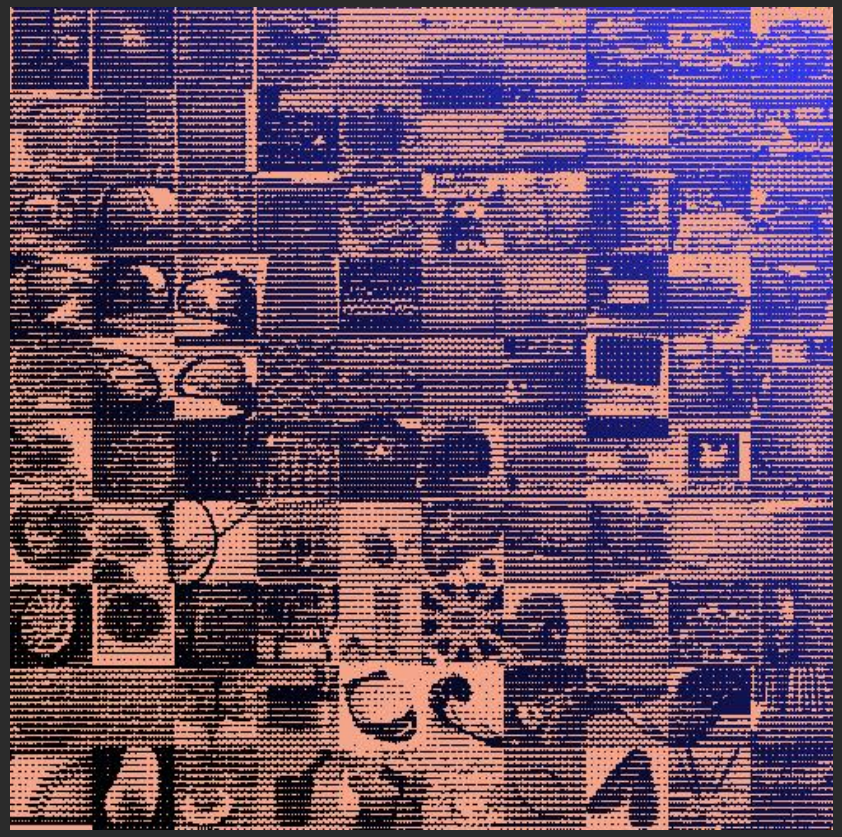
# Introduction to Machine Learning and Convolutions for Computer Vision

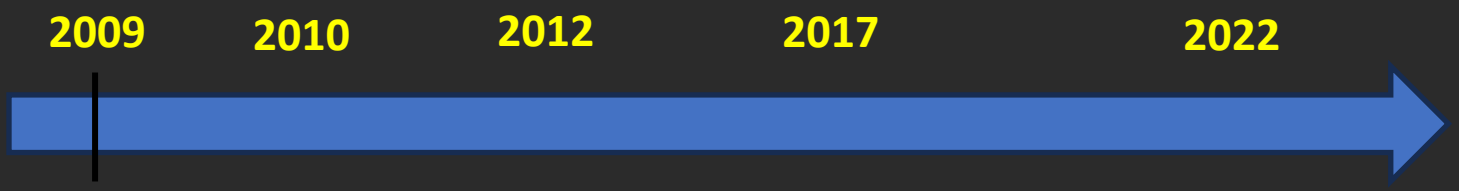
**Thomas Basikolo, ITU**

28 April 2024

# Unleashing Collective Intelligence



ImageNet




**IMAGENET**

ImageNet Challenge (ILSVRC)

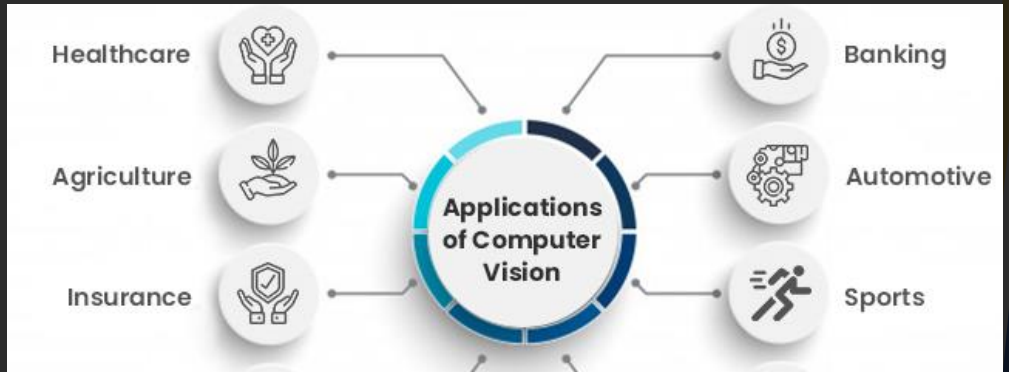
- ALEXNET (deep CNN) kickstarting the current boom in computer vision

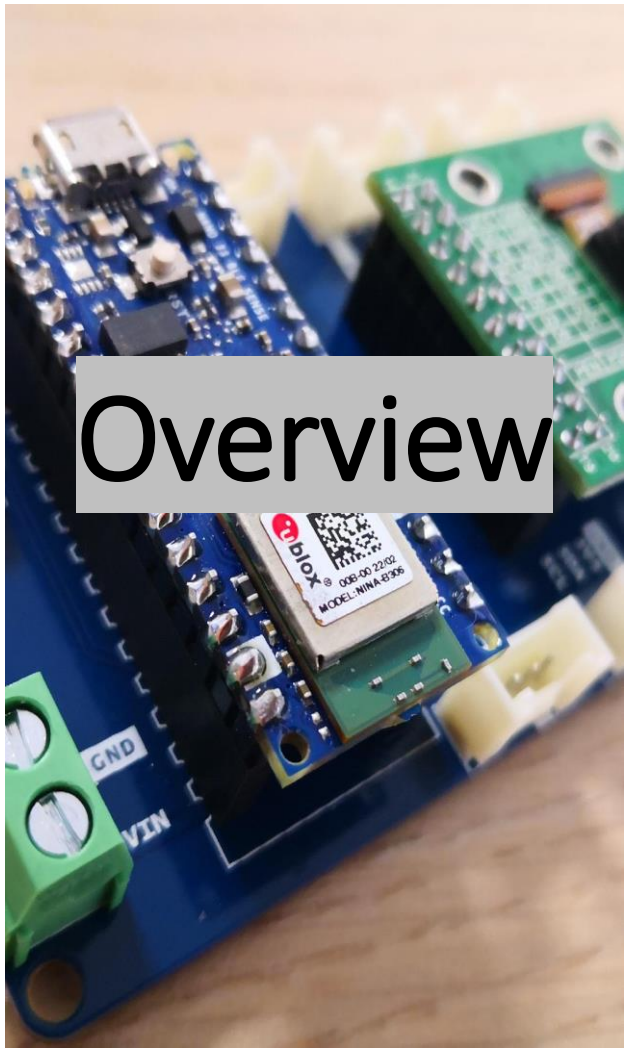
29 of 38 teams > 95% accuracy

**TRANSFORMER**

ChatGPT

GoogLeNet/Inception(2014), VGGNet (2014), ResNet(2015)






## 1. Machine Learning

- Overview of Machine Learning
- ML Models

## 2.Computer Vision

- Image Classification
- Object Detection

## 3. Convolutions

- ConvNets
- Padding & Strides
- Transfer Learning


Alexa, play some rock music!

Playing the Rock Hits playlist

# ElephantEdge

Building The World's Most Advanced **Wildlife Tracker**.

smart parks · hackster.io · EDGE IMPULSE · Microsoft · Western Digital. · NORDIC SEMICONDUCTOR · TAOGLAS. · ublox · WWF · EARTHRANGER

Source: https://wildlabs.net/resources/competition/challenge-elephantedge

# ElephantEdge

## Risk Monitoring

"Know when an elephant is moving into a high-risk area and send real-time notifications to park rangers."

## Conflict Monitoring

"Sense and alert when an elephant is heading into an area where farmers live."

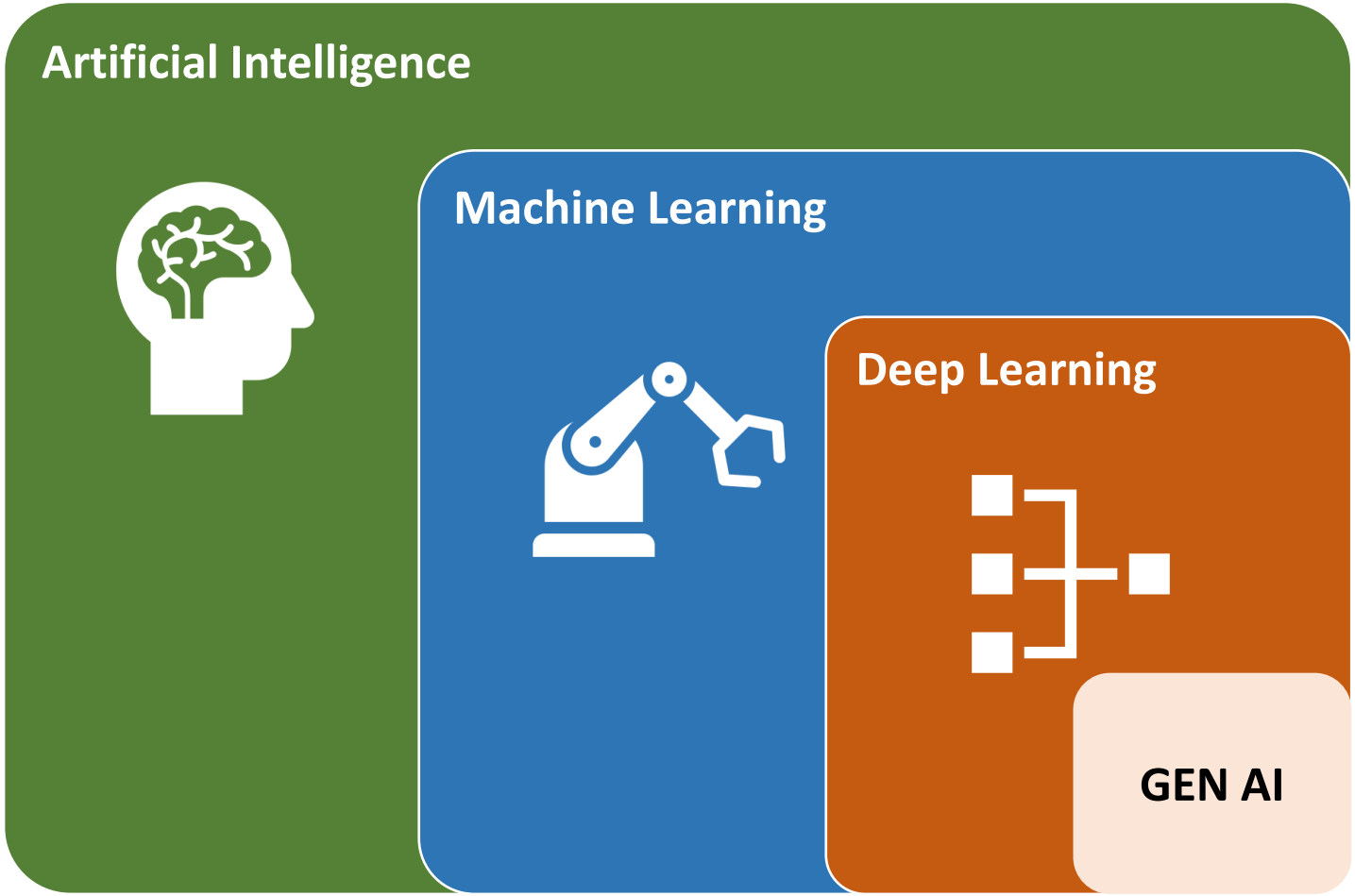
## Activity Monitoring

"Classify the general behavior of the elephant, such as when it is drinking, eating, sleeping, etc."

## Communication Monitoring

"Listen for vocal communications between elephants via the onboard microphone."

# AI vs. ML vs. DL




**Artificial intelligence (AI):** any technique that enables computers to mimic human intelligence.

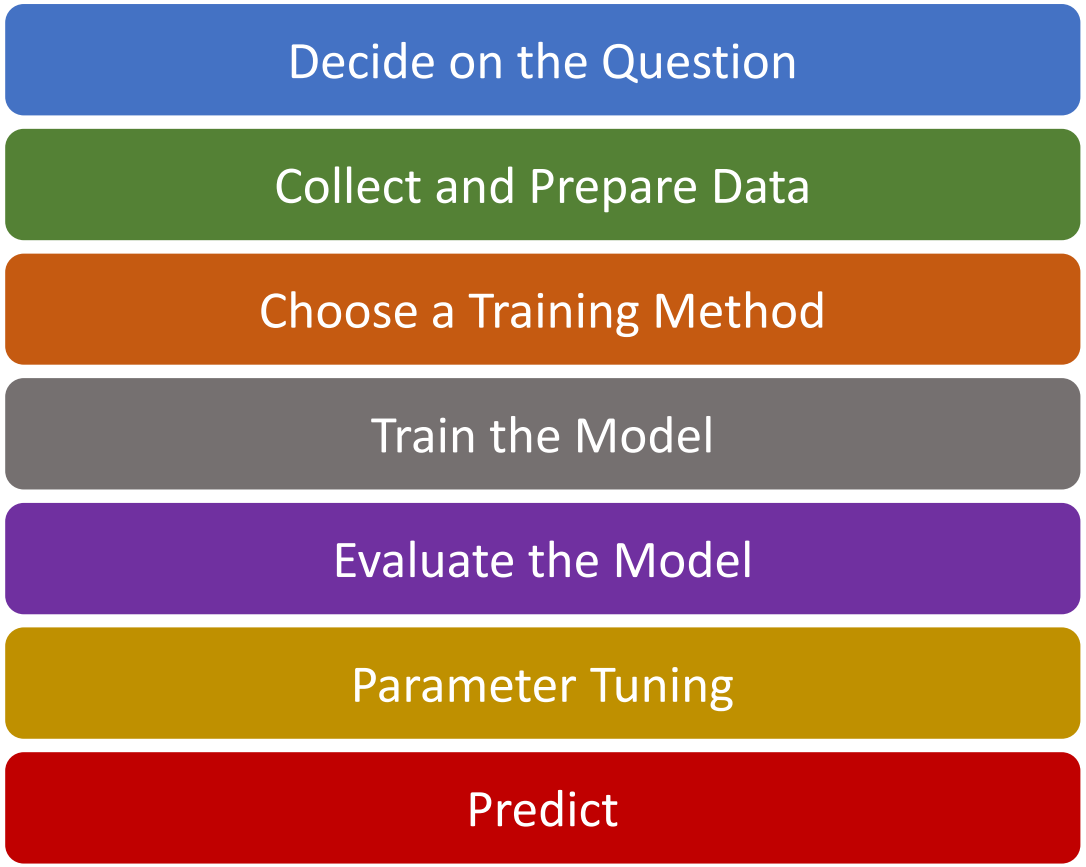
**Machine learning (ML):** a subset of AI that uses techniques that enable machines to use experience to improve at tasks.

**Deep learning (DL):** a subset of ML based on artificial neural networks (ANN). The learning process is deep because of its structure.

Based on https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

# General Steps for Machine Learning

On a high level, the craft of creating machine learning (ML) processes is comprised of several steps:

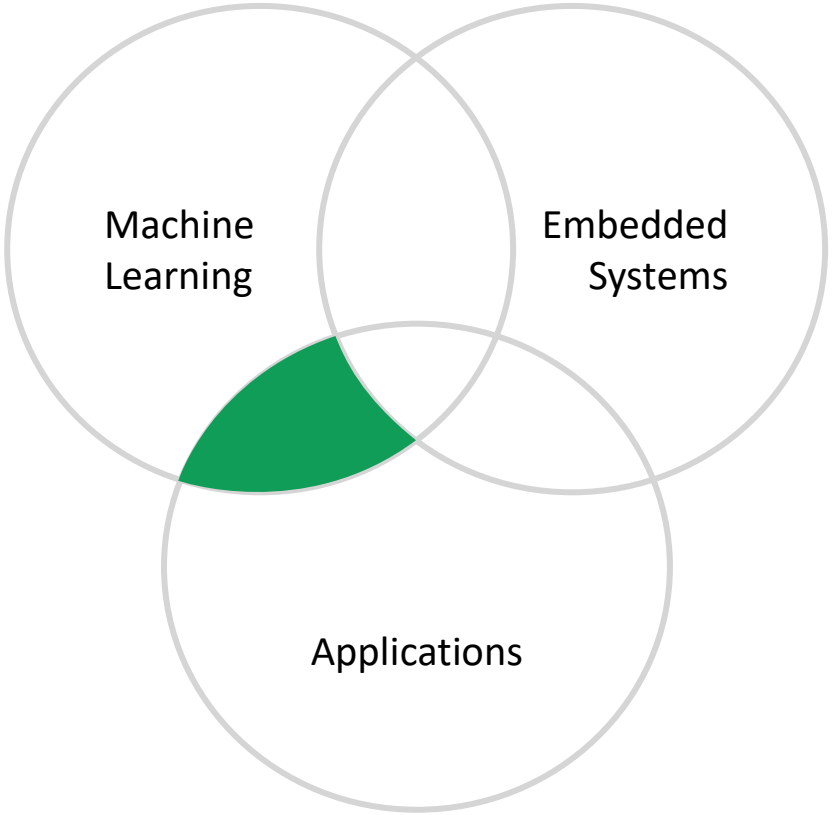



Based on https://microsoft.github.io/ML-For-Beginners/#/1-Introduction/4-techniques-of-ML/README?id=techniques-of-machine-learning

# 3 main components

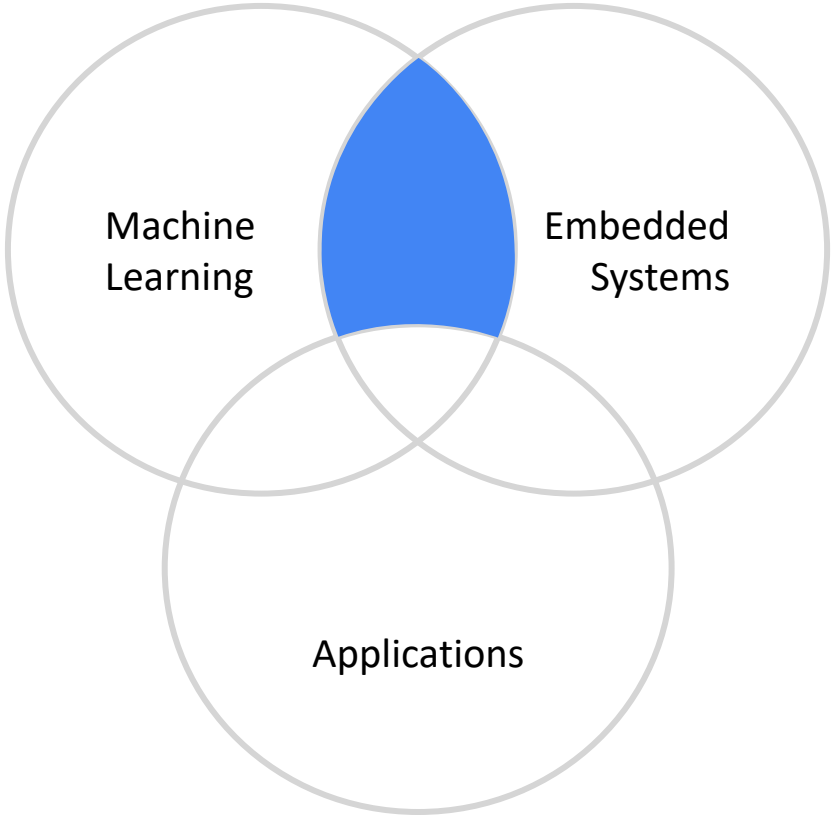
# Interactions

How **machine learning** can enable new and interesting **TinyML applications**?



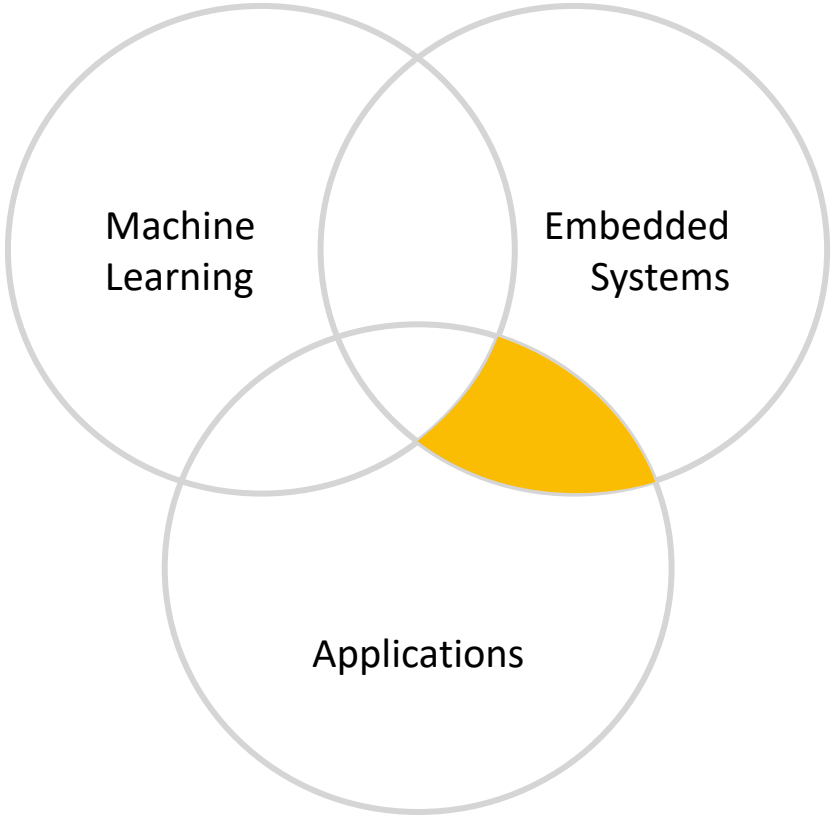
# Interactions

What are the **challenges** with enabling **machine learning** on **tiny**, resource-constrained **embedded devices**?



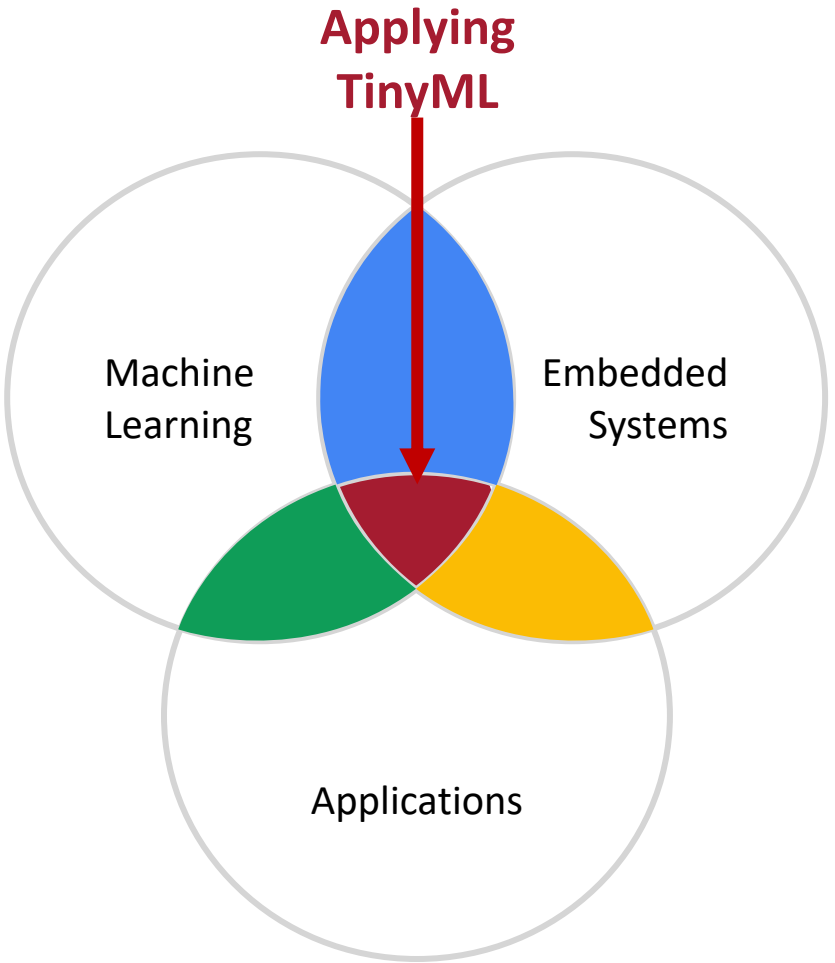
# Interactions

What type of new **use cases** can we possibly enable on **embedded systems** that we could not otherwise do before?



# Interactions

Given your understanding of things at these various intersections, you will have a deep understanding for **how to apply TinyML**



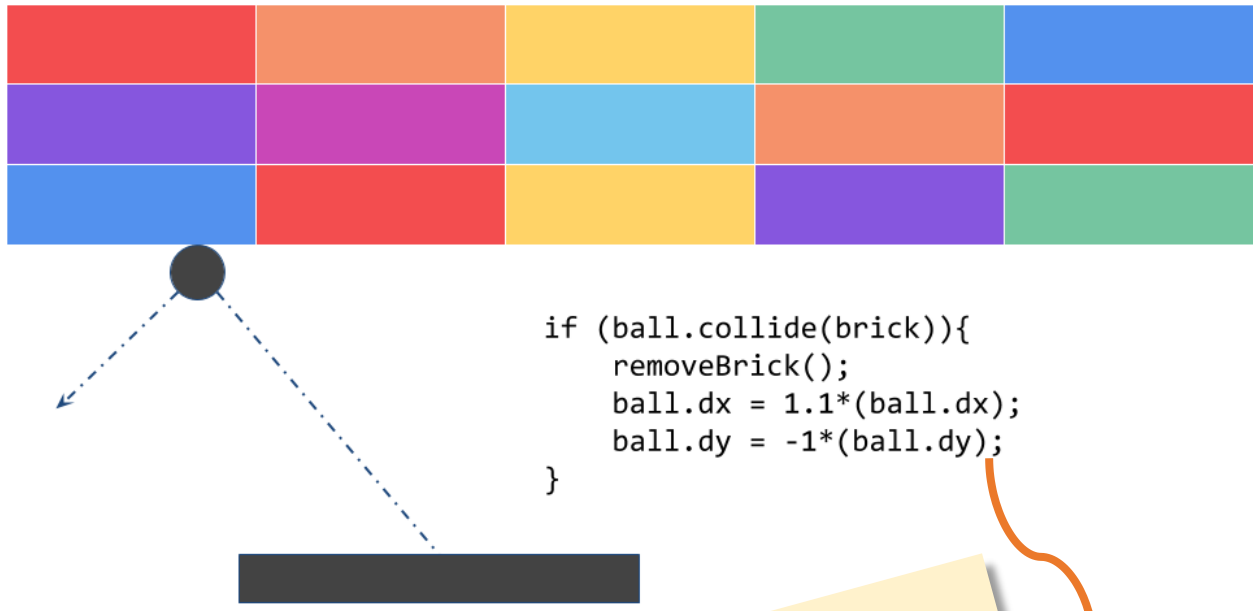
# We will run through this long process


This is a **first encounter with ML**, but many things will be left to be **experimented or developed.**

# The Machine Learning Paradigm

# Explicit Coding

- **Defining rules** that determine behavior of a program
- Everything is **pre-calculated and pre-determined** by the programmer
- **Scenarios are limited** by program complexity




https://en.wikipedia.org/wiki/Breakout_(video_game)

# The Traditional Programming Paradigm

$$y = f(x) \leftarrow x$$

# Consider Activity Detection

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// ???
```

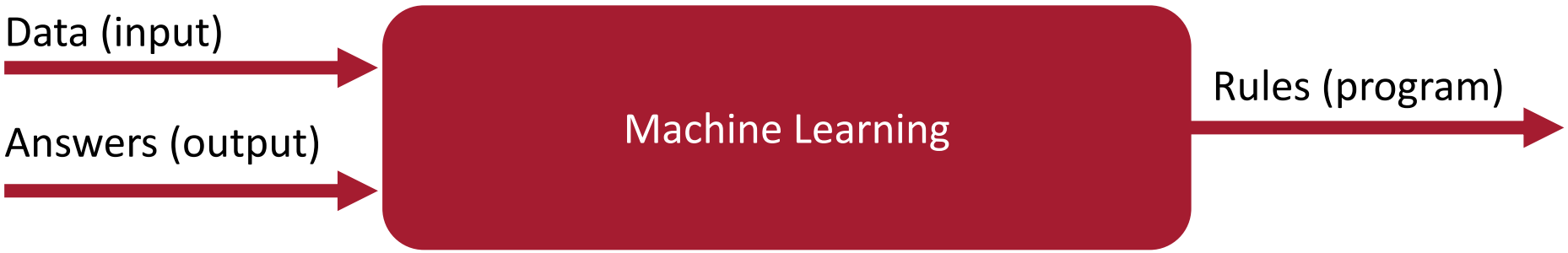
Way too complex to code!

# The Machine Learning Paradigm

Training set:

Input $x$ Output $y$

$$y = f(x)$$



# Activity Detection with Machine Learning

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING

# The Machine Learning Paradigm

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

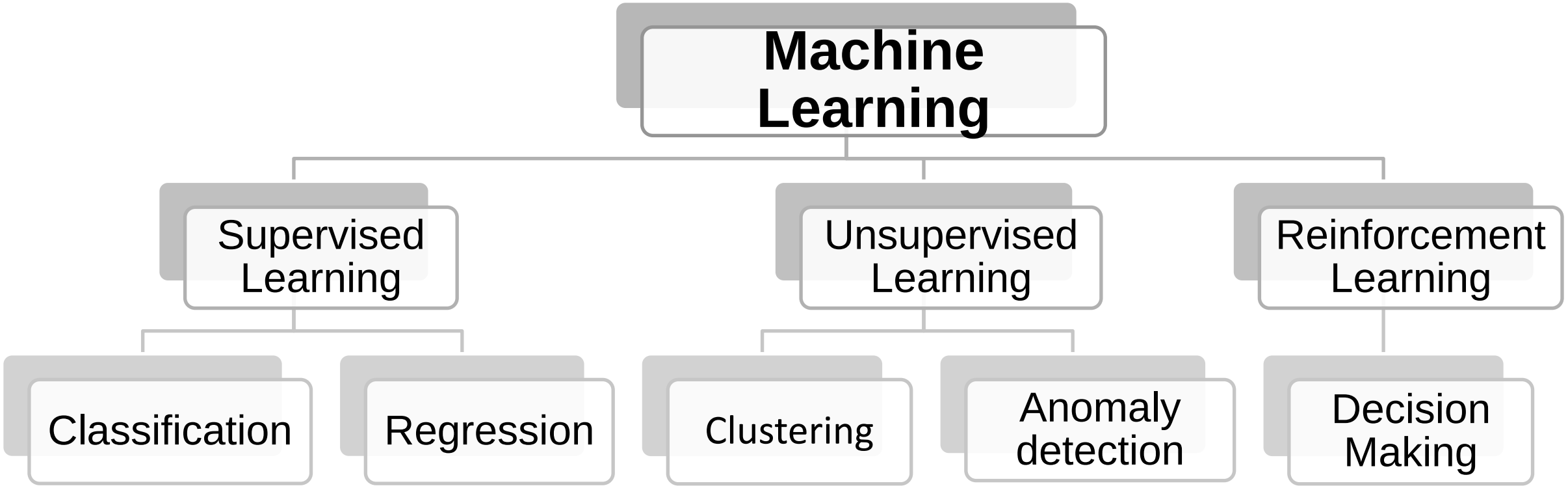
```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING

# Two Approaches



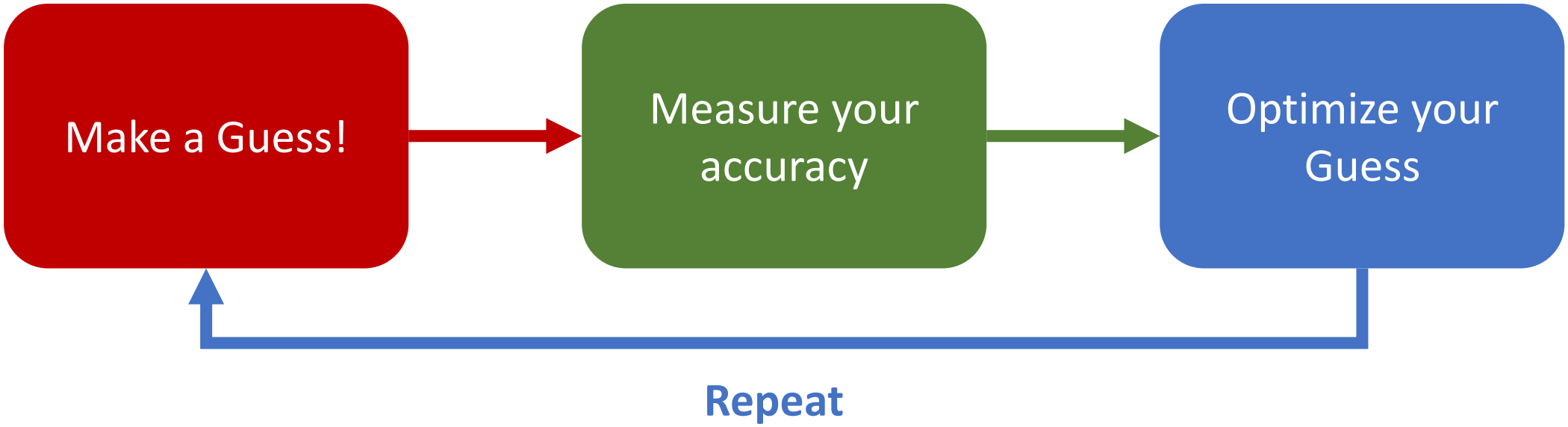

*Task driven*

*Data driven*

*Learning from mistakes*

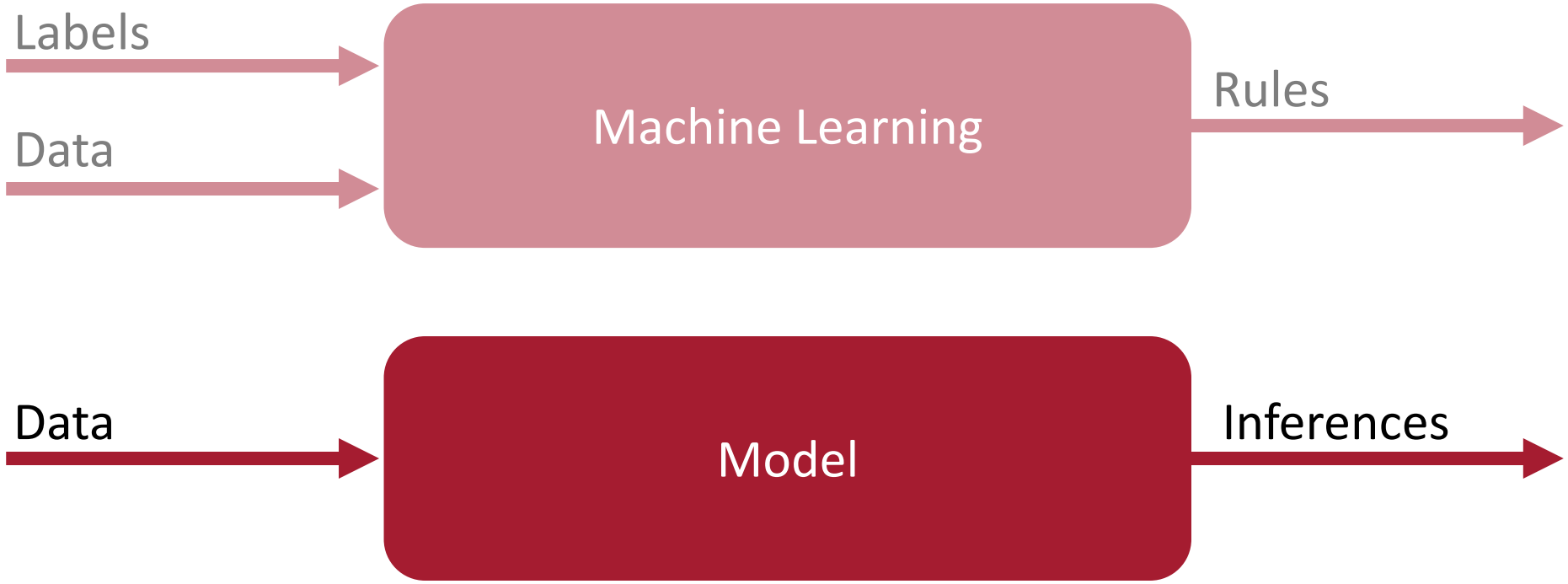
***this is not an exhaustive list*

# The Machine Learning Paradigm



# The Machine Learning Paradigm

# The Machine Learning Paradigm

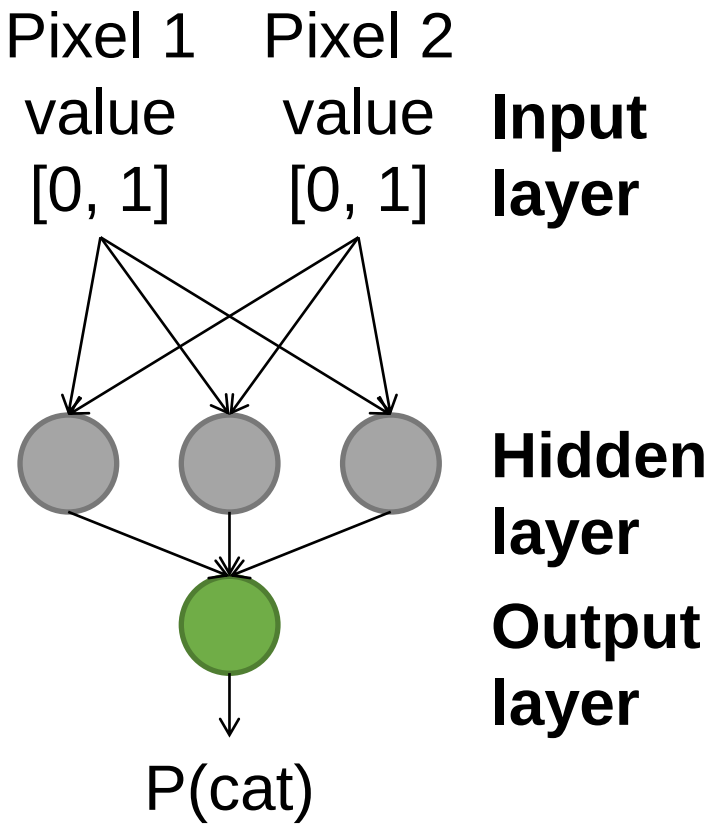


# Machine Learning Models

- **Logistic Regression**: determine if an input belongs to a certain group or not
- **SVM**: create coordinates for each object in an n-dimensional space and uses a hyperplane to group objects by common features
- **Decision Trees:** determine what category an input falls into by traversing the leafs and nodes of a tree
- **kNN:** grouping the closest objects in a dataset and finding the most frequent or average characteristics among the objects.
- **Random Forest**: collection of many decision trees from random subsets of the data, resulting in a combination of trees that may be more accurate in prediction than a single decision tree.
- **Boosting algorithms**: [Gradient Boosting Machine, XGBoost, and LightGBM] use ensemble learning. They combine the predictions from multiple algorithms (such as decision trees) while considering error from the previous algorithm.

# Neural Network (NN)

type of ML process that uses interconnected nodes or neurons in a layered structure that resembles the human brain



# How good is your model?

a way to measure your perfomance

# Performance Metrics in Machine Learning

- **Regression** tasks commonly use metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and $R^2$ (R-Squared).

$$MAE = \frac{1}{n}\sum|y - \hat{y}|$$

- **Classification** tasks use metrics like Accuracy, Confusion Matrix, Precision and Recall, and F1-score

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy - ratio of the correctly predicted instances to the total instances in the dataset

# Confusion Matrix

- **Four Quadrant Measurement on "Performance" of a model.**

Number correctly predicted not as the class (e.g., not a dog)

Number incorrectly predicted as the class (e.g. dog), when it is not that class.

| | Predicted (False) | Predicted (True) |
|---|---|---|
| Actual (False) | True Negative (TN) | False Positive (FP) |
| Actual (True) | False Negative (FN) | True Positive (TP) |

Number incorrectly predicted as not the class (e.g. not a dog)

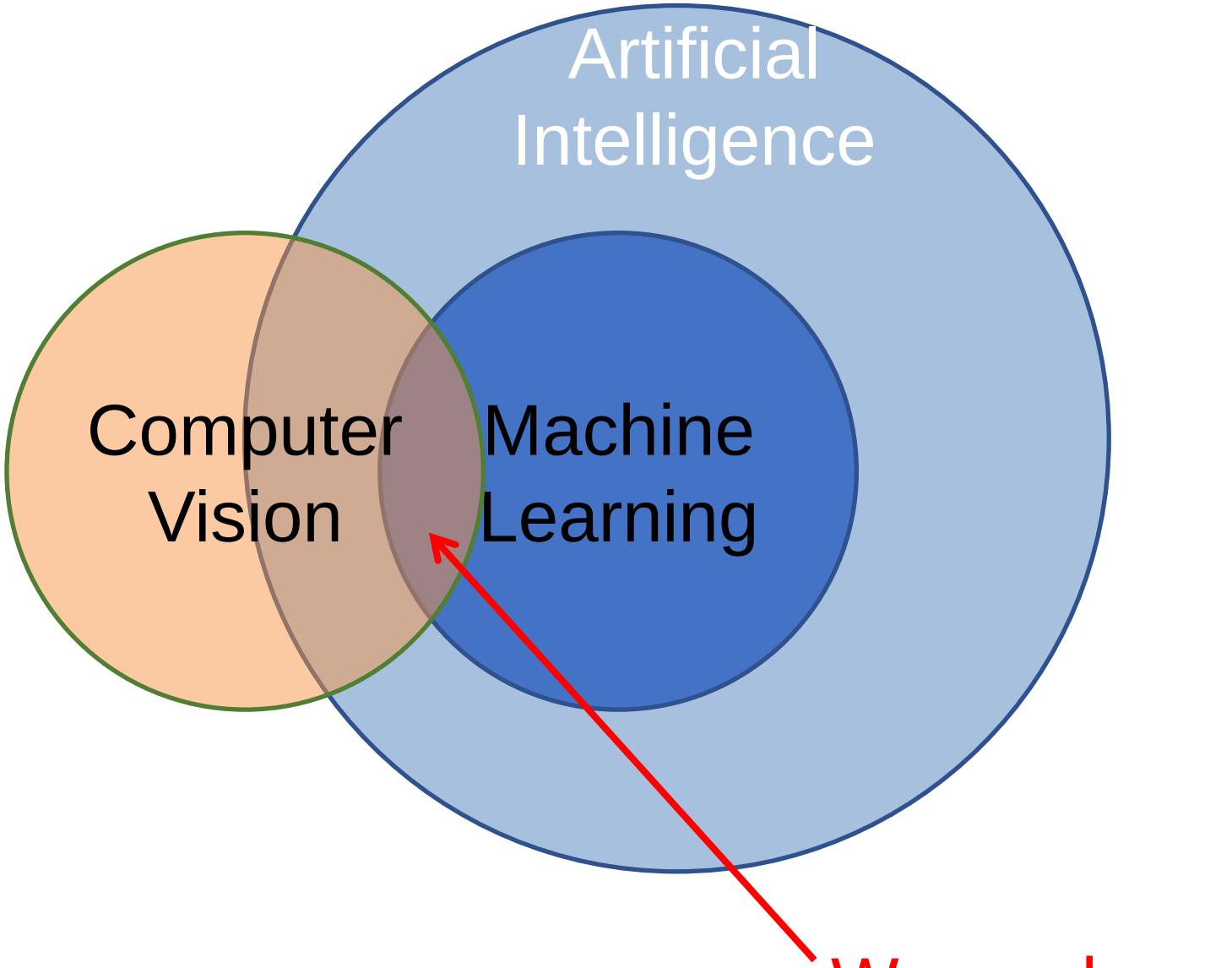
Number correctly predicted as the class (e.g., dog)

- **Accuracy = ( TP + TN ) / N**
- **Misclassification = ( FP + FN ) / N**
- **Precision = TP / ( TP + FP )**

Example Measurements

# Computer Vision

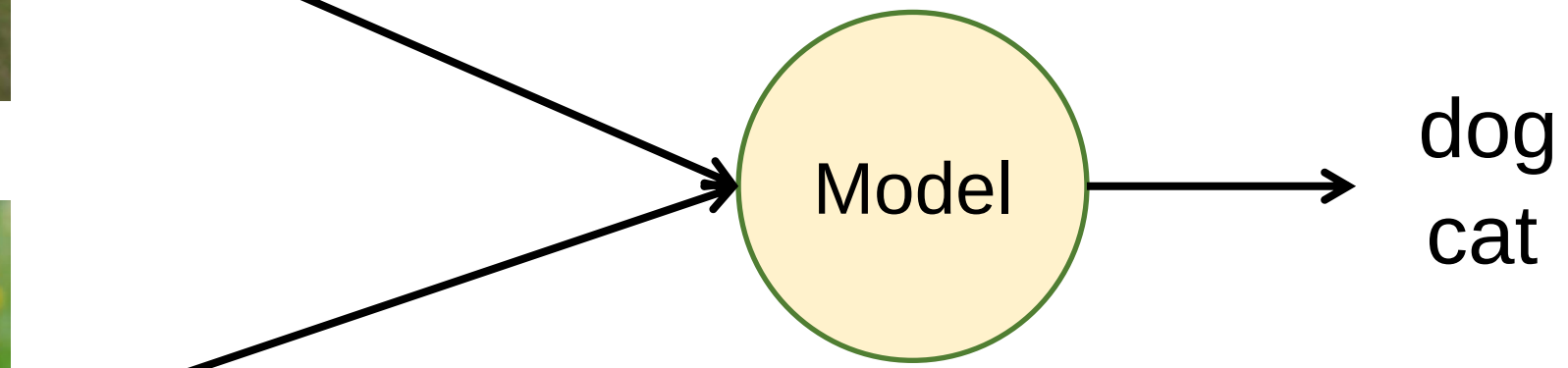
- Image Processing
- Pattern Recognition
- Signal processing
- Physics
- Mathematics
- AI (ML/DL)



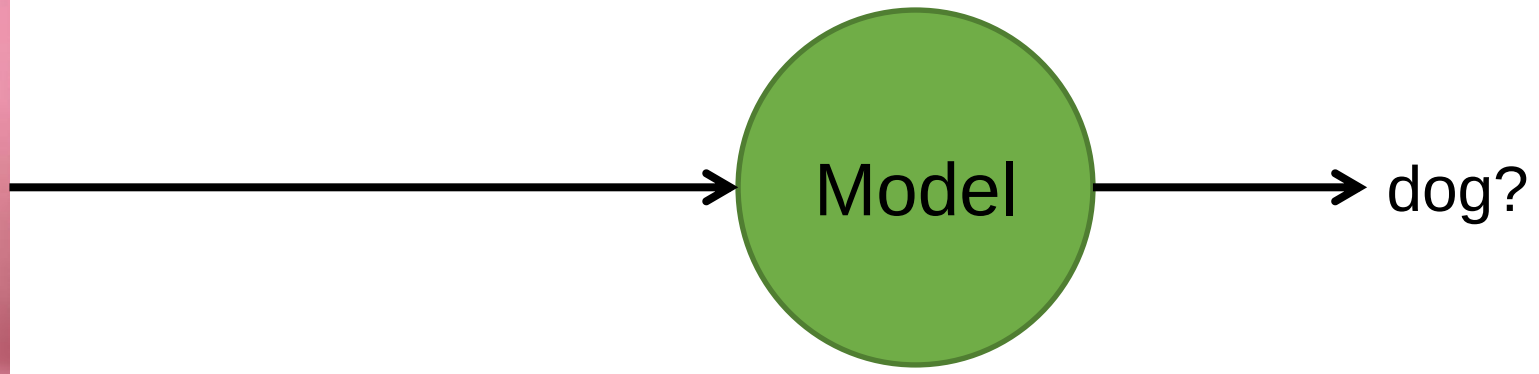
# Definitions

- **Image Classification**: Predict class of object in an image (comprehend an image)
- **Object Localization**: Locate presence of object(s) in an image
- **Object Detection**: Locate and classify object(s) in an image

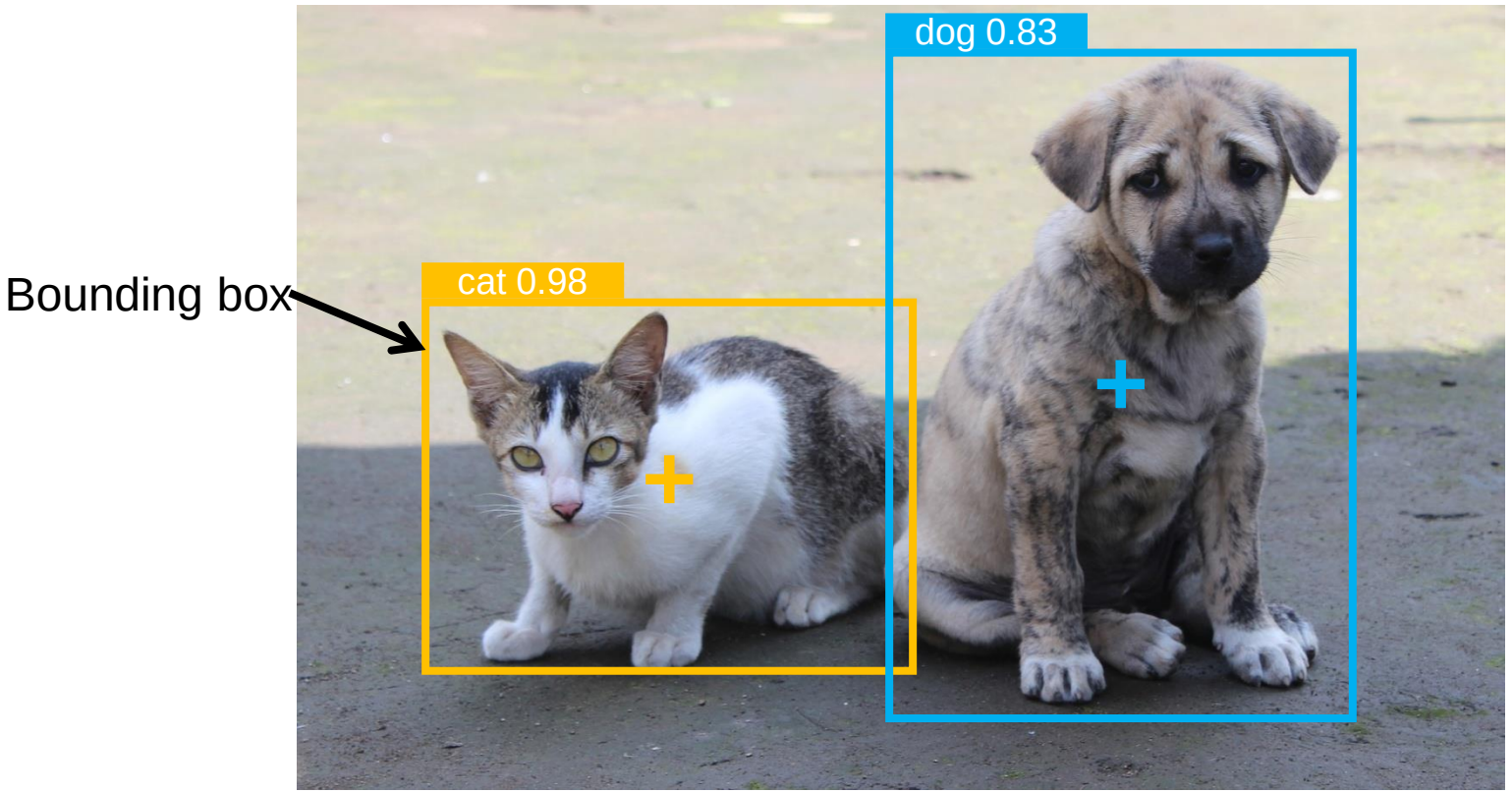
# Image Classification



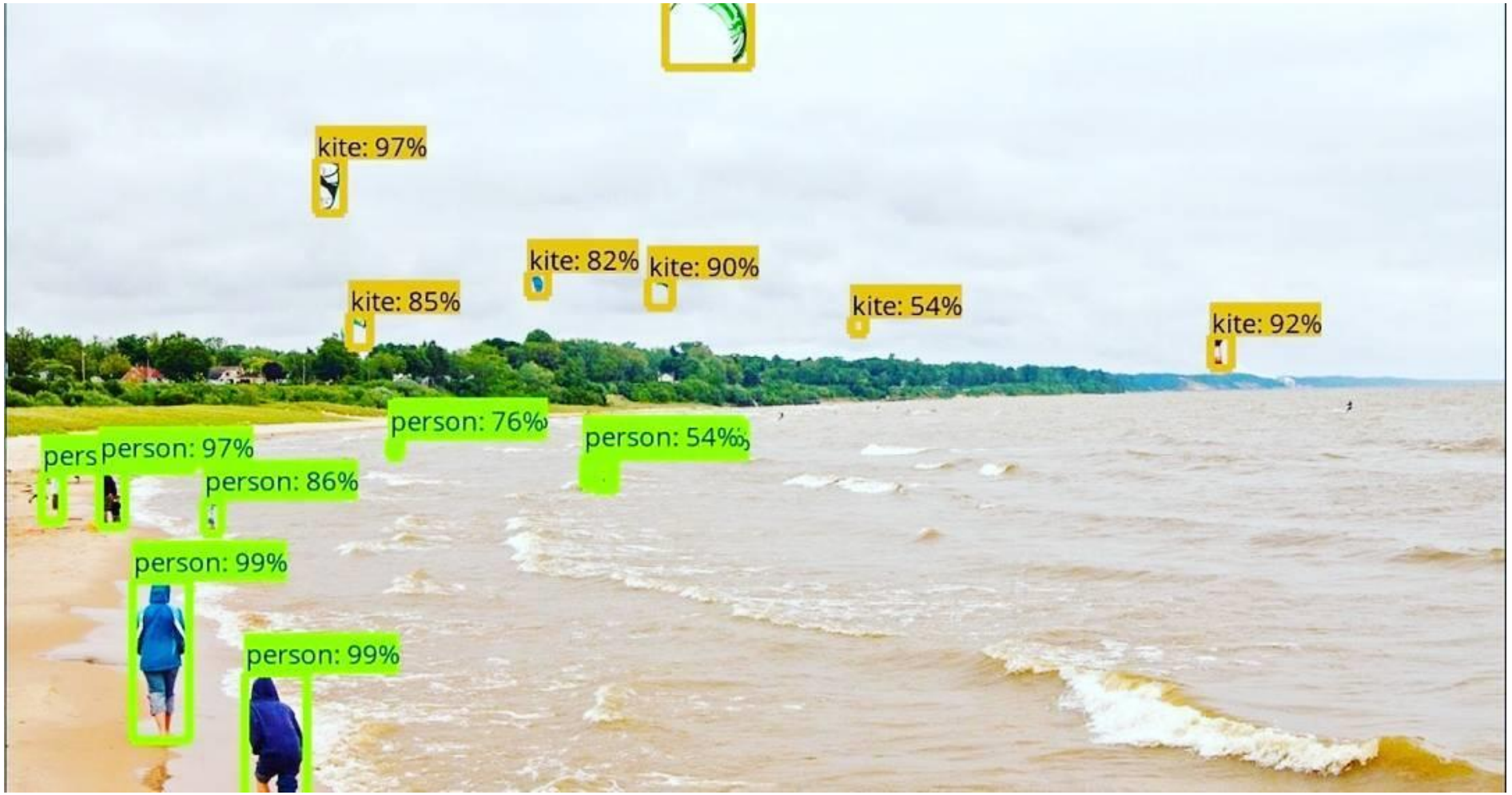
# Image Classification



# Object Localization



# Object Detection



# Rationale for using tinyML

"OpenMV H7 Camera", by SparkFun Electronics (CC BY 2.0)

240 x 240 x 8 x 30 x 1 = 13.824 Mbps

Number of pixels

Bits per pixel

Num. cameras

Frames per second

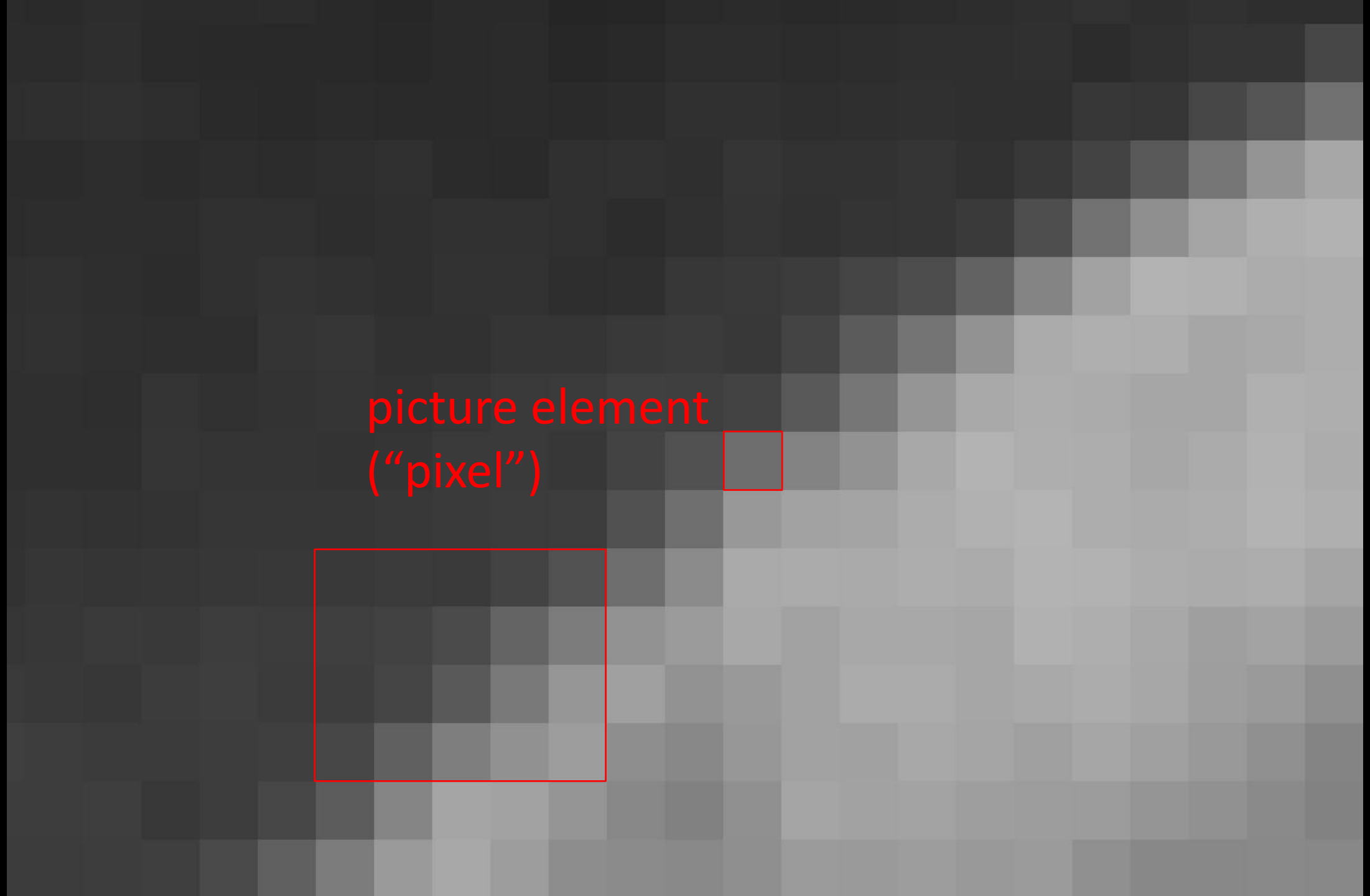
1 x 30 x 1 = 30 bps

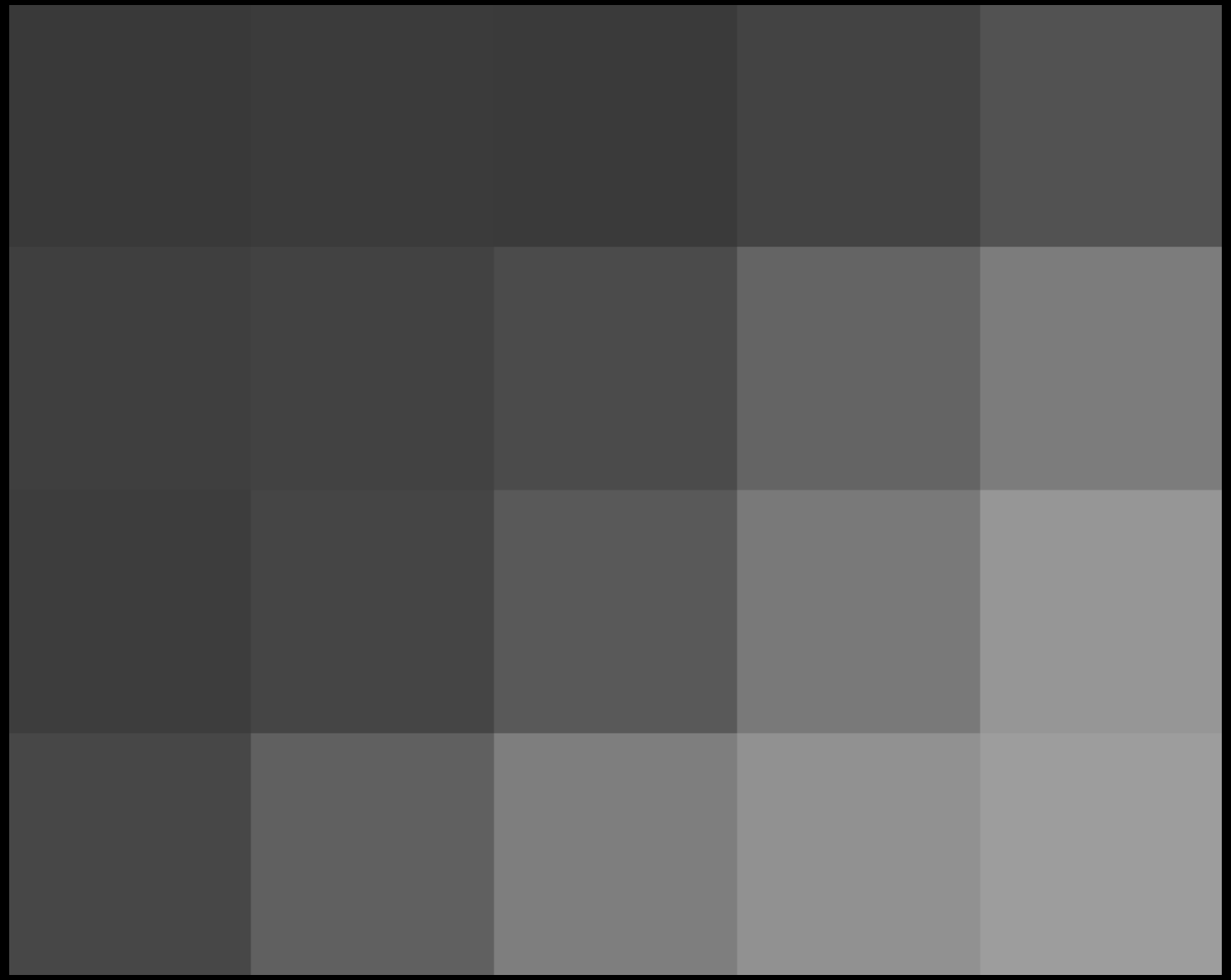
Person or not person

Num. cameras

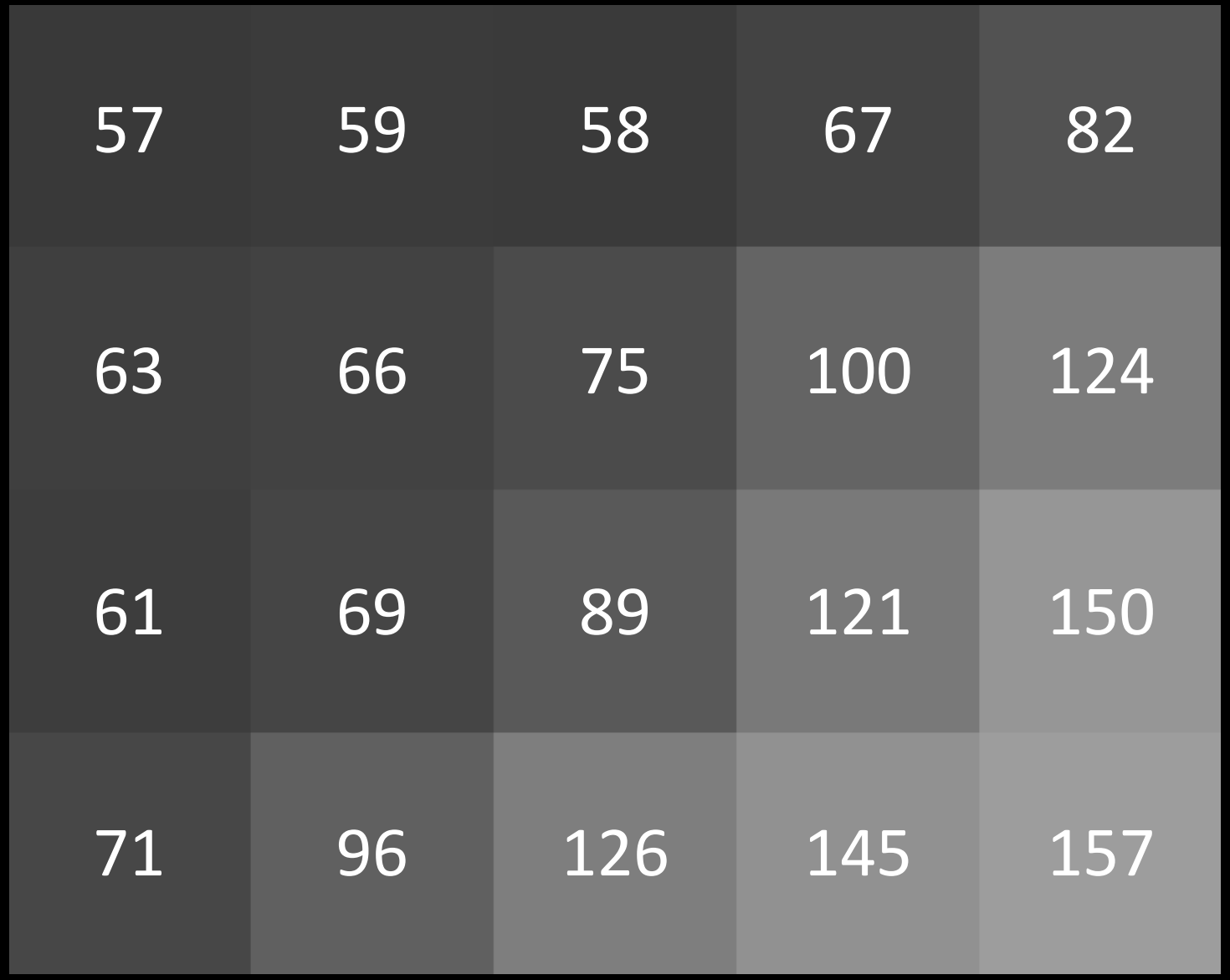
Frames per second

# Overview of Digital Images


picture element
(“pixel”)

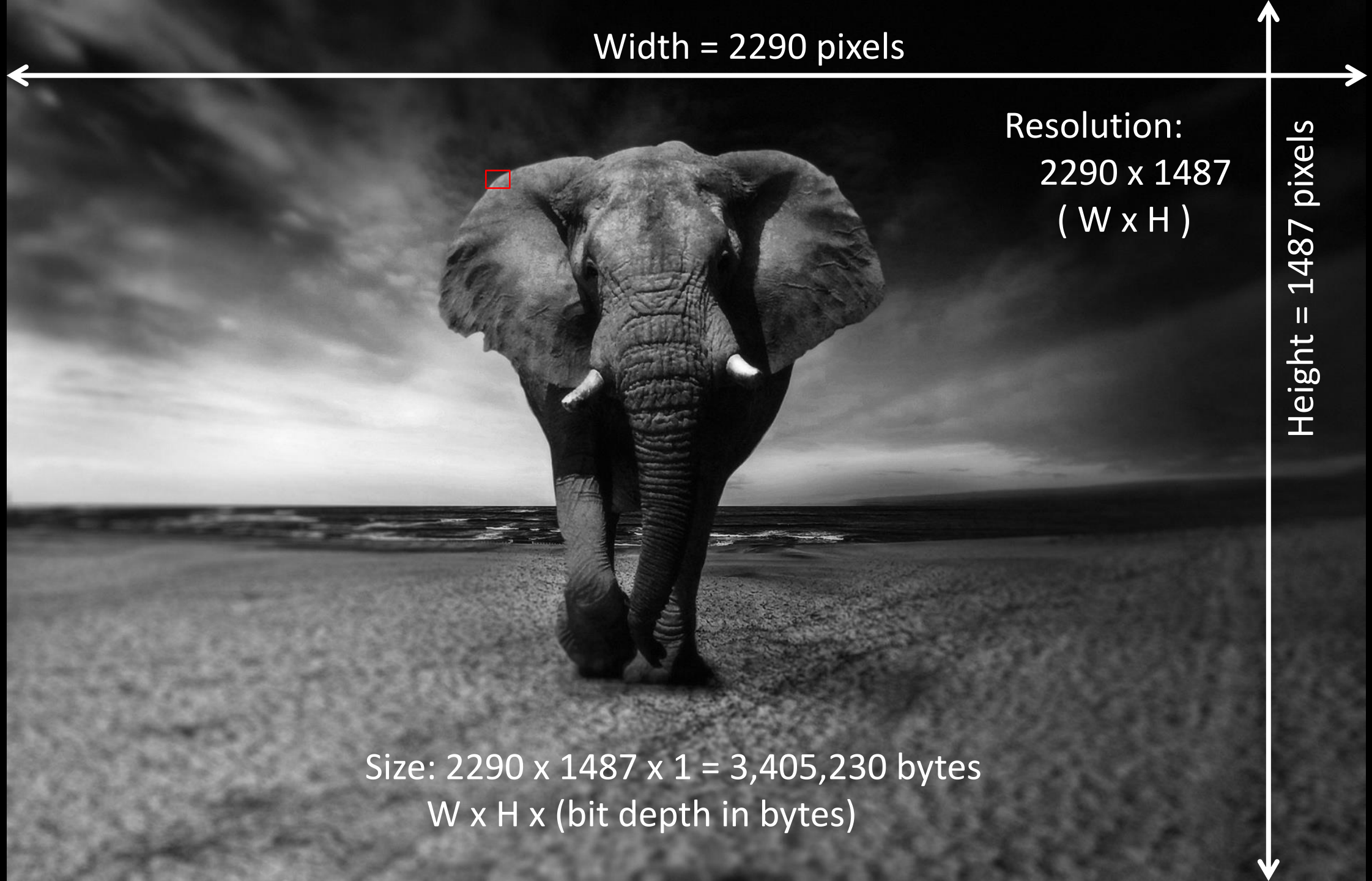


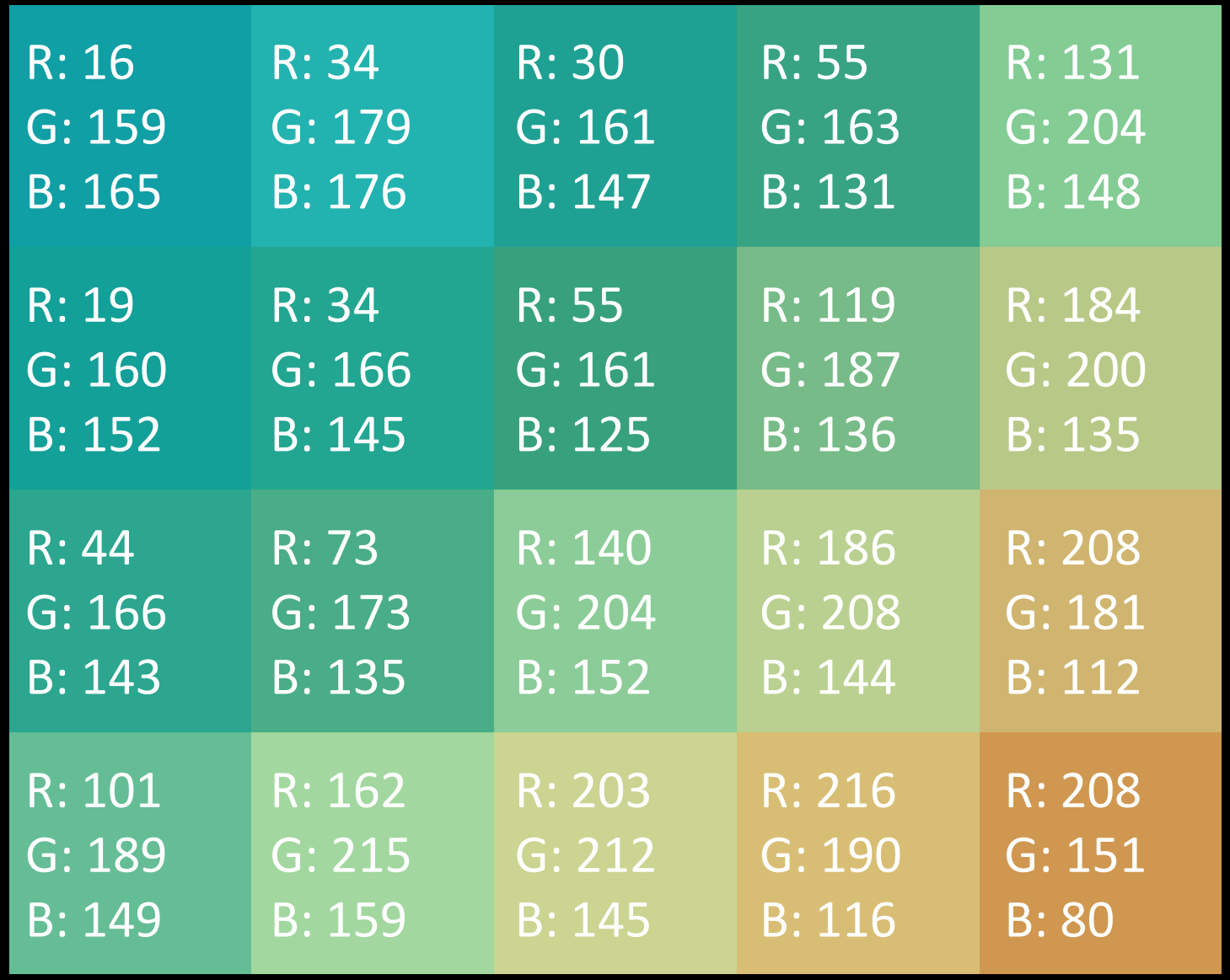
| | | | | |
|---|---|---|---|---|
| 0.22 | 0.23 | 0.23 | 0.26 | 0.32 |
| 0.25 | 0.26 | 0.29 | 0.39 | 0.49 |
| 0.24 | 0.27 | 0.35 | 0.47 | 0.59 |
| 0.28 | 0.38 | 0.49 | 0.57 | 0.62 |




Bit depth: 8 bits

- 0 = black
- 255 = white


Width = 2290 pixels
Height = 1487 pixels
Resolution:
2290 x 1487
( W x H )
Size: 2290 x 1487 x 1 = 3,405,230 bytes
W x H x (bit depth in bytes)

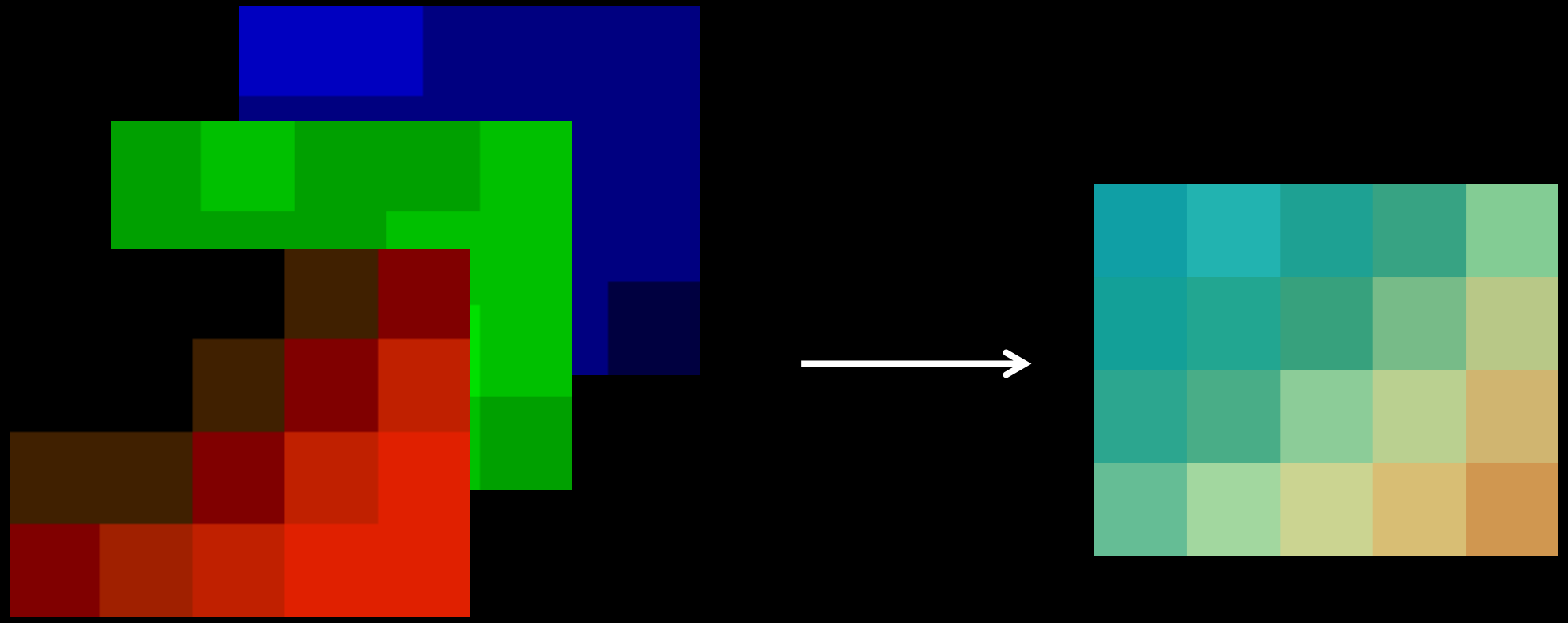



Bit depth: 24 bits

- 1 byte: Red
- 1 byte: Green
- 1 byte: Blue

Optional:

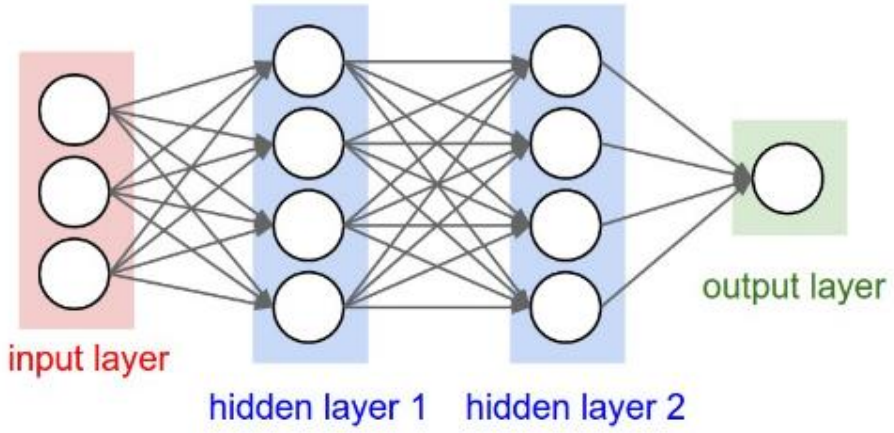
- 1 byte: Alpha

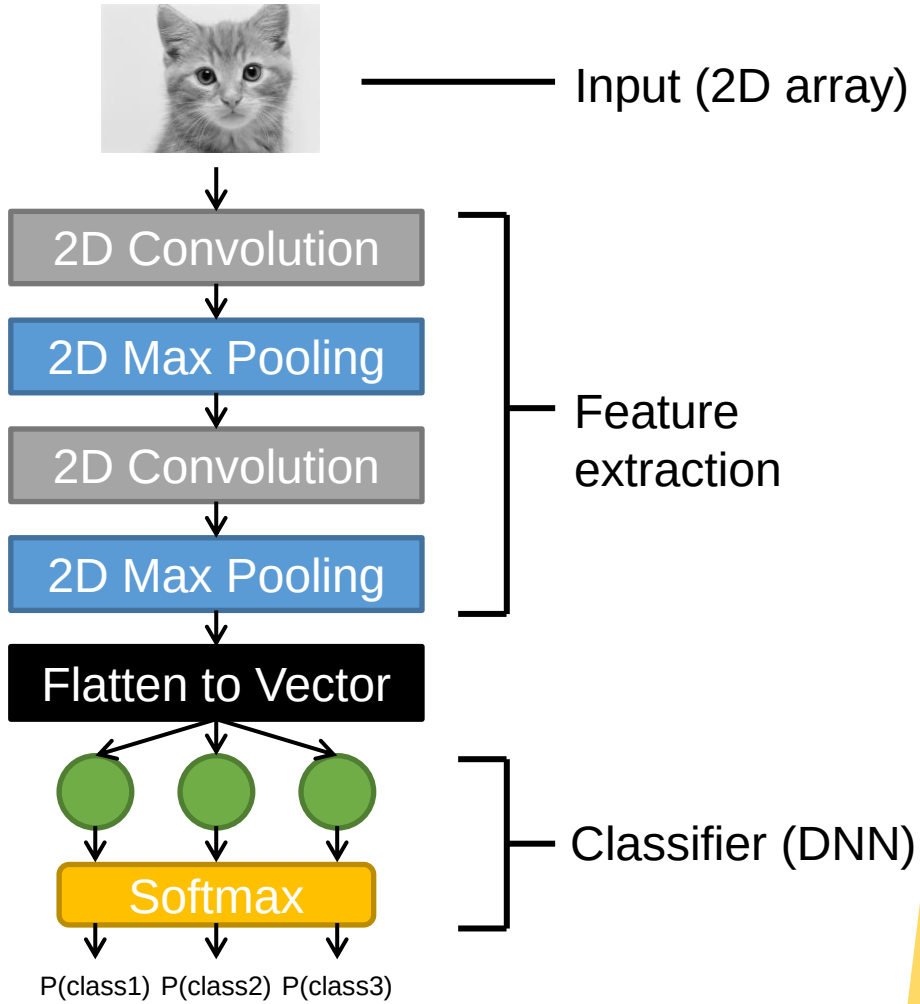


# Image Convolution and Filtering

# Convolutional Neural Networks (CNN)

Dense neural network vs Convolutional neural network

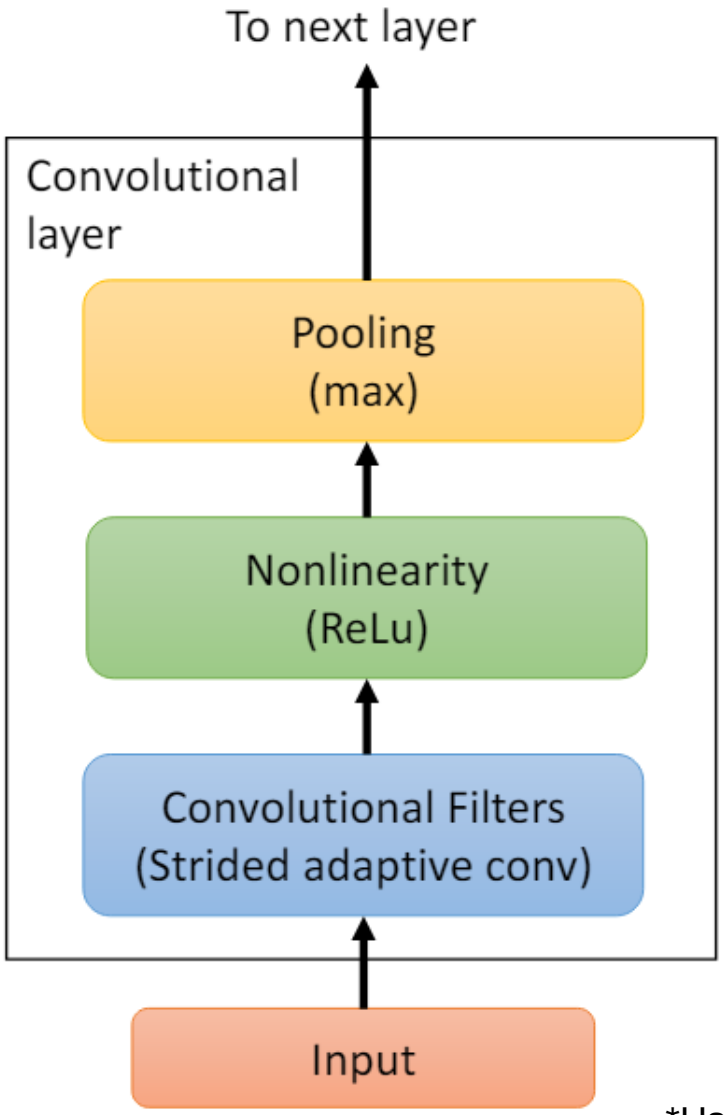


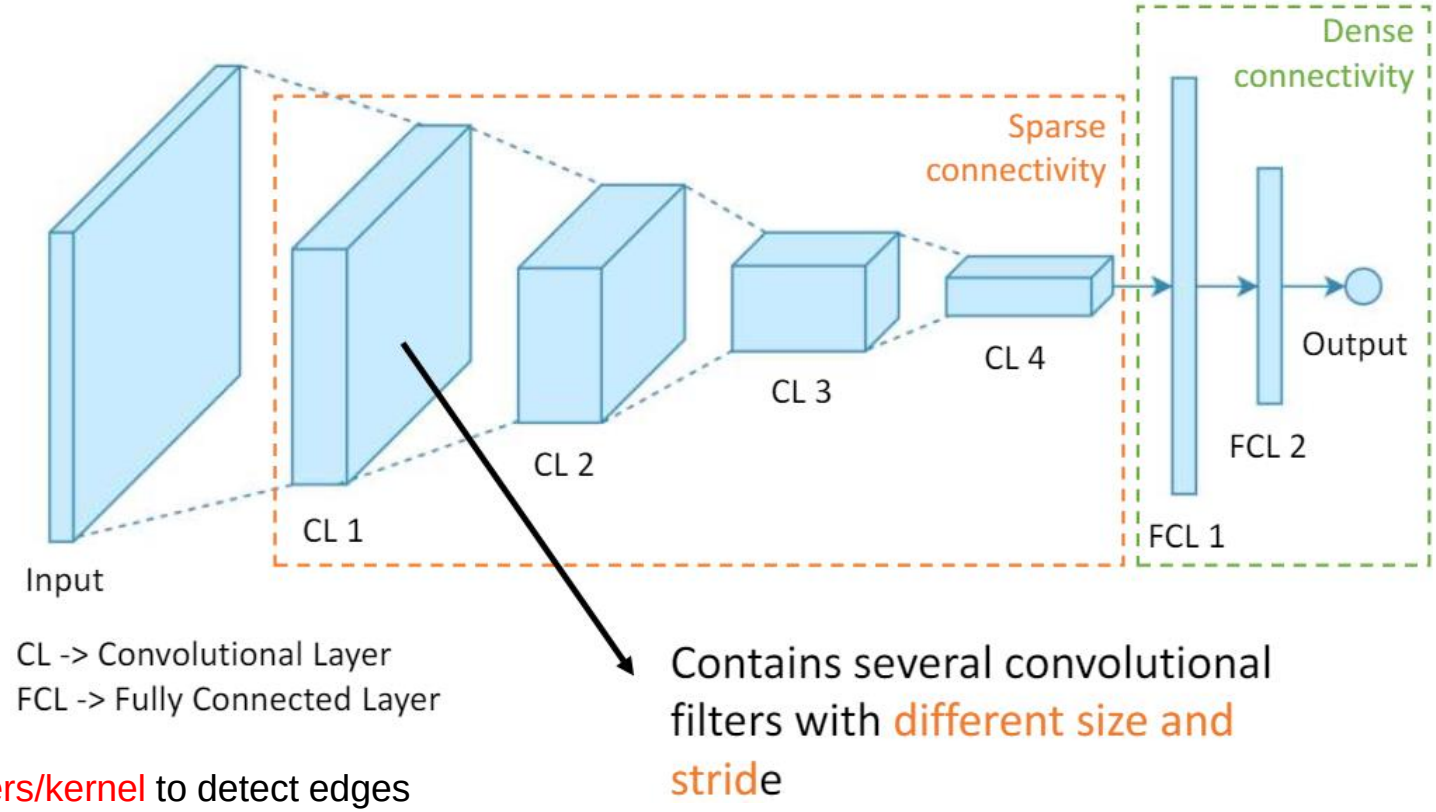

Convolution operation in computer vision

- Helps us to work with large images
- Reduce compute & computational complexity

# The Convolutional Architecture



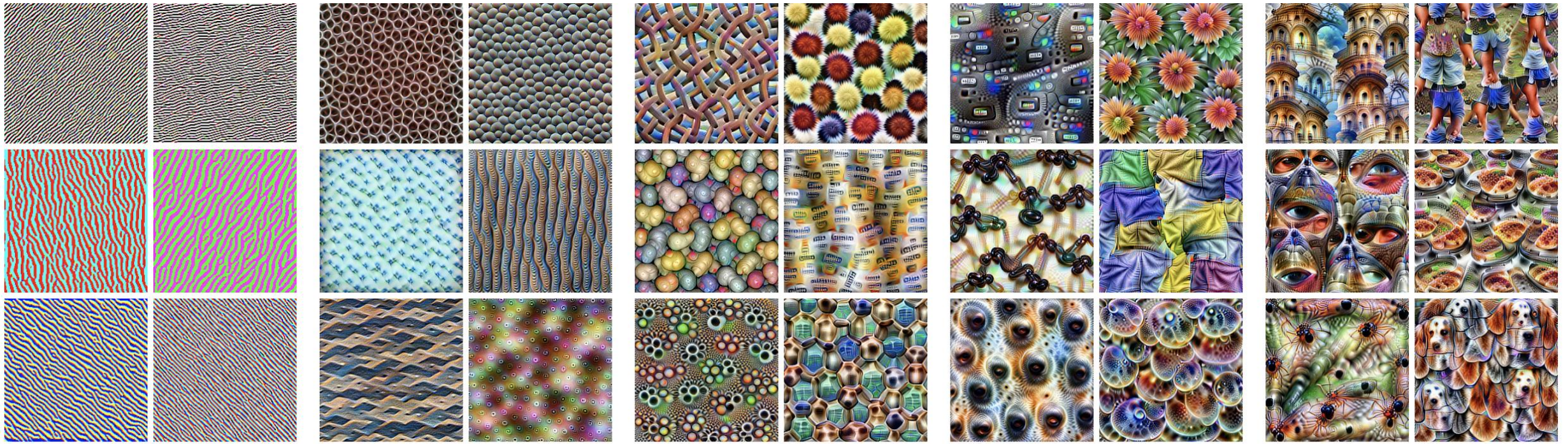

- An architecture made by a hierarchical composition of the basic elements
- Convolution layer is an abstraction for the composition of the 3 basic operations
- Network parameters are in the convolutional component




*Use filters/kernel to detect edges

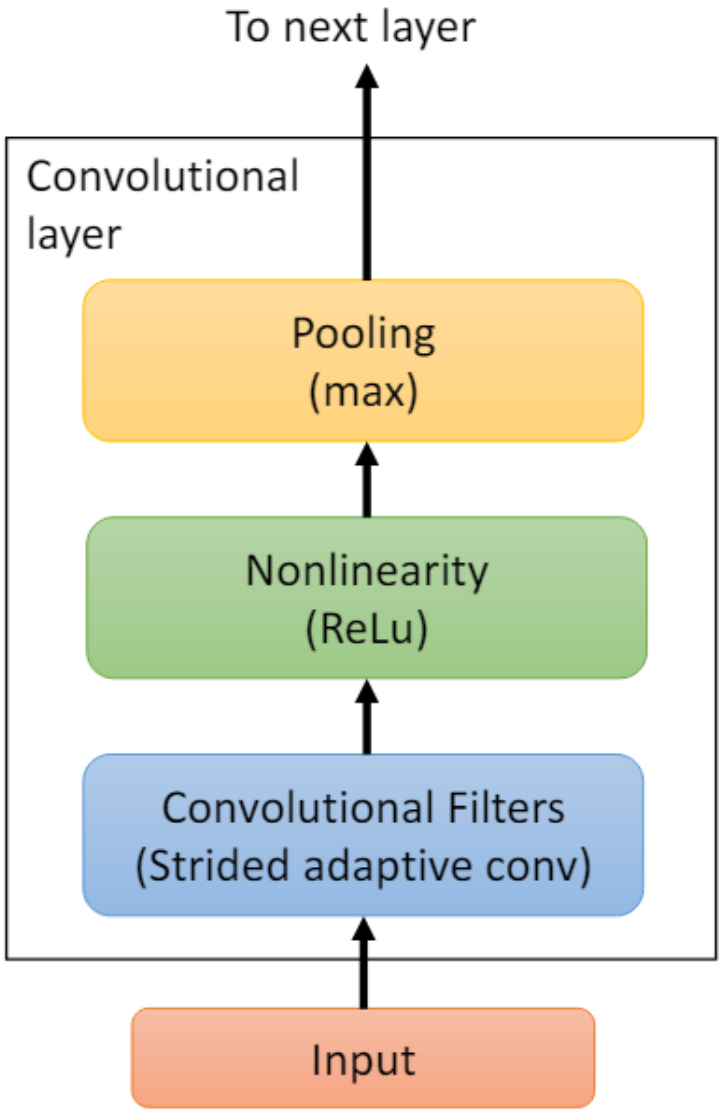
# Learning in convolutions

Convolutional neural networks learn abstract features and concepts from raw image pixels



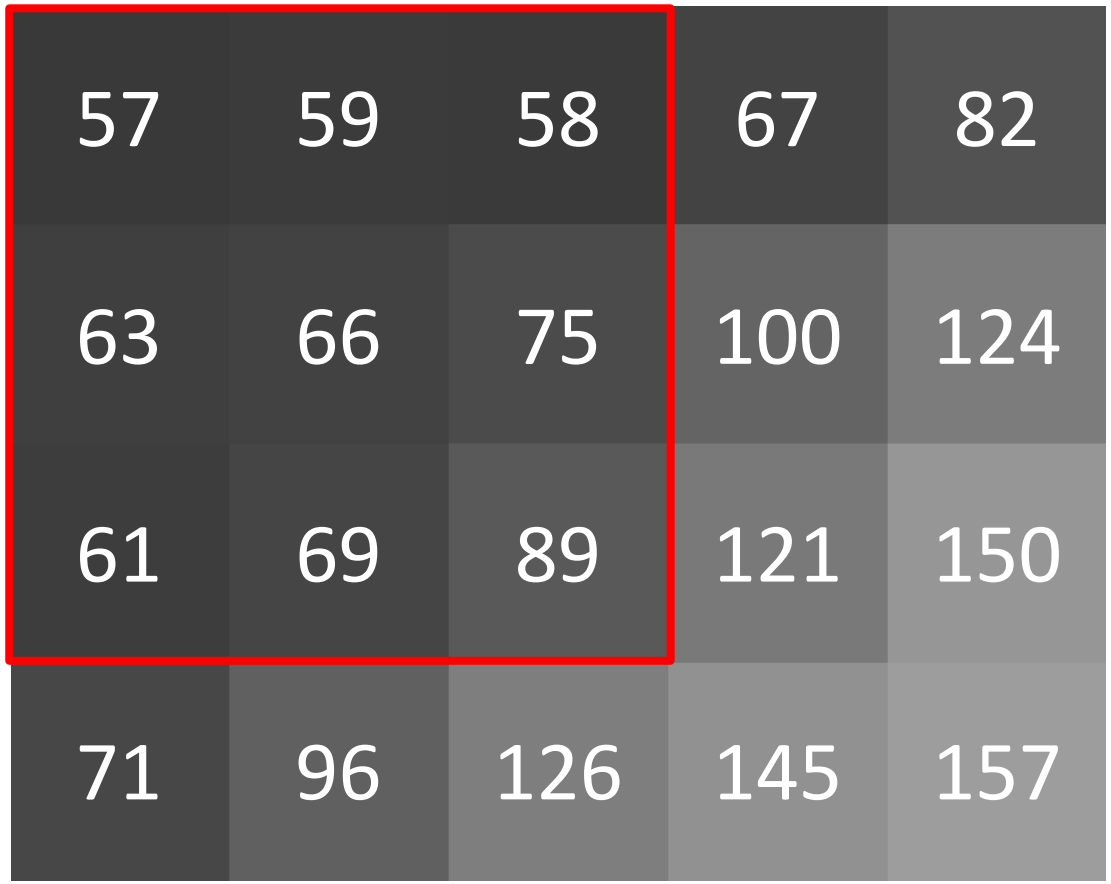
Edges Textures Patterns Parts Objects

# Convolution Process




- main task is called convolution- applying of a sliding window function to a matrix of pixels representing an image.
- The sliding function applied to the matrix is called kernel or filter.
- Several filters of equal size are applied, each filter recognize a specific pattern from the image

Image




Kernel

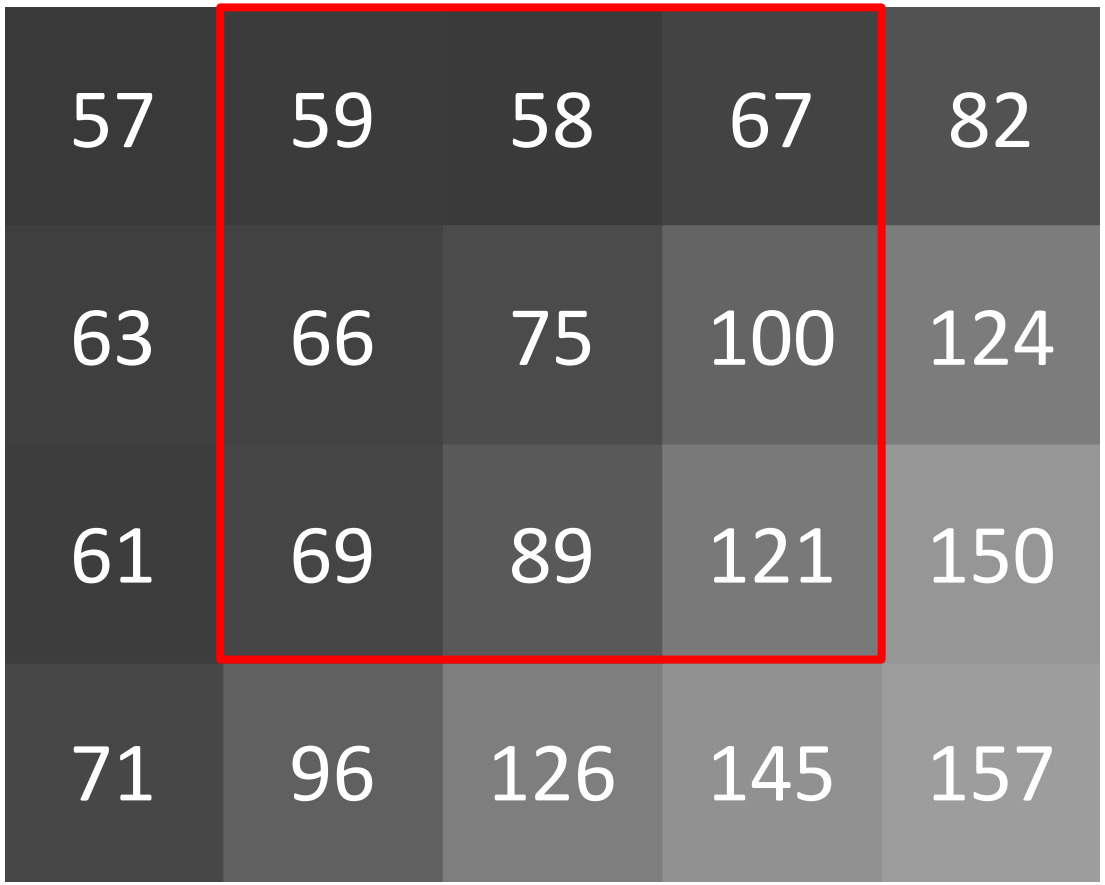
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(57 \cdot 0) + (59 \cdot -1) + (58 \cdot 0) +$
$(63 \cdot -1) + (66 \cdot 5) + (75 \cdot -1) +$
$(61 \cdot 0) + (69 \cdot -1) + (89 \cdot 0) = 64$

Output

| | | |
|---|---|---|
| 64 | | |
| | | |

Image




Kernel

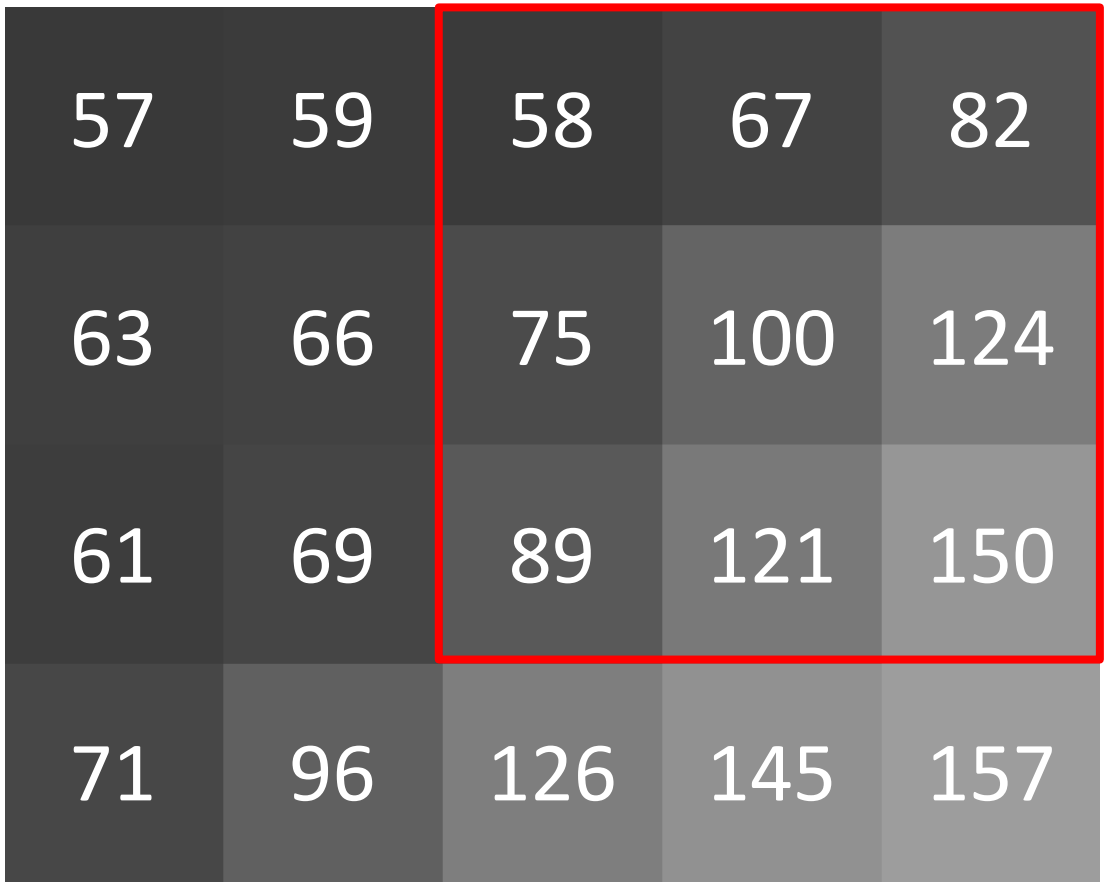
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(59 \cdot 0) + (58 \cdot -1) + (67 \cdot 0) +$
$(66 \cdot -1) + (75 \cdot 5) + (100 \cdot -1) +$
$(69 \cdot 0) + (89 \cdot -1) + (121 \cdot 0) = 62$

Output

| | | |
|---|---|---|
| 64 | 62 | |
| | | |

Image




Kernel

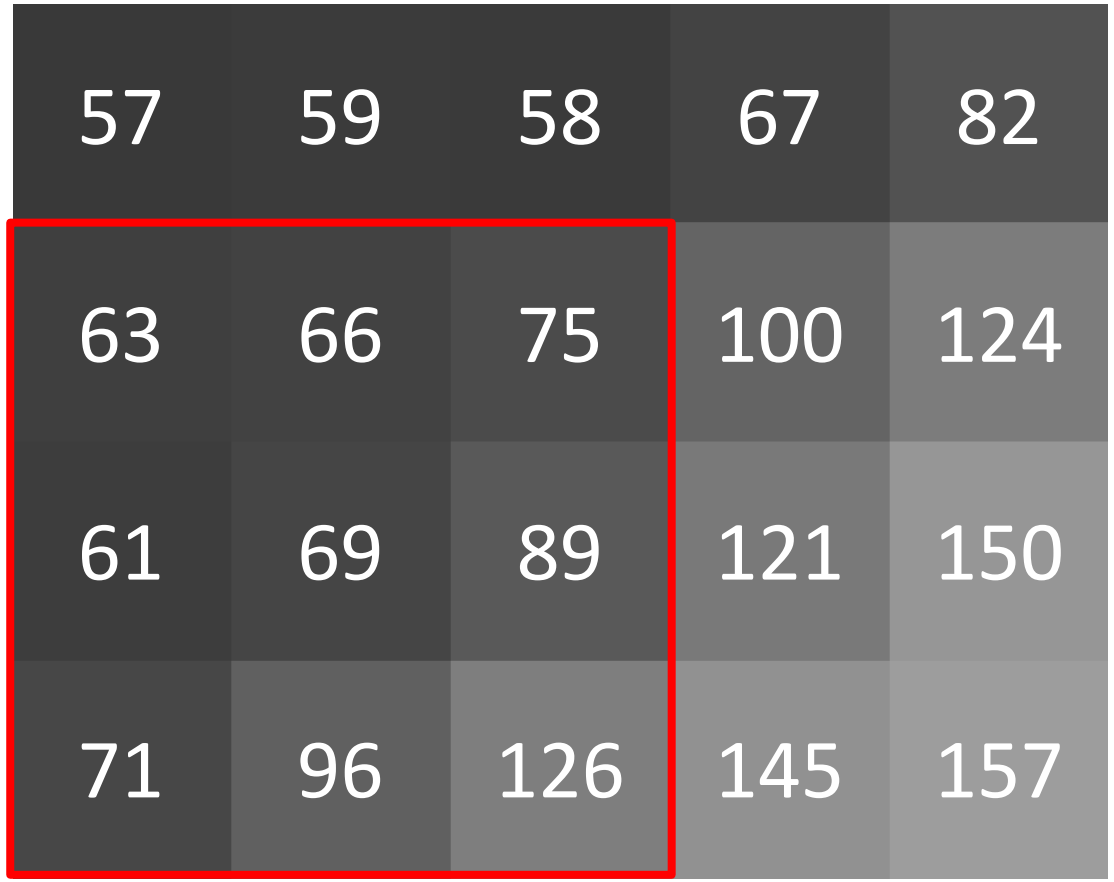
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(58 \cdot 0) + (67 \cdot -1) + (82 \cdot 0) +$
$(75 \cdot -1) + (100 \cdot 5) + (124 \cdot -1) +$
$(89 \cdot 0) + (121 \cdot -1) + (150 \cdot 0) = 113$

Output

| | | |
|---|---|---|
| 64 | 62 | 113 |
| | | |

Image




Kernel

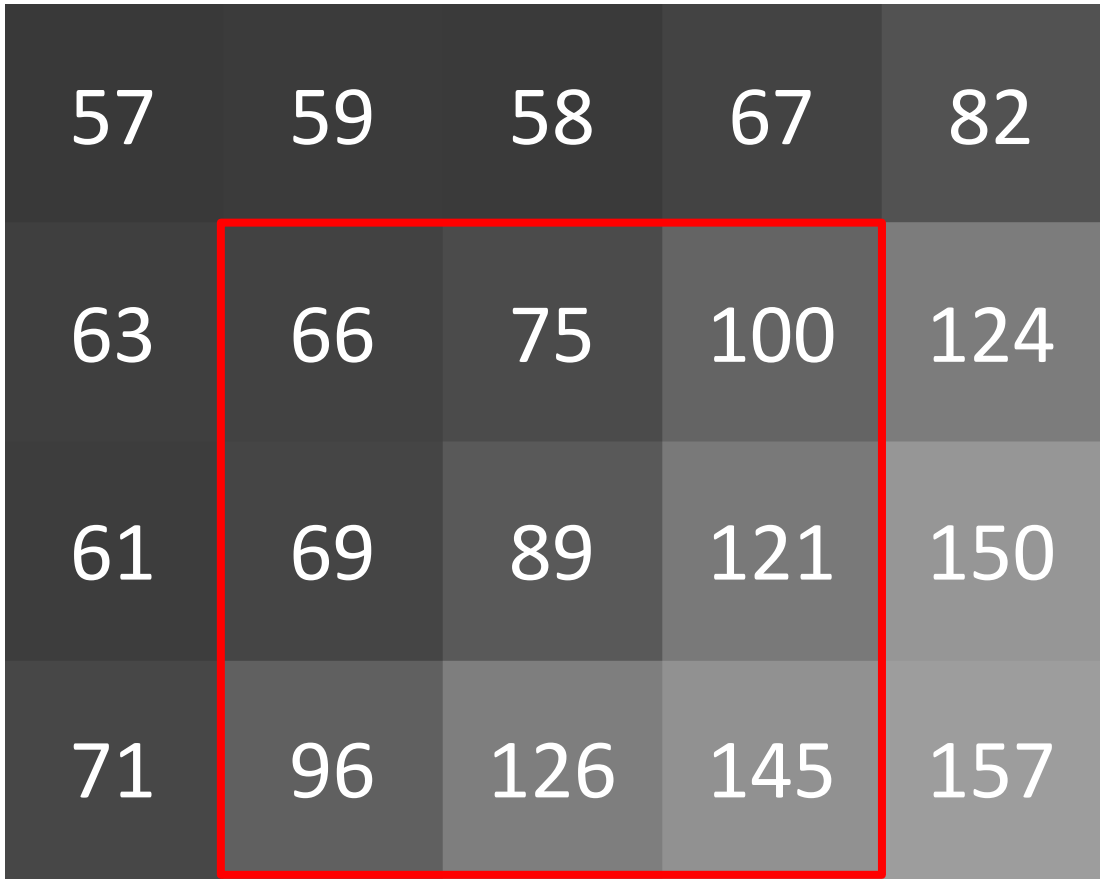
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(63 \cdot 0) + (66 \cdot -1) + (75 \cdot 0) +$
$(61 \cdot -1) + (69 \cdot 5) + (89 \cdot -1) +$
$(71 \cdot 0) + (96 \cdot -1) + (126 \cdot 0) = 33$

Output

| | | |
|---|---|---|
| 64 | 62 | 113 |
| 33 | | |

Image




Kernel

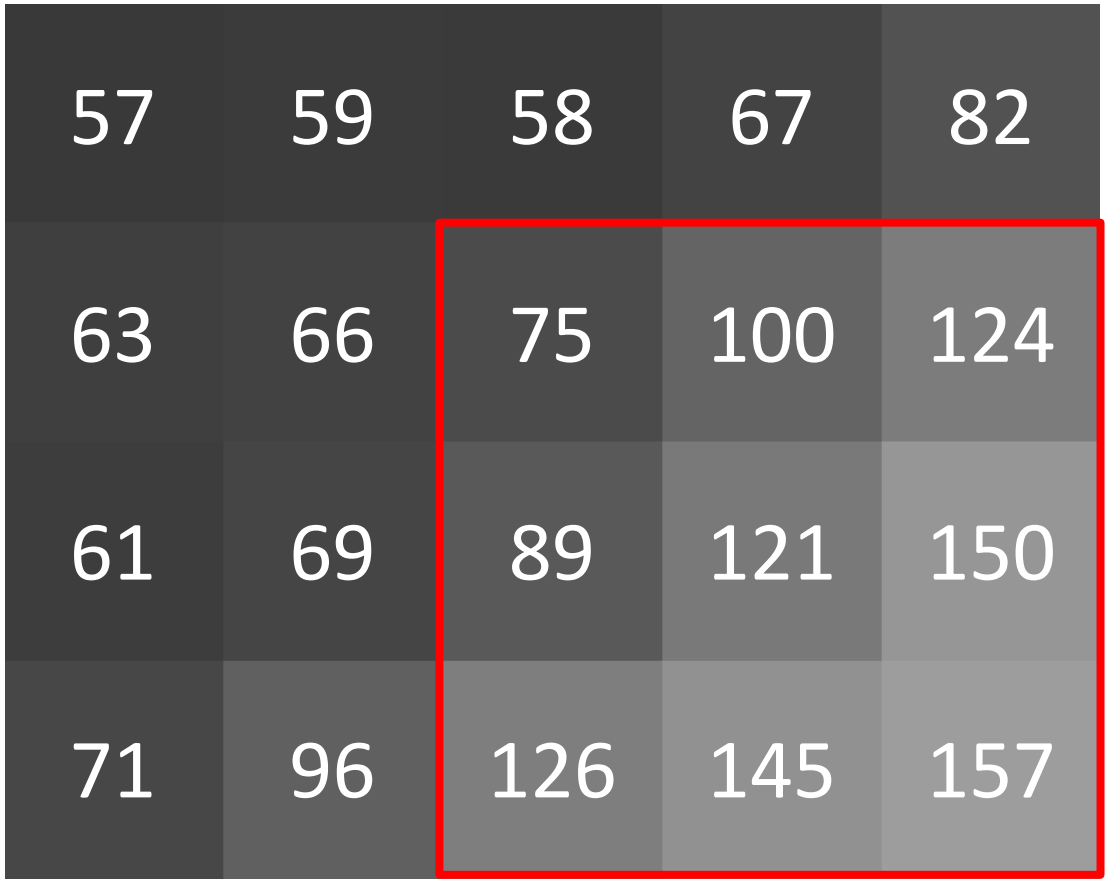
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(66 \cdot 0) + (75 \cdot -1) + (100 \cdot 0) +$
$(69 \cdot -1) + (89 \cdot 5) + (121 \cdot -1) +$
$(96 \cdot 0) + (126 \cdot -1) + (145 \cdot 0) = 54$

Output

| | | |
|---|---|---|
| 64 | 62 | 113 |
| 33 | 54 | |

Image




stride = 1

Kernel

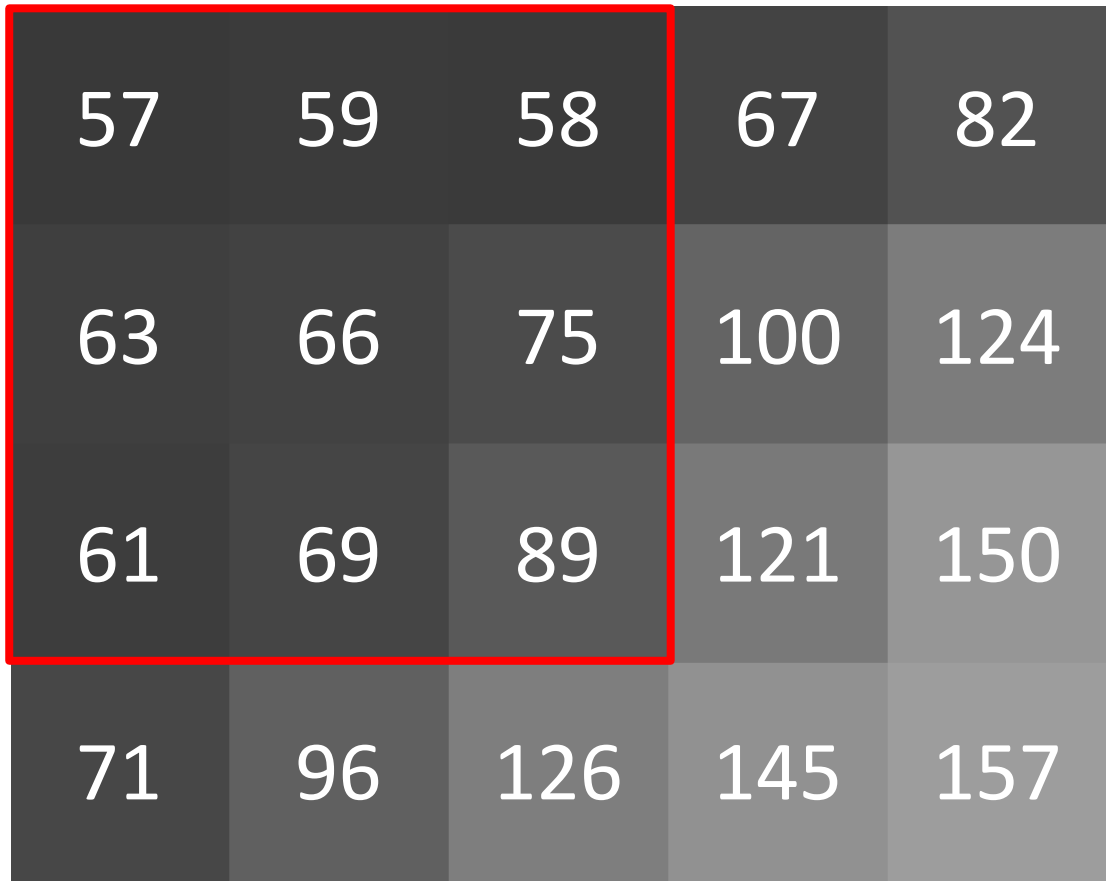
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(75 \cdot 0) + (100 \cdot -1) + (124 \cdot 0) +$
$(89 \cdot -1) + (121 \cdot 5) + (150 \cdot -1) +$
$(126 \cdot 0) + (145 \cdot -1) + (157 \cdot 0) = 121$

Output

| | | |
|---|---|---|
| 64 | 62 | 113 |
| 33 | 54 | 121 |

Image




stride = 2

Kernel

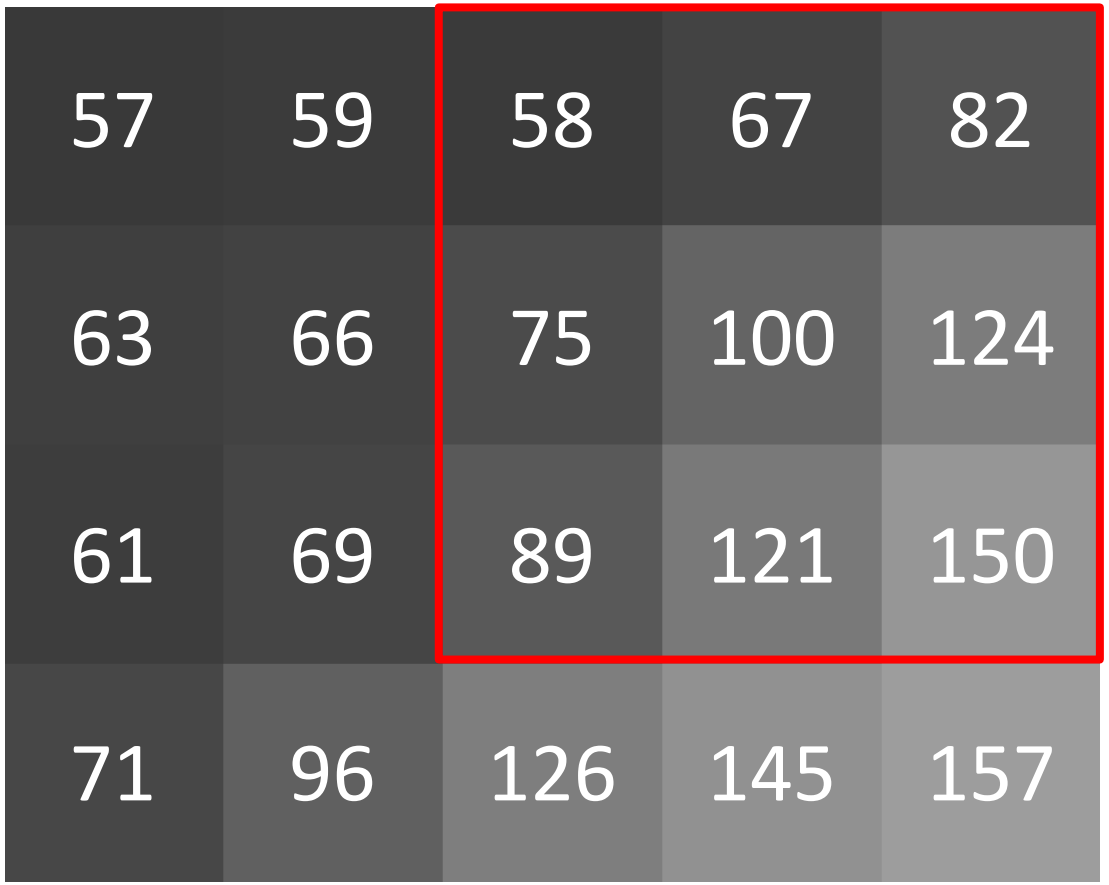
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(57 \cdot 0) + (59 \cdot -1) + (58 \cdot 0) +$
$(63 \cdot -1) + (66 \cdot 5) + (75 \cdot -1) +$
$(61 \cdot 0) + (69 \cdot -1) + (89 \cdot 0) = 64$

Output

| | |
|---|---|
| 64 | |

Image




stride = 2

Kernel

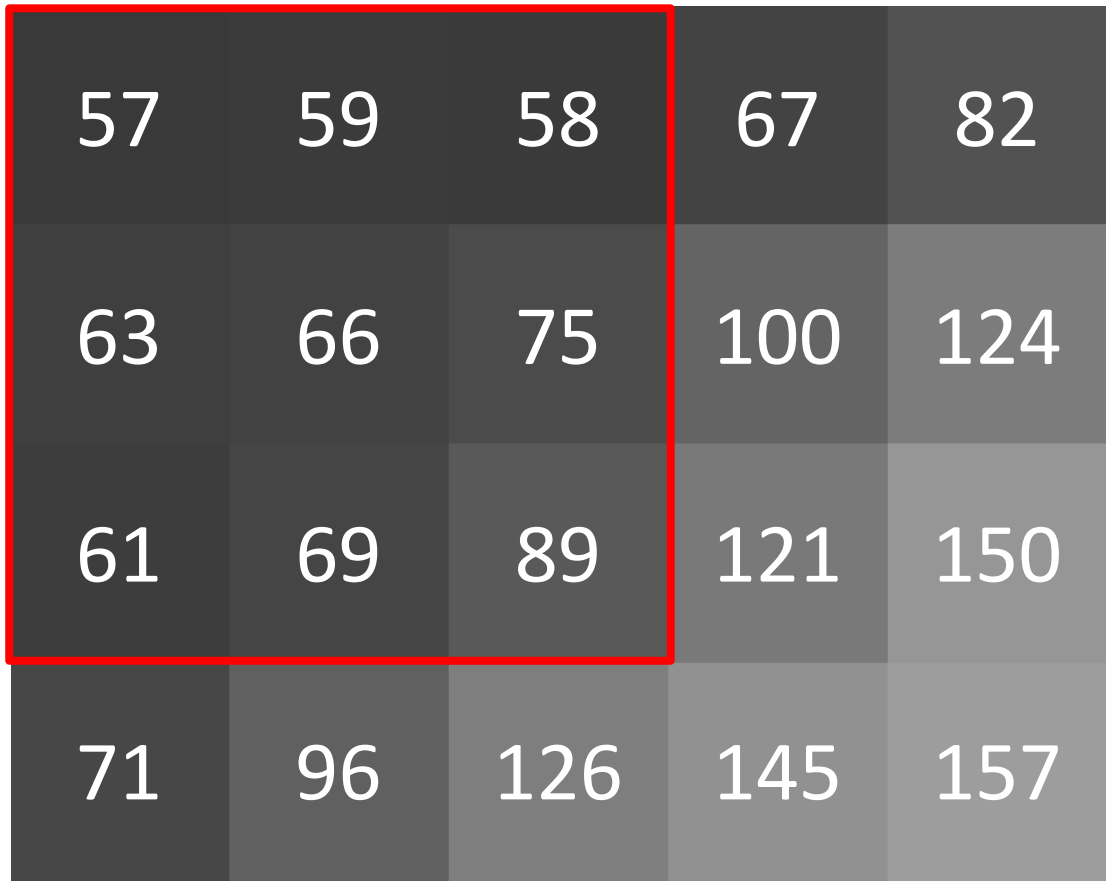
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$(58 \cdot 0) + (67 \cdot -1) + (82 \cdot 0) +$
$(75 \cdot -1) + (100 \cdot 5) + (124 \cdot -1) +$
$(89 \cdot 0) + (121 \cdot -1) + (150 \cdot 0) = 113$

Output

| | |
|---|---|
| 64 | 113 |

Image




8-bit pixel values must be between 0 and 255!

Kernel

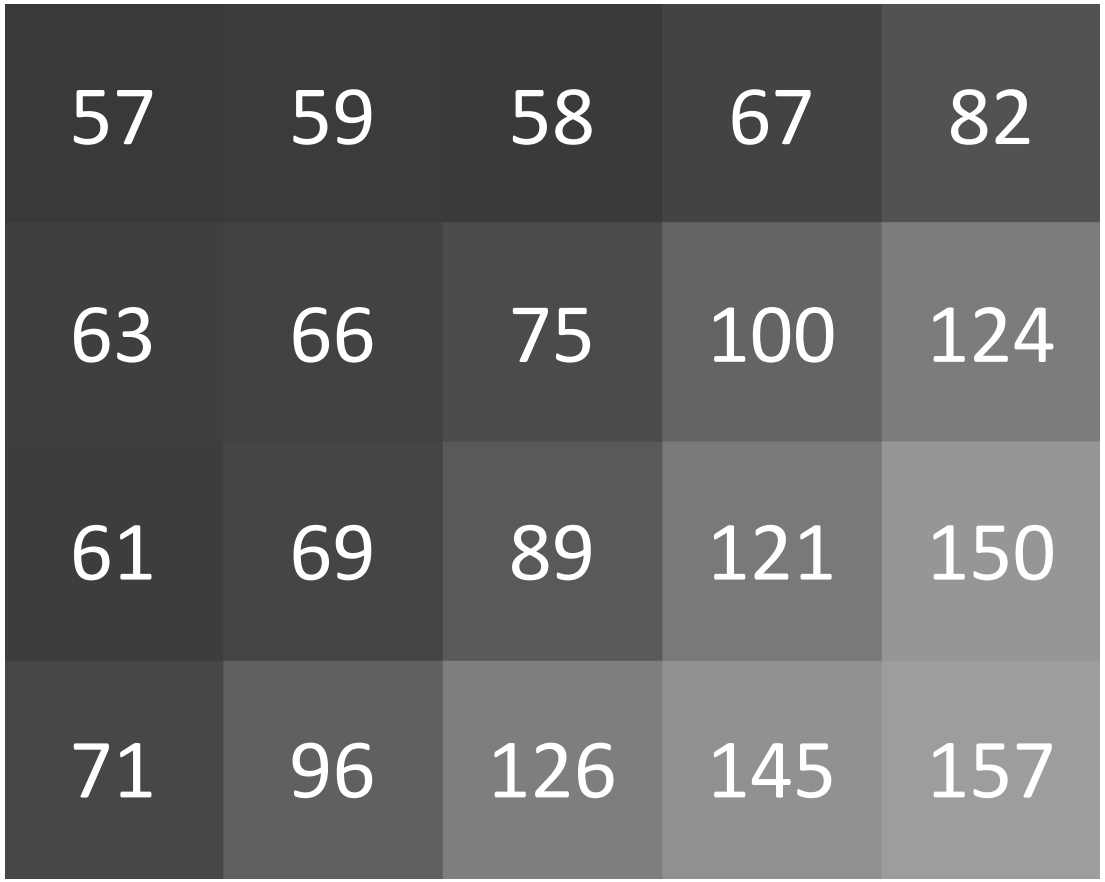
| | | |
|---|---|---|
| 1 | 5 | 1 |
| 5 | 5 | 5 |
| 1 | 5 | 1 |

$$(57 \cdot 1) + (59 \cdot 5) + (58 \cdot 1) +$$
$$(63 \cdot 5) + (66 \cdot 5) + (75 \cdot 5) +$$
$$(61 \cdot 1) + (69 \cdot 5) + (89 \cdot 1) = 1925$$

Output

| | | |
|---|---|---|
| 255 | | |
| | | |

Image




8-bit pixel values must be between 0 and 255!

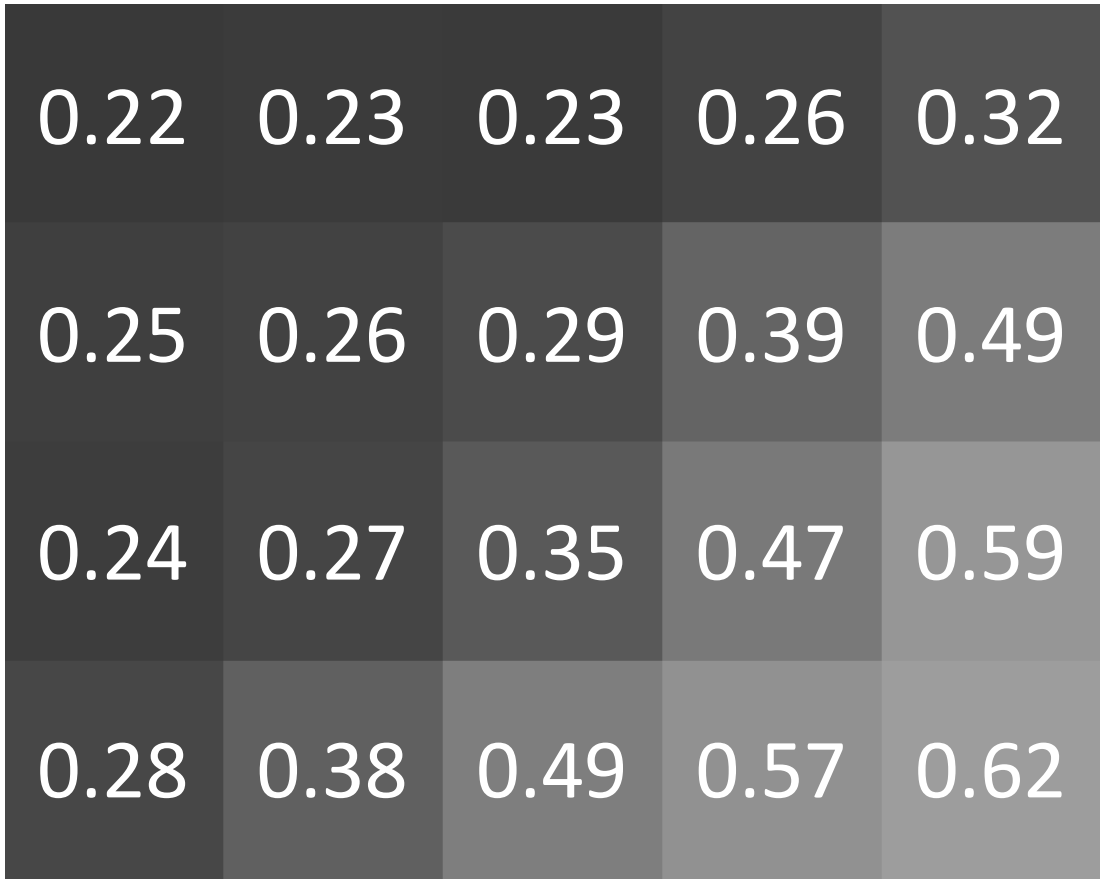
Kernel

| | | |
|---|---|---|
| 1 | 5 | 1 |
| 5 | 5 | 5 |
| 1 | 5 | 1 |

Output

| | | |
|---|---|---|
| 255 | 255 | 255 |
| 255 | 255 | 255 |

Image




Floating point pixel values will likely be between 0.0 and 1.0

Kernel

| | | |
|---|---|---|
| 1 | 5 | 1 |
| 5 | 5 | 5 |
| 1 | 5 | 1 |

Output

| | | |
|---|---|---|
| 1.0 | 1.0 | 1.0 |
| 1.0 | 1.0 | 1.0 |

Original image (200x130)

Gaussian blur

| | | |
|---|---|---|
| 1/16 | 1/8 | 1/16 |
| 1/8 | 1/4 | 1/8 |
| 1/16 | 1/8 | 1/16 |

Outline

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

Sharpen

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

Left Sobel

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Emboss

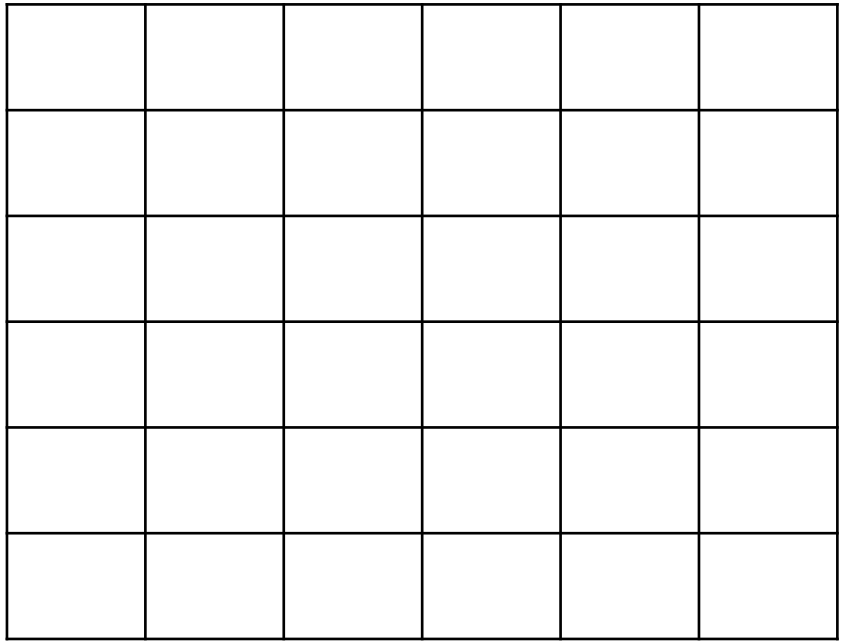
| | | |
|---|---|---|
| -2 | -1 | 0 |
| -1 | 1 | 1 |
| 0 | 1 | 2 |

Top Sobel

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

# Convolution Process

Image



*n* x *n*

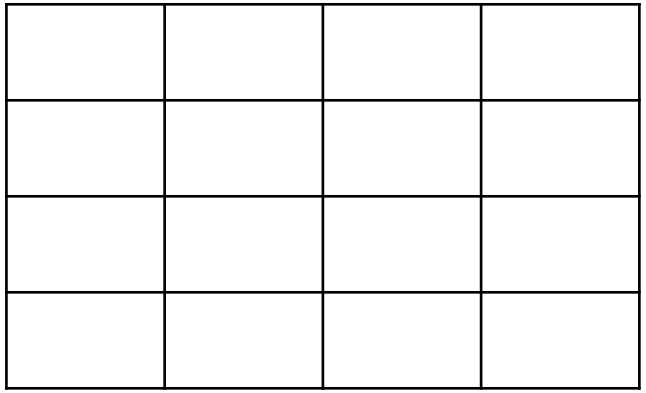
*

Kernel

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

*f* x *f*

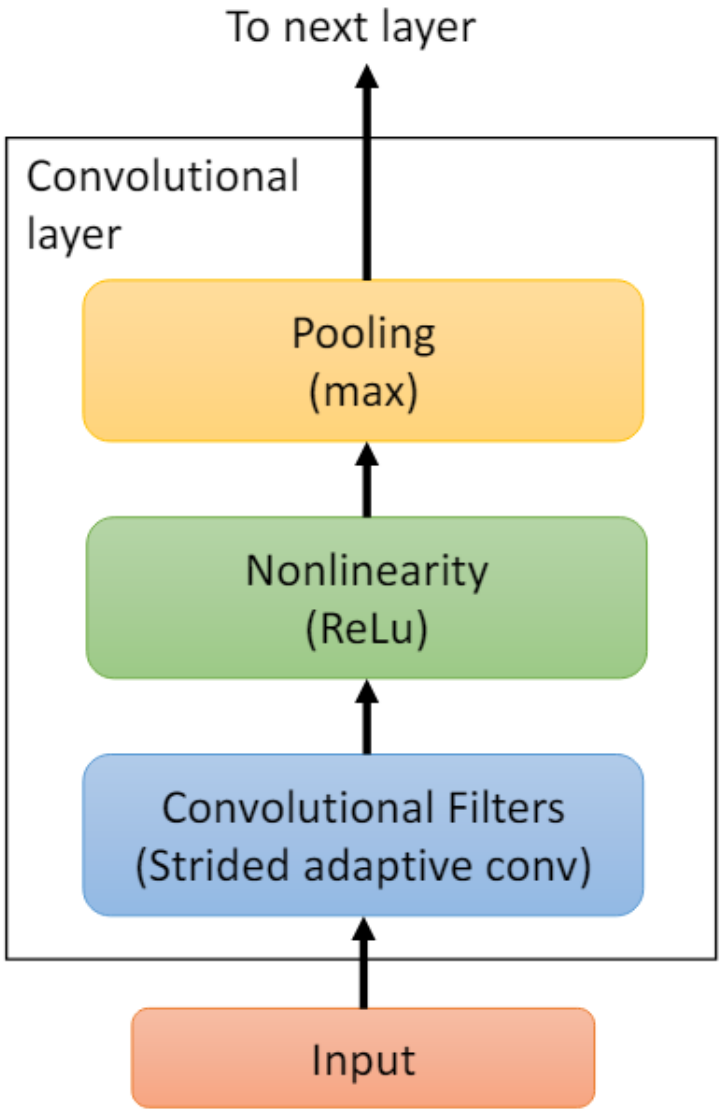
Output



*n-f+1* x *n-f+1*

- Image shrink after conv*
- Throwing info from edges

# Padding

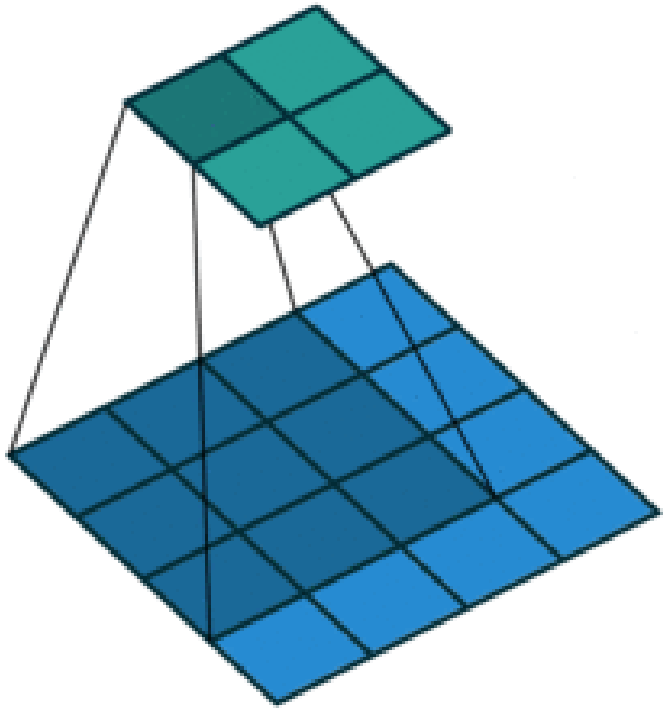



- Technique used to preserve the spatial dimensions of the input image after convolution operations on a feature map
- Removes aggregation bias from the convolution operation
- Types

1. **Same Padding**: add zeros around the input image. To keep the output size equal to the input size after convolution.
2. **Valid Padding**: 'no padding,' doesn't add any extra pixels to the input image. Causing the output feature map to shrink compared to the input.
3. **Causal Padding**: NLP and time-series analysis -adds padding only to the left side of the input sequence.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 57 | 59 | 58 | 67 | 82 | 0 |
| 0 | 63 | 66 | 75 | 100 | 124 | 0 |
| 0 | 61 | 69 | 89 | 121 | 150 | 0 |
| 0 | 71 | 96 | 126 | 145 | 157 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

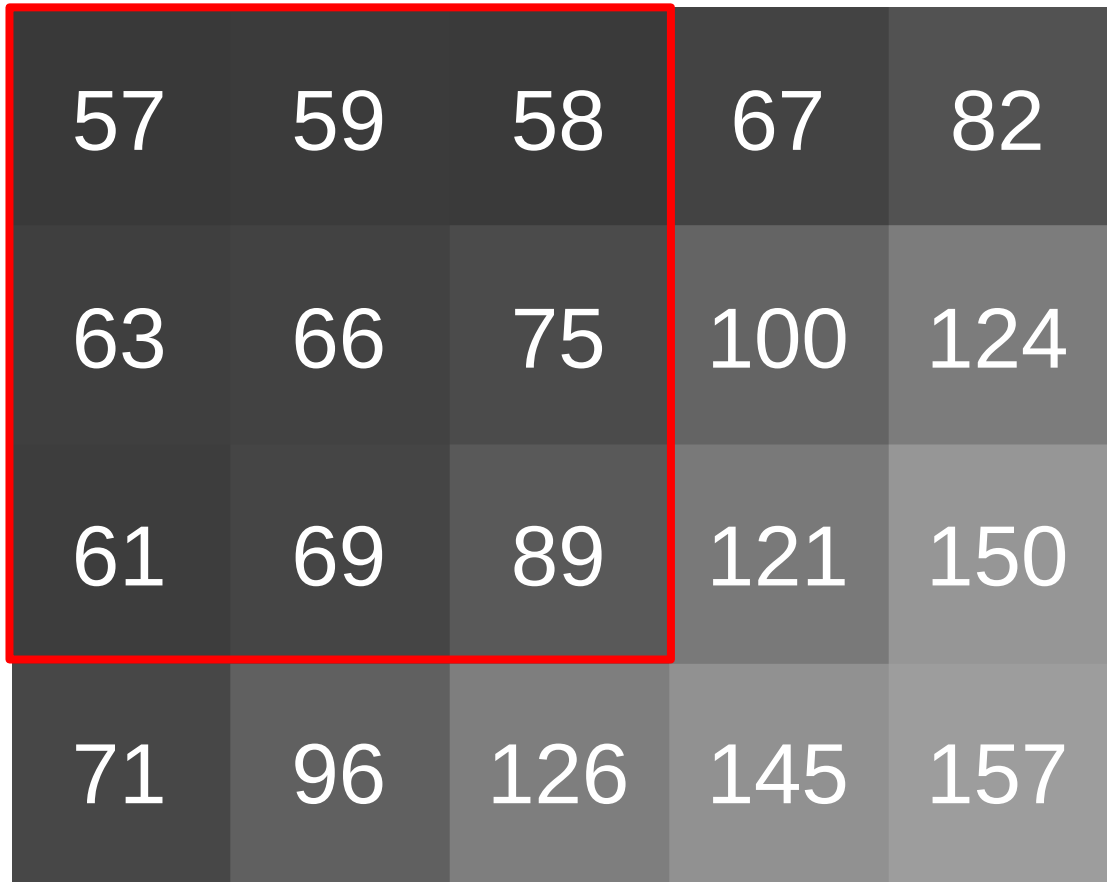
# Valid Padding



Animations from github.com/vdumoulin/conv_arithmetic (MIT license)

# Valid Padding

Image




stride = 2

Kernel

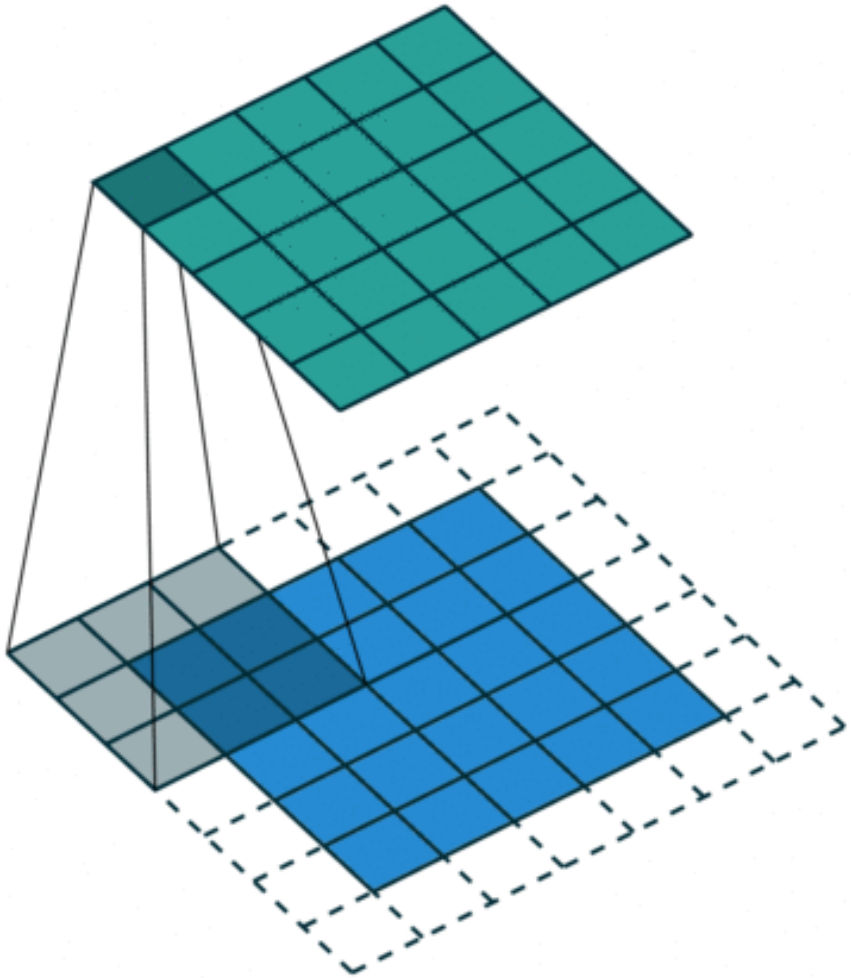
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

Output

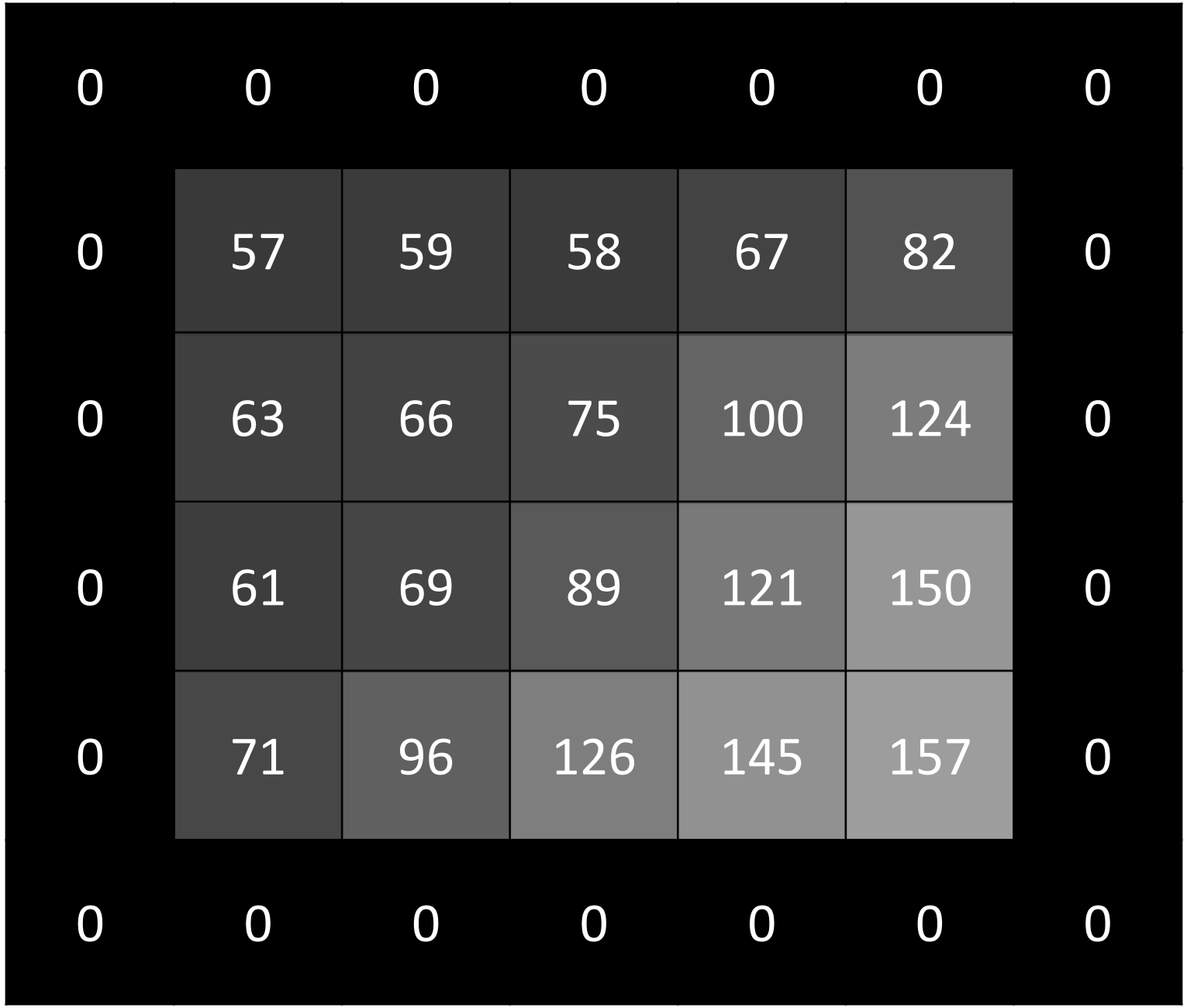
| | |
|---|---|
| 64 | 113 |

# Same Padding

Output same size as input image



Animations from github.com/vdumoulin/conv_arithmetic (MIT license)


0 0 0 0 0 0 0
0 57 59 58 67 82 0
0 63 66 75 100 124 0
0 61 69 89 121 150 0
0 71 96 126 145 157 0
0 0 0 0 0 0 0

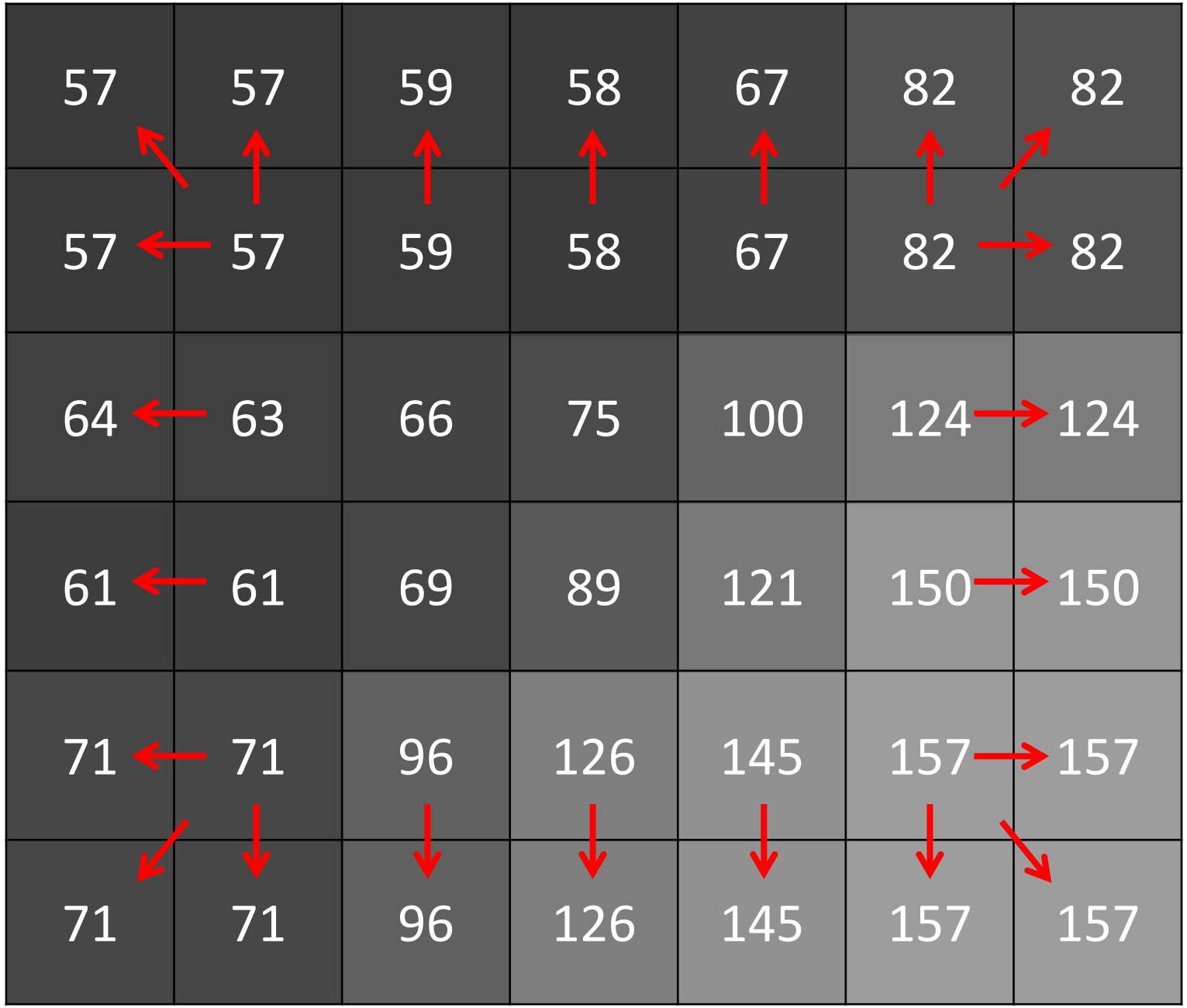
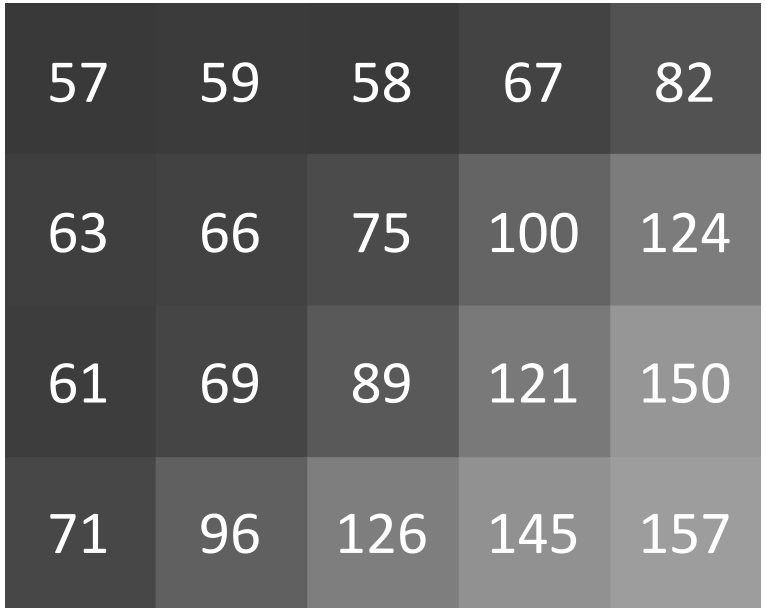

57 57 59 58 67 82 82
57 57 59 58 67 82 82
64 63 66 75 100 124 124
61 61 69 89 121 150 150
71 71 96 126 145 157 157
71 71 96 126 145 157 157

Image (without padding)




stride = 1

Kernel

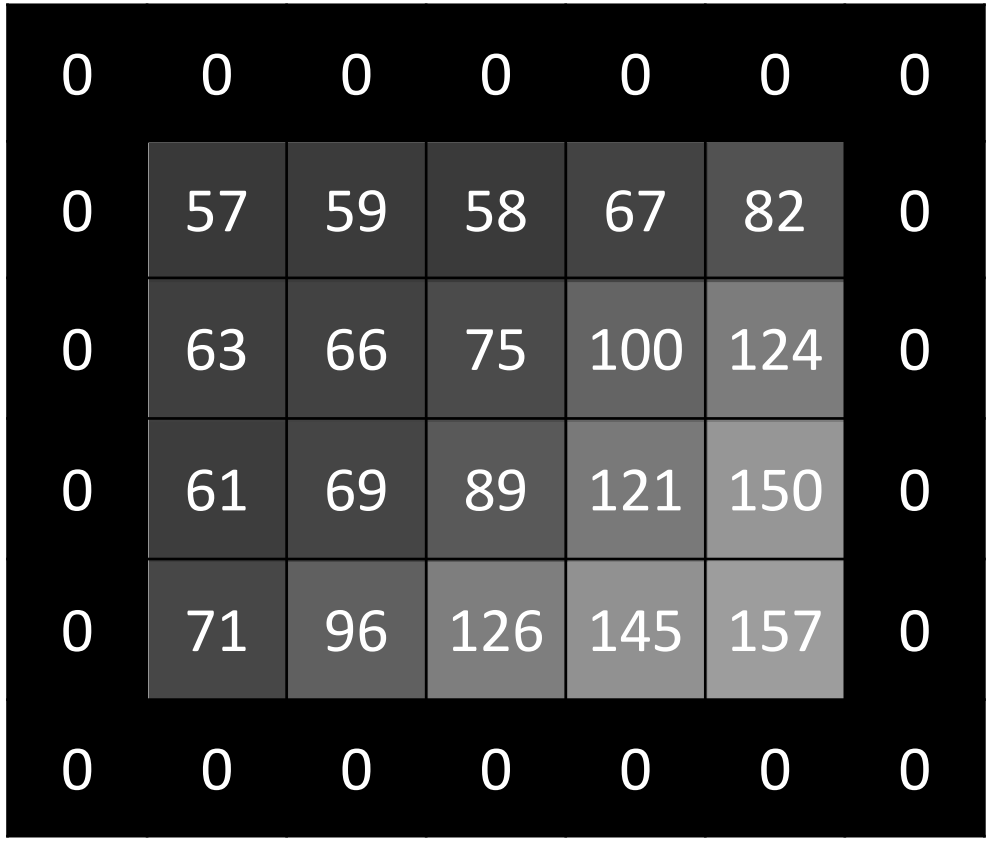
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

Without padding:
- Smaller output
- Information on borders is lost

Output

| | | |
|---|---|---|
| 64 | 62 | 113 |
| 33 | 54 | 121 |

Input Image (with padding)




stride = 1

Kernel

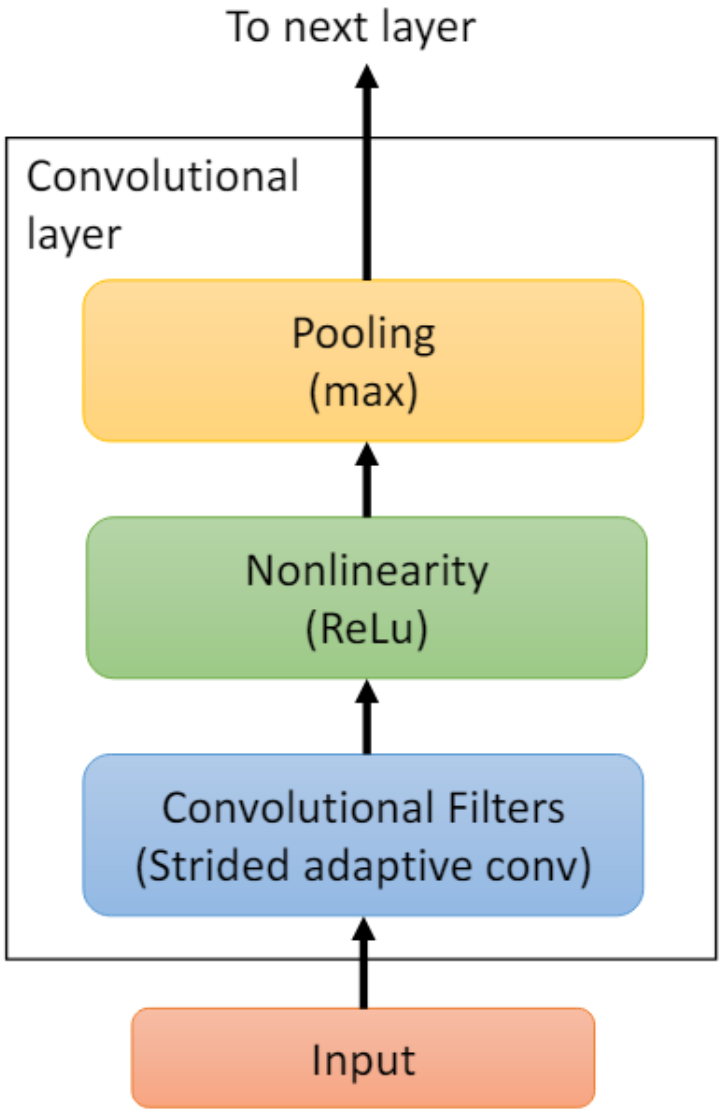
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

With padding:
- Larger output matrix
- Information on borders is maintained

Output

| | | | | |
|---|---|---|---|---|
| 163 | 114 | 89 | 95 | 219 |
| 131 | 64 | 62 | 113 | 255 |
| 102 | 33 | 54 | 121 | 255 |
| 198 | 214 | 255 | 255 | 255 |

# Summary on Convolutions




*n x n* image

*f x f* filter/kernel

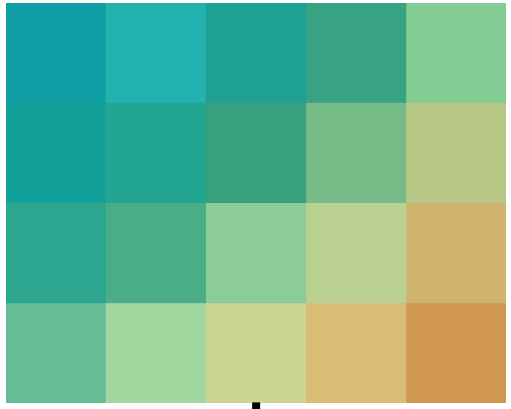
padding *p*

stride *s*

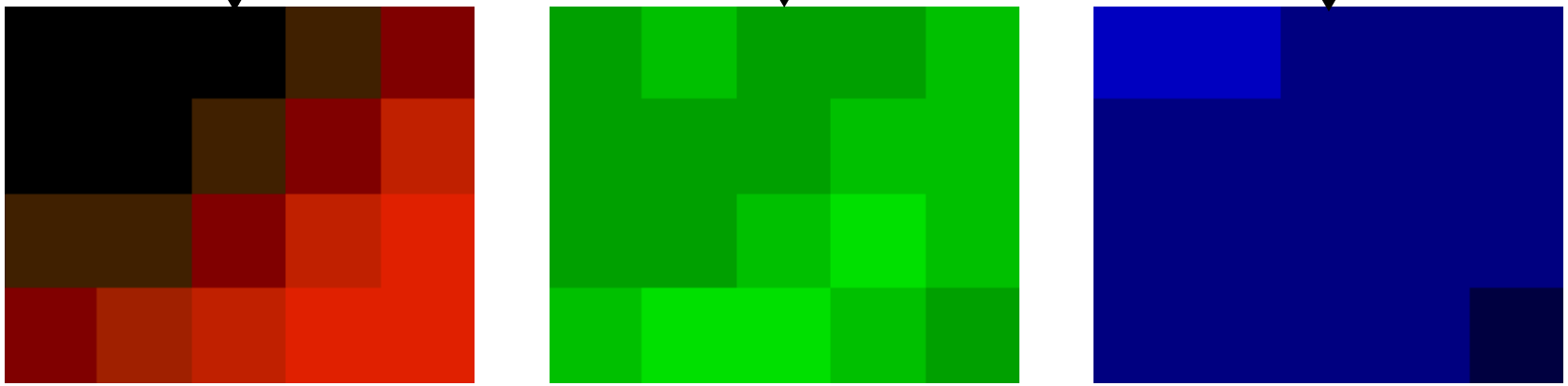
$$\left[\frac{n+2p-f}{s}+1\right]\times\left[\frac{n+2p-f}{s}+1\right]$$

Cross-correlation vs convolution

Original image



RGB channels



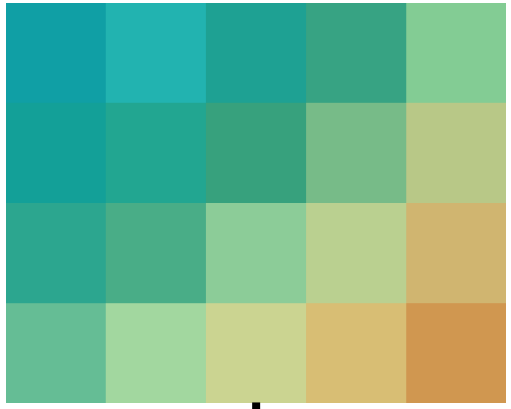
Kernel

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

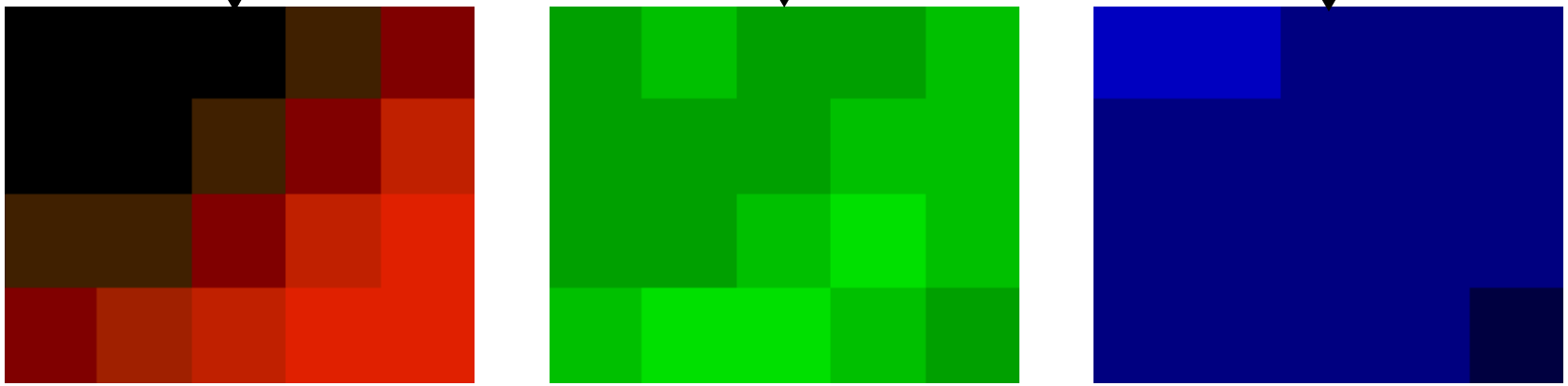
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

Original image



RGB channels



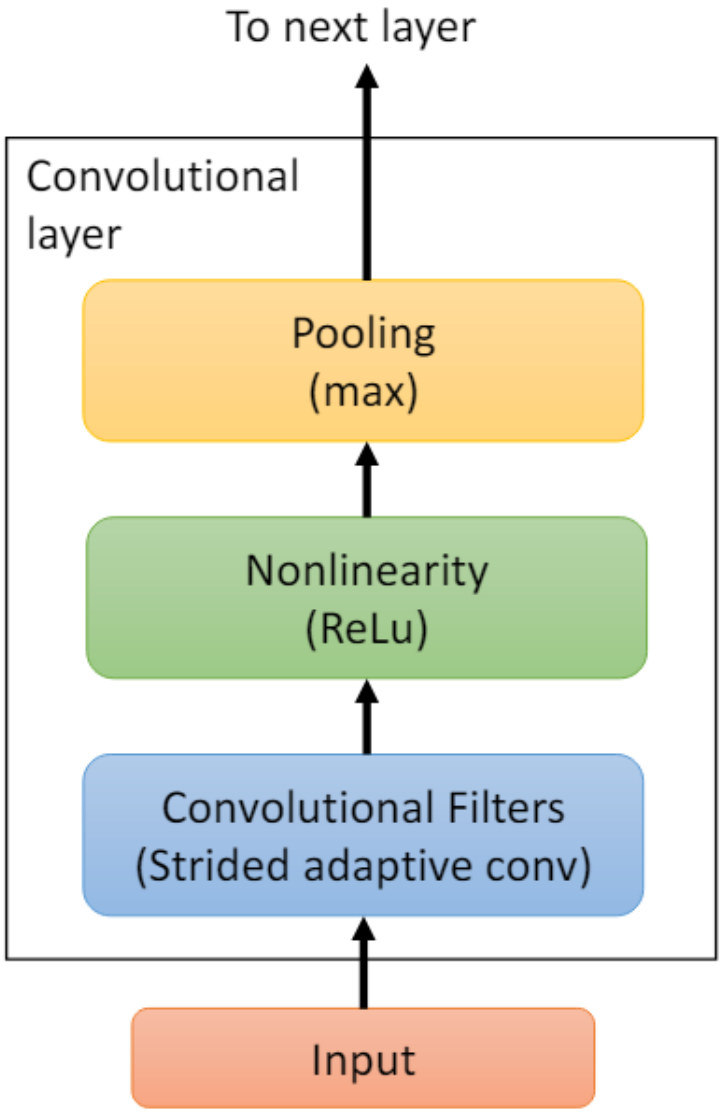
Kernel

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

# Pooling




- Subsampling pixels (does not change the object)
- consolidates the features identified by CNNs.
- gradually shrink the model's spatial dimension to reduce the network's parameters and computations.

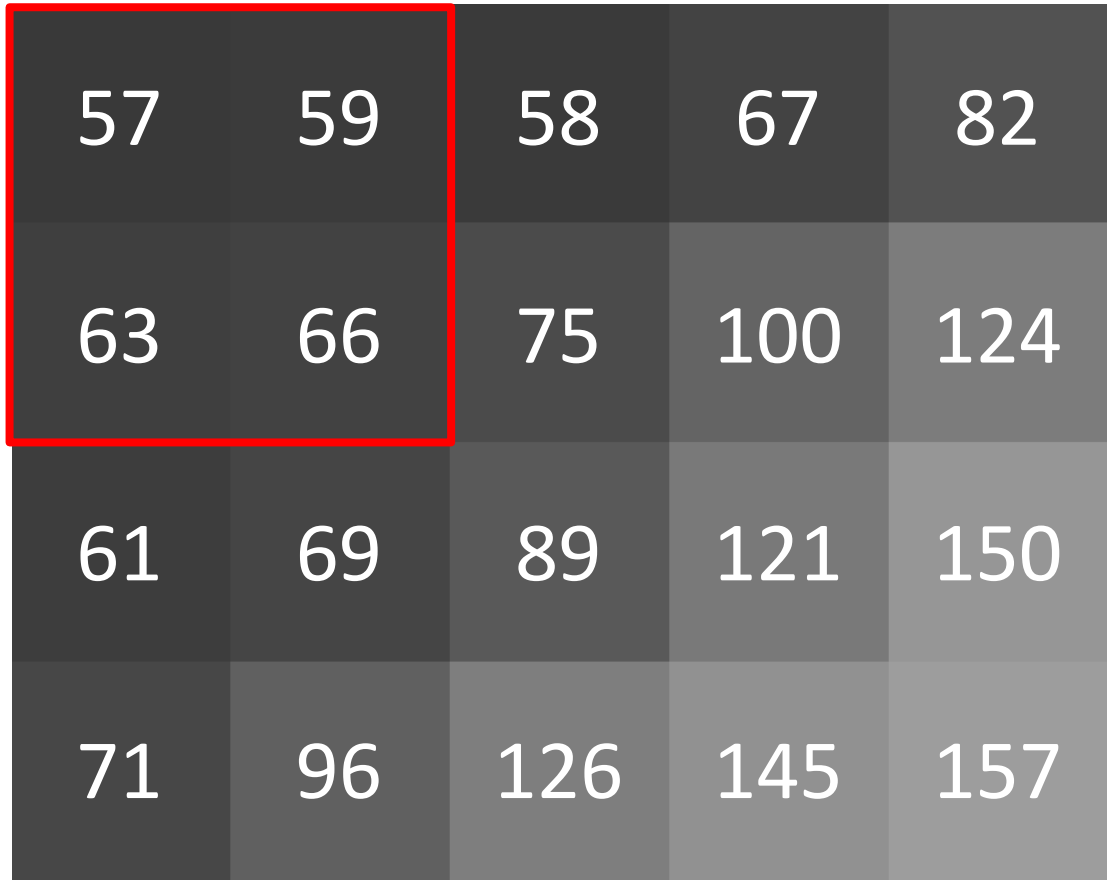
bird

Subsampling

bird

# Average Pooling

Image




Window: 2x2
Stride: 2

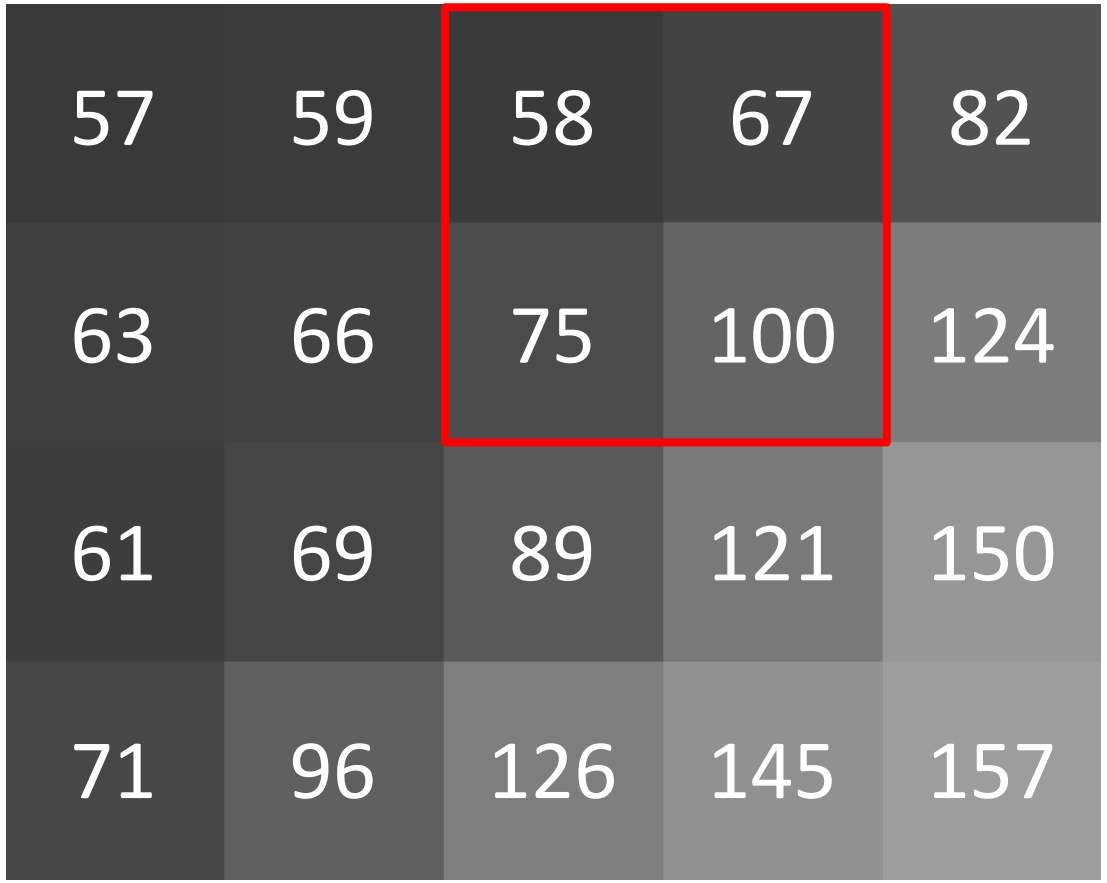
Find average under window:
(57 + 59 + 63 + 66) / 4 = 61.25

Output

| | |
|---|---|
| 61 | |
| | |

# Average Pooling

Image




Window: 2x2
Stride: 2

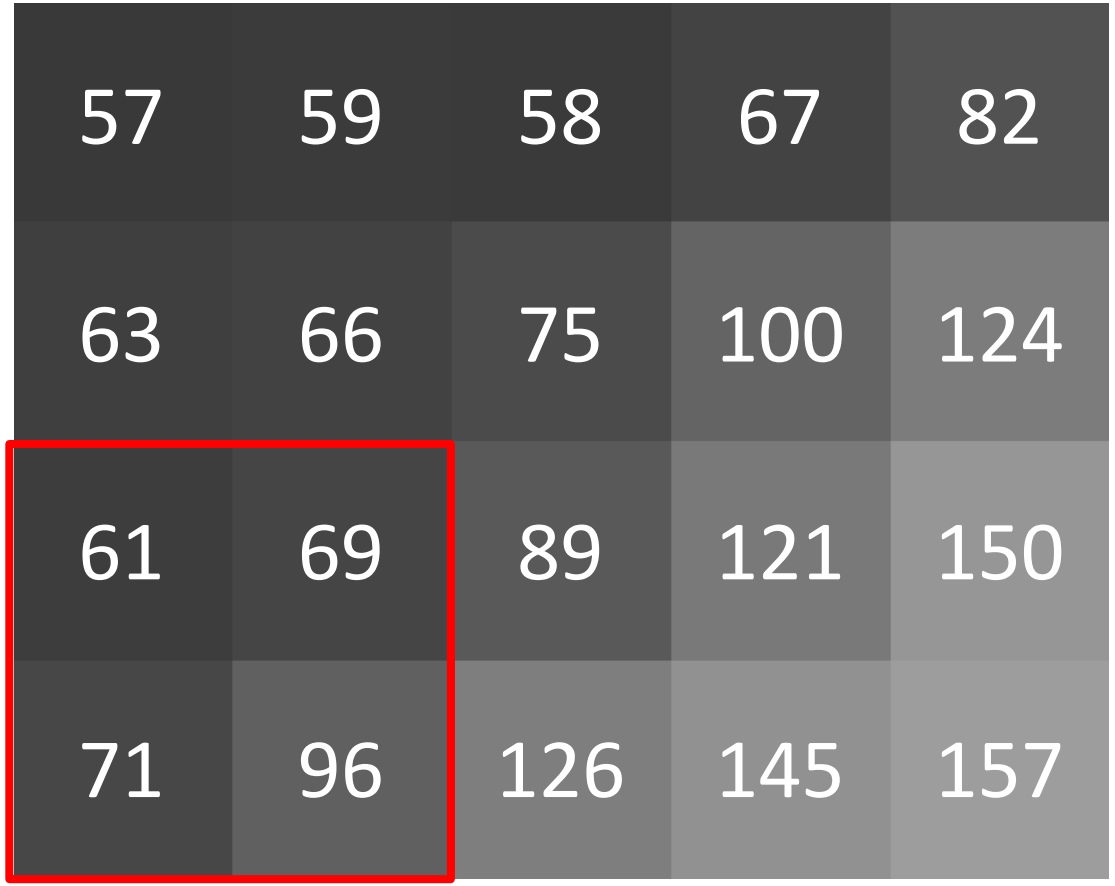
Find average under window:
(58 + 67 + 75 + 100) / 4 = 75

Output

| | |
|---|---|
| 61 | 75 |
| | |

# Average Pooling

Image




Window: 2x2
Stride: 2

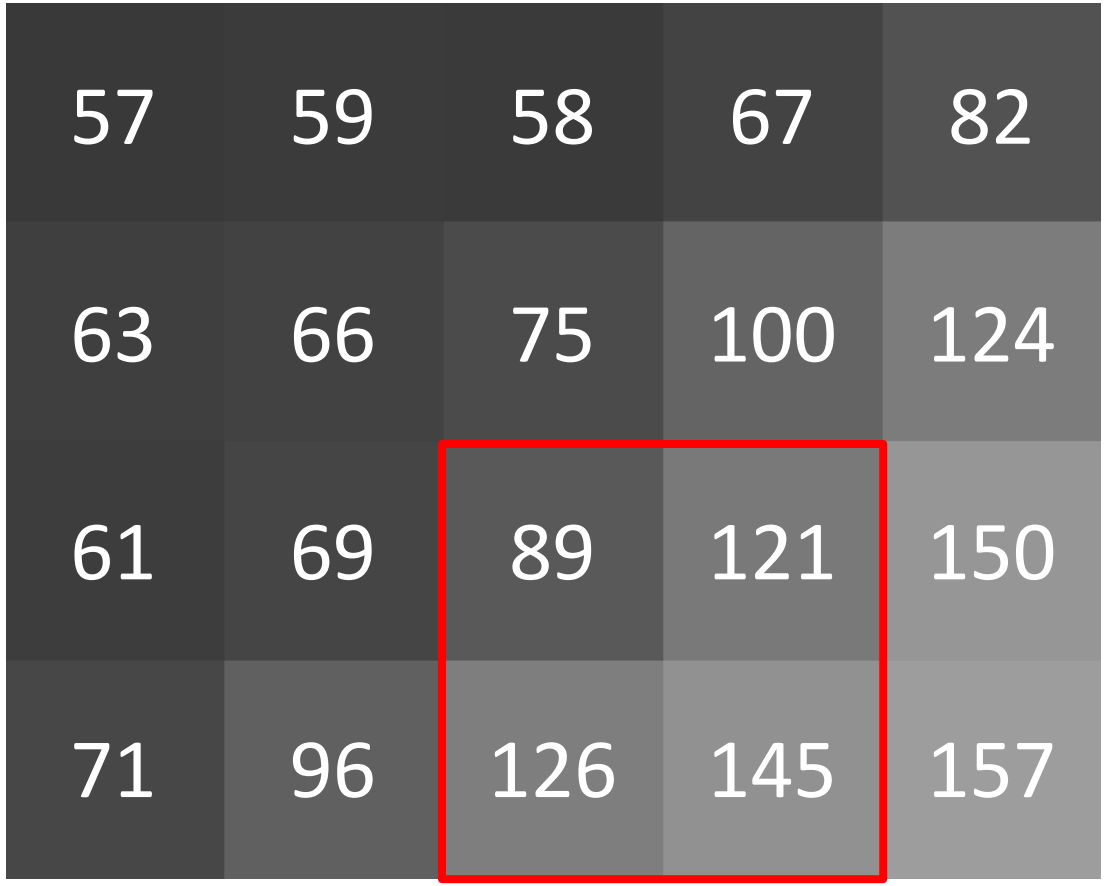
Find average under window:
(61 + 69 + 71 + 96) / 4 = 74.25

Output

| | |
|---|---|
| 61 | 75 |
| 74 | |

# Average Pooling

Image




Window: 2x2
Stride: 2

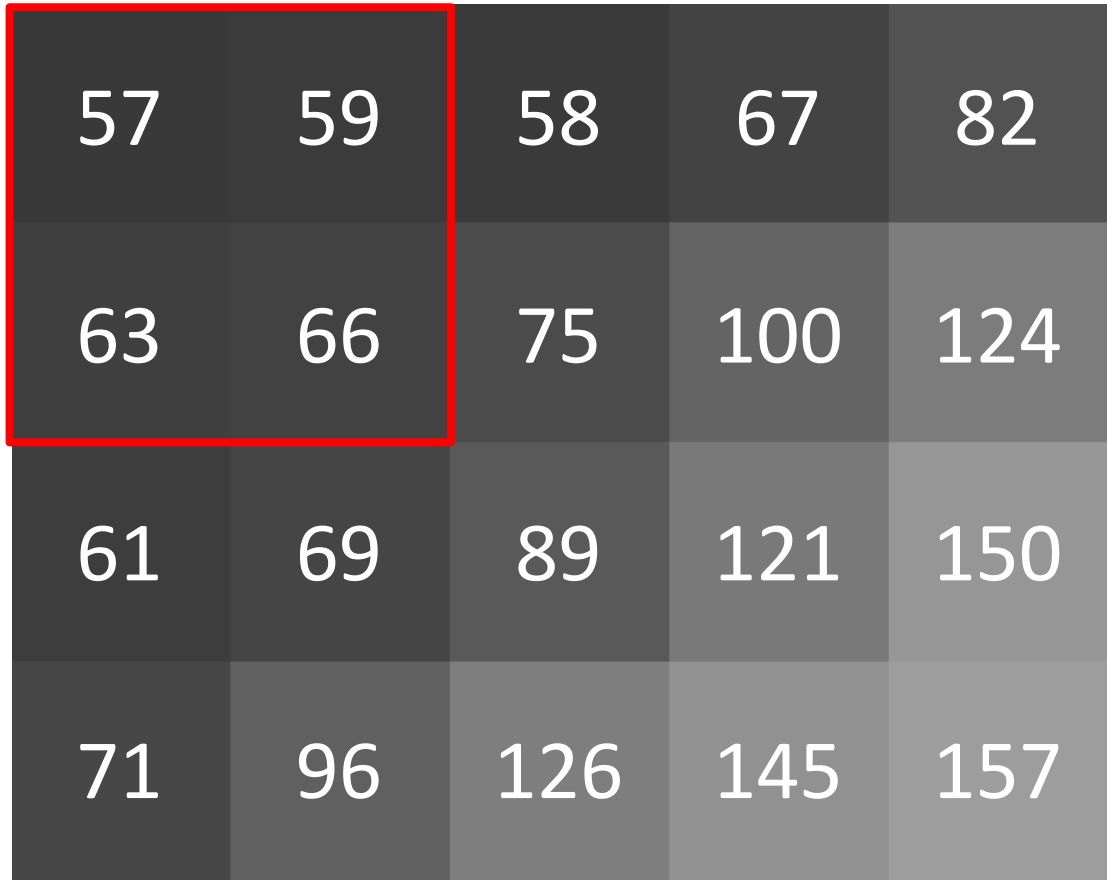
Find average under window:
(89 + 121 + 126 + 145) / 4 = 120.25

Output

| | |
|---|---|
| 61 | 75 |
| 74 | 120 |

# Max Pooling

Image




Window: 2x2
Stride: 2

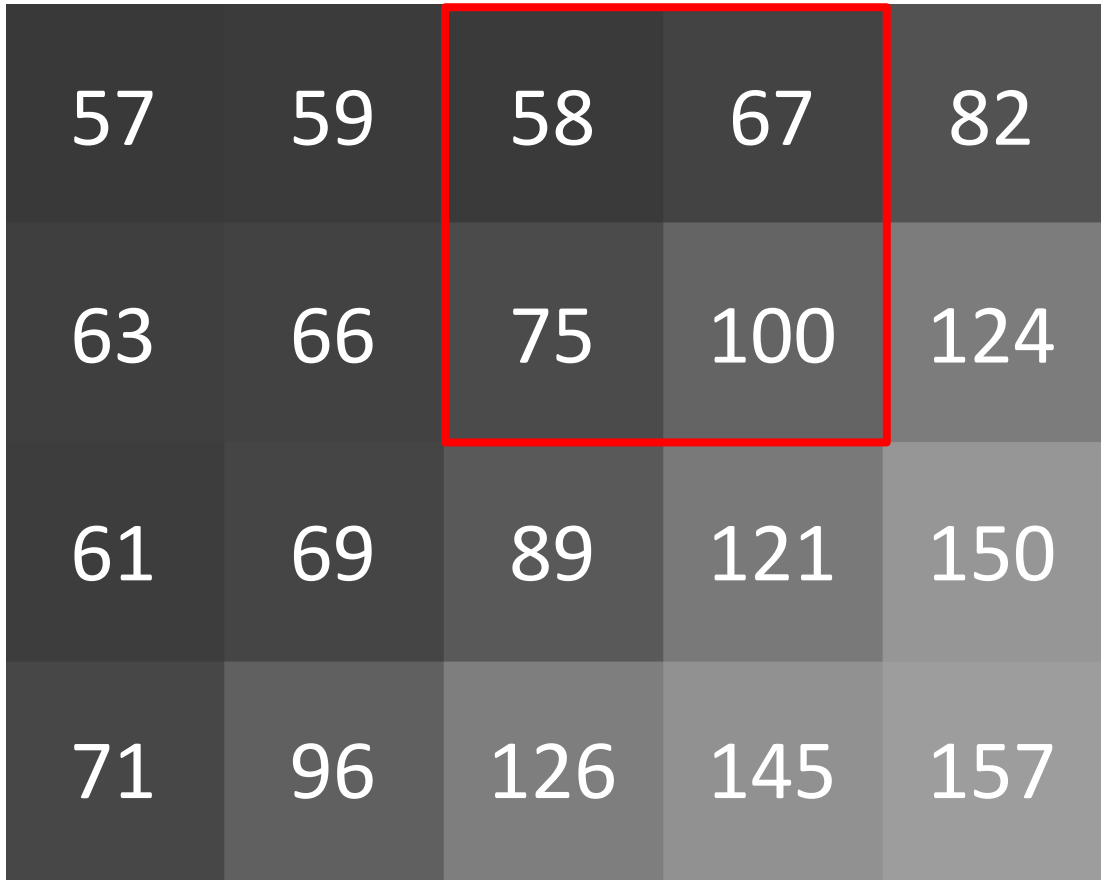
Find maximum value under window: 66

Output

| | |
|---|---|
| 66 | |
| | |

# Max Pooling

Image




Window: 2x2
Stride: 2

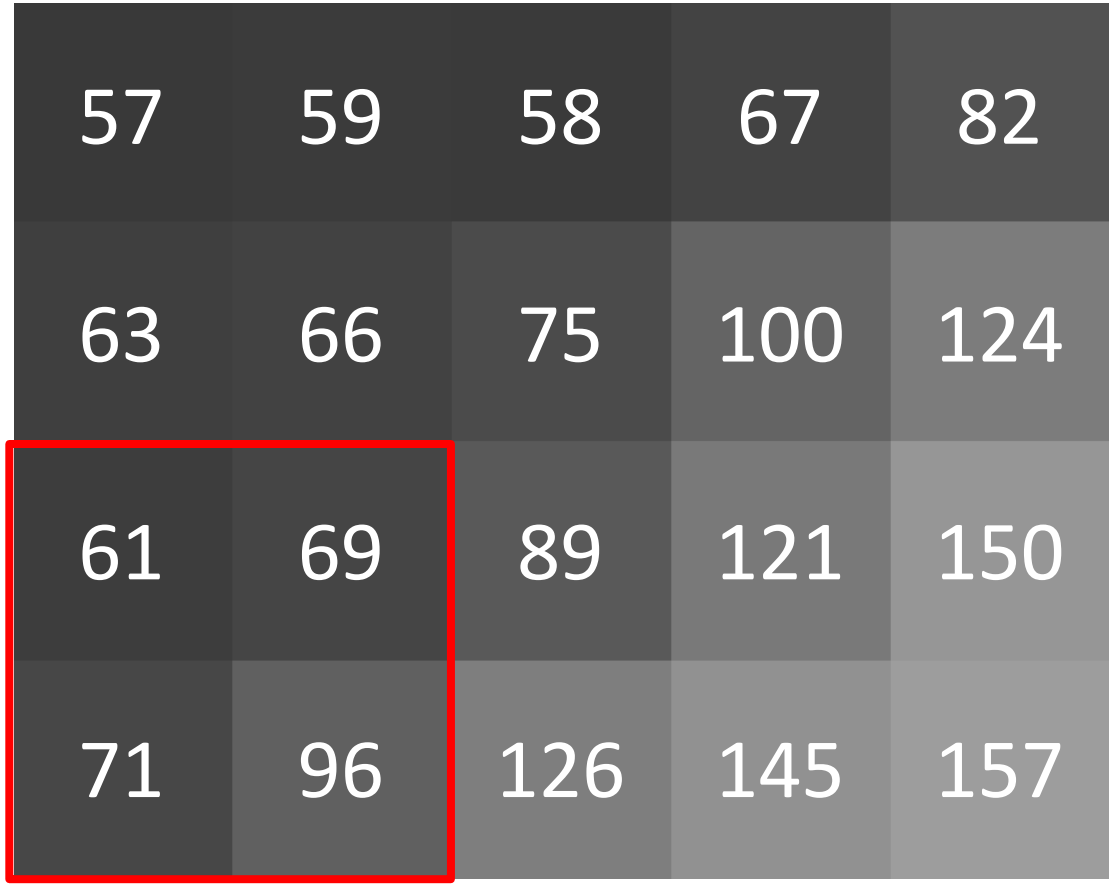
Find maximum value under window: 100

Output

| | |
|---|---|
| 66 | 100 |
| | |

# Max Pooling

Image




Window: 2x2
Stride: 2

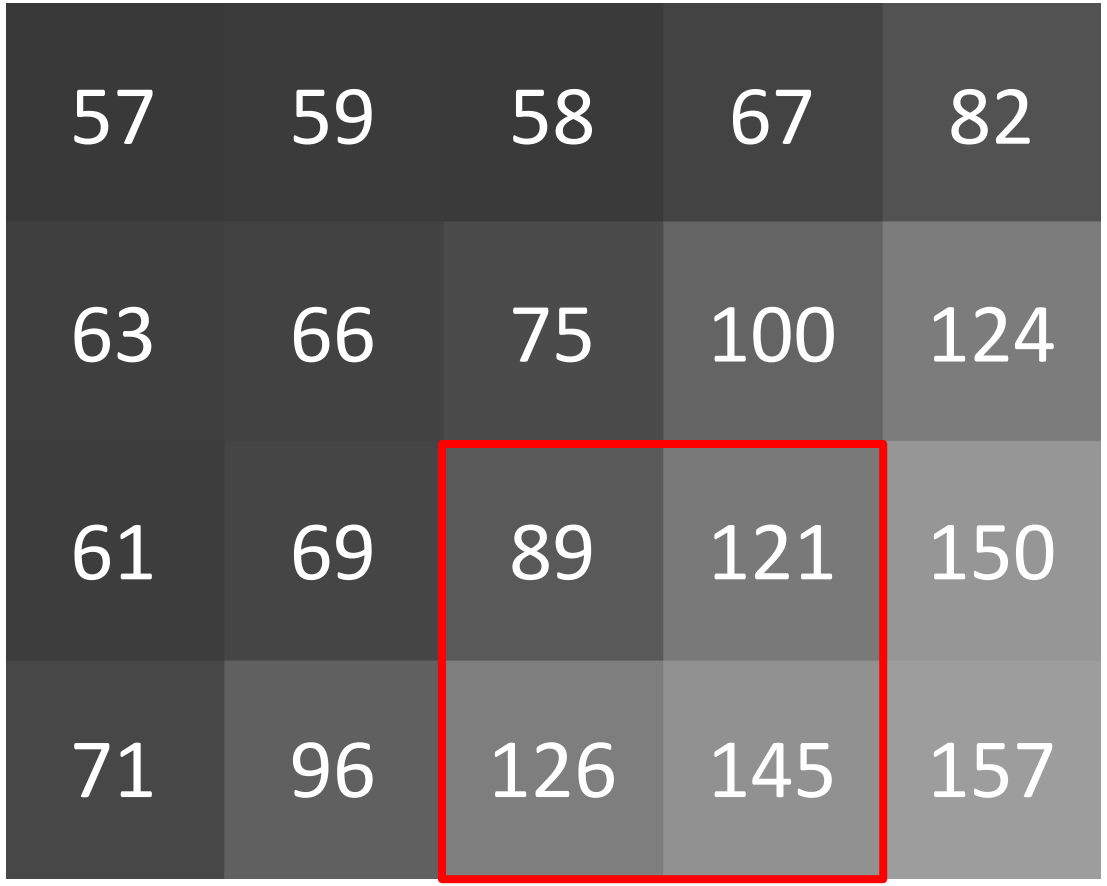
Find maximum value under window: 96

Output

| | |
|---|---|
| 66 | 100 |
| 96 | |

# Max Pooling

Image




Window: 2x2
Stride: 2

Find maximum value under window: 145

Output

| | |
|---|---|
| 66 | 100 |
| 96 | 145 |

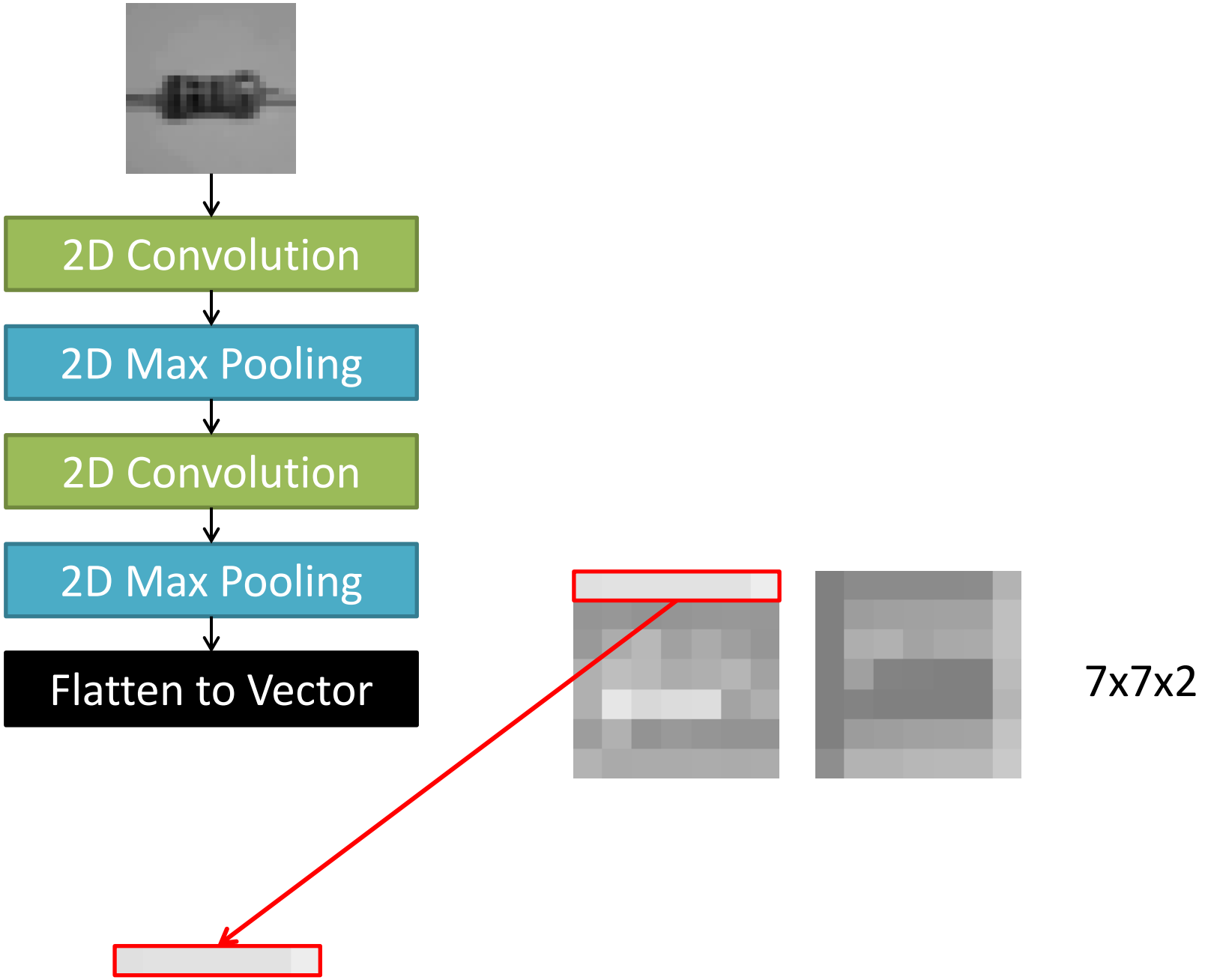
Filtered image
(198x128)

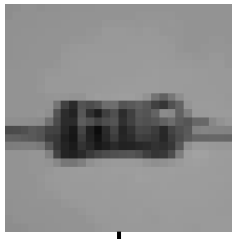
Average Pooling

(99x64)

Max Pooling

(99x64)


2D Convolution
2D Max Pooling
2D Convolution
2D Max Pooling
Flatten to Vector
7x7x2



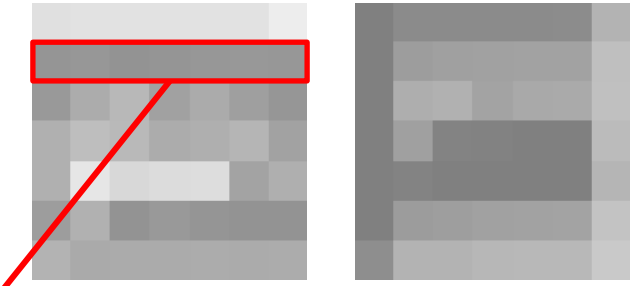
2D Convolution

2D Max Pooling

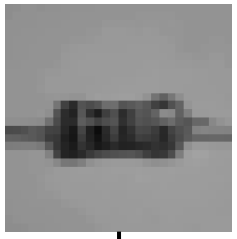
2D Convolution

2D Max Pooling

Flatten to Vector



7x7x2



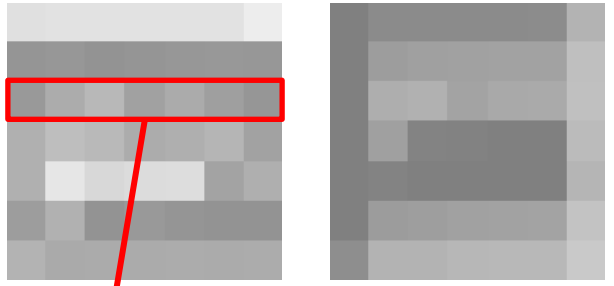
2D Convolution

2D Max Pooling

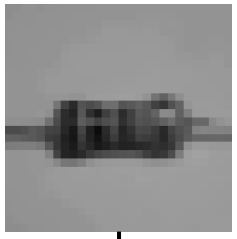
2D Convolution

2D Max Pooling

Flatten to Vector



7x7x2



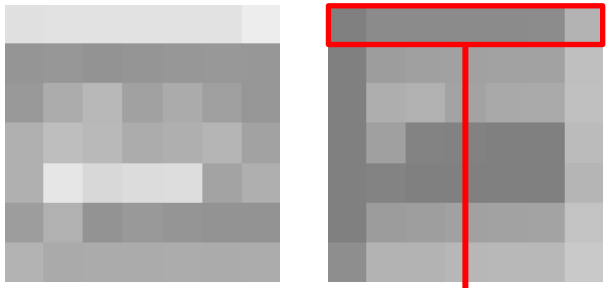
2D Convolution

2D Max Pooling

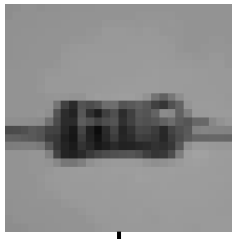
2D Convolution

2D Max Pooling

Flatten to Vector



7x7x2



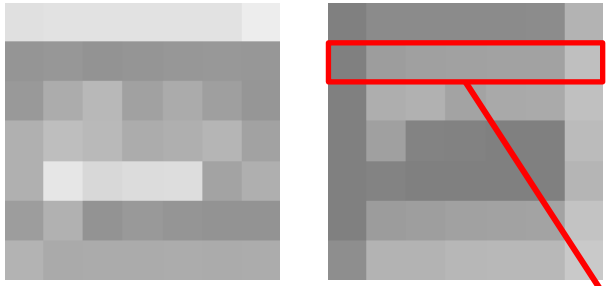
2D Convolution

2D Max Pooling

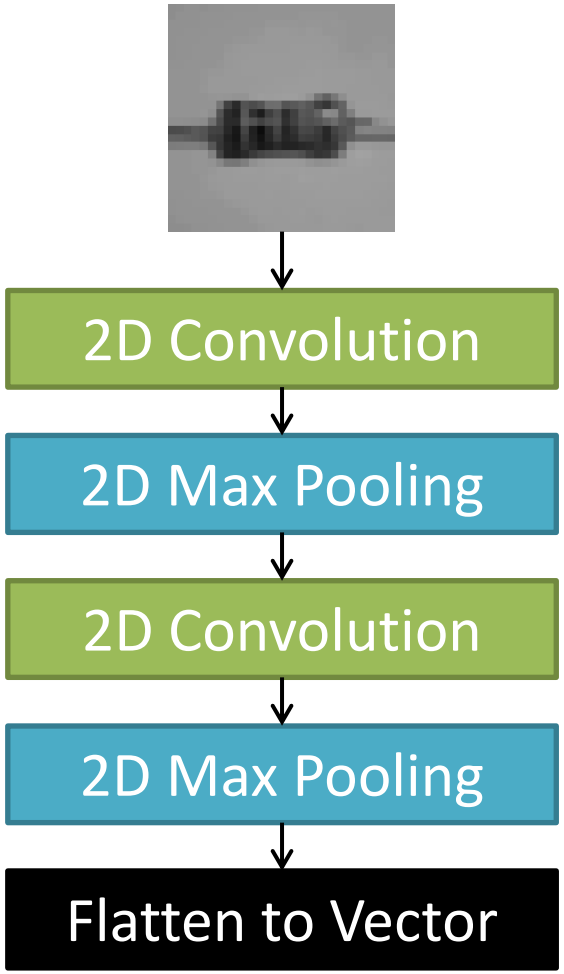
2D Convolution

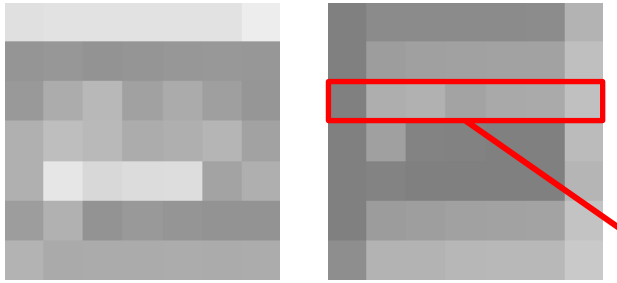
2D Max Pooling

Flatten to Vector



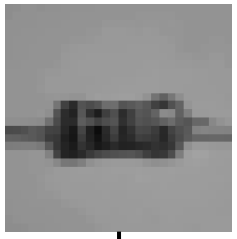
7x7x2


2D Convolution
2D Max Pooling
2D Convolution
2D Max Pooling
Flatten to Vector



7x7x2

1x98



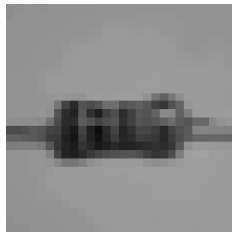
2D Convolution

2D Max Pooling

2D Convolution

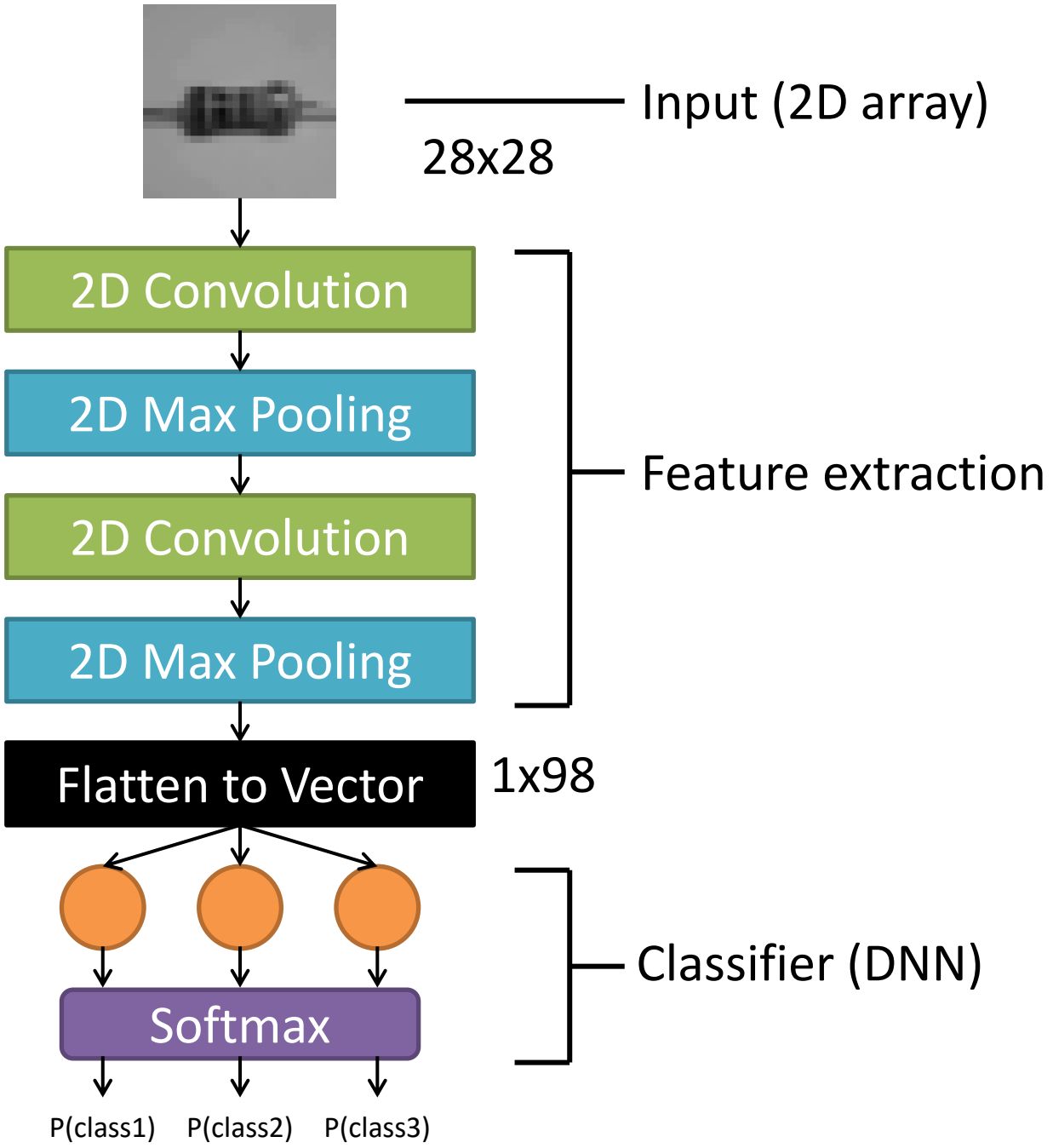
2D Max Pooling

Flatten to Vector



Flatten original image: 28x28 = 784

1x98


Input (2D array)
28x28
2D Convolution
2D Max Pooling
2D Convolution
2D Max Pooling
Feature extraction
Flatten to Vector
1x98
Softmax
Classifier (DNN)
P(class1)
P(class2)
P(class3)

# Model Evaluation

# Confusion Matrix

Predicted Label

| Actual Label | Ball | Dog | Toy |
|---|---|---|---|
| Ball | 205 | 10 | 1 |
| Dog | 6 | 199 | 0 |
| Toy | 9 | 17 | 223 |

**Precision, Positive Predictive Value (PPV):**
What proportion of positive predictions was actually correct?

$$Precision = \frac{TP}{TP + FP} = \frac{\square}{\square + \square}$$

$$= \frac{199}{199 + 27} = 0.881$$

# Confusion Matrix

Predicted Label

| Actual Label | Ball | Dog | Toy |
|---|---|---|---|
| Ball | 205 | 10 | 1 |
| Dog | 6 | 199 | 0 |
| Toy | 9 | 17 | 223 |

**Recall, True Positive Rate (TPR):**
What proportion of actual positives was identified correctly?

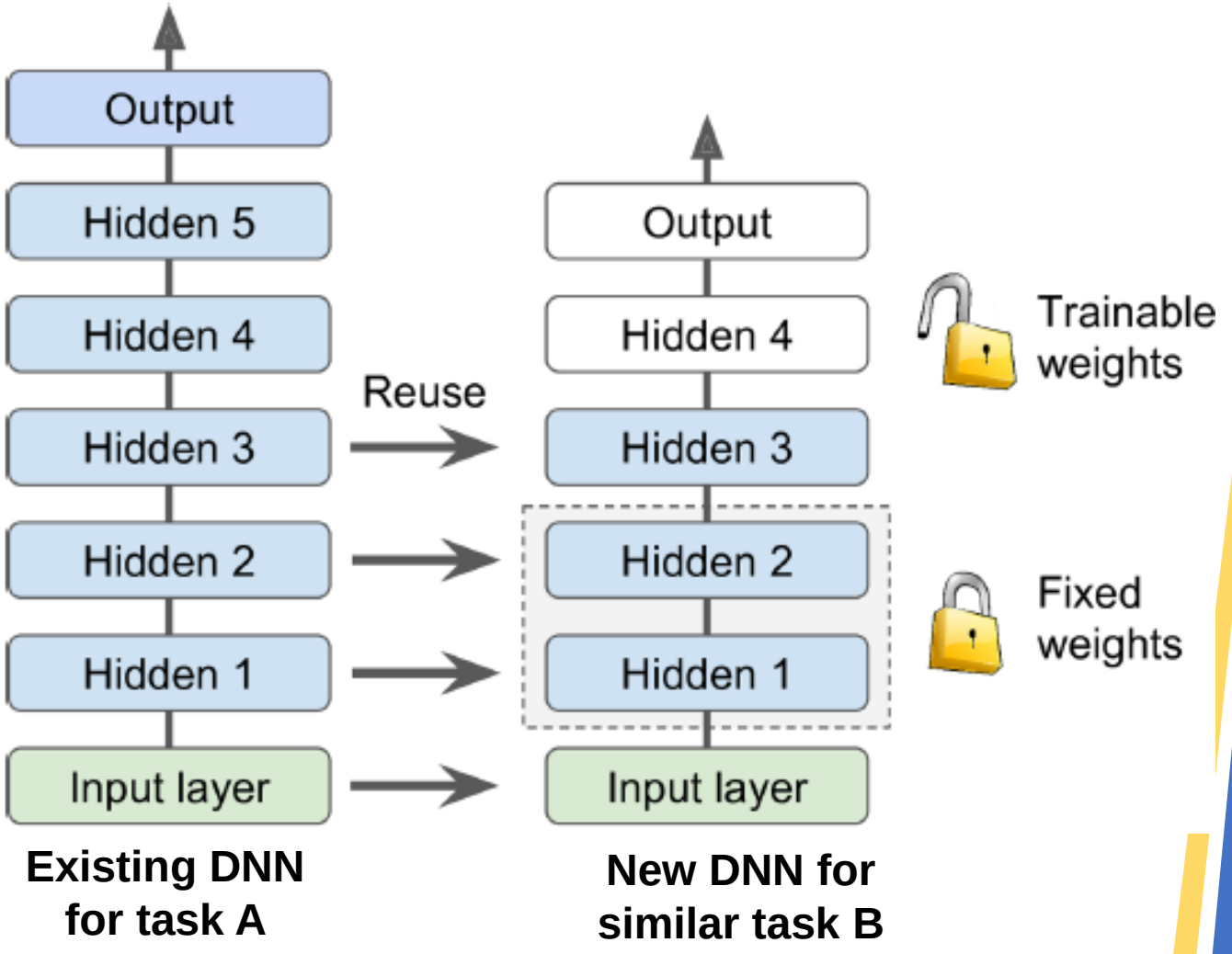
$$Recall = \frac{TP}{TP + FN} = \frac{\square}{\square + \square}$$

$$= \frac{199}{199 + 6} = 0.971$$

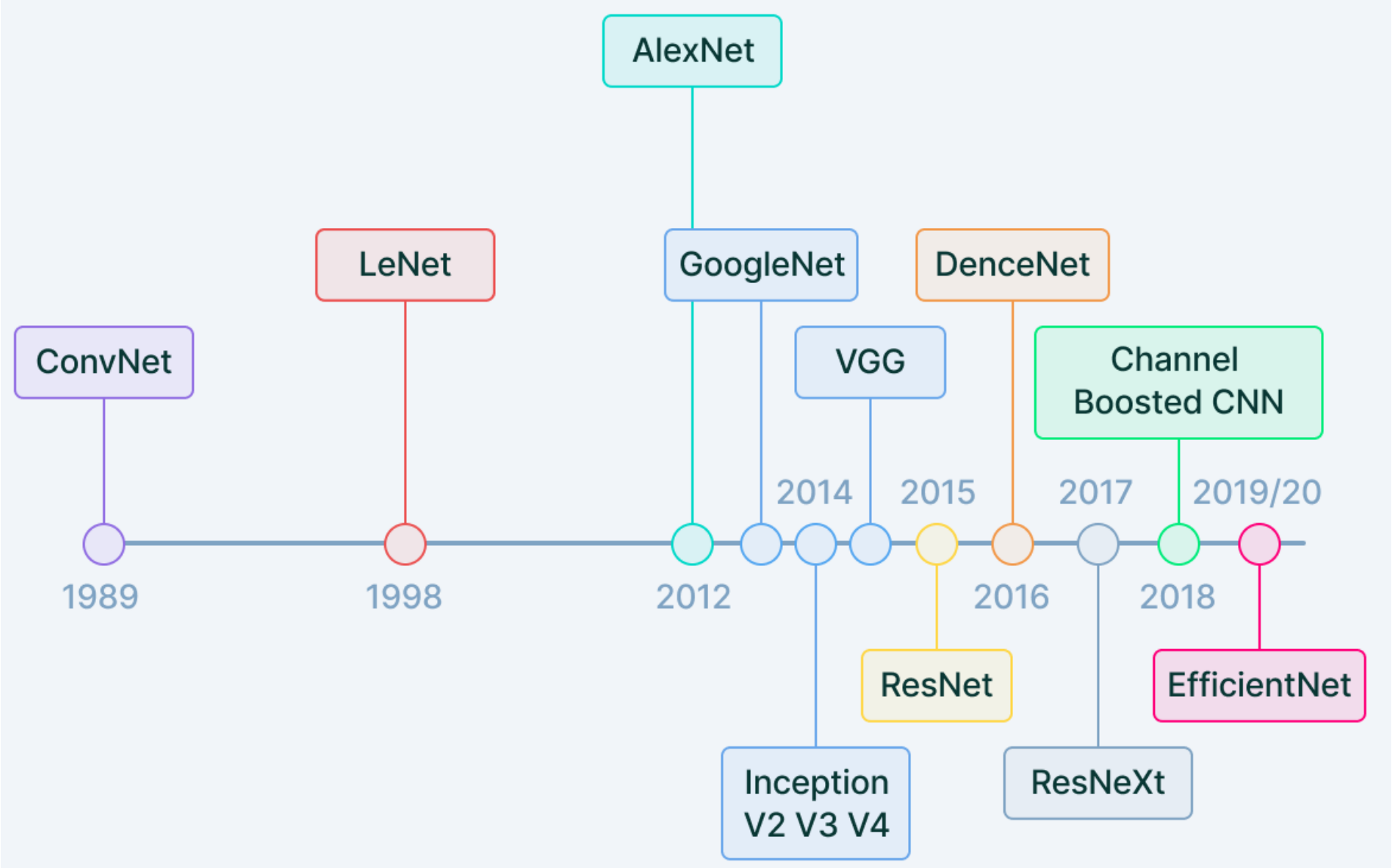
# Transfer learning

technique in ML in which knowledge learned from a task is re-used to boost performance on a related task

- Faster
- Smaller training dataset



# CNN Landscape



# Thank you!