without padding:

$$\frac{N - F}{\text{stride}} + 1$$

Now, we should adjust the formula considering padding P :

$$\frac{N + 2P - F}{\text{stride}} + 1$$

Figure adapted from [2]

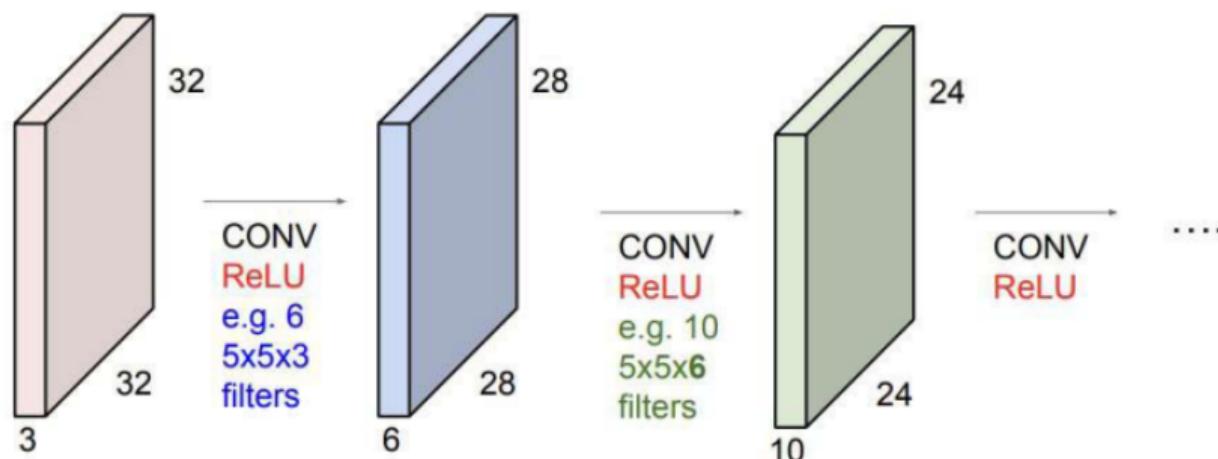
Padding

- Zero-padding is used not only for stride > 1 , but also **to prevent a reduction in output size even when $S = 1$** .
- For stride > 1 , zero padding is adjusted to ensure that the size of the convolved output is: $\left\lceil \frac{N}{S} \right\rceil$

This is achieved by zero padding the image with: $P = S \left\lceil \frac{N}{S} \right\rceil - N$

- For an F width filter:
 - **Odd F** : Pad on both left and right with $\frac{F-1}{2}$ columns of zeros.
 - **Even F** : Pad one side with $\frac{F}{2}$ columns of zeros, and the other with $\frac{F}{2}-1$ columns of zeros.
 - The resulting image is width: $N + F - 1$
- The top & bottom zero padding follows the same rules to maintain map height after convolution.

Convnet



- **ConvNet:** a sequence of convolution layers **interspersed with activation functions.**

Figure adapted from [2]

Convnet

Question: What do convolutional filters at different levels of a ConvNet learn?

Convnet

Question: What do convolutional filters at different levels of a ConvNet learn?

Answer:

- Filters in the **early layers** typically detect **simple features** such as edges, textures, and basic shapes.
- As we move **deeper** into the network, the filters learn **more complex and abstract features**, such as specific parts of objects (e.g., a nose, eyes, or other high-level patterns).

Convolution Output

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyper-parameters:
 - Number of filters K
 - Their spatial extent F
 - The stride S
 - The amount of zero padding P
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = \left(\frac{W_1 - F + 2P}{S}\right) + 1$
 - $H_2 = \left(\frac{H_1 - F + 2P}{S}\right) + 1$
 - $D_2 = K$

Parameter Setting

- With parameter sharing, convolution introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.

Common settings:

- $K = \text{powers of 2 (e.g., 32, 64, \dots)}$
- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = \text{adjusted accordingly.}$
- $F = 1, S = 1, P = 0$

Example

Question: Given an input volume of $32 \times 32 \times 3$, we apply **10** filters, each of size **5×5** , with a stride of **1** and padding of **2**.

- What is the output volume size of this convolutional layer?
- How many parameters are required for this convolutional layer?

Example

Output Volume Size:

$$\left(\frac{32 + 2 \times 2 - 5}{1} \right) + 1 = 32 \quad (\text{spatial dimensions}), \quad 32 \times 32 \times 10$$

Number of Parameters:

Each filter parameters: $5 \times 5 \times 3 + 1 = 76 \quad (+1 \text{ for bias}).$
Therefore, total parameters: $76 \times 10 = 760.$

1 Motivation

2 CNN vs. FCN

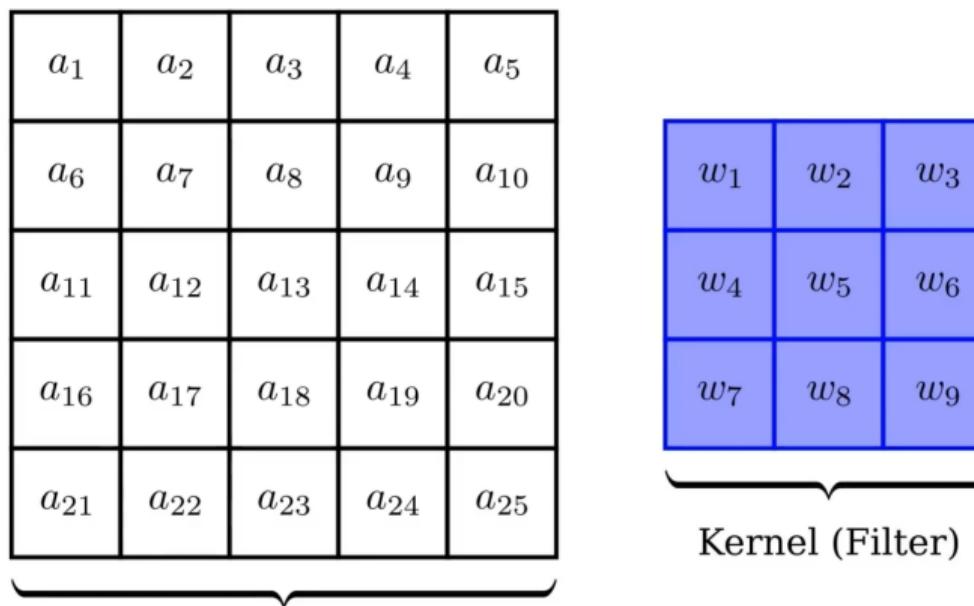
3 Convolution

4 Backpropagation

5 References

Example Overview

- we will tackle this problem with an example using stride = 2 and padding = 0.



Figures adapted from [7]

Input Layer

Convolution

Recall: convolution process

Convolution

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Convolution

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

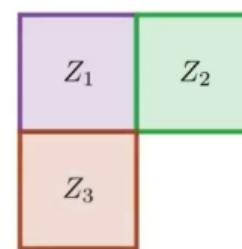
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Convolution

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

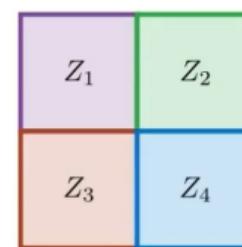
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Convolution

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Another view

$$z_1 = w_1 \times a_1 + w_2 \times a_2 + w_3 \times a_3 + w_4 \times a_4 + \dots + w_9 \times a_{13}$$

$$z_2 = w_1 \times a_3 + w_2 \times a_4 + w_3 \times a_5 + w_4 \times a_8 + \dots + w_9 \times a_{15}$$

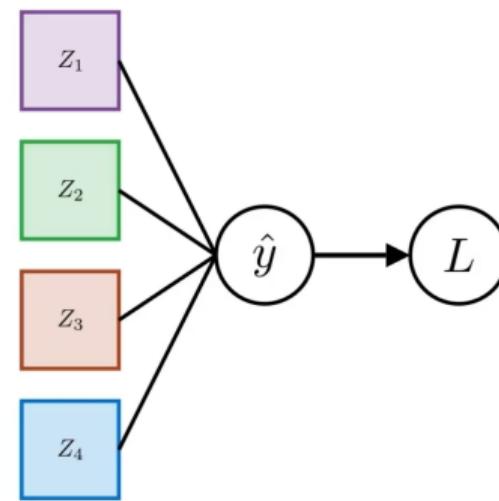
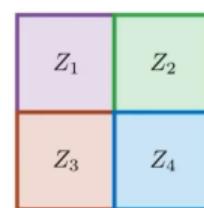
$$z_3 = w_1 \times a_{11} + w_2 \times a_{12} + w_3 \times a_{13} + w_4 \times a_{16} + \dots + w_9 \times a_{23}$$

$$z_4 = w_1 \times a_{13} + w_2 \times a_{14} + w_3 \times a_{15} + w_4 \times a_{18} + \dots + w_9 \times a_{25}$$

Whole Network

- For easier computation, we will assume one layer of convolution and a perceptron as our whole network.
- Recall:** $w_i^* = w_i - \alpha \times \frac{\partial L}{\partial w_i}$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Gradient

- We can easily calculate the gradients:

$$\begin{aligned}\frac{\partial L}{\partial w_1} &= \frac{\partial z_1}{\partial w_1} \frac{\partial L}{\partial z_1} + \frac{\partial z_2}{\partial w_1} \frac{\partial L}{\partial z_2} + \frac{\partial z_3}{\partial w_1} \frac{\partial L}{\partial z_3} + \frac{\partial z_4}{\partial w_1} \frac{\partial L}{\partial z_4} \\ &= a_1 \frac{\partial L}{\partial z_1} + a_3 \frac{\partial L}{\partial z_2} + a_{11} \frac{\partial L}{\partial z_3} + a_{13} \frac{\partial L}{\partial z_4}\end{aligned}$$

Gradient

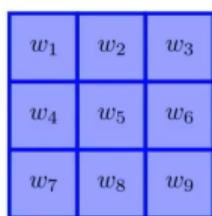
Do the same for each w

$$\frac{\partial L}{\partial w_1} = a_1 \frac{\partial L}{\partial z_1} + a_3 \frac{\partial L}{\partial z_2} + a_{11} \frac{\partial L}{\partial z_3} + a_{13} \frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_2} = a_2 \frac{\partial L}{\partial z_1} + a_4 \frac{\partial L}{\partial z_2} + a_{12} \frac{\partial L}{\partial z_3} + a_{14} \frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_3} = a_3 \frac{\partial L}{\partial z_1} + a_5 \frac{\partial L}{\partial z_2} + a_{13} \frac{\partial L}{\partial z_3} + a_{15} \frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_4} = a_6 \frac{\partial L}{\partial z_1} + a_8 \frac{\partial L}{\partial z_2} + a_{16} \frac{\partial L}{\partial z_3} + a_{18} \frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_5} = a_7 \frac{\partial L}{\partial z_1} + a_9 \frac{\partial L}{\partial z_2} + a_{17} \frac{\partial L}{\partial z_3} + a_{19} \frac{\partial L}{\partial z_4}$$

...

Overview

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}



$$\begin{aligned}
 \frac{\partial L}{\partial w_1} &= a_1 \frac{\partial L}{\partial z_1} + a_3 \frac{\partial L}{\partial z_2} + a_{11} \frac{\partial L}{\partial z_3} + a_{13} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_2} &= a_2 \frac{\partial L}{\partial z_1} + a_4 \frac{\partial L}{\partial z_2} + a_{12} \frac{\partial L}{\partial z_3} + a_{14} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_3} &= a_3 \frac{\partial L}{\partial z_1} + a_5 \frac{\partial L}{\partial z_2} + a_{13} \frac{\partial L}{\partial z_3} + a_{15} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_4} &= a_6 \frac{\partial L}{\partial z_1} + a_8 \frac{\partial L}{\partial z_2} + a_{16} \frac{\partial L}{\partial z_3} + a_{18} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_5} &= a_7 \frac{\partial L}{\partial z_1} + a_9 \frac{\partial L}{\partial z_2} + a_{17} \frac{\partial L}{\partial z_3} + a_{19} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_6} &= a_8 \frac{\partial L}{\partial z_1} + a_{10} \frac{\partial L}{\partial z_2} + a_{18} \frac{\partial L}{\partial z_3} + a_{20} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_7} &= a_{11} \frac{\partial L}{\partial z_1} + a_{13} \frac{\partial L}{\partial z_2} + a_{21} \frac{\partial L}{\partial z_3} + a_{23} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_8} &= a_{12} \frac{\partial L}{\partial z_1} + a_{14} \frac{\partial L}{\partial z_2} + a_{22} \frac{\partial L}{\partial z_3} + a_{24} \frac{\partial L}{\partial z_4} \\
 \frac{\partial L}{\partial w_9} &= a_{13} \frac{\partial L}{\partial z_1} + a_{15} \frac{\partial L}{\partial z_2} + a_{23} \frac{\partial L}{\partial z_3} + a_{25} \frac{\partial L}{\partial z_4}
 \end{aligned}$$

$$\begin{array}{ccc}
 \begin{matrix} a_1 & a_2 & a_3 \\ a_6 & a_7 & a_8 \\ a_{11} & a_{12} & a_{13} \end{matrix} & \times \frac{\partial L}{\partial z_1} & + \quad \begin{matrix} a_3 & a_4 & a_5 \\ a_8 & a_9 & a_{10} \\ a_{13} & a_{14} & a_{15} \end{matrix} & \times \frac{\partial L}{\partial z_2} & + \quad \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} \end{matrix} & \times \frac{\partial L}{\partial z_3} & + \quad \begin{matrix} a_{13} & a_{14} & a_{15} \\ a_{18} & a_{19} & a_{20} \\ a_{23} & a_{24} & a_{25} \end{matrix} & \times \frac{\partial L}{\partial z_4} = & \begin{matrix} \frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_4} & \frac{\partial L}{\partial w_5} & \frac{\partial L}{\partial w_6} \\ \frac{\partial L}{\partial w_7} & \frac{\partial L}{\partial w_8} & \frac{\partial L}{\partial w_9} \end{matrix}
 \end{array}$$

Result

$$\begin{matrix} w_1^* & w_2^* & w_3^* \\ w_4^* & w_5^* & w_6^* \\ w_7^* & w_8^* & w_9^* \end{matrix} = \begin{matrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{matrix} - \alpha \times \begin{matrix} \frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_4} & \frac{\partial L}{\partial w_5} & \frac{\partial L}{\partial w_6} \\ \frac{\partial L}{\partial w_7} & \frac{\partial L}{\partial w_8} & \frac{\partial L}{\partial w_9} \end{matrix}$$

$$w_i^* = w_i - \alpha \times \frac{\partial L}{\partial w_i}$$

What's next?

Proceed with gradient descent.

So Far

- The highly structured nature of image data.
- Local correlations are important.
- CNNs have local and shared connections.
- Strides & padding.

Up Next

Learn more details about CNNs!

1 Motivation

2 CNN vs. FCN

3 Convolution

4 Backpropagation

5 References

Contributions

- This slide has been prepared thanks to:
 - Ali Aghayari
 - Donya Navabi

- [1] M. Soleymani Baghshah, "Deep learning." Lecture slides.
- [2] F.-F. Li, Y. Li, and R. Gao, "Cs231n: Deep learning for computer vision." Lecture slides.
- [3] B. Raj, R. Singh, and B. Dhingra, "11-785: Introduction to deep learning." Lecture slides.
- [4] M. Kellis, "6.874 computational systems biology: Deep learning in the life sciences." Lecture slides.
- [5] H. Li, "Eleg 5491: Introduction to deep learning." Lecture slides.
- [6] M. Elgendi, *Deep Learning for Vision Systems*.
Manning Publications, 2020.
- [7] DeepMind, "Deep learning for ai with geoffrey hinton, yoshua bengio, and yann lecun." YouTube video.