# Lecture: Convolutional Neural Network (CNN)
## An Introduction

Dr. Goutam Chakraborty

**SAS® Professor of Marketing Analytics**

**Director of MS in Business Analytics and Data Science*** (http://analytics.okstate.edu/mban/ )
**Director of Graduate Certificate in Business Data Mining** (http://analytics.okstate.edu/certificate/grad-data-mining/ )
**Director of Graduate Certificate in Marketing Analytics** (http://analytics.okstate.edu/certificate/grad-marketing-analytics/ )

- *Name change pending internal approval.
- 
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

1

## Outline

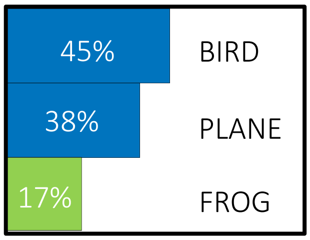
- Typical applications of CNN
- Motivations behind using CNN

2

## Applications of CNN: Image Classification and Object Detection

### Image Classification

"It's a bird!
It's a plane!
It's a frog!"



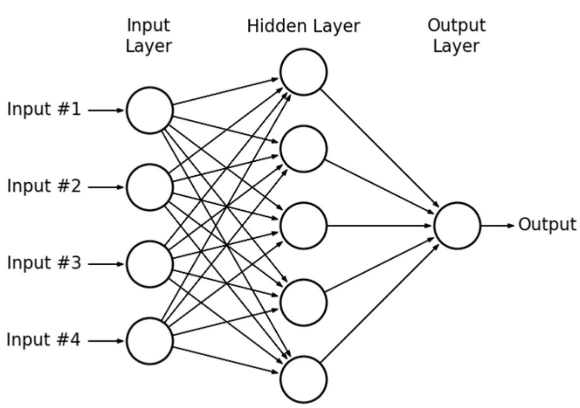

### Object Detection


### Natural Language Processing

"That was a funny joke……not!"

3

## Why Convolution?



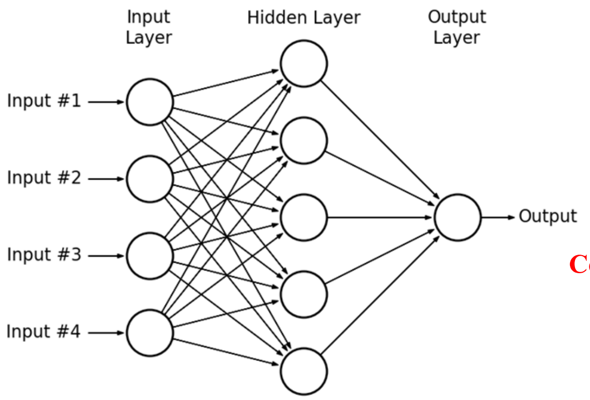

In MNIST data set the images were 28 X 28 pixels i.e., 784 input features

With a 100 unit hidden layer we need about 79,510 parameters in the the model

In real life images could be 512 X 512 pixels i.e., > 250K input features which means > 25m parameters

4

# Why Convolution? (Contd.)




In MNIST data set the images were 28 X 28 pixels i.e. 784 input features

With a 100 unit hidden layer we have about 79,510 parameters in the the model

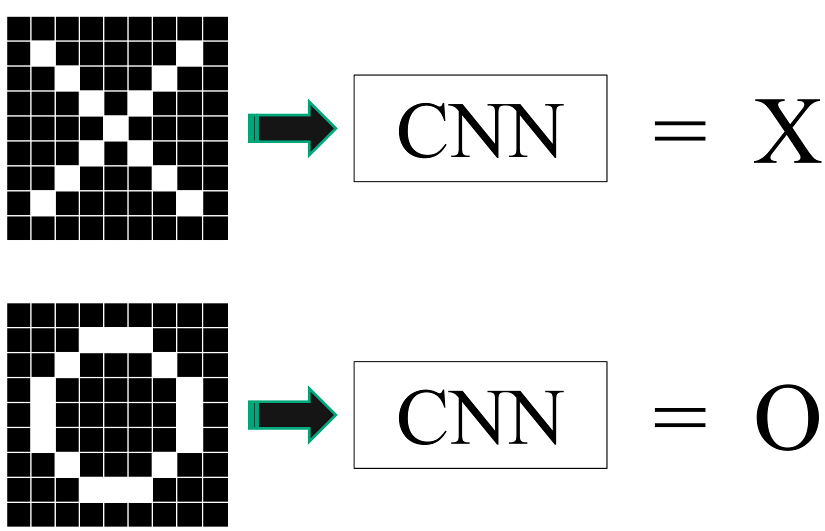
In real life images could be 512 X 512 pixels i.e. > 250K input features which means > 25m parameters

**Convolution Networks solve the following problems:**

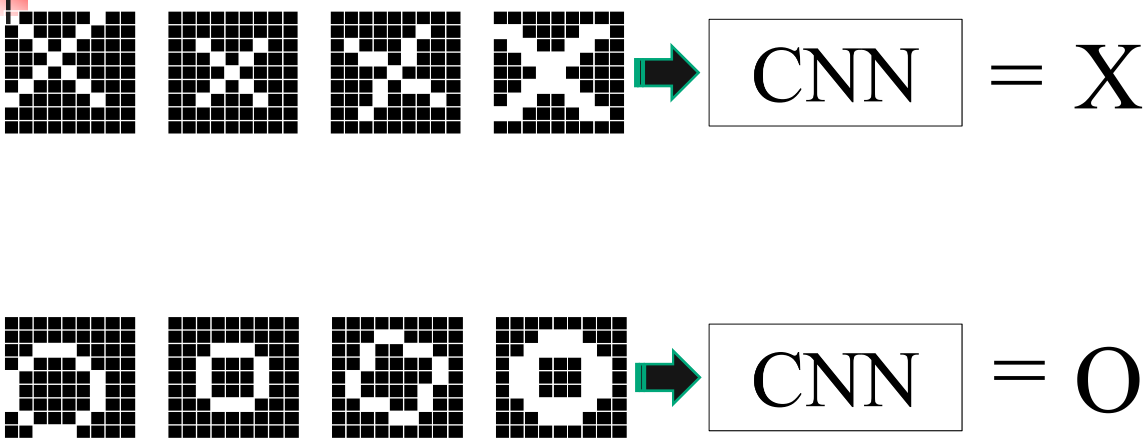
1. Reduce the number of parameters
2. Preserve spatial information – we do not need to flatten the inputs



# Output of Convolution



## Output of Convolution (Contd.)




7

## What is CNN?

- Convolutional neural networks (CNNs) are a type of neural network that is viewed to be **computationally and statistically efficient** (Goodfellow, Bengio, and Courville).
- A typical CNN consists of five types of layers: input, convolution, pooling, fully connected, and output.
  - Each type of layer has its own specific properties and functionalities.
- *Convolution layers* require fewer parameters than a fully connected layer because the parameters are shared across columns thereby improving computation efficiency.
- *Pooling layers* themselves do not contain parameters, but instead they combine columns with an output summary.

8

# Lecture: CNN
# Input and Convolutional Layers

Dr. Goutam Chakraborty

**SAS® Professor of Marketing Analytics**

**Director of MS in Business Analytics and Data Science*** (http://analytics.okstate.edu/mban/ )
**Director of Graduate Certificate in Business Data Mining** (http://analytics.okstate.edu/certificate/grad-data-mining/ )
**Director of Graduate Certificate in Marketing Analytics** (http://analytics.okstate.edu/certificate/grad-marketing-analytics/ )

- *Name change pending internal approval.
- 
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

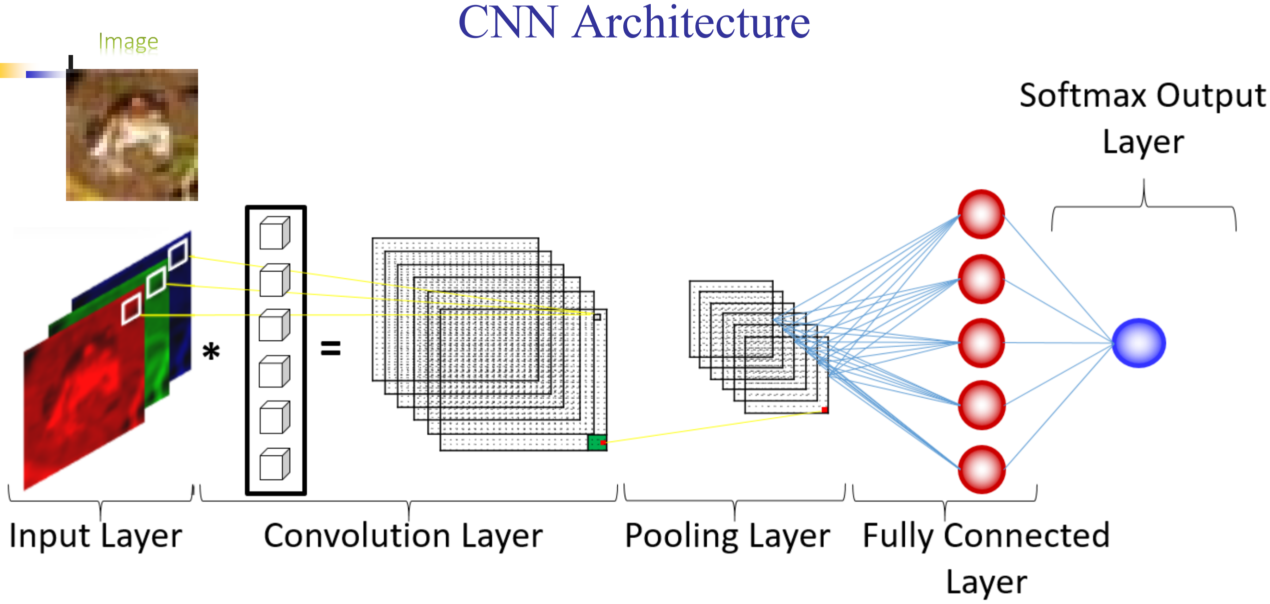
9

## Outline

- Basic CNN Architecture
- Input layers for grayscale vs. color
- Convolutional layers
  - Filters

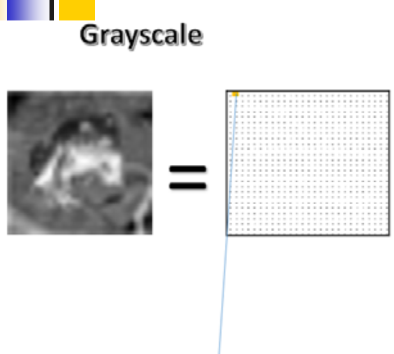
10

# CNN Architecture

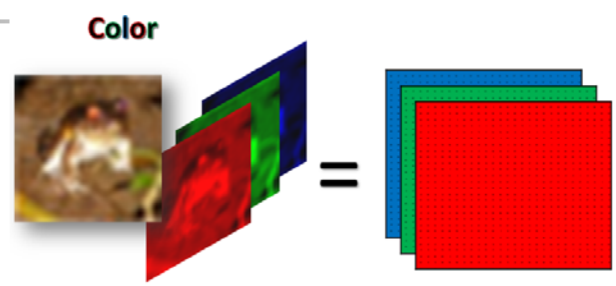



11

# CNN: Input Layer




| Label | Column 1 | Column 2 | .... | Column *N* |
|---|---|---|---|---|
| Frog | 223 | 225 | .... | 82 |



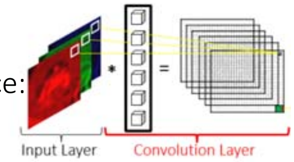

| Label | Column 1 | Column 2 | .... | Column *N* |
|---|---|---|---|---|
| Frog | 144 | 37 | .... | 180 |

12

## Convolutional Layers

- Convolutional layers use *filters*, which do the following:
  - capture edges
  - include learnable parameters
  - introduce new hyperparameters: **width, height, and stride**
  - The number of filter channels equals the number of channels present in the incoming information
- Convolutional layers are equivariant to translation.
- The cross-correlation operation is applied to the incoming information and the kernel filter.

Reference:



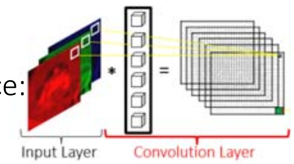



## Filters

- A typical convolution layer can consist of many filters.
- The filters *slide across the input surface in parallel*, capturing meaningful characteristics.
  - Therefore, the parameters in each filter are shared by multiple columns in the data. Parameter sharing decreases the number of parameters needed to translate the input space.
- Each filter creates **equivariant** representations of the input, which means that changes in the input space are represented in the output.

## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Bias

+ 10 =

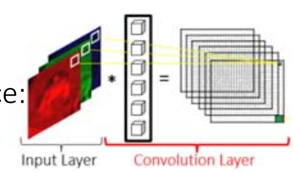
Feature Map

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |



## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Bias

+ 10 =

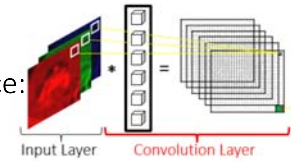
Feature Map

| | | | |
|---|---|---|---|
| ? | | | |
| | | | |
| | | | |
| | | | |

$245 + 97 + 239 + 0 + 0 + 0 - 212 - 175 - 160 + 10 = 44$

# Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

+ Bias 10 =

Feature Map

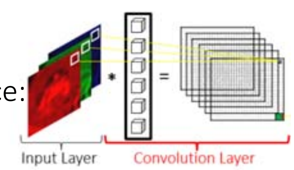
| | | | |
|---|---|---|---|
| 44 | | | |
| | | | |
| | | | |
| | | | |

245 + 97 + 239 + 0 + 0 + 0 - 212 - 175 - 160 + 10 = 44

17

# Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


Stride = 1

| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

+ Bias 10 =

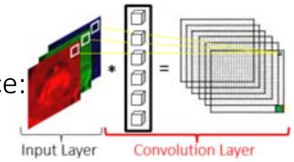
Feature Map

| | | | |
|---|---|---|---|
| 44 | ? | | |
| | | | |
| | | | |
| | | | |

97 + 239 + 187 + 0 + 0 + 0 - 175 - 160 - 11 + 10 = 187

18

## Convolutional Layers

Reference: 


Single Input Channel: 6 * 6

| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Bias

+ 10 =

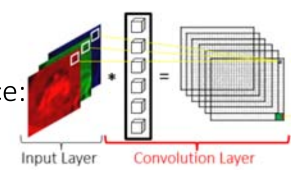
Feature Map

| | | | |
|---|---|---|---|
| 44 | 187 | | |
| | | | |
| | | | |
| | | | |

97 + 239 + 187 + 0 + 0 + 0 - 175 - 160 - 11 + 10 = 187

19

## Convolutional Layers

Reference: 


Single Input Channel: 6 * 6

| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Bias

+ 10 =

Feature Map

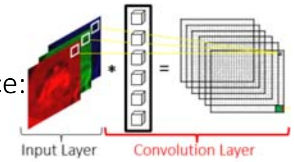
| | | | |
|---|---|---|---|
| 44 | 187 | 433 | |
| | | | |
| | | | |
| | | | |

239 + 187 + 195 + 0 + 0 + 0 - 160 - 11 - 27 + 10 = 433

20

## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

\+ Bias 10 =

Feature Map

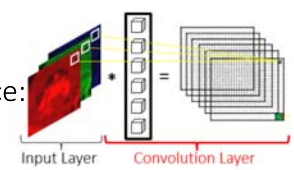
| | | | |
|---|---|---|---|
| 44 | 187 | 433 | 606 |
| | | | |
| | | | |
| | | | |

187 + 195 + 254 + 0 + 0 + 0 – 11 – 27 – 2 + 10 = 606

21

## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

\+ Bias 10 =

Feature Map

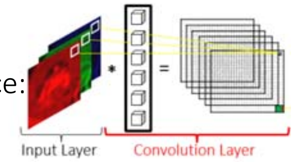
| | | | |
|---|---|---|---|
| 44 | 187 | 433 | 606 |
| 378 | | | |
| | | | |
| | | | |

185 + 207 + 11 + 0 + 0 + 0 – 33 – 2 – 0 + 10 = 378

22

## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

+ Bias 10 =

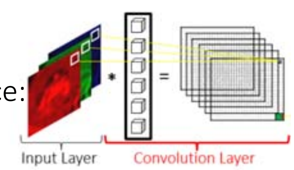
Feature Map

| | | | |
|---|---|---|---|
| 44 | 187 | 433 | 606 |
| 378 | 303 | 191 | 426 |
| 417 | 356 | 204 | 44 |
| -382 | -207 | -250 | -448 |

23

## Convolutional Layers

Single Input Channel: 6 * 6

Reference: 


| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Stride = 1

Filter: 3 * 3

| | | |
|---|---|---|
| w1 | w2 | w3 |
| w4 | w5 | w6 |
| w7 | w8 | w9 |

+ Bias $\beta$ =

Feature Map

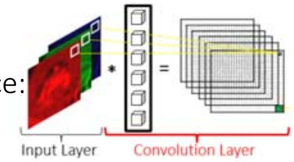
| | | | |
|---|---|---|---|
| 44 | 187 | 433 | 606 |
| 378 | 303 | 191 | 426 |
| 417 | 356 | 204 | 44 |
| -382 | -207 | -250 | -448 |

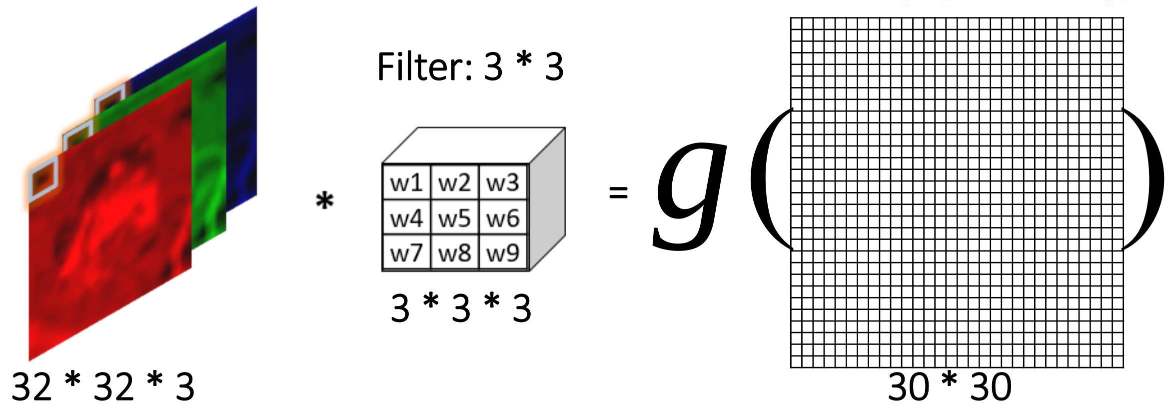
24

## Convolutional Layers

Three Channel Input:

Reference:




25

# Lecture: CNN

## Padding, Pooling, Fully Connected and Output Layers

Dr. Goutam Chakraborty

**SAS® Professor of Marketing Analytics**

**Director of MS in Business Analytics and Data Science*** (http://analytics.okstate.edu/mban/ )
**Director of Graduate Certificate in Business Data Mining** (http://analytics.okstate.edu/certificate/grad-data-mining/ )
**Director of Graduate Certificate in Marketing Analytics** (http://analytics.okstate.edu/certificate/grad-marketing-analytics/ )

- *Name change pending internal approval.
- 
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

26

## Outline

- Padding in convolutional layer
- Pooling layer
- Fully connected layer
- Output layer



## Padding

- Padding increases the relevance of pixels existing on the edge of an image.
- It retains information that would otherwise be disregarded.
- Padding is calculated with the goal of producing an output image in a size that will be "reasonable" based on the original input image size and filter size.
  - The key issues are whether the input image dimensions are even or odd, whether the filter dimensions are even or odd, and the value of the user-specified stride.

## Reasonable Padding Rules

- The horizontal and vertical padding sizes are independent of each other. Changing the horizontal dimension of an image or filter has no effect on the vertical padding size.
- If the stride is 1, then the output image size is the same as the input image size. When discussing convolutions, this is sometimes referred to as same padding.
- If the stride is 2, then the output image size is about one half the input image size.
- Padding is first added to the right side of the image and then the left. So if padding is unequal, the right side will have more padding than the left.
- Images are padded with zeros.

29



## Padding

Single Input Channel: 6 * 6

| | | | | | |
|---|---|---|---|---|---|
| 245 | 97 | 239 | 187 | 195 | 254 |
| 185 | 207 | 11 | 77 | 93 | 246 |
| 212 | 175 | 160 | 11 | 27 | 2 |
| 33 | 2 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 4 | 2 |
| 226 | 152 | 49 | 18 | 193 | 247 |

*

Without Padding

Filter: 3 * 3

| | | |
|---|---|---|
| w1 | w2 | w3 |
| w4 | w5 | w6 |
| w7 | w8 | w9 |

Stride = 1

=

Feature Map: 4 * 4

| | | | |
|---|---|---|---|
| n | n | n | n |
| n | n | n | n |
| n | n | n | n |
| n | n | n | n |

30

# Padding

Single Input Channel: 6 * 6

With Padding

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 245 | 97 | 239 | 187 | 195 | 254 | 0 |
| 0 | 185 | 207 | 11 | 77 | 93 | 246 | 0 |
| 0 | 212 | 175 | 160 | 11 | 27 | 2 | 0 |
| 0 | 33 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 140 | 0 | 0 | 0 | 4 | 2 | 0 |
| 0 | 226 | 152 | 49 | 18 | 193 | 247 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*

Filter: 3 * 3

| | | |
|---|---|---|
| w1 | w2 | w3 |
| w4 | w5 | w6 |
| w7 | w8 | w9 |

Stride = 1

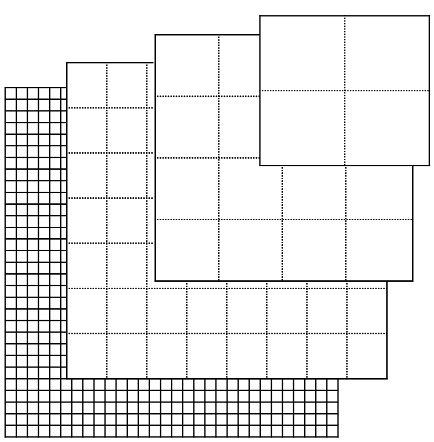
=

Feature Map: 6 * 6

| | | | | | |
|---|---|---|---|---|---|
| n | n | n | n | n | n |
| n | n | n | n | n | n |
| n | n | n | n | n | n |
| n | n | n | n | n | n |
| n | n | n | n | n | n |
| n | n | n | n | n | n |



# Feature Map Dimensions



Output

$$\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$$

X

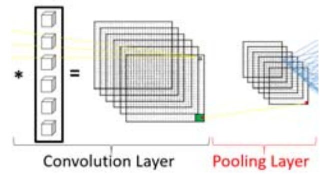
Output

$$\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$$

where

$n$ is the input dimension.
$p$ is the padding dimension.
$f$ is the filter dimension.
$s$ is the stride.

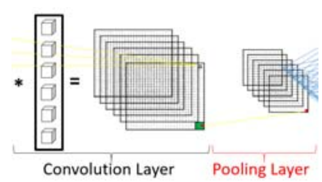
## Pooling Layers

Reference: 


- increase invariance of a convolutional neural network
- are essential for handling inputs of varying sizes
- provide a localized summary
  - MAX, AVERAGE, MIN
- improve computational efficiency.

33

## Pooling Layers

Reference: 


### Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

### Maximum Pooling

| | |
|---|---|
| ? | |
| | |

### Average Pooling

| | |
|---|---|
| ? | |
| | |

### Minimum Pooling

| | |
|---|---|
| ? | |
| | |

### Hyperparameters

Width = 2
Height = 2
Stride = 2

34

# Pooling Layers

Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

Maximum Pooling

| | |
|---|---|
| 368 | |
| | |

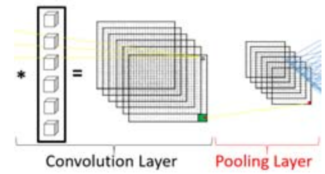
Average Pooling

| | |
|---|---|
| 216.5 | |
| | |

Minimum Pooling

| | |
|---|---|
| 34 | |
| | |

Reference:




Hyperparameters

Width = 2
Height = 2
Stride = 2

35

# Pooling Layers

Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

Maximum Pooling

| | |
|---|---|
| 368 | |
| | |

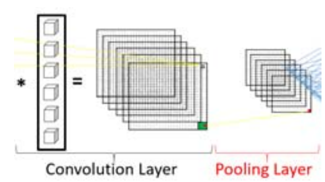
Average Pooling

| | |
|---|---|
| 216.5 | |
| | |

Minimum Pooling

| | |
|---|---|
| 34 | |
| | |

Reference:




Hyperparameters

Width = 2
Height = 2
Stride = 2

36

## Pooling Layers

### Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

### Maximum Pooling

| | |
|---|---|
| 368 | ? |
| | |

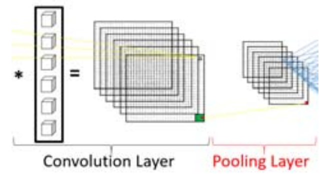
### Average Pooling

| | |
|---|---|
| 216.5 | ? |
| | |

### Minimum Pooling

| | |
|---|---|
| 34 | ? |
| | |

Reference:




### Hyperparameters

Width = 2
Height = 2
Stride = 2

37

## Pooling Layers

### Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

### Maximum Pooling

| | |
|---|---|
| 368 | 519 |
| | |

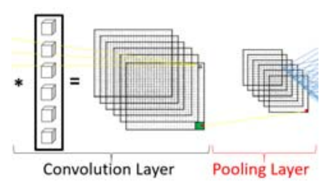
### Average Pooling

| | |
|---|---|
| 216.5 | 383.5 |
| | |

### Minimum Pooling

| | |
|---|---|
| 34 | 181 |
| | |

Reference:



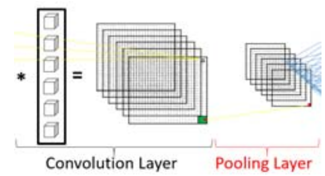

### Hyperparameters

Width = 2
Height = 2
Stride = 2

38

# Pooling Layers

Reference:




## Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

## Maximum Pooling

| | |
|---|---|
| 368 | 519 |
| 407 | |

## Average Pooling

| | |
|---|---|
| 216.5 | 383.5 |
| 35.8 | |

## Minimum Pooling

| | |
|---|---|
| 34 | 181 |
| -293 | |

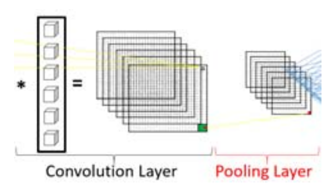
## Hyperparameters

Width = 2
Height = 2
Stride = 2

39

# Pooling Layers

Reference:




## Feature Map

| | | | |
|---|---|---|---|
| 34 | 171 | 418 | 519 |
| 368 | 293 | 181 | 416 |
| 407 | 346 | 198 | 40 |
| -393 | -217 | -260 | -485 |

## Maximum Pooling

| | |
|---|---|
| 368 | 519 |
| 407 | 198 |

## Average Pooling

| | |
|---|---|
| 216.5 | 383.5 |
| 35.8 | -126.8 |

## Minimum Pooling

| | |
|---|---|
| 34 | 181 |
| -293 | -485 |

## Hyperparameters

Width = 2
Height = 2
Stride = 2

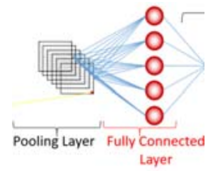
40

## Fully Connected Layer

A fully connected layer

- uses conventional layer-wise connections to map features to outputs via matrix multiplication
- incorporates a large number of parameters and therefore is expensive to train.

Reference:




41

## Output Layer

- The output layer uses a softmax activation function.
- Cross or relative entropy resolves to the Bernoulli error function when the target is binary:

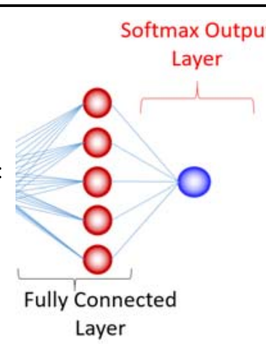
$$Q(\mathbf{w}) = -2\sum_{i}^{n}[\log(\hat{p}) + (1-y)\log(1-\hat{p})]$$

- Cross or relative entropy for more than two classes:

$$\sum_{i}^{n}\sum_{c}^{C} -y_{true}^{(c)} \log(\hat{p}_{predicted}^{(c)})$$

where $c$ is the class label for observation $i$.

Reference:




42

# Lecture: CNN
# Skip Layers and Architectural Design Strategies

Dr. Goutam Chakraborty

**SAS® Professor of Marketing Analytics**

**Director of MS in Business Analytics and Data Science*** (http://analytics.okstate.edu/mban/ )
**Director of Graduate Certificate in Business Data Mining** (http://analytics.okstate.edu/certificate/grad-data-mining/ )
**Director of Graduate Certificate in Marketing Analytics** (http://analytics.okstate.edu/certificate/grad-marketing-analytics/ )

- *Name change pending internal approval.
- 
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

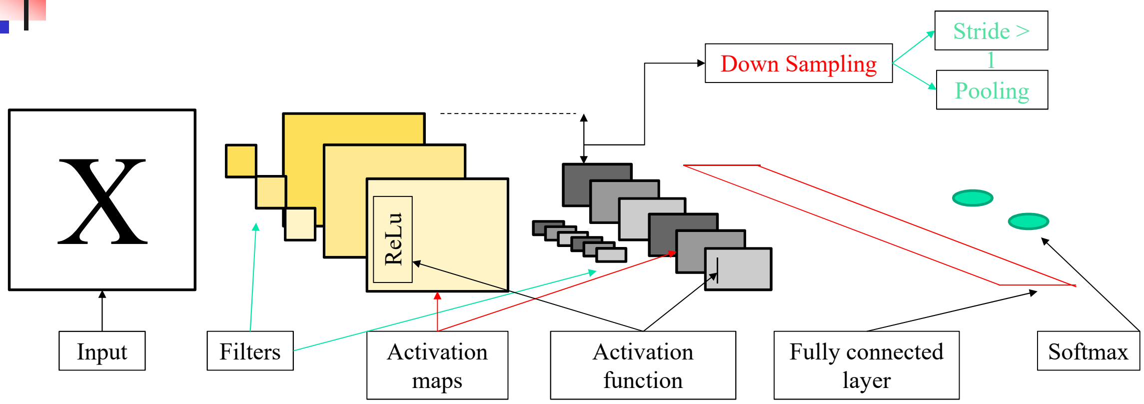
43

## Outline

- The end-to-end process of convolutional layers
- Skip Layers
  - Concatenation layer
  - Residual layer
- Architectural design strategies

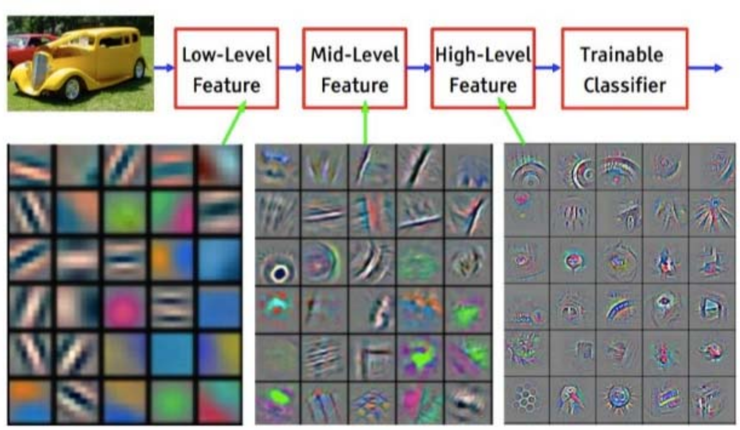
44

# The End-to-End Process of Convolution




45

# How Each Convolutional Layer 'Sees/Learns" an Image

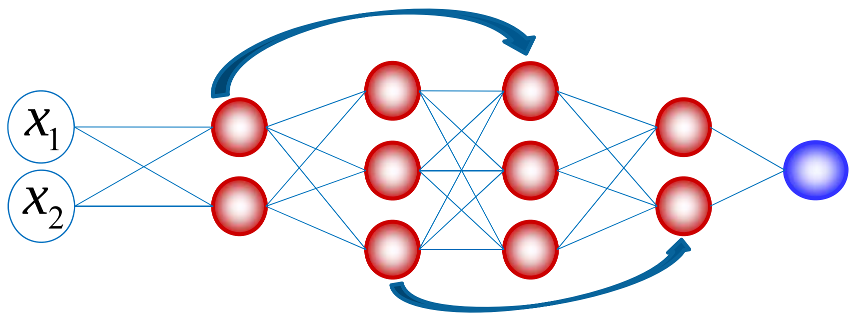



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

46

## Skip-Layer Connections

- Skip-layer connections help mitigate "forgotten/vanishing" gradients.
- They combine previously extracted feature with a current set of features.
- The concatenation layer and residual layer are used to create skip-layer connections.





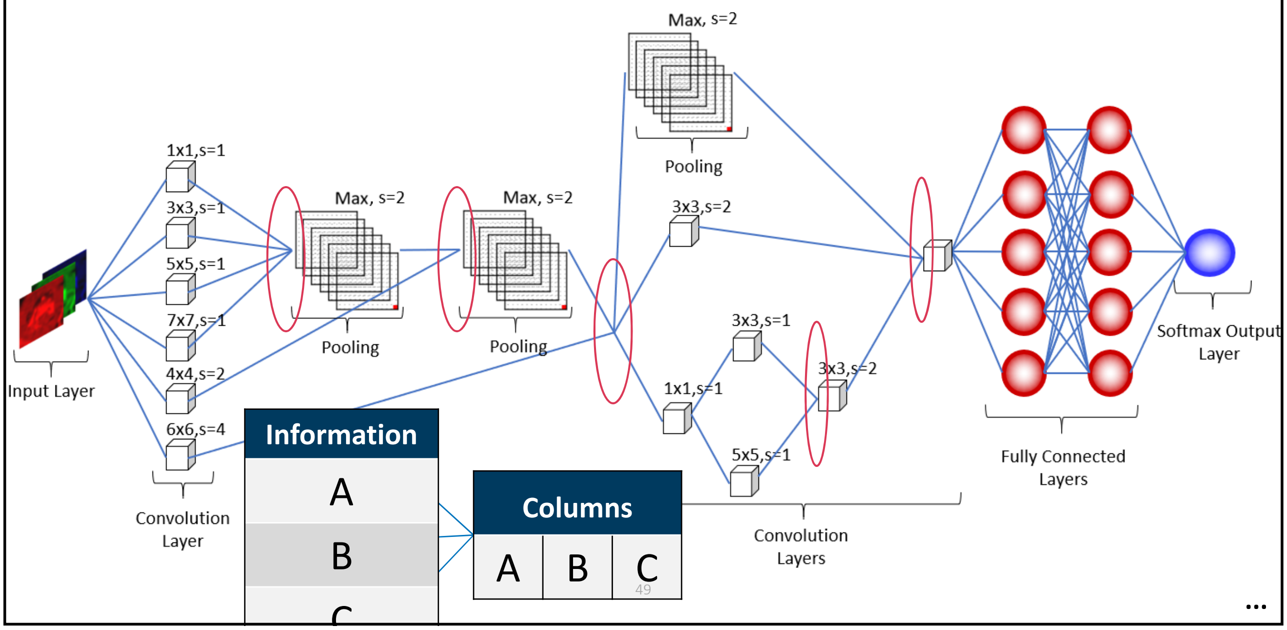

...

## Skip-Layer Connections

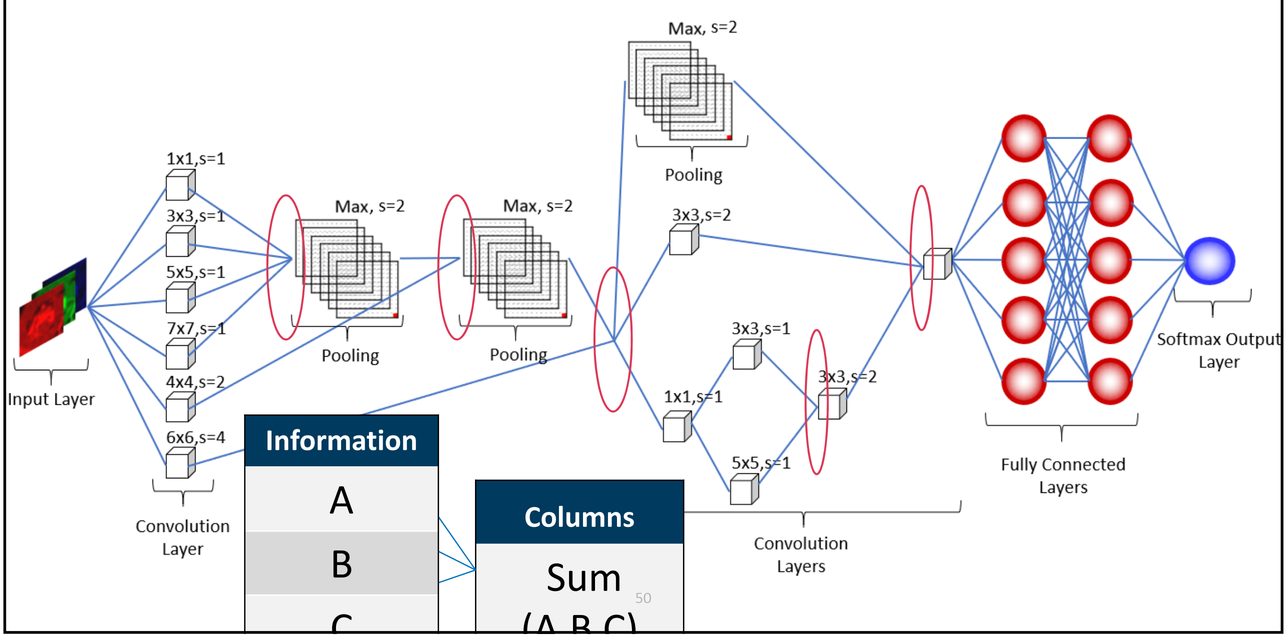
- Concatenation Layer
  - The concatenation layer combines multiple inputs by concatenating them along an axis.
  - This results in a "fatter" network.

- Residual Layer
  - The residual layer sums information through identity mapping.
  - This results in a "thinner" network.

## Concatenation Layers

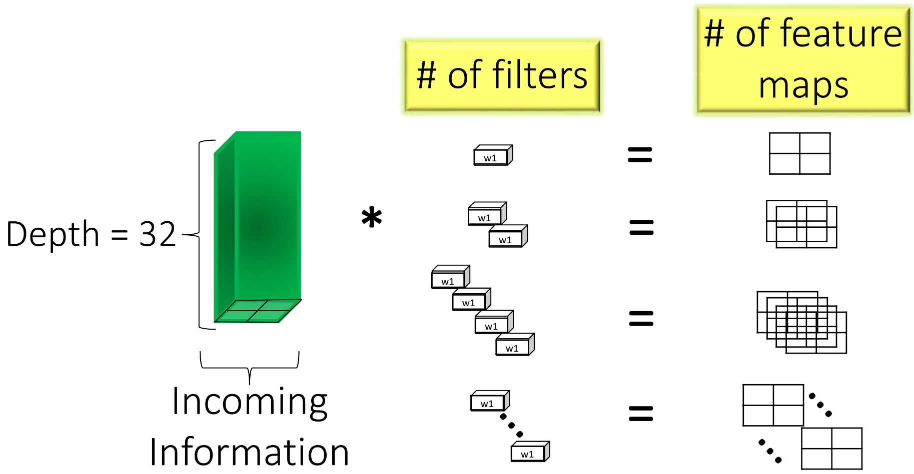



## Residual Layers



## One-by-One Convolutions

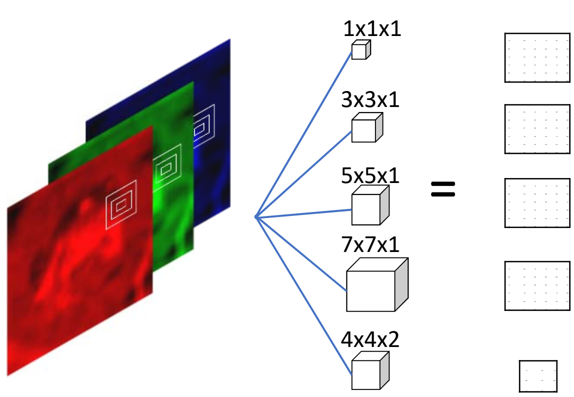
- Used to change the dimensional depth of the data



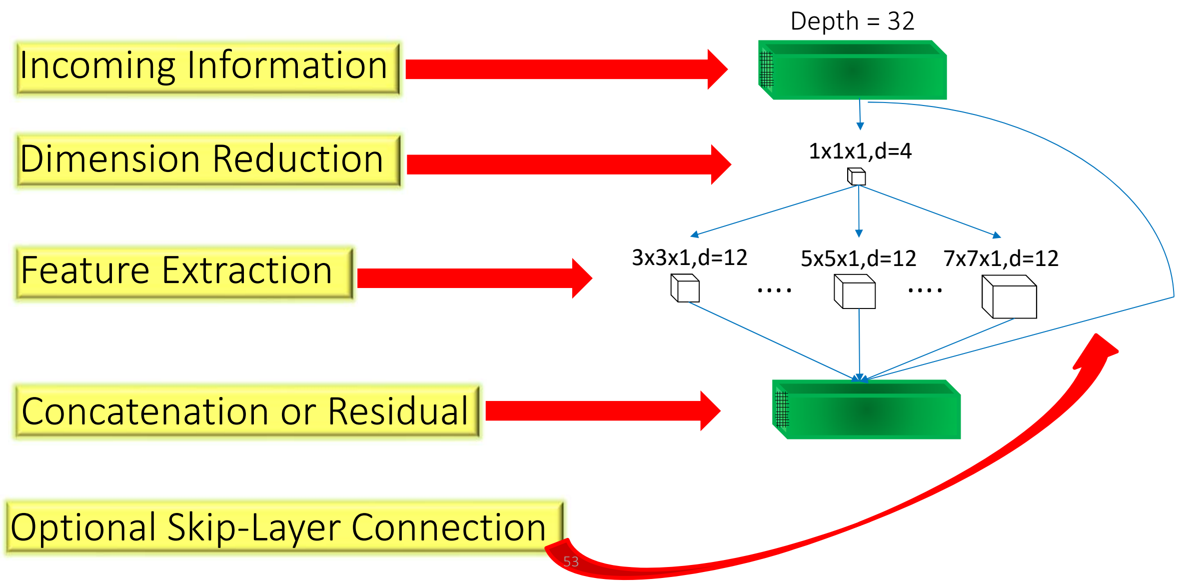



## Spatial Exploration

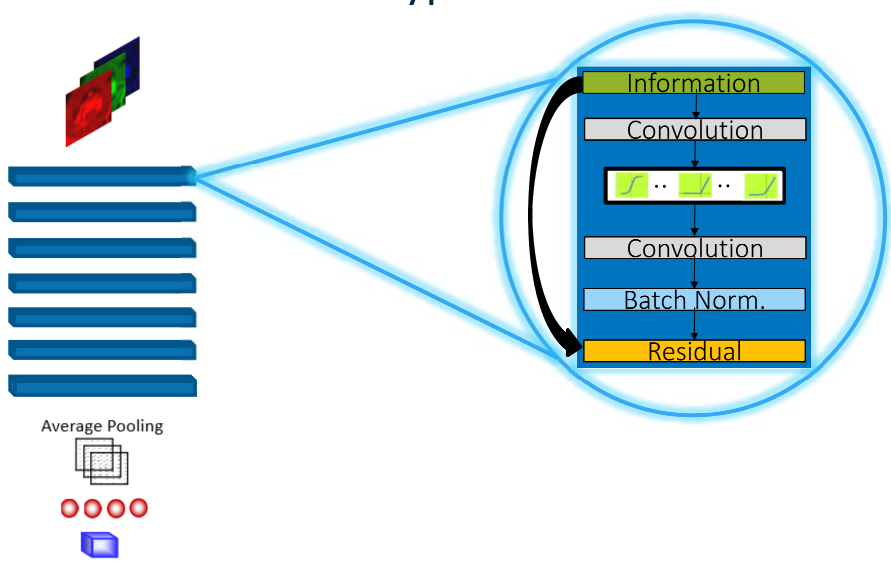
- Use different-sized filters to explore varying levels of granularity.
- Use different stride values to impact the exploration scheme.
  - The width and high of output feature maps are reduced by larger stride values.



## Creating Blocks

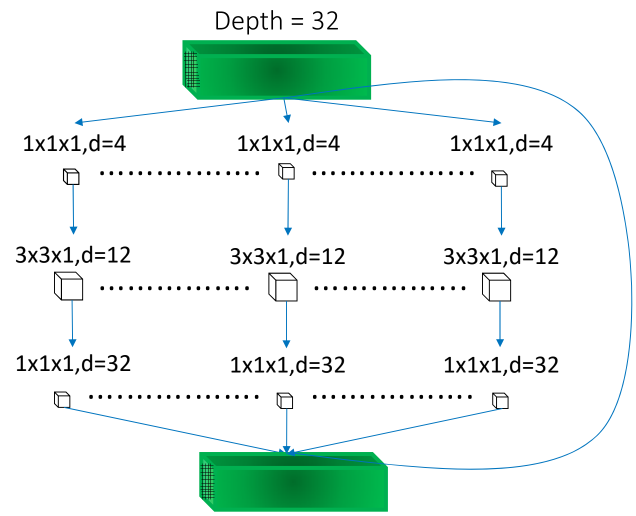



## ResNet Type



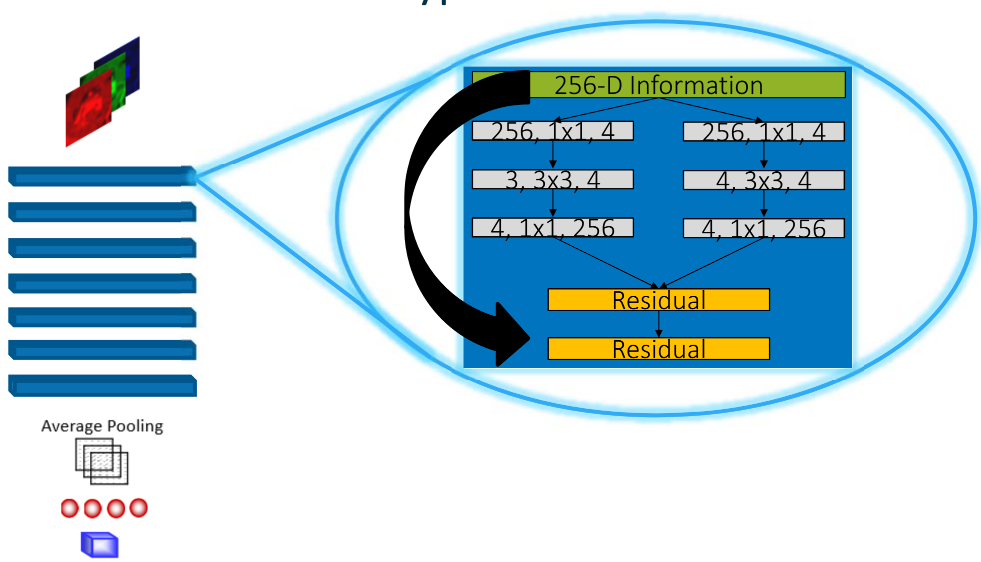
# Cardinality

- *Cardinality* is the number of transformation sets within a block.
  - It is an alternative to going deeper or wider for increased accuracy.
  - It can be expensive to train.
  - If skip layers are used, residual connections are recommended to reduce training cost.




55

# ResNeXt Type




56

57