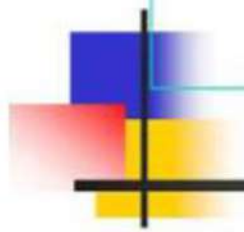
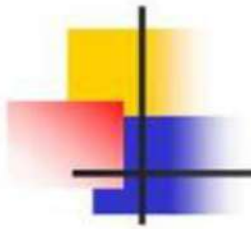


Goodfellow: Chap 5 Machine Learning Basics

Dr. Charles Tappert

Introduction

- Machine Learning
  - Form of applied statistics with extensive use of computers to estimate complicated functions
- Two central approaches
  - Frequentist estimators and Bayesian inference
- Two categories of machine learning
  - Supervised learning and unsupervised learning
- Most machine learning algorithms based on stochastic gradient descent optimization
- Focus on building machine learning algorithms



1 Learning Algorithms

- A machine learning algorithm learns from data
- A computer program learns from
  - experience *E*
  - with respect to some class of tasks *T*
  - and performance measure *P*
  - if it's performance *P* on tasks *T* improves with experience *E*

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? Mitchell (1997) provides the definition "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." One can imagine a very wide variety of experiences E, tasks T, and performance measures P, and we do not make any attempt in this book to provide a formal definition of what may be used for each of these entities. Instead, the following sections provide intuitive descriptions and examples of the different kinds of tasks, performance measures and experiences that can be used to construct machine learning algorithms.

1.1 The Task, *T*

- Machine learning tasks are too difficult to solve with fixed programs designed by humans
- Machine learning tasks are usually described in terms of how the system should process an example where
  - An example is a collection of features measured from an object or event
- Many kinds of tasks
  - Classification, regression, transcription, anomaly detection, imputation of missing values, etc.

Task

- Machine learning tasks are usually described in terms of how the machine learning system should process an **example.**
- An example is a collection of **features** that have been quantitatively measured from some object or event that we want the machine learning system to process.
- Eg: program the robot to learn to walk, driverless car

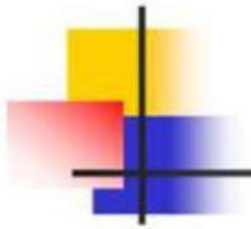
- Many kinds of tasks can be solved with machine learning. Some of the most common machine learning tasks include the following:
- Classification
- Regression
- Machine Translation
- Anomaly detection

1.2 The Performance Measure, *P*

- Performance *P* is usually specific to the Task *T*
- For classification tasks, *P* is usually accuracy
  - The proportion of examples correctly classified
  - Here we are interested in the accuracy on data not seen before – a test set

Performance

- In order to evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually this performance measure *P* is specific to the task *T* being carried out by the system.
- Accuracy- proportion of examples for which the model produces the correct output
- Error rate- the proportion of examples for which the model produces an incorrect output
- Task classification- measures accuracy



1.3 The Experience, *E*

- Most learning algorithms in this book are allowed to experience an entire dataset
- Unsupervised learning algorithms
  - Learn useful structural properties of the dataset
- Supervised learning algorithms
  - Experience a dataset containing features with each example associated with a label or target
    - For example, target provided by a teacher
- Blur between supervised and unsupervised

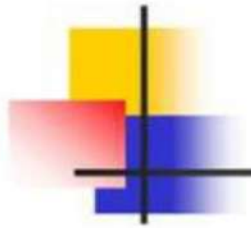
- Supervised algorithm- experience a dataset containing features, but each example is also associated with a **label** or **target**
- **Unsupervised algorithm-** experience a dataset containing many features, then learn useful properties of the structure of this dataset.
- Reinforcement learning- interact with an environment, so there is a feedback loop between the learning system and its experiences

1.4 Example: Linear Regression

- Linear regression takes a vector x as input and predicts the value of a scalar y as it's output
  - Task: predict scalar y from vector x $\quad \hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$
  - Experience: set of vectors X and vector of targets y
  - Performance: mean squared error => normal eqs
  - Solution of the normal equations is

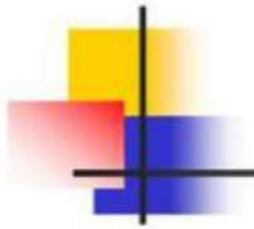
$$\boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})}$$

2 Capacity, Overfitting and Underfitting



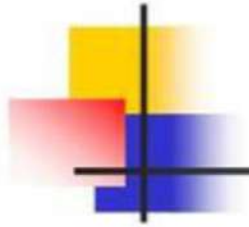
- The central challenge in machine learning is to perform well on new, previously unseen inputs, and not just on the training inputs
- We want the generalization error (test error) to be low as well as the training error
- This is what separates machine learning from optimization

2 Capacity, Overfitting and Underfitting



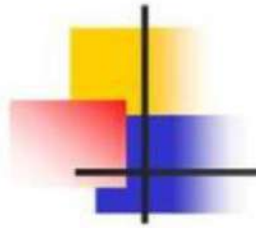
- How can we affect performance on the test set when we get to observe only the training set?
- Statistical learning theory provides answers
- We assume train and test sets are generated independent and identically distributed – i.i.d.
- Thus, expected train and test errors are equal

2 Capacity, Overfitting and Underfitting



- Because the model is developed from the training set the expected test error is usually greater than the expected training error
- Factors determining machine performance are
  - Make the training error small
  - Make the gap between train and test error small
- These two factors correspond to the two challenges in machine learning
  - Underfitting and overfitting

2 Capacity, Overfitting and Underfitting



- Underfitting and overfitting can be controlled somewhat by altering the model's capacity
  - Capacity = model's ability to fit variety of functions
    - Low capacity models struggle to fit the training data
    - High capacity models can overfit the training data
- One way to control the capacity of a model is by choosing its hypothesis space
  - For example, simple linear vs quadratic regression

2 Capacity, Overfitting and Underfitting

- While simpler functions are more likely to generalize (have a small gap between training and test error) we still want low training error
- Training error usually decreases until it asymptotes to the minimum possible error
- Generalization error usually has a U-shaped curve as a function of model capacity

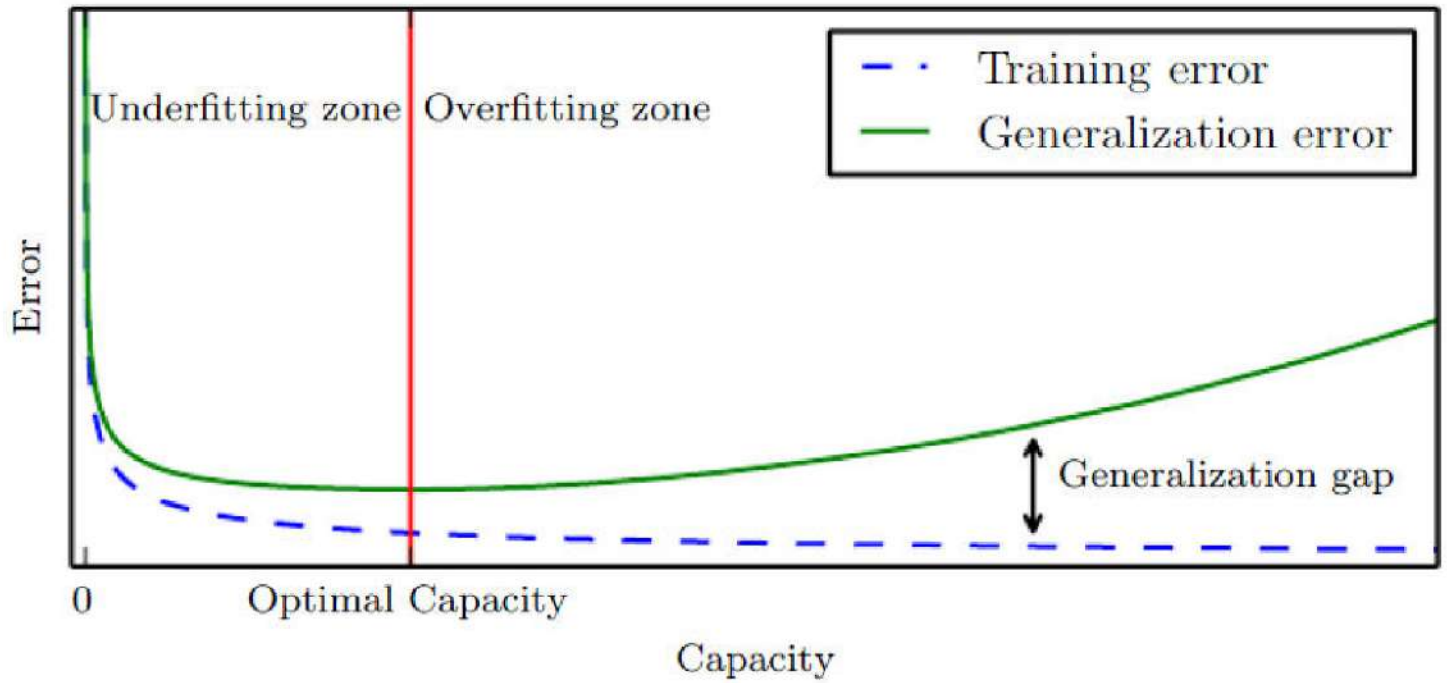
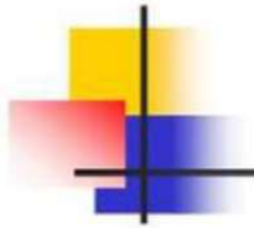


Figure 5.3: Typical relationship between capacity and error. Training and test error behave differently. At the left end of the graph, training error and generalization error are both high. This is the **underfitting regime**. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the **overfitting regime**, where capacity is too large, above the **optimal capacity**.



2 Capacity, Overfitting and Underfitting

- Non-parametric models can reach arbitrarily high capacities
- For example, the complexity of the nearest neighbor algorithm is a function of the training set size
- Training and generalization error vary as the size of the training set varies
  - Test error decreases with increased training set size

Parameters in ML

- Parameter is a value that a model learns from the training data.
- It is used to control the behavior of the algorithm and to make predictions on new data.
- Eg:, in linear regression, the parameters are the coefficients that represent the slope and intercept of the regression line.

Functions

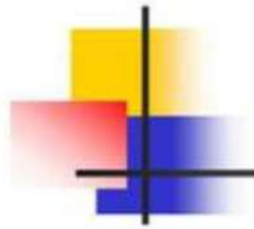
- A function, on the other hand, is a mathematical relationship between the input data and the output data.
- In machine learning, functions can be used to make predictions based on the input data.
- For example, in a decision tree, a function is used to determine the path to follow through the tree to reach a predicted outcome.
- In summary, parameters are the values learned by a machine learning algorithm to make predictions, while functions are the mathematical relationships used by the algorithm to map input data to output data.

Function Estimation As we mentioned above, sometimes we are interested in performing function estimation (or function approximation). Here we are trying to predict a variable $\boldsymbol{y}$ given an input vector $\boldsymbol{x}$. We assume that there is a function $f(\boldsymbol{x})$ that describes the approximate relationship between $\boldsymbol{y}$ and $\boldsymbol{x}$. For example, we may assume that $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ stands for the part of $\boldsymbol{y}$ that is not predictable from $\boldsymbol{x}$. In function estimation, we are interested in approximating f with a model or estimate $\hat{f}$. Function estimation is really just the same as estimating a parameter $\boldsymbol{\theta}$; the function estimator $\hat{f}$ is simply a point estimator in function space. The linear regression example (discussed above in section 5.1.4) and the polynomial regression example (discussed in section 5.2) are both examples of scenarios that may be interpreted either as estimating a parameter $\boldsymbol{w}$ or estimating a function $\hat{f}$ mapping from $\boldsymbol{x}$ to y.

- Parametric machine learning models make assumptions about the underlying distribution of the data and have a fixed number of parameters, which are estimated from the training data. Examples of parametric models include linear regression, logistic regression, and Naive Bayes.
- Non-parametric machine learning models, on the other hand, do not make assumptions about the underlying distribution of the data and can have a variable number of parameters, which are not fixed prior to training. Eg: nearest neighbor

2.1 No Free Lunch Theorem

- Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points
- In other words, no machine learning algorithm is universally any better than any other
- However, by making good assumptions about the encountered probability distributions, we can design high-performing learning algorithms
  - This is the goal of machine learning research



3 Hyperparameters and Validation Sets

- Most machine learning algorithms have several settings, called hyperparameters, to control the behavior of the algorithm

3 Hyperparameters and Validation Sets

- To avoid overfitting during training, the training set is often split into two sets
  - Called the training set and the validation set
  - Typically, 80% of the training data is used for training and 20% for validation
- Cross-Validation
  - For small datasets, *k*-fold cross validation partitions the data into *k* non-overlapping subsets
    - *k* trials are run, train on *(k-1)/k* of the data, test on *1/k*
    - Test error is estimated by averaging error on each run

Hyperparameters

- Hyperparameters are the settings or configurations of a machine learning algorithm that cannot be learned from the data during training.
- Instead, they need to be set by the user prior to training.
- The choice of hyperparameters can significantly affect the performance of the model, and tuning them is an important part of the machine learning workflow.
- Eg: Neural Networks: number of layers, number of neurons per layer, activation function, learning rate, regularization strength
- Decision Trees: maximum depth of the tree, minimum number of samples required to split a node, minimum number of samples required to be at a leaf node

Cross validation

- *k*-fold cross-validation
- partition of the dataset by splitting it into *k* non-overlapping subsets.
- The test error may then be estimated by taking the average test error across *k* trials.
- On trial *i*, the *i* -th subset of the data is used as the test set and the rest of the data is used as the training set

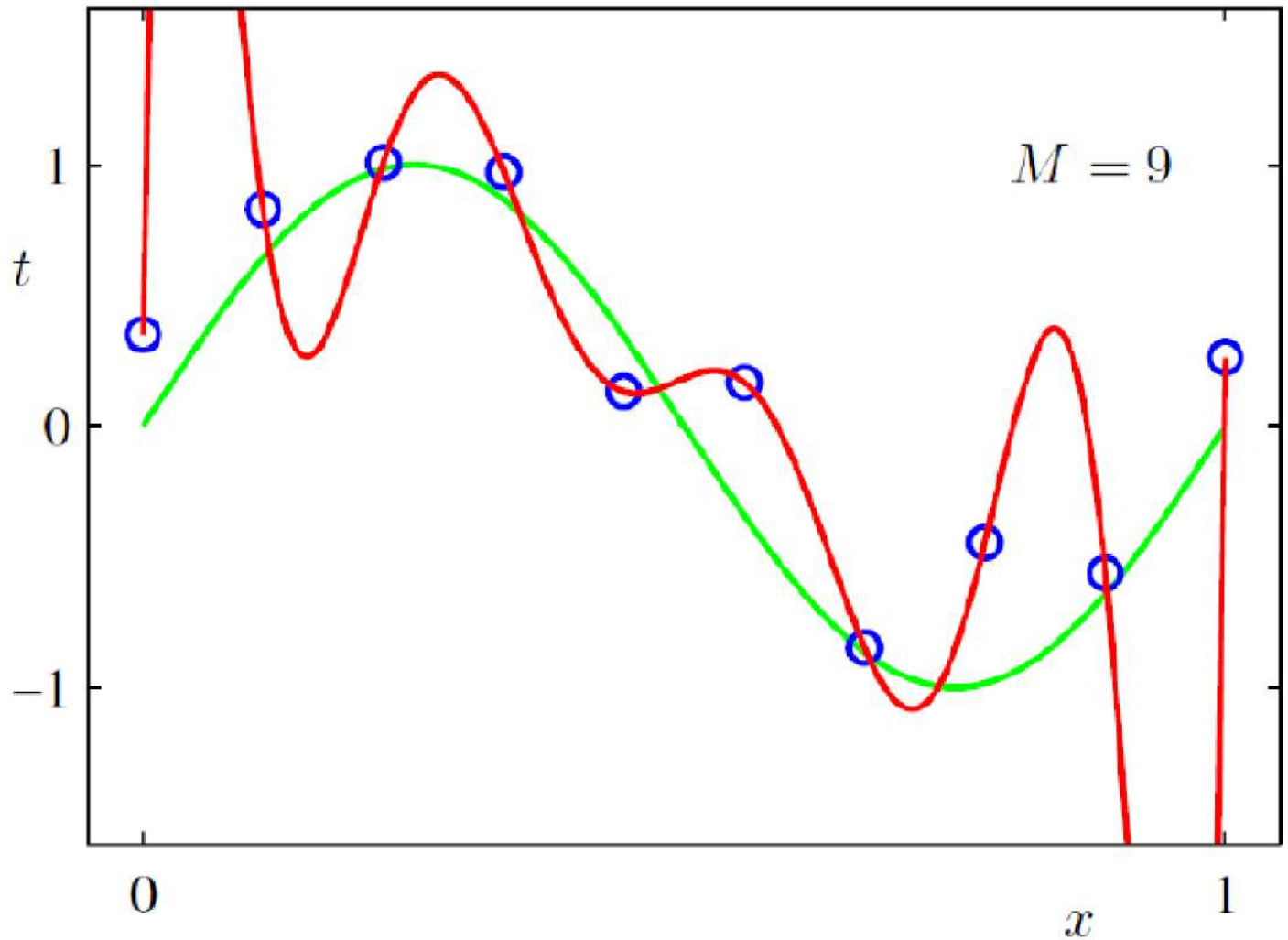
Regularization

- When building models, data scientists and statisticians often talk about penalty, regularization and shrinkage. What do these terms mean and why are they important?

- **The goal of machine learning is to model the pattern and ignore the noise. Anytime an algorithm is trying to fit the noise in addition to the pattern, it is overfitting.**

Overfitting example

$$t = \sin(2\pi x) + \epsilon$$



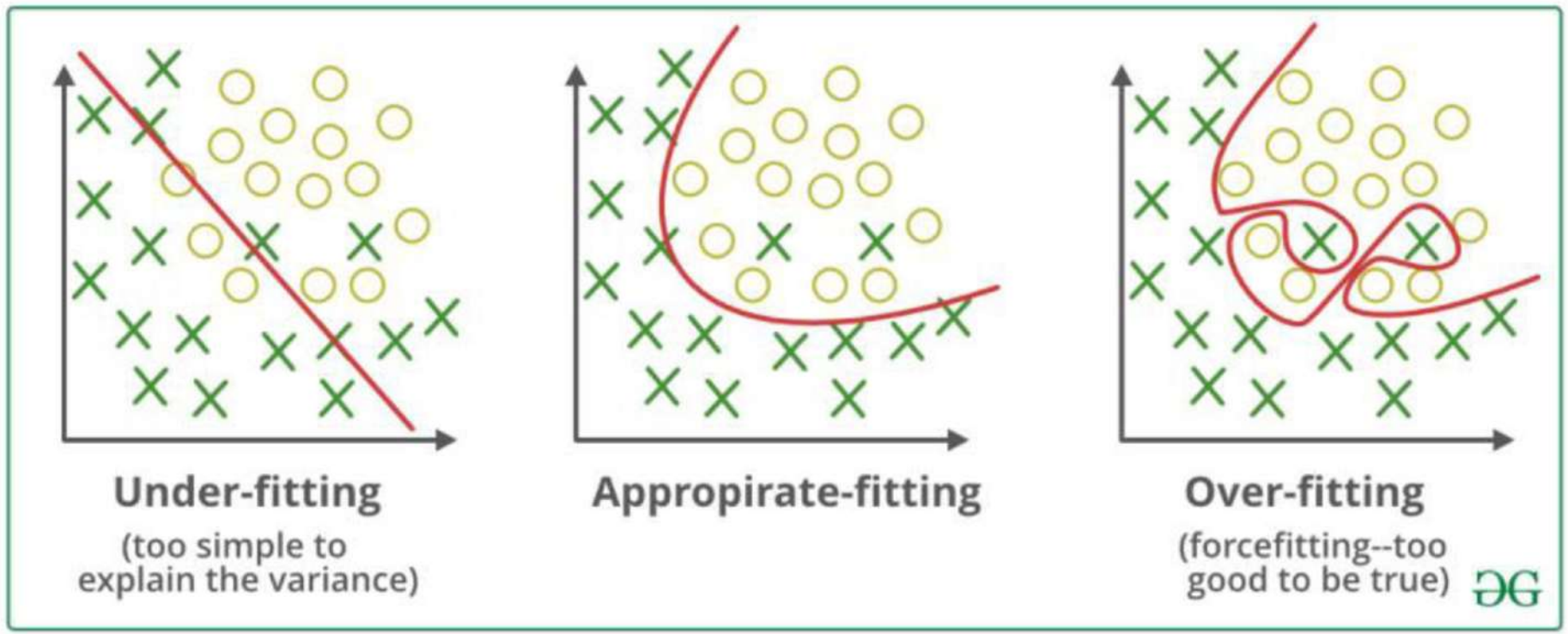
- **Regularization "refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent over fitting.**

- This information usually comes in the form of a penalty for complexity, such as restrictions for smoothness or bounds on the vector space norm."

  - In general: any method to prevent overfitting or help the optimization

  - Specifically: additional terms in the training optimization objective to prevent overfitting or help the optimization

Overfitting is a phenomenon that occurs when a Machine Learning model is constraint to training set and not able to perform well on unseen data.



Regularization

- The main idea behind the regularization technique is **to penalize complex models, i.e.- to define a penalty function to quantify complexity of the model.** (the more complex models will have a greater penalty associated with them). Since most of the training algorithms are considered an optimization problem where the loss is minimized we add the penalty term and minimize the whole expression together.

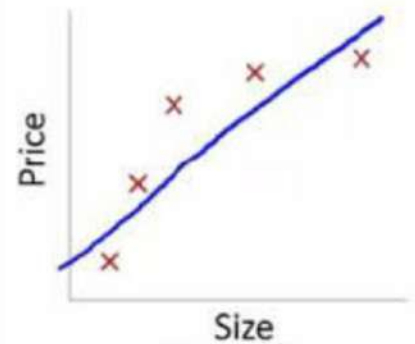
Bias-Variance

Bias-Variance Tradeoff

Bias is the amount of error introduced by approximating real-world phenomena with a simplified model.

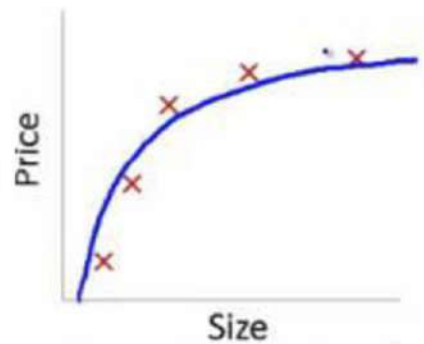
Variance is how much your model's test error changes based on variation in the training data. It reflects the model's sensitivity to the idiosyncrasies of the data set it was trained on.

As a model increases in complexity and it becomes more wiggly (**flexible**), its bias decreases (it does a good job of explaining the training data), but variance increases (it doesn't generalize as well). **Ultimately, in order to have a good model, you need one with low bias and low variance.**



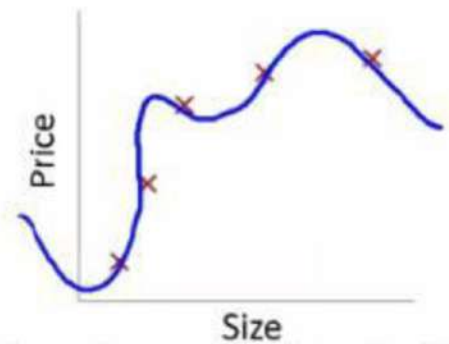
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

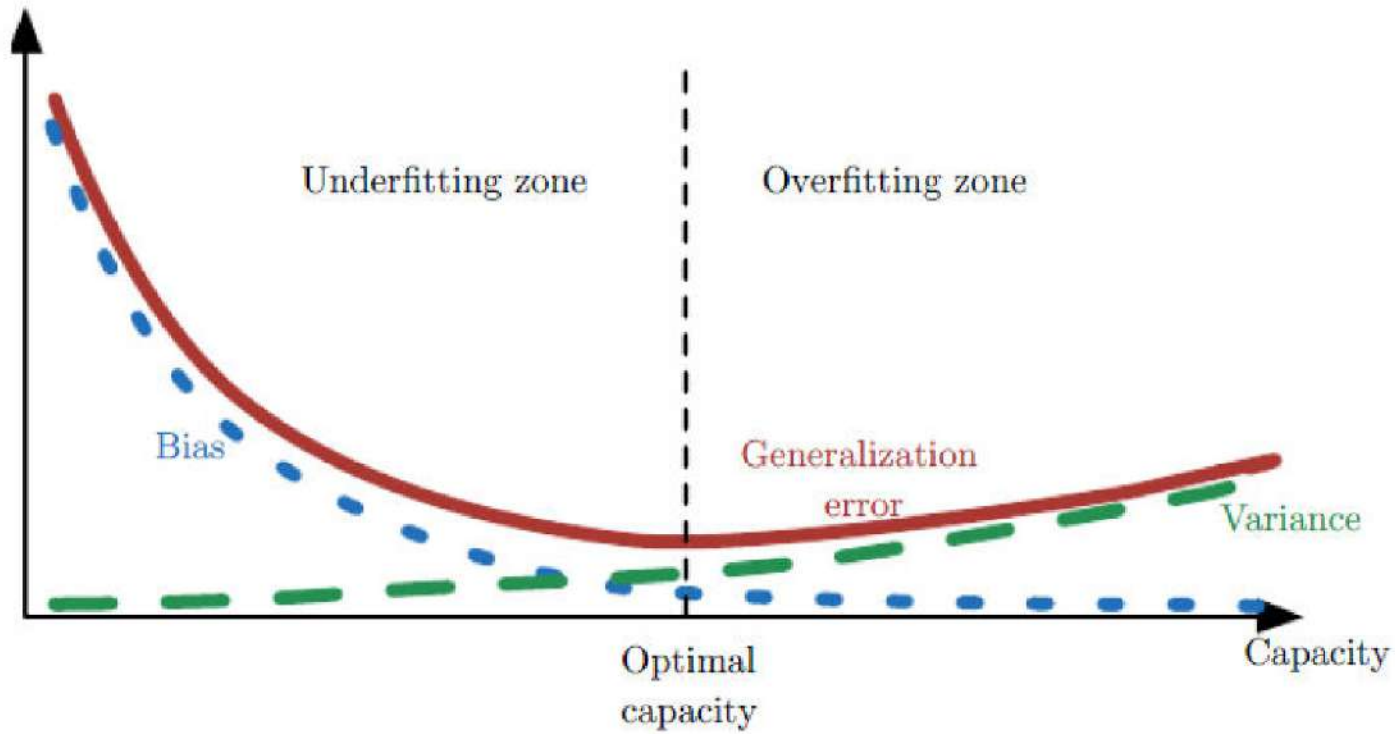
"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

- Bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between the model's ability to fit the training data (bias) and its ability to generalize to new, unseen data (variance)
- A model with high bias tends to underfit the data, while a model with high variance tends to overfit the data.
- The goal is to find the optimal balance between bias and variance to achieve the best possible predictive performance on new data.
- Techniques such as regularization, cross-validation, can help to mitigate the bias-variance tradeoff.


Underfitting zone
Overfitting zone
Bias
Generalization
error
Variance
Optimal
capacity
Capacity

Introduction to Convolution

Contents

- Image
- Filter/mask/kernel/template/window
- Convolution
- Padding
- Stride
- Types of filters

What's an image?

"A lady is crossing the road "

6 words, 3 stop words, 13 char sequence

"An image is worth 1000 words"

"An image is worth 1,00,000 words"

600X300 image,3 channels, 5,40,000 pixel intensity values

Image

255	144	…	120
57	100	…	32
…	…	…	…
251	41	…	23
57	209	…	118
27	59	…	19
…	…	…	…
4	203	…	69
35	170	…	199
17	11	…	97
…	…	…	…
145	165	…	210

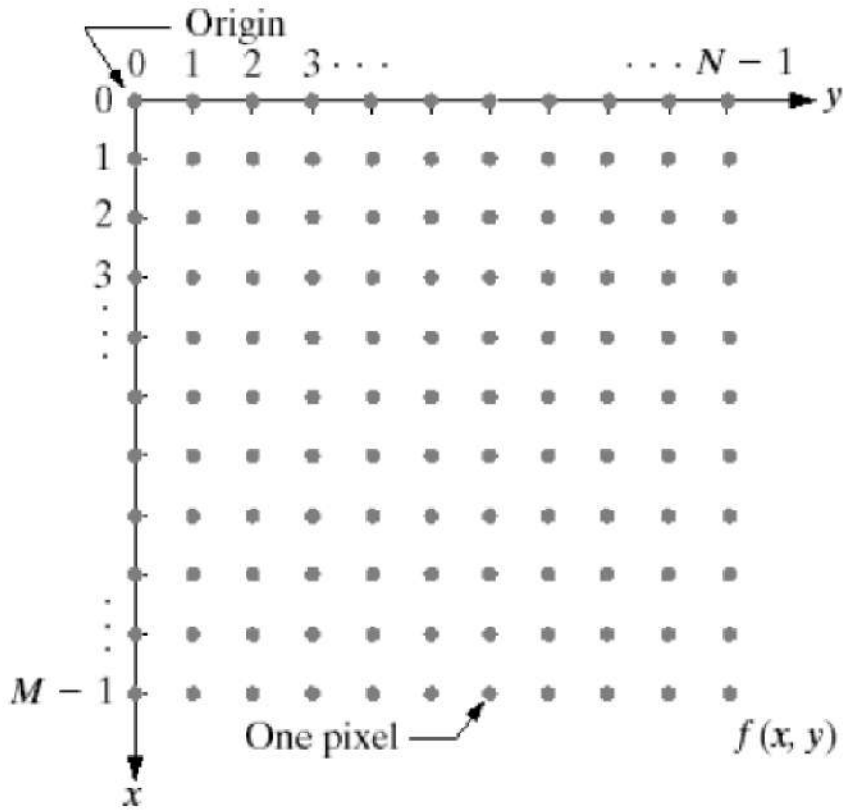
What is an image

- An image is nothing more than a two dimensional signal.
- It is defined by the mathematical function f(x,y) where x and y are the two co-ordinates horizontally and vertically.
- The value of f(x,y) at any point gives the pixel value at that point of an image.

Fundamentals of Digital Images

- **An image**: **a multidimensional function of spatial coordinates**.
- **Spatial coordinate**: (x,y) for 2D case such as photograph,
  (x,y,z) for 3D case such as CT scan images
  (x,y,t) for movies
- The function f may represent intensity (for monochrome images) or color (for color images) or other associated values.

Conventional Coordinate for Image Representation



- This image is a two dimensional array of numbers ranging between 0 and 255.

128	30	123
232	123	321
123	77	89
80	255	255

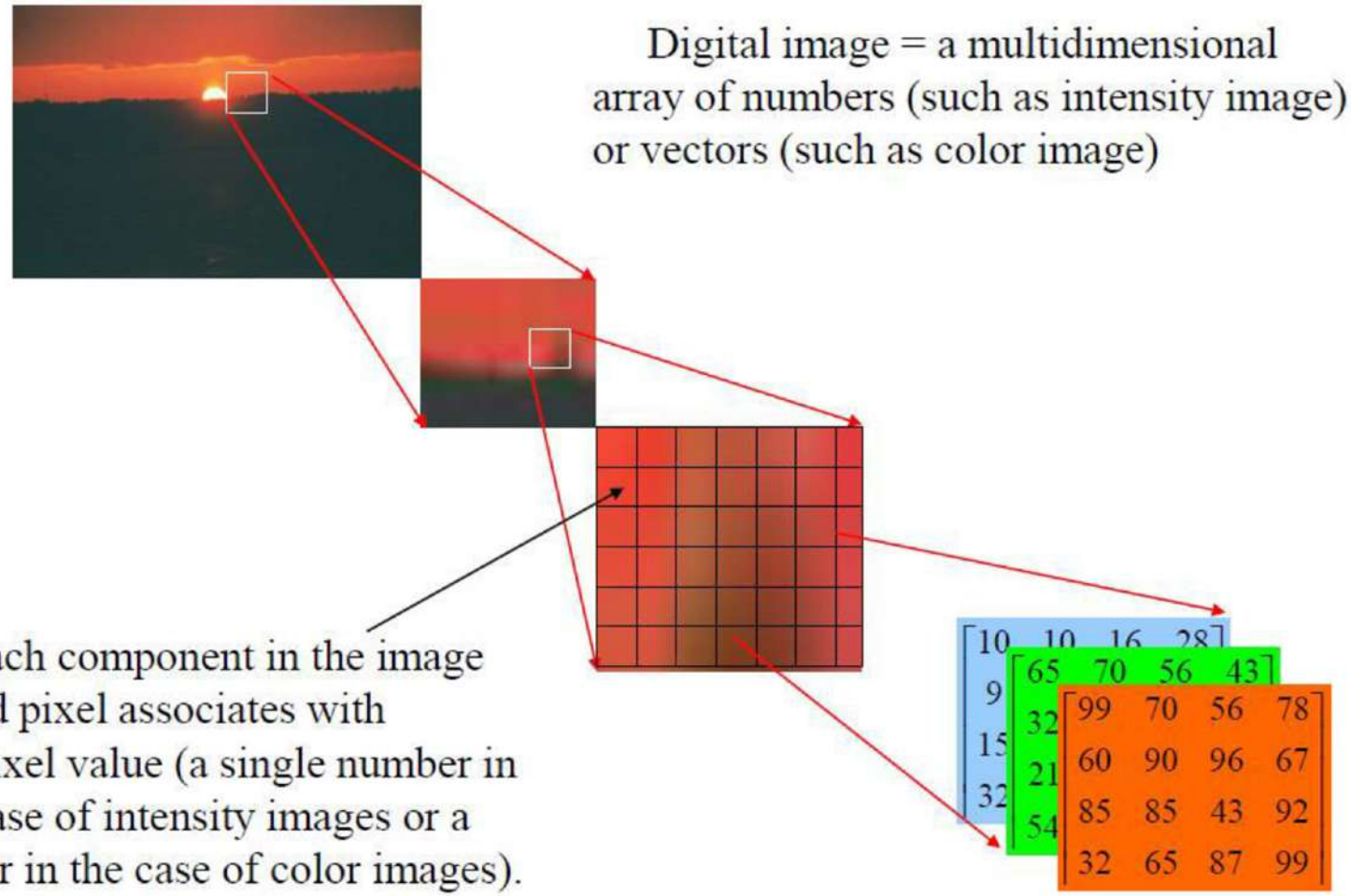
- Each number represents the value of the function f(x,y) at any point. In this case the value 128 , 230 ,123 each represents an individual pixel value.
- The dimensions of the picture is actually the dimensions of this two dimensional array.

What We See

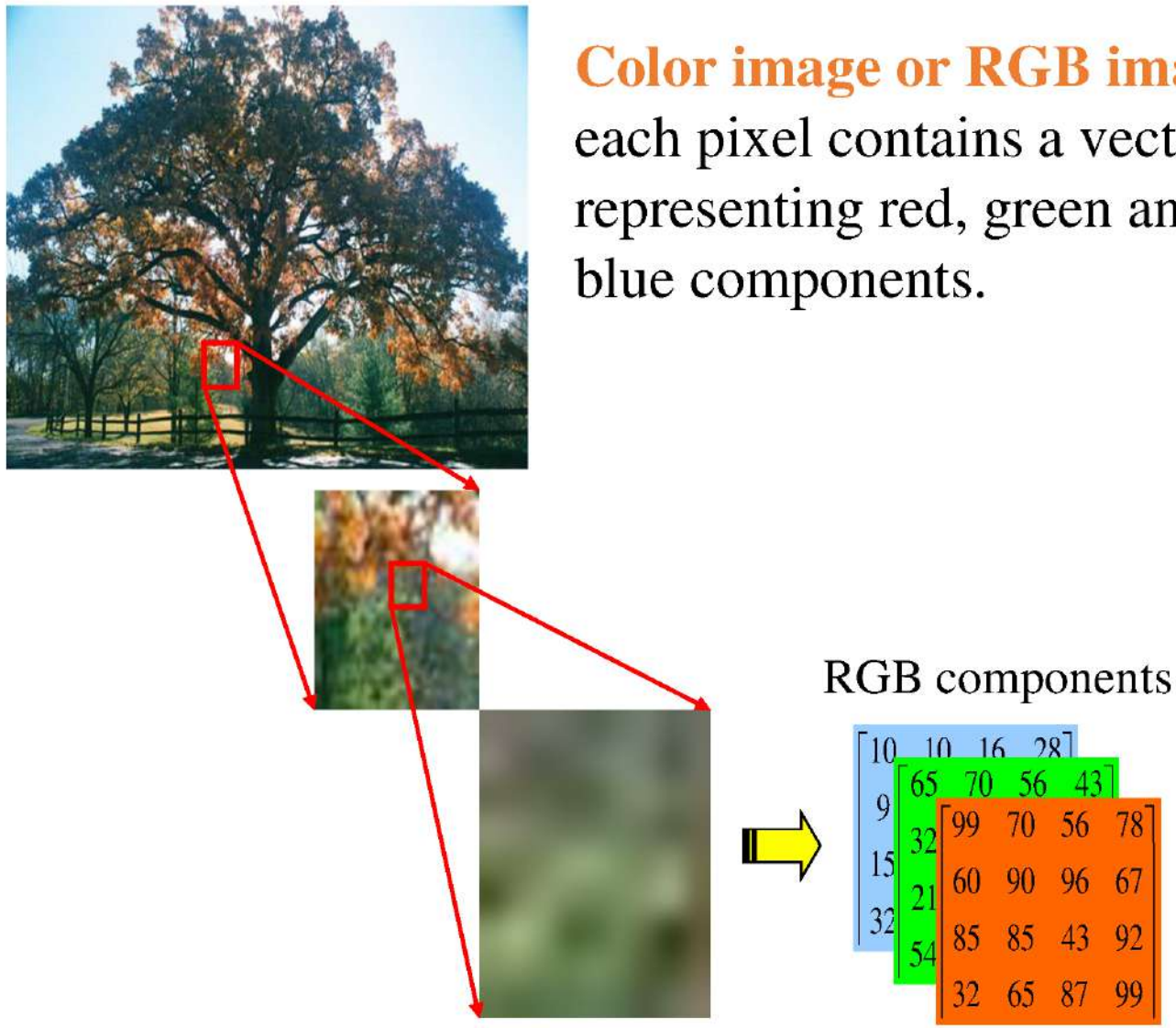
```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

Digital Image



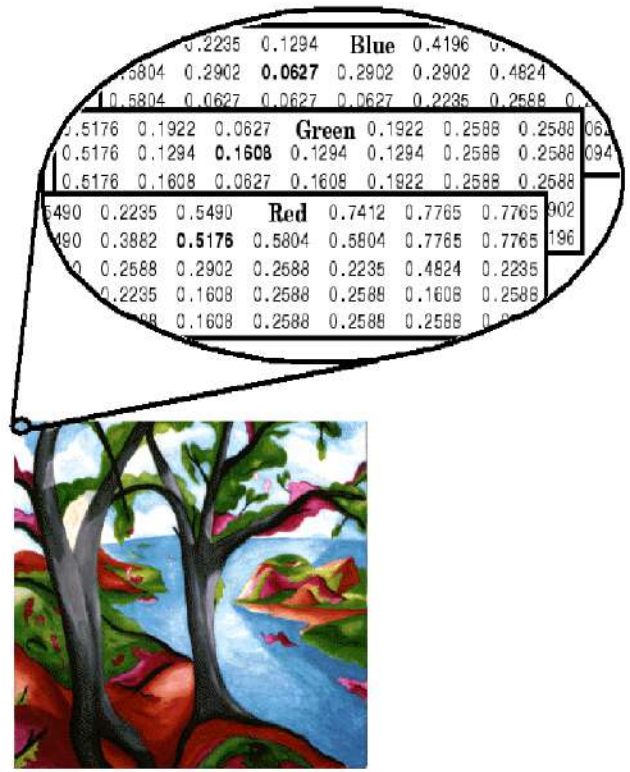
Digital Image Types : RGB Image

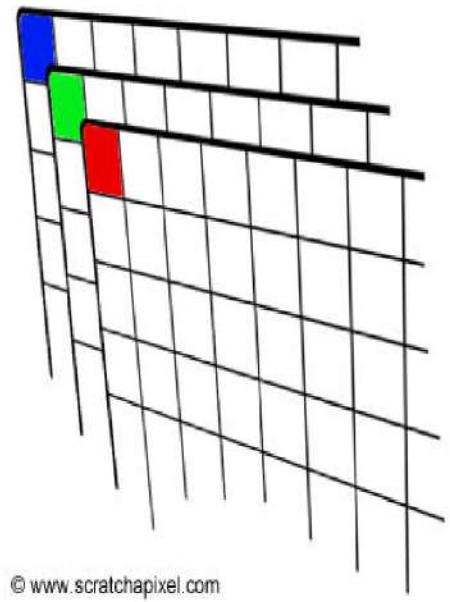


Color image or RGB image: each pixel contains a vector representing red, green and blue components.

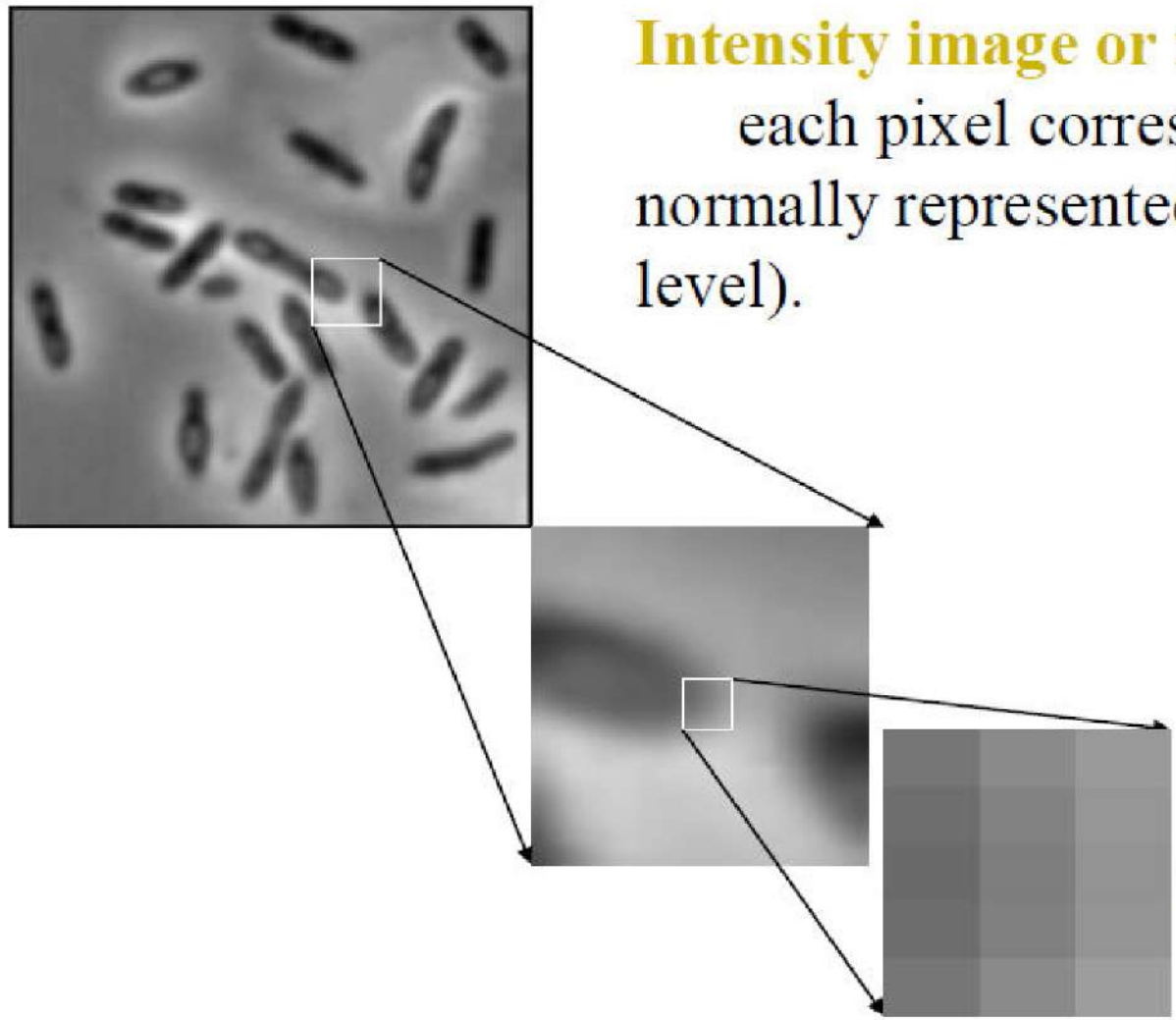
RGB image

- Image – 32 X 32 X 3





Digital Image Types : Intensity Image



Intensity image or monochrome image

each pixel corresponds to light intensity normally represented in gray scale (gray level).

Gray scale values

$$\begin{bmatrix} 10 & 10 & 16 & 28 \\ 9 & 6 & 26 & 37 \\ 15 & 25 & 13 & 22 \\ 32 & 15 & 87 & 39 \end{bmatrix}$$

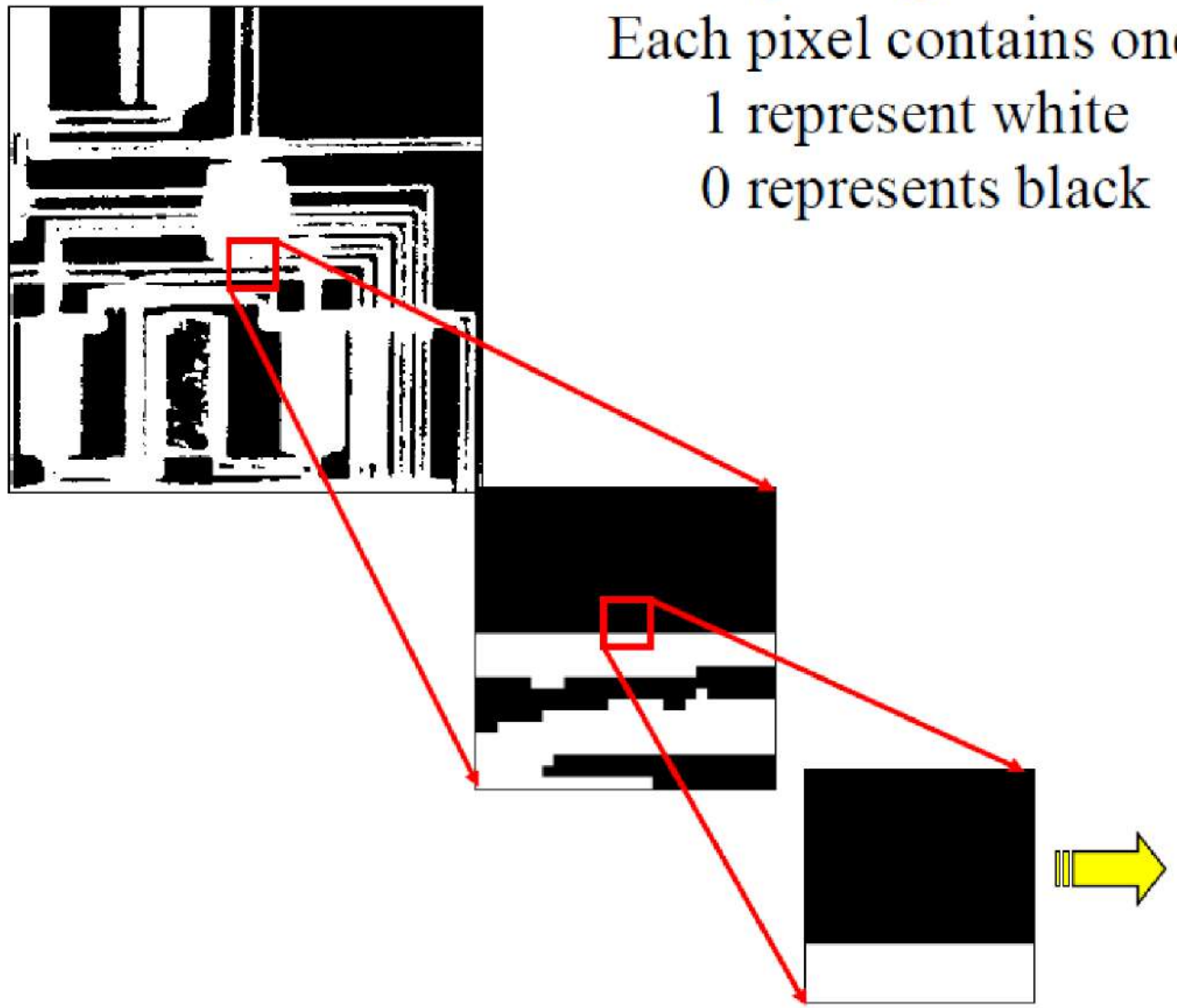
Image Types : Binary Image

Binary image or black and white image

Each pixel contains one bit :

1 represent white

0 represents black



Binary data

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

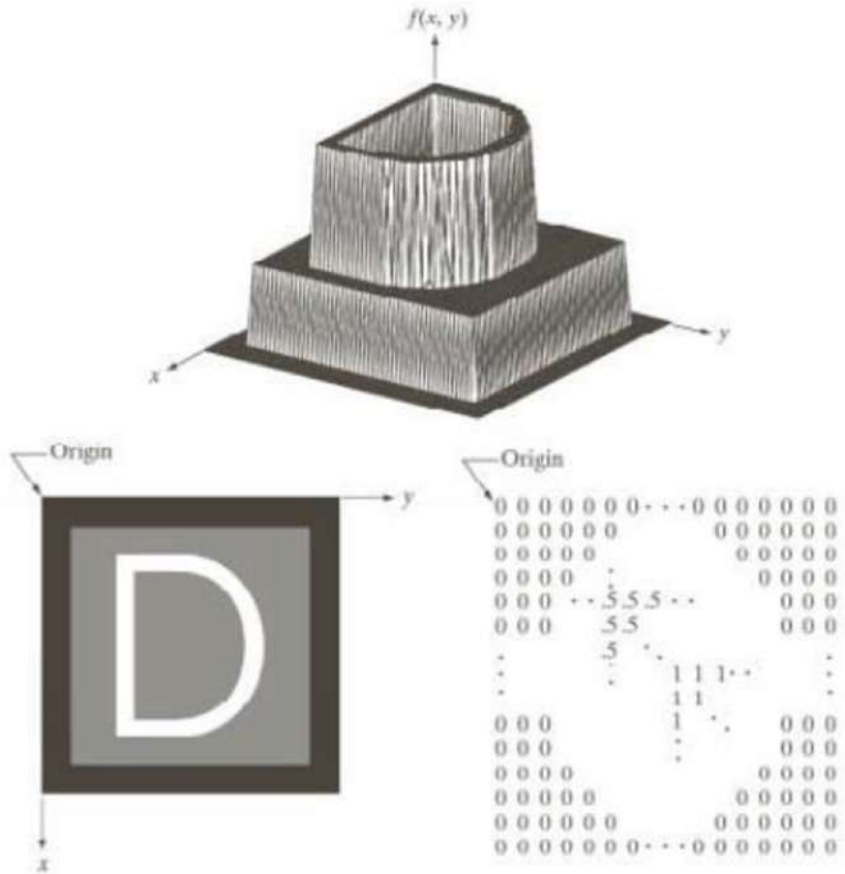
A simple image formation model

- Images denoted by two-dimensional functions $f(x,y)$
- Value of amplitude of f *at spatial coordinate* (x,y) *is a positive scalar quantity and finite.*
- Image generated by physical process, its intensity values are proportional to the energy radiated by a physical source =>
- $0 < f(x,y) < \infty$
- $f(x,y)$ *may be characterized by 2 components:*
  - **The amount of source illumination *incident on the scene:*** *illumination* $i(x,y)$
  - **The amount of illumination *reflected by the objects of the scene:*** *reflectance* $r(x,y)$
- $f(x,y) = i(x,y)\, r(x,y)$, *where* $0 < i(x,y) < \infty$ *and* $0 < r(x,y) < 1$

Image sampling and quantization

- To create a digital image, we need to convert the continuous sensed data **into digital form**
- This involves 2 processes: **Sampling and Quantization**
- An image is continuous in its **coordinates** and also in its **amplitude**(intensity values)
- Digitizing coordinate values- **sampling**
- Digitizing amplitude values- **quantization**
- for sensing strip- sampling **in one direction**
- For a sensing array- sensing in **both the directions**
- The real plane spanned by coordinates of an image is called **spatial domain**.
- X and Y being referred to as **spatial variables** or **spatial coordinates**

Representing digital image



Representing digital image

- Image is a 2 D array f(x,y)
- M rows and N Columns
- *(x,y) = discrete coordinates, x = 0, 1, 2,…, M-1 and y = 0, 1, 2…, N-1*
- Value of the image at any spatial coordinate x,y is f(x,y)
- L, **intensity levels**= 2^k
- K is the number of bits to represent a pixel
- Range of intensity values is **0 to L-1**
- **Dynamic range of an imaging system** – Ratio of the maximum measurable intensity to the minimum detectable intensity level in the system
- Upper limit is determined by **saturation**
- Lower limit is determined by **noise**
- **Contrast-** The difference in intensity between the highest and lowest intensity levels in an image
- Image which has high dynamic range has **high contrast**
- Image which has low dynamic range has dull, washed out gray look

- The number of bits required to store a digitized image is
  - $b = M \times N \times k$
- when M=N, it becomes $b = N^2 k$
- Image with 2^k intensity levels => *"k-bit image" (ex: 256 → 8-bit image)*

TABLE 2.1
Number of storage bits for various values of N and k.

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

2.5 Some Basic Relationships between Pixels

Given an image *f(x,y) and pixels p or q*

2.5.1 Neighbours of a pixel

• A pixel p at *(x,y) has 4 horizontal and vertical neighbours, whose coordinates are:*

(x+1,y), (x-1,y), (x,y+1), (x,y-1) → set N4(p) (4-neighbours of p)

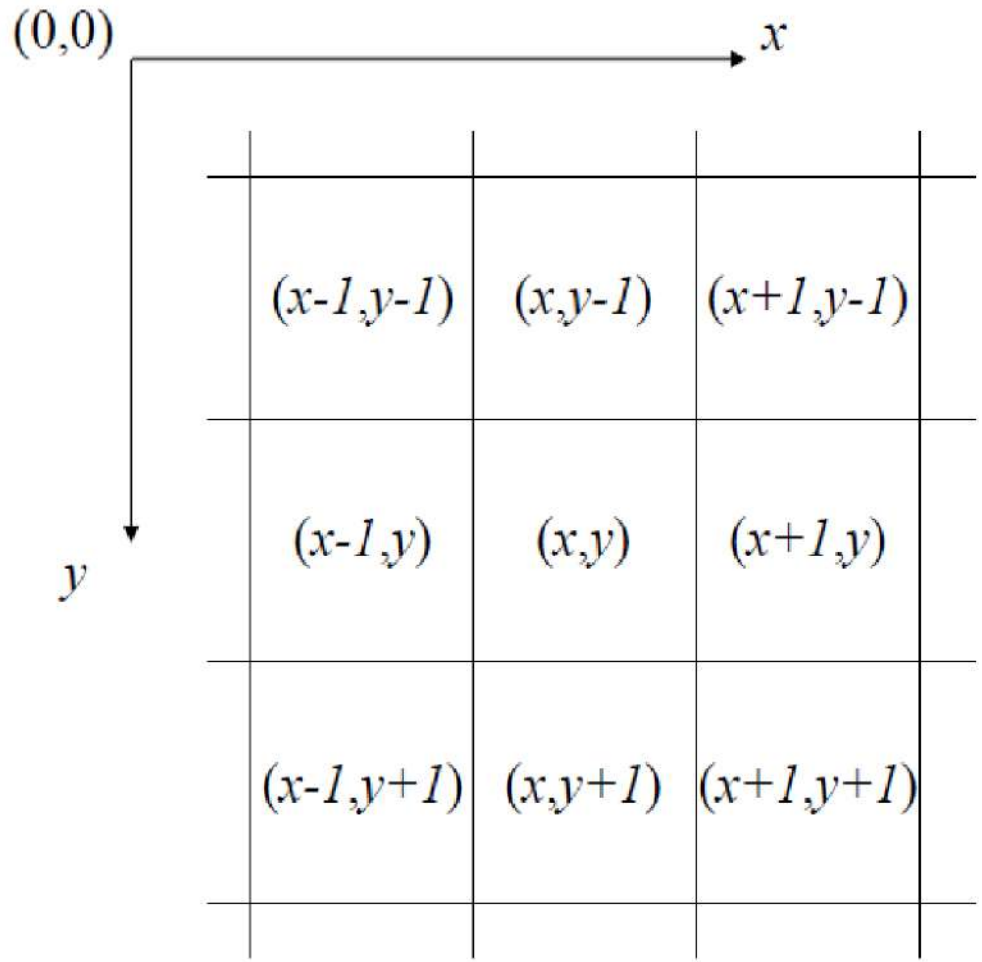
NB: each is a unit distance from p, and some of these locations lie outside the image (borders)

• The 4 *diagonal neighbours of p have coordinates:*

(x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1) → set ND(p)

• N4(p) ∪ ND(p) = N8(p) : the set *of 8-neighbours of p*

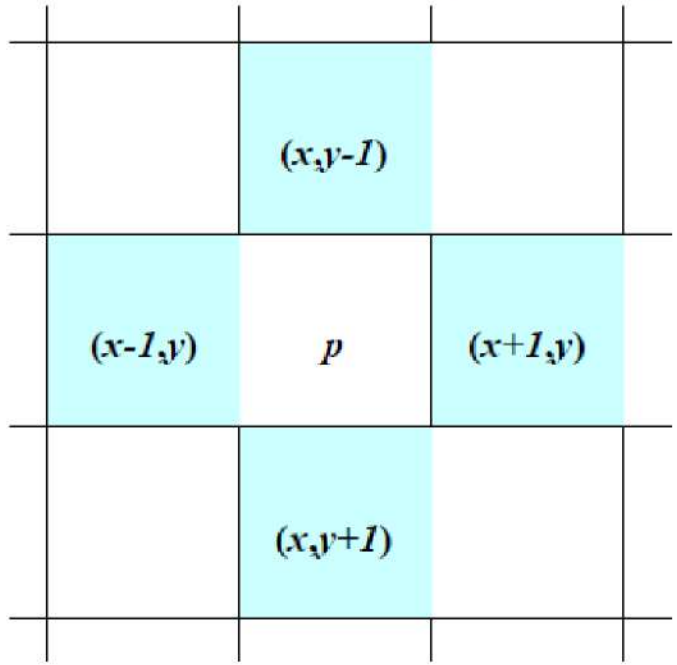
Basic Relationship of Pixels



Conventional indexing method

Neighbors of a Pixel

Neighborhood relation is used to tell adjacent pixels. It is useful for analyzing regions.



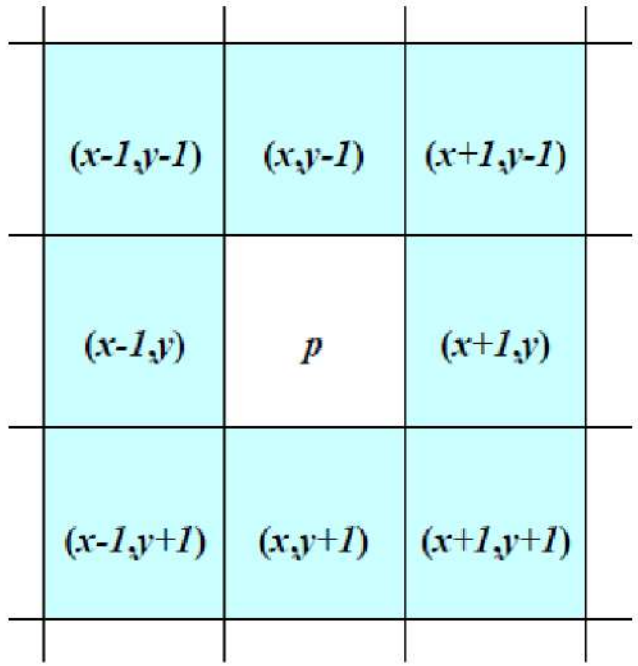
4-neighbors of *p*:

$$N_4(p) = \{ (x-1,y), (x+1,y), (x,y-1), (x,y+1) \}$$

4-neighborhood relation considers only vertical and horizontal neighbors.

Note: $q \in N_4(p)$ implies $p \in N_4(q)$

Neighbors of a Pixel (cont.)

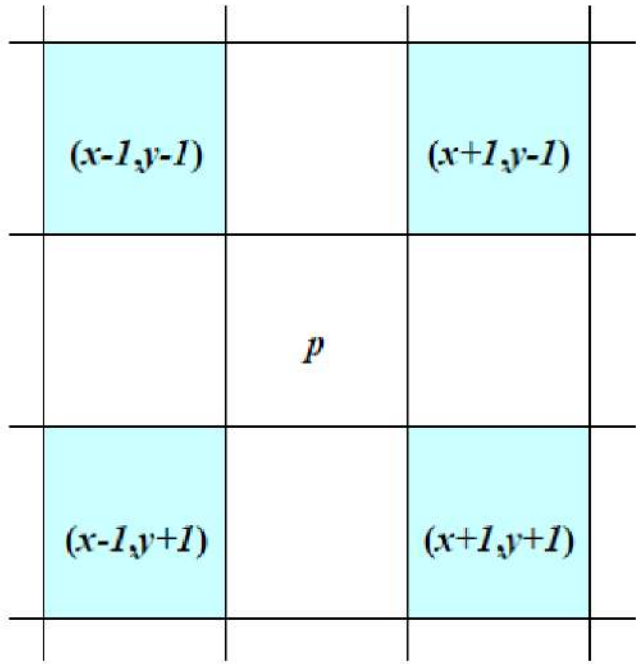


8-neighbors of *p*:

$$N_8(p) = \begin{Bmatrix} (x-1,y-1) \\ (x,y-1) \\ (x+1,y-1) \\ (x-1,y) \\ (x+1,y) \\ (x-1,y+1) \\ (x,y+1) \\ (x+1,y+1) \end{Bmatrix}$$

8-neighborhood relation considers all neighbor pixels.

Neighbors of a Pixel (cont.)



Diagonal neighbors of *p*:

$$N_D(p) = \{(x-1,y-1), (x+1,y-1), (x-1,y+1), (x+1,y+1)\}$$

Diagonal -neighborhood relation considers only diagonal neighbor pixels.

Convolution

- A convolution is essentially sliding a filter over the input
- Usually applied on image to extract features from that
- “A convolution can be thought as “looking at a function’s surroundings to make better/accurate predictions of its outcome.”
- Rather than looking at an entire image at once to find certain features it can be more effective to look at smaller portions of the image.

Filter/mask/kernel/window

- In image processing, a **kernel**, **convolution matrix**, or **mask** is a small matrix.
- It is used for blurring, sharpening, edge detection, edge enhancement, and more. This is accomplished by doing a convolution between a kernel and an image.

2	3	1	4	5
7	3	2	0	9
4	3	2	1	7
6	4	9	0	2
1	3	4	9	8

Image

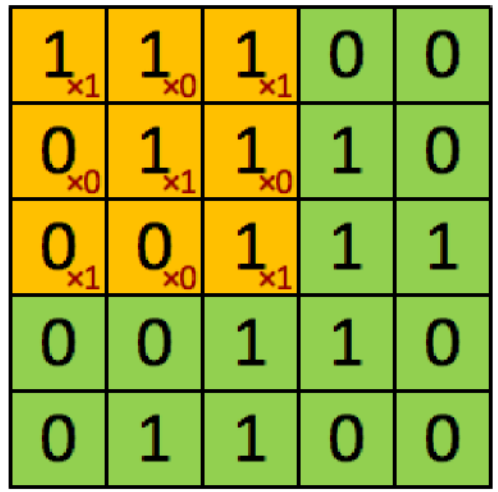
5 X 5

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

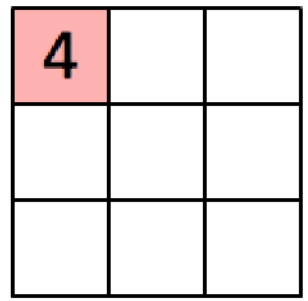
3 x 3

filter

- **Padding-** To handle the edge pixels there are several approaches:
- Padding with zero value pixels
- Reflection padding
- Losing the edge pixels



Image



Convolved Feature

- The array you end up with is called a **feature map** or an **activation map**!
- It's smaller than the original input image if no padding is applied
- Stride- By how many pixels the filter is sliding through the image

- In convolutional network terminology, the first argument to the convolution is often referred to as the **input**
- Second argument as the **kernel**.
- The output is sometimes referred to as the **feature map**.

Depth of the filter

- Depth of the filter depends on (and is equal to) the number of channels of the input image.
- So, we apply a **3X3X1** convolution filter on gray-scale images (the number of channels = 1)
- we apply a **3X3X3** convolution filter on a colored (RGB) image (the number of channels = 3).

Fundamentals of Spatial Filtering

- Spatial filters (also called spatial *masks, kernels, templates or windows)*
- *Filtering refers to accepting(passing) or rejecting certain frequency components.*
- *Filter that passes low frequency is called lowpass filter*

Example : linear spatial filter using a 3x3 neighbourhood:

At any point *(x,y)* in the original image, response *g(x,y)* of the filter:

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots$$
$$+w(0,0)f(x,y) + \cdots + w(1,1)f(x+1,y+1)$$

Centre coefficient of filter

3.4.1 The Mechanics of Spatial Filtering

For a mask of size mxn, we assume:

$m = 2a + 1$ and $n = 2b + 1$

where a and b are positive integers (*odd size*)

General expression of a linear spatial filtering of an image of size M x N with a filter of size m x n :

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

(each pixel in w visiting every pixel in f)

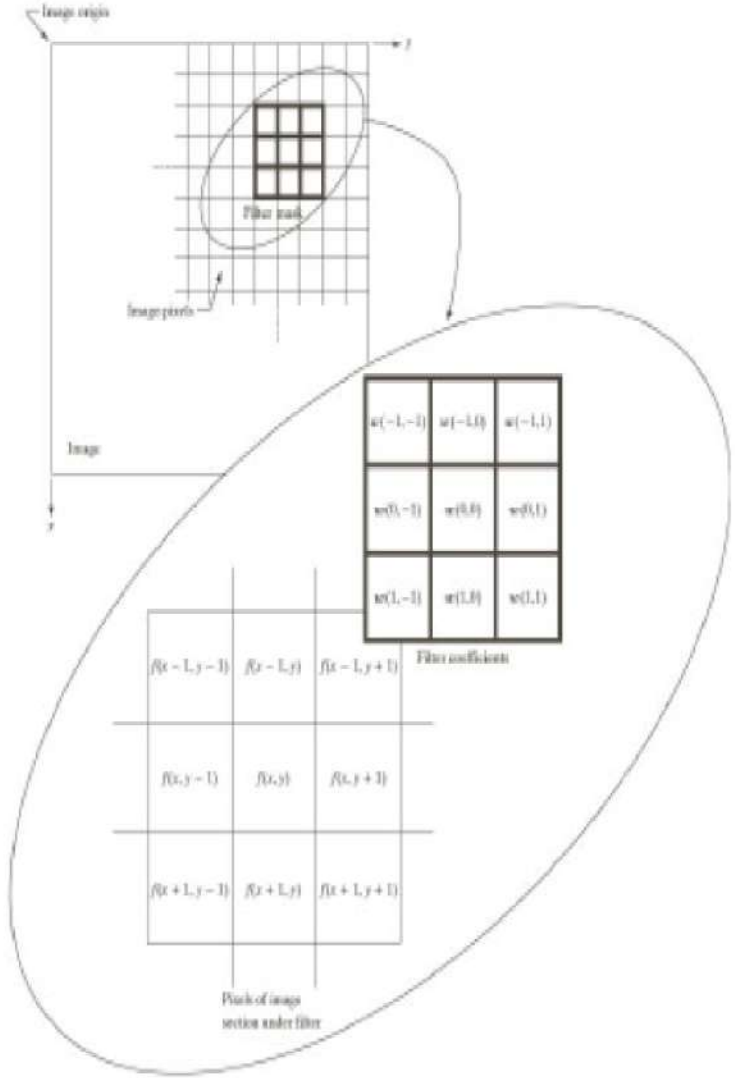
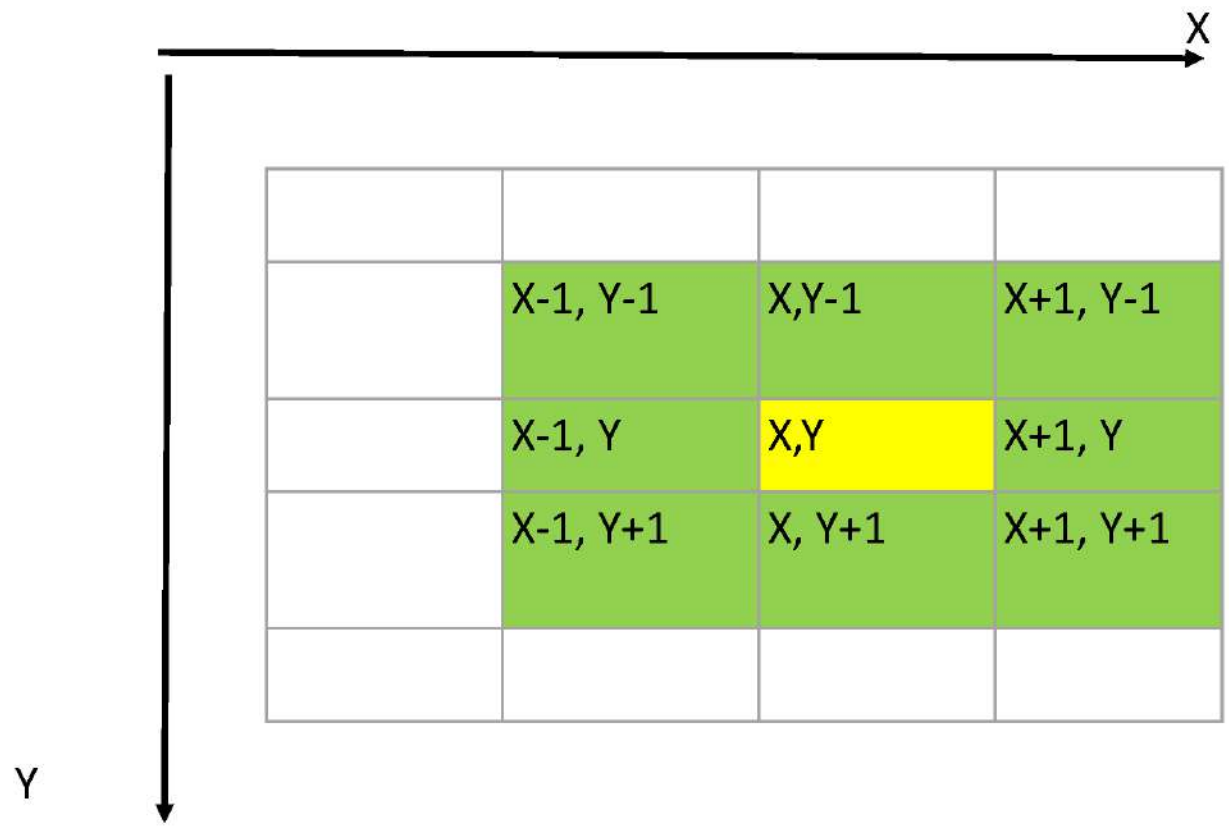
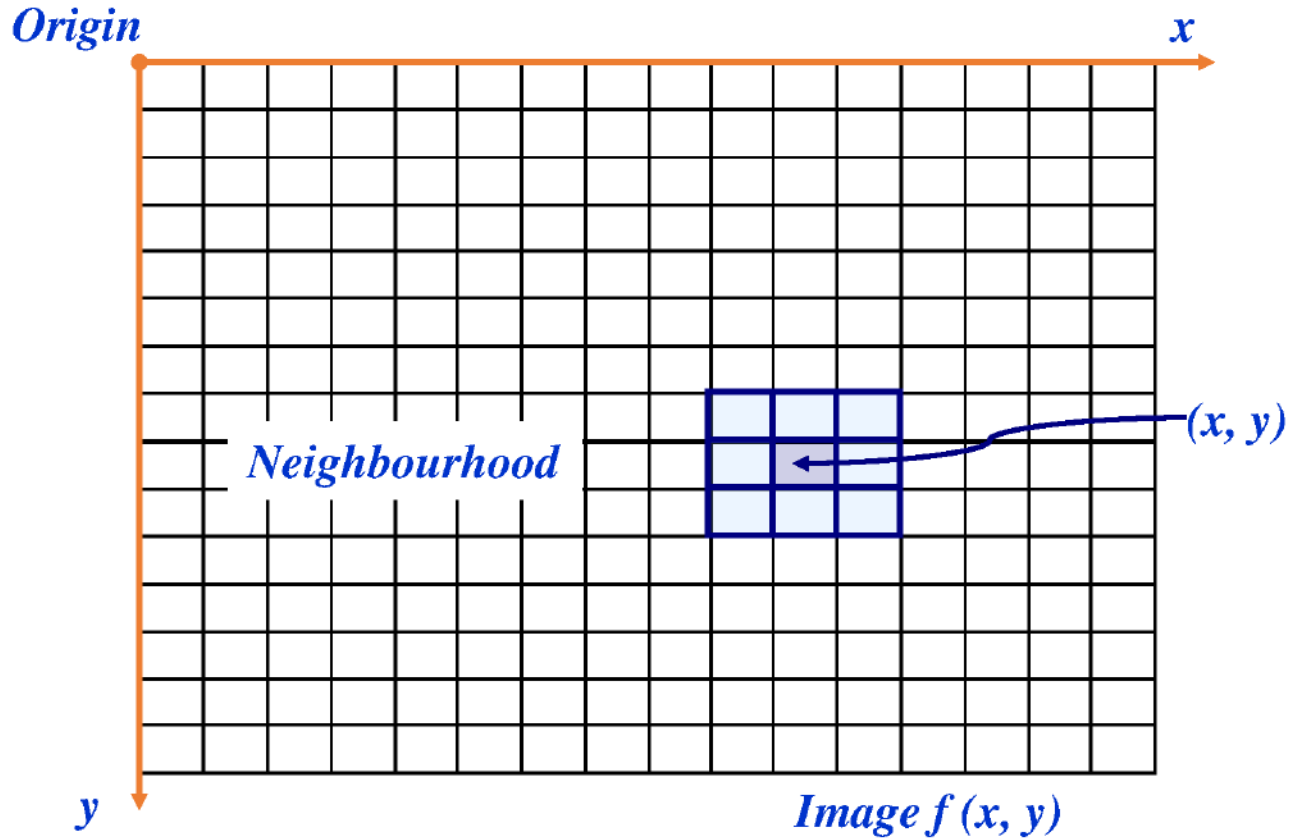


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

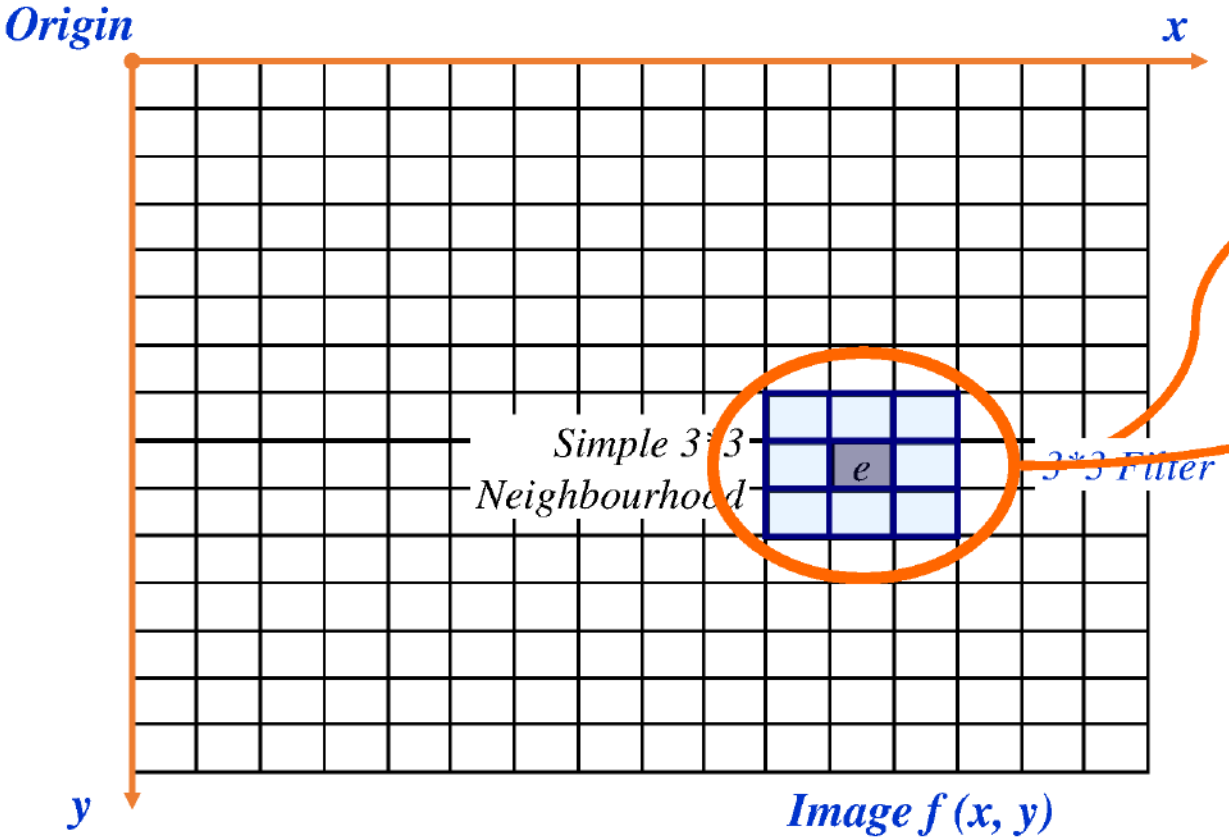

X
Y
X-1, Y-1
X,Y-1
X+1, Y-1
X-1, Y
X,Y
X+1, Y
X-1, Y+1
X, Y+1
X+1, Y+1

Neighbourhood Operations

- Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations
- Neighbourhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible



The Spatial Filtering Process



a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

Original Image Pixels

*

r	*s*	*t*
u	*v*	*w*
x	*y*	*z*

Filter

$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image

- How to relate the convolution with the neural network?
- image intensity values – inputs
- Filter values- weights

Different types of filters

- Low pass- to extract high frequency component – used to blur the image
- High pass – to extract high frequency component like edges, lines etc.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

0	0	2	0	0
0	0	2	0	0
0	0	2	0	0
0	0	2	0	0
0	0	2	0	0

Image

5 X 5

0	**-4**	**8**	**-4**	**0**
0	-6	12	-6	0
0	-6	12	-6	0
0	-6	12	-6	0
0	-4	8	-4	0

Convolutions have been used for a while

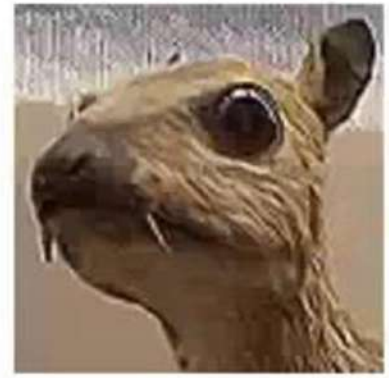
Original image

Kernel

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Edge detection

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



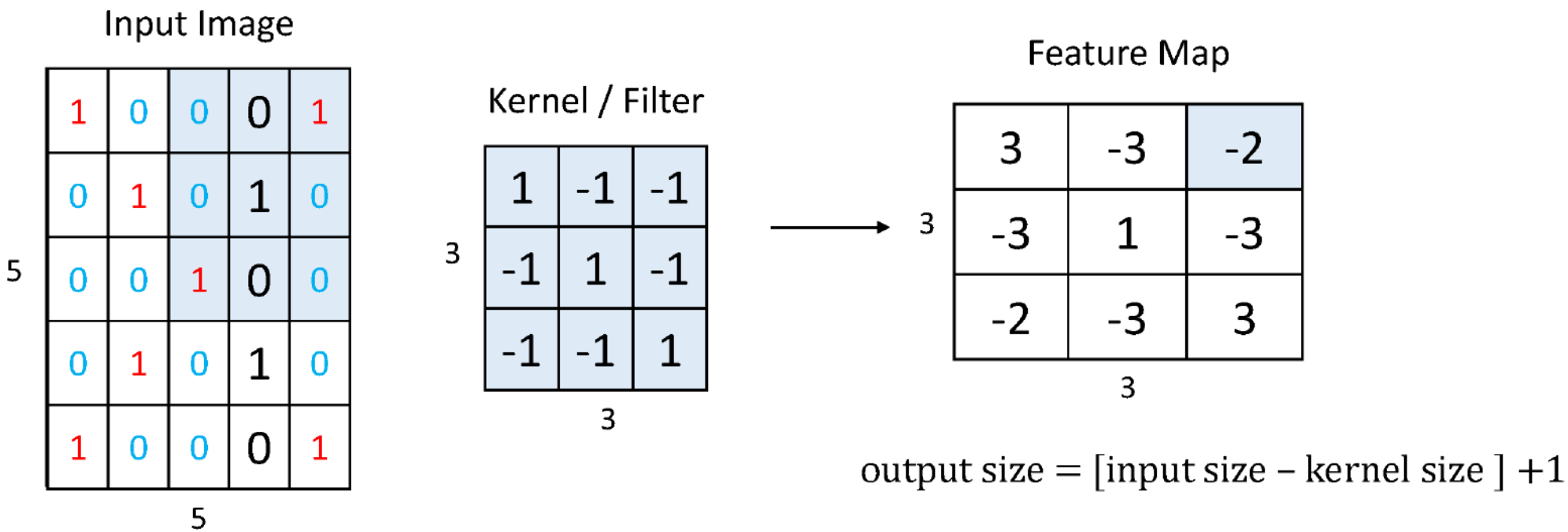
Sharpening

$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Blurring

Step 2: Convolution

- Convolution is an element-wise multiplication of two matrices (sub-matrix) followed by a sum
- Eg: without padding



$$\text{output size} = [\text{input size} - \text{kernel size}\ ] + 1$$

$$= [5 - 3] + 1$$

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

1	-1	-1
-1	1	-1
-1	-1	1

Stride = 1

3	-3	-2
-3	1	-3
-2	-3	3

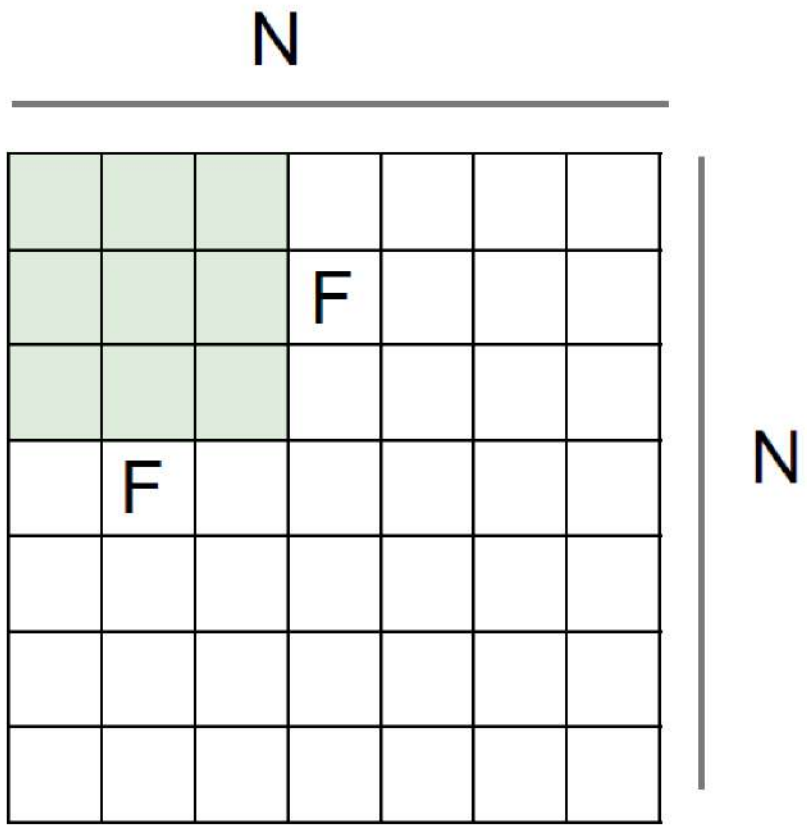
1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

1	-1	-1
-1	1	-1
-1	-1	1

Stride = 2

3	-2
-2	3

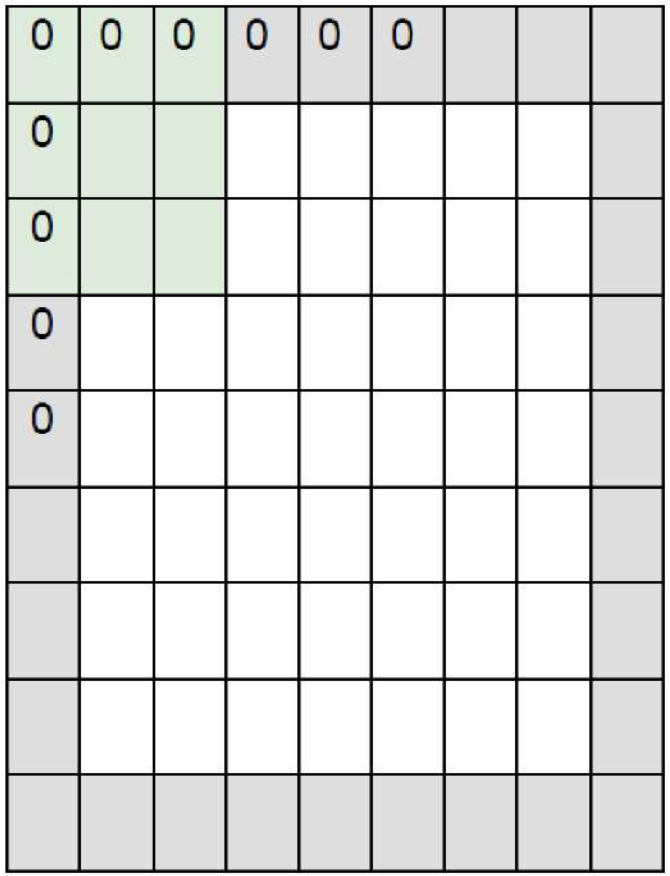
$$\text{Output size(without padding)} = \frac{[\text{input size} - \text{kernel size}]}{\text{stride}} + 1$$



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

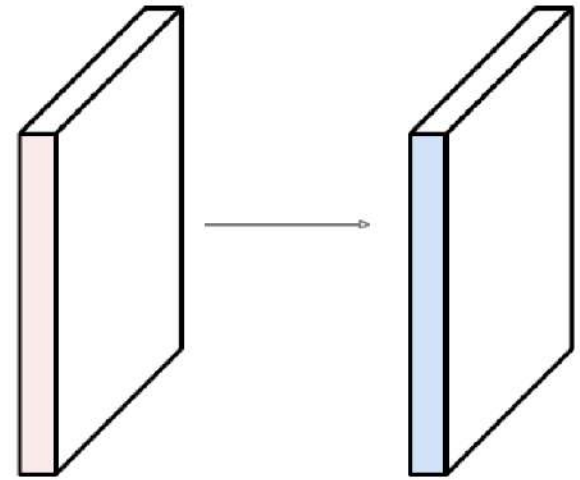
Zero-Padding: common to the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Examples time:

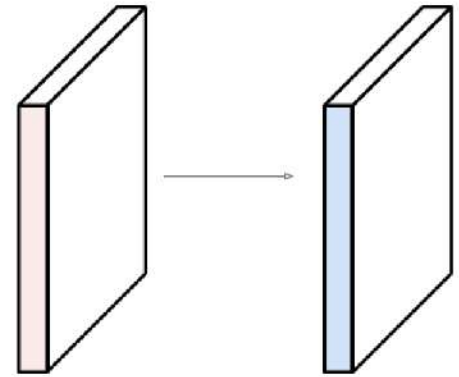


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

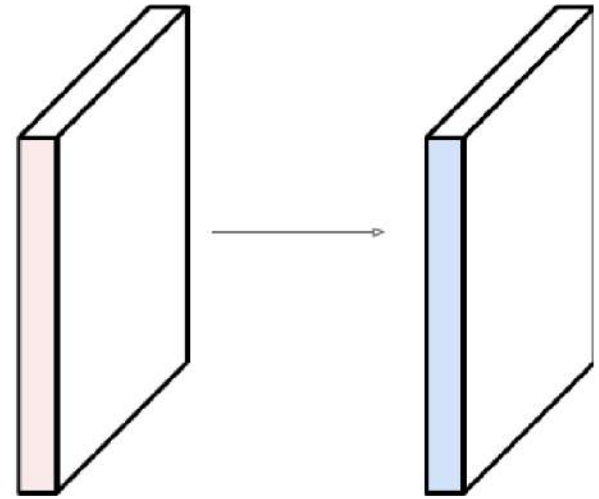
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Examples time:



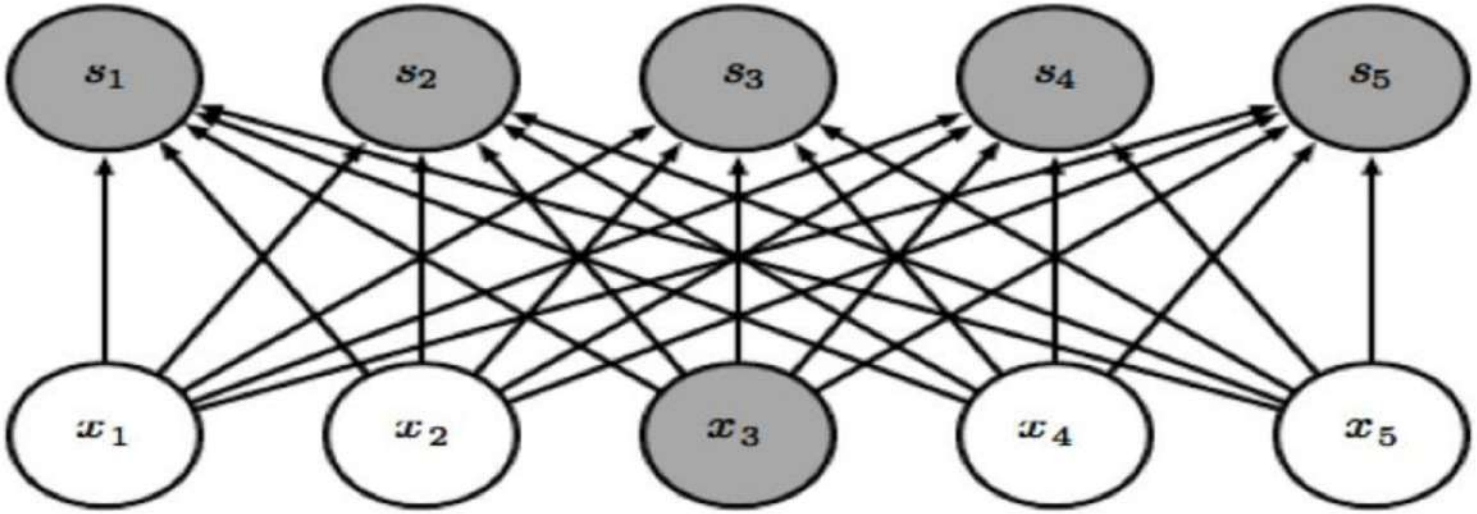
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

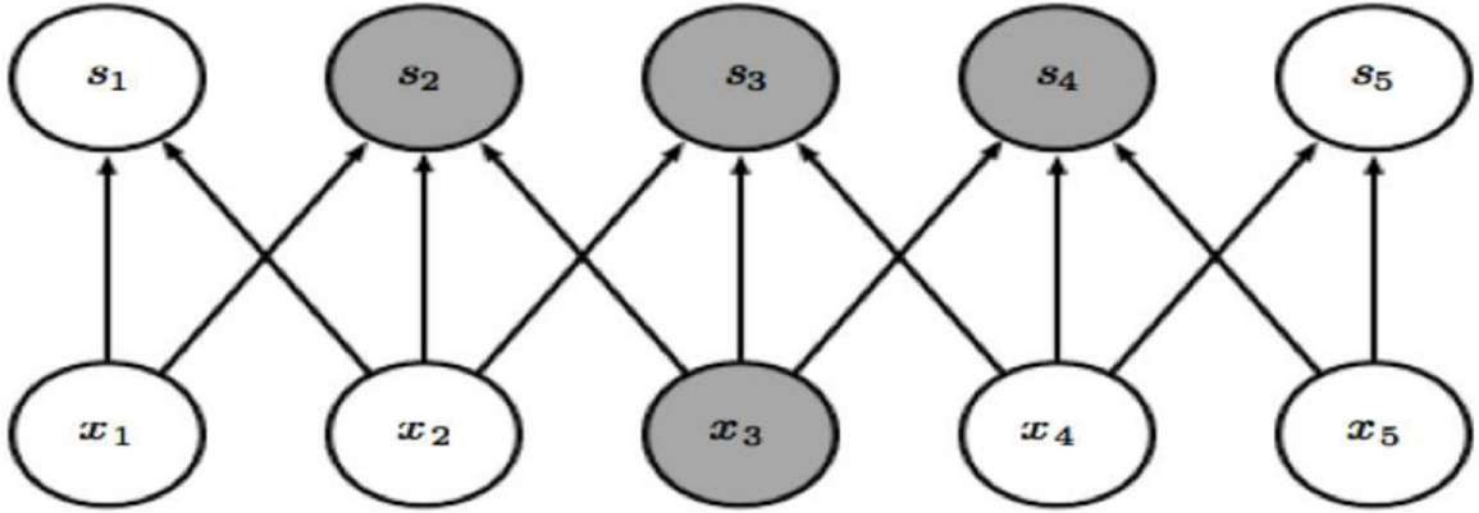
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

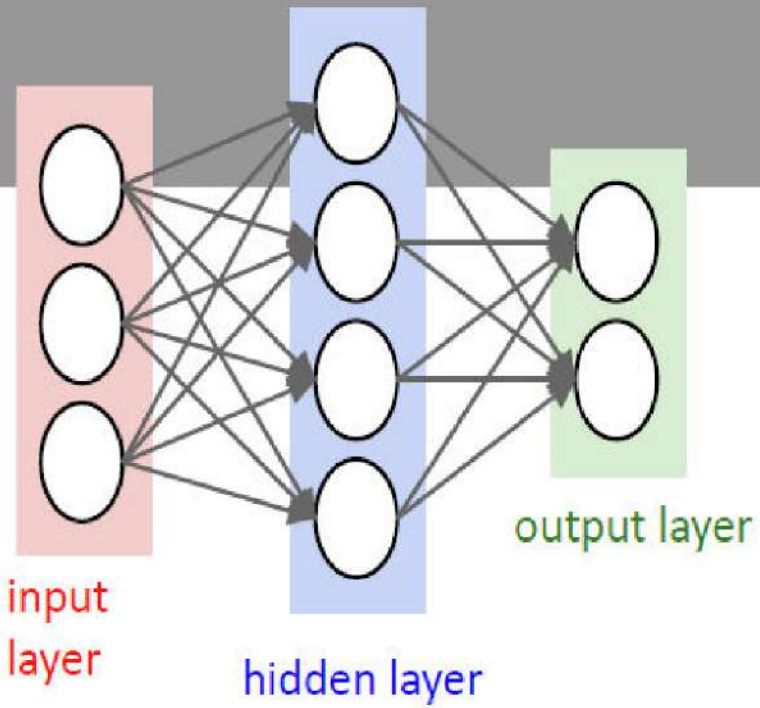
3 reasons why convolution is cool

Reason 1 : Sparse Connectivity

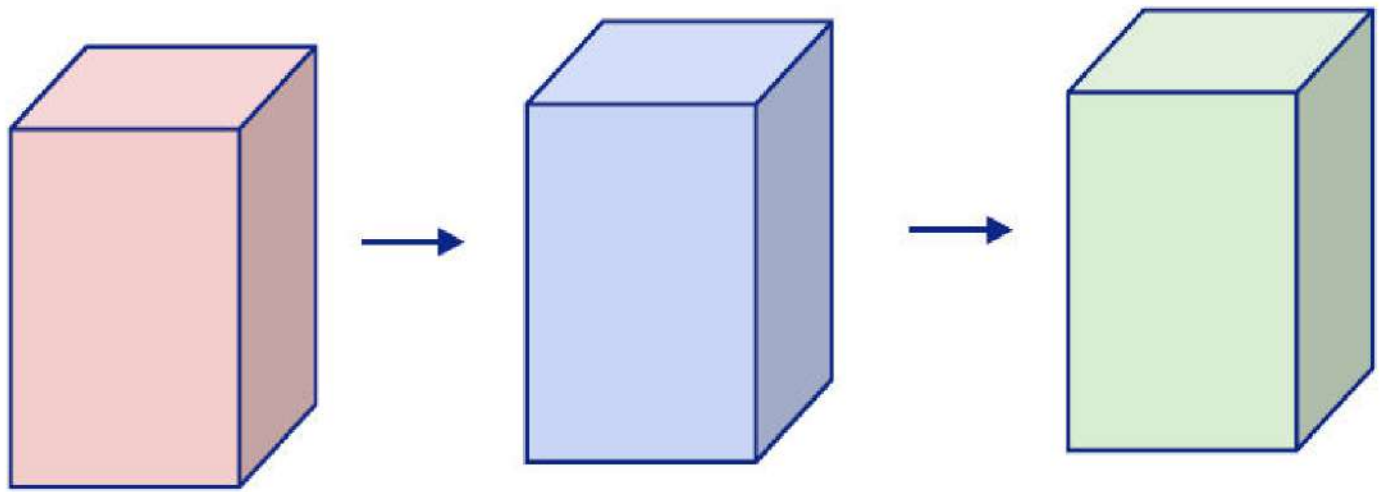




Fully connected

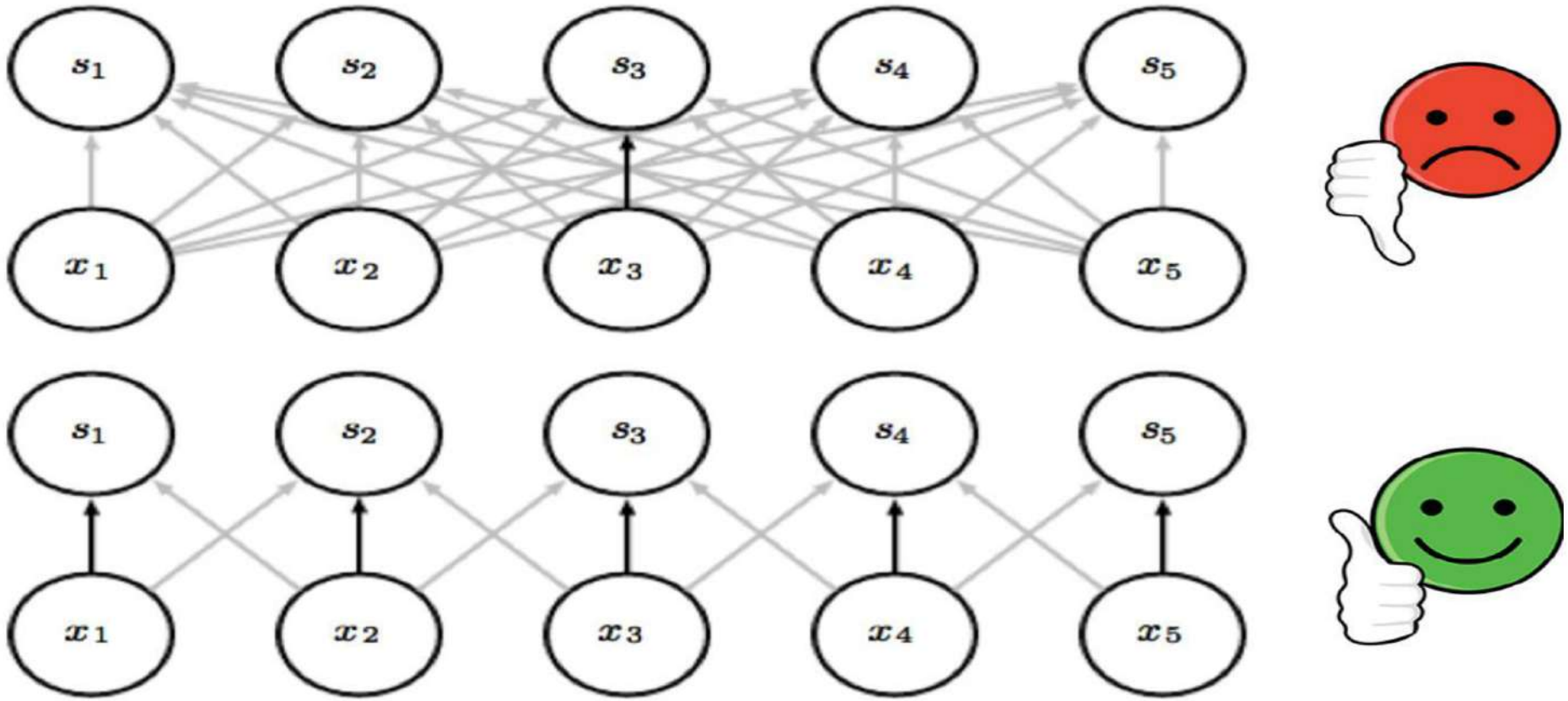


Convolutional
:



- Traditional neural network layers use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each input unit and each output unit.
- This means **every output unit interacts** with every input unit.
- CNN has **sparse interactions** (also referred to as sparse connectivity or sparse weights).
- This is accomplished by making the **kernel smaller than the input**.
- When processing an image, the input image might have thousands or millions of pixels, but **we can detect small, meaningful features** such as edges with kernels that occupy only tens or hundreds of pixels.
- This means that we need to store fewer parameters, which both **reduces the memory requirements** of the model and improves its statistical efficiency.

Reason 2 : Parameter sharing



- Parameter sharing refers to using the **same parameter for more than one function in a model**
- In a traditional neural net, **each element of the weight matrix is used exactly once** when computing the output of a layer.
- It is multiplied by one element of the input and then never revisited. a network has **tied weights**, because the value of the weight applied to one input is tied to the value of a weight applied elsewhere.
- In CNN, each member of the kernel is used at every position
- of the input

Reason 3 : Equivariant Representations

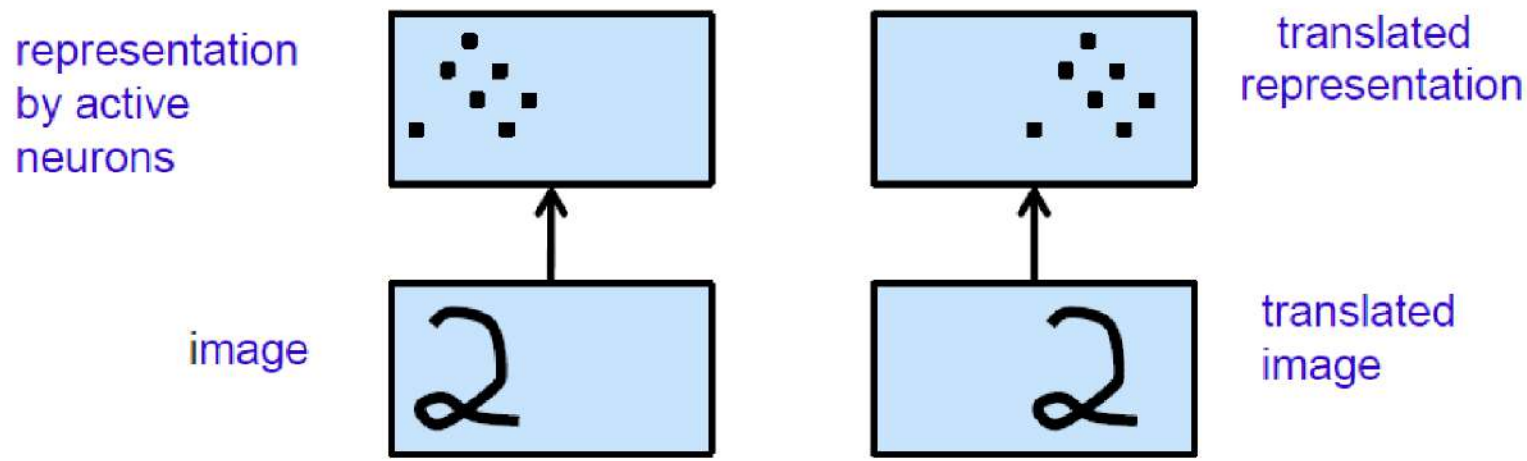
When the input changes -> output changes in the same way

Eg. Let I be a function giving images brightness at integer coordinates
Let g be a function mapping one image function to another image function,
such that $I' = g(I)$ is the image function with $I'(x,y) = I(x - 1,y)$.
This shifts every pixel of I one unit to the right.
If we apply this transformation to I, then apply convolution,
the result will be the same as if we applied convolution to I',
then applied the transformation g to the output.

Reason 3 : Equivariant Representations

When the input changes -> output changes in the same way

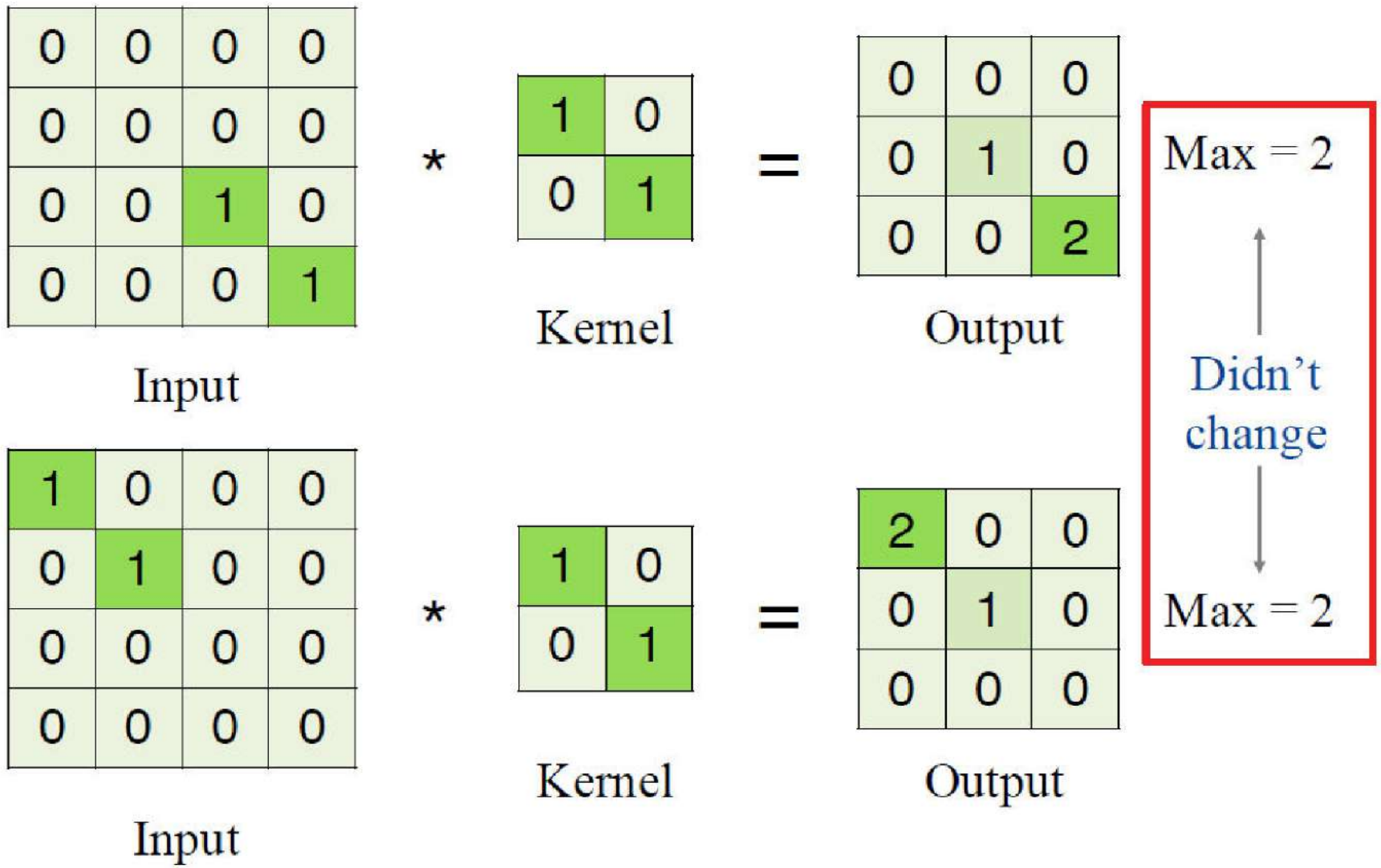
Eg. Let I be a function giving images brightness at integer coordinates
Let g be a function mapping one image function to another image function,
such that $I' = g(I)$ is the image function with $I'(x,y) = I(x - 1,y)$.
This shifts every pixel of I one unit to the right.
If we apply this transformation to I, then apply convolution,
the result will be the same as if we applied convolution to I',
then applied the transformation g to the output.



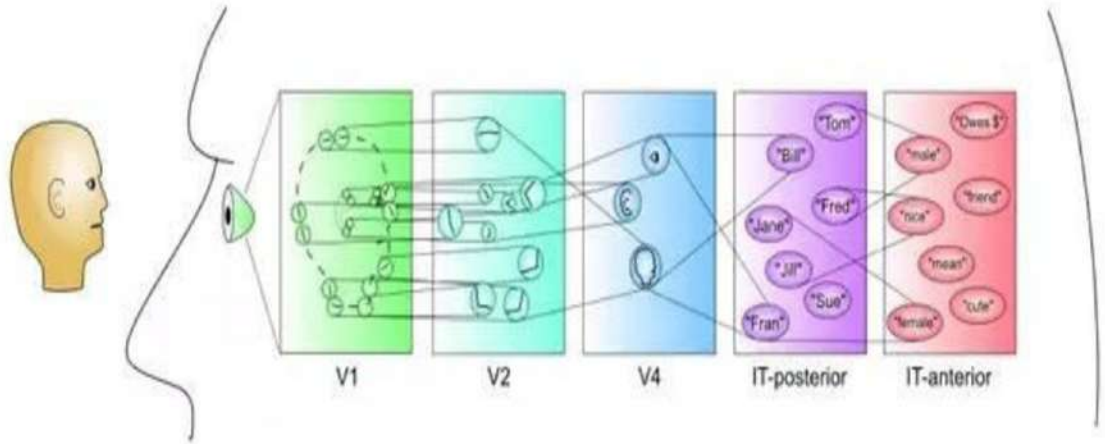
- Invariant knowledge: If a feature is useful in some locations during training, detectors for that feature will be available in all locations during testing.

Reason 3 contd.

How do we maintain translation invariance?



What inspired convolution neural networks?



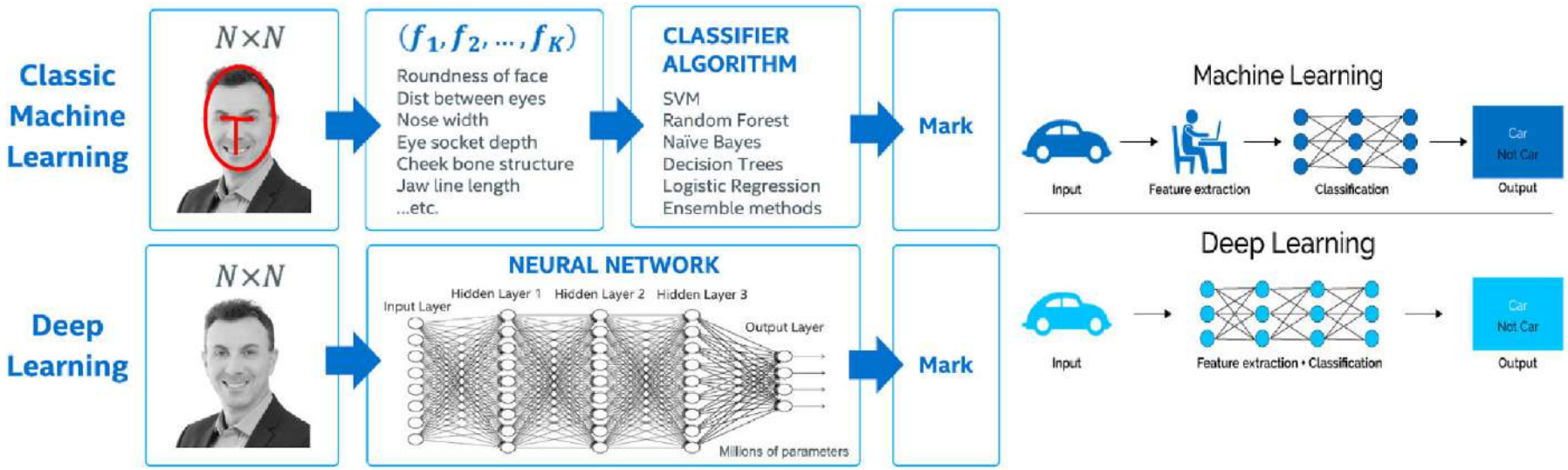
The architecture of deep convolutional neural networks was inspired by the ideas mentioned above

- local connections

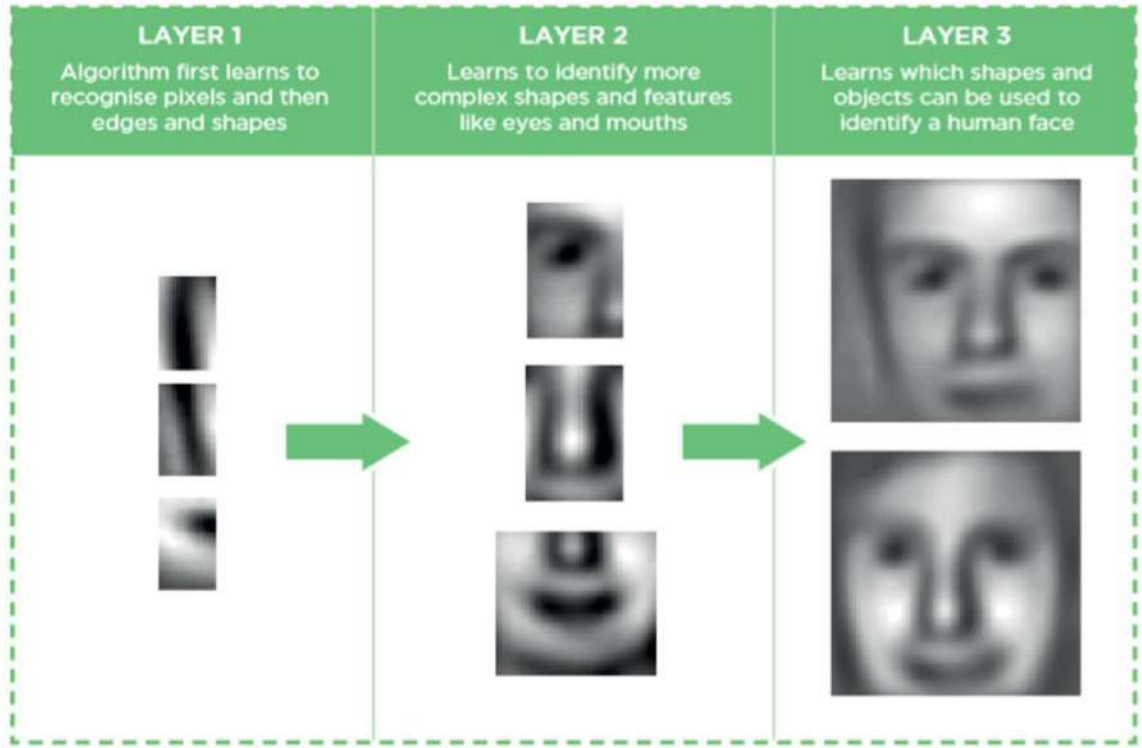
- layering

- spatial invariance (shifting the input signal results in an equally shifted output signal. , most of us are able to recognize specific faces under a variety of conditions because we learn abstraction These abstractions are thus invariant to size, contrast, rotation, orientation

Neural feature extraction vs classical ML

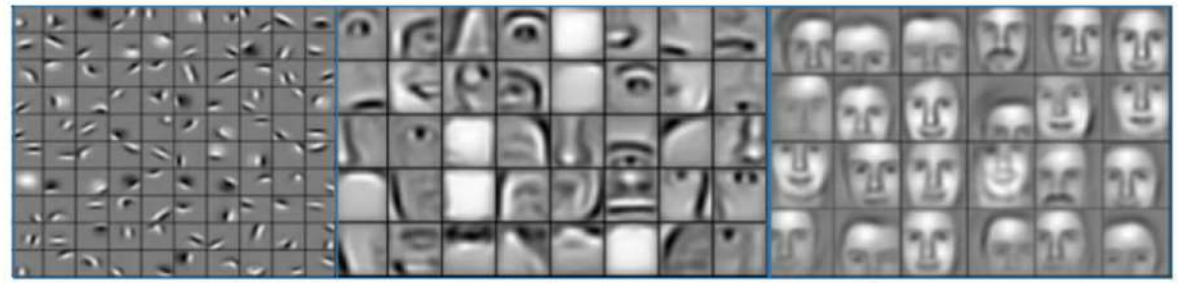


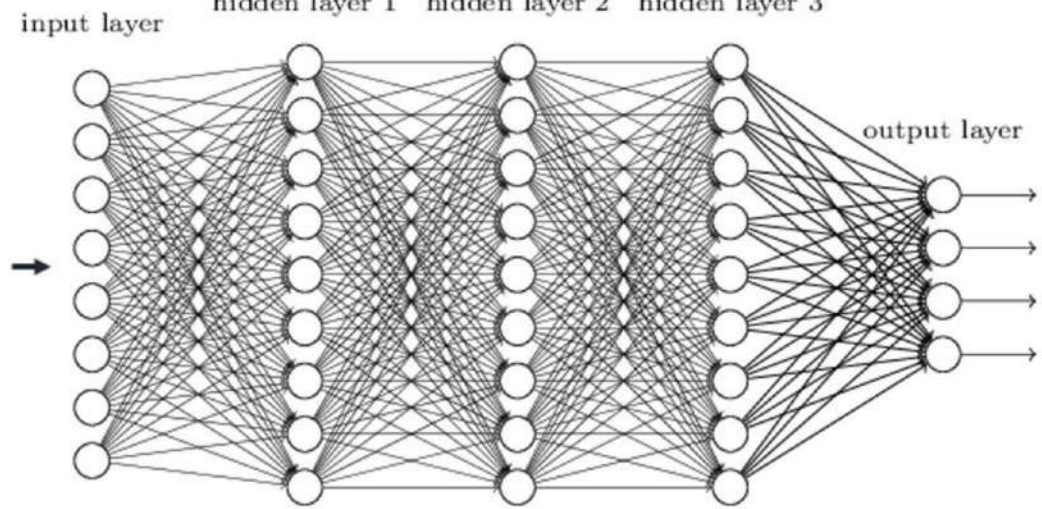
What are layers and what happens inside?



What are layers and what happens inside?

Deep neural networks learn hierarchical feature representations

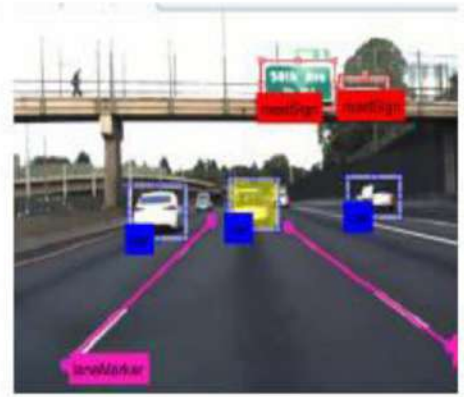
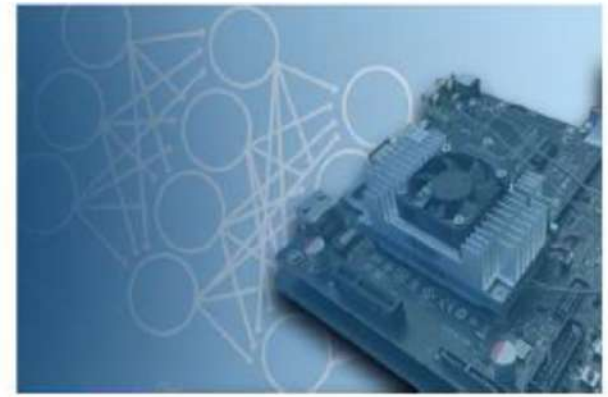




Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image



What is Deep Learning?




Deep learning is a type of machine learning in which a model learns to perform tasks directly from images, text, or sound.

Deep learning is usually implemented using a **neural network.**

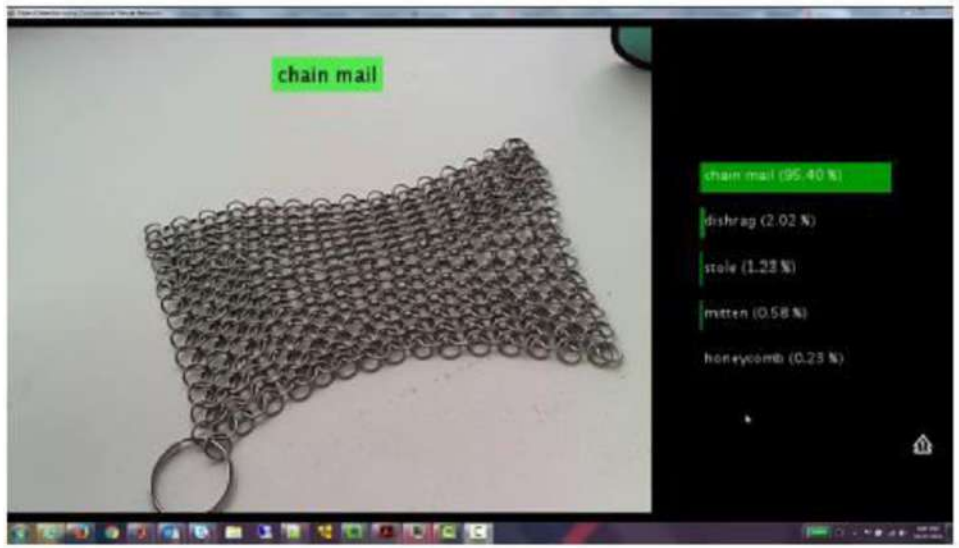
The term “deep” refers to the **number of layers** in the network—the more layers, the deeper the network.

How Companies Use CNNs

Data, data, data.

- The companies that have lots of this magic 4 letter word are the ones that have an inherent advantage over the rest of the competition.
- The more training data that you can give to a network, the more training iterations you can make, the more weight updates you can make, and the better tuned to the network is when it goes to production.
- Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

Object recognition using deep learning



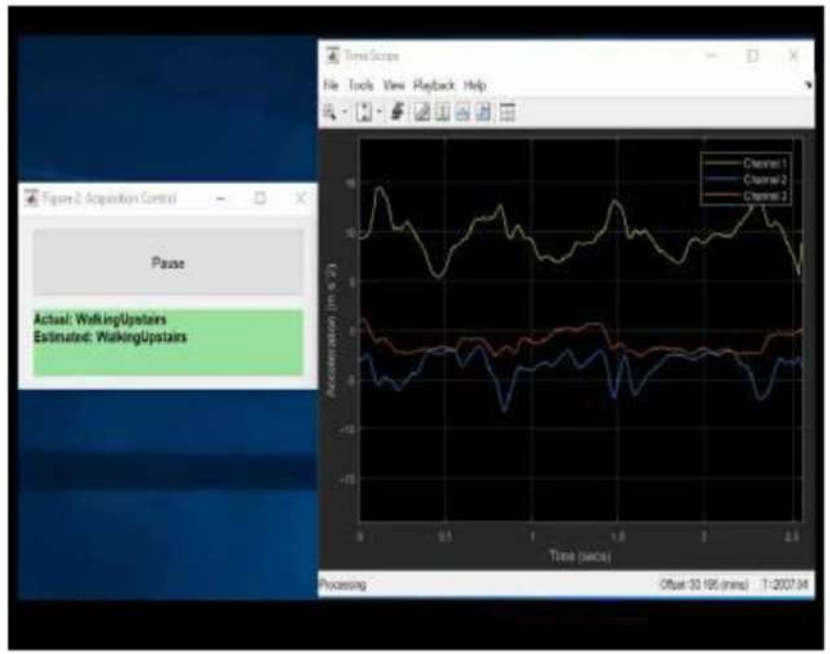
Training (GPU)	Millions of images from 1000 different categories
Prediction	Real-time object recognition using a webcam connected to a laptop

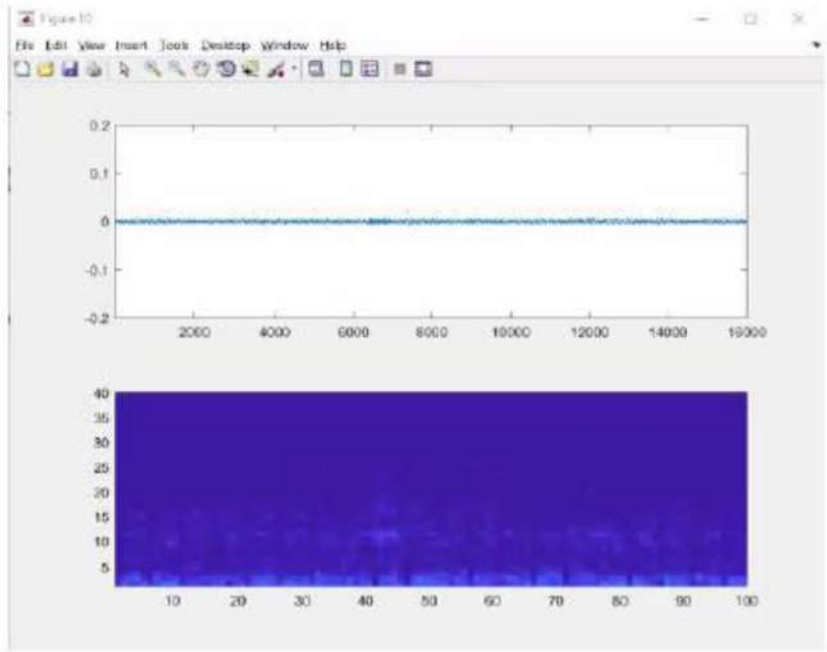
Detection and localization using deep learning

Regions with Convolutional Neural Network Features (R-CNN)

Semantic Segmentation using SegNet

Analyzing signal data using deep learning





Signal Classification using LSTMs

Speech Recognition using CNNs

Example: Food classifier using deep transfer learning

Hot dog →

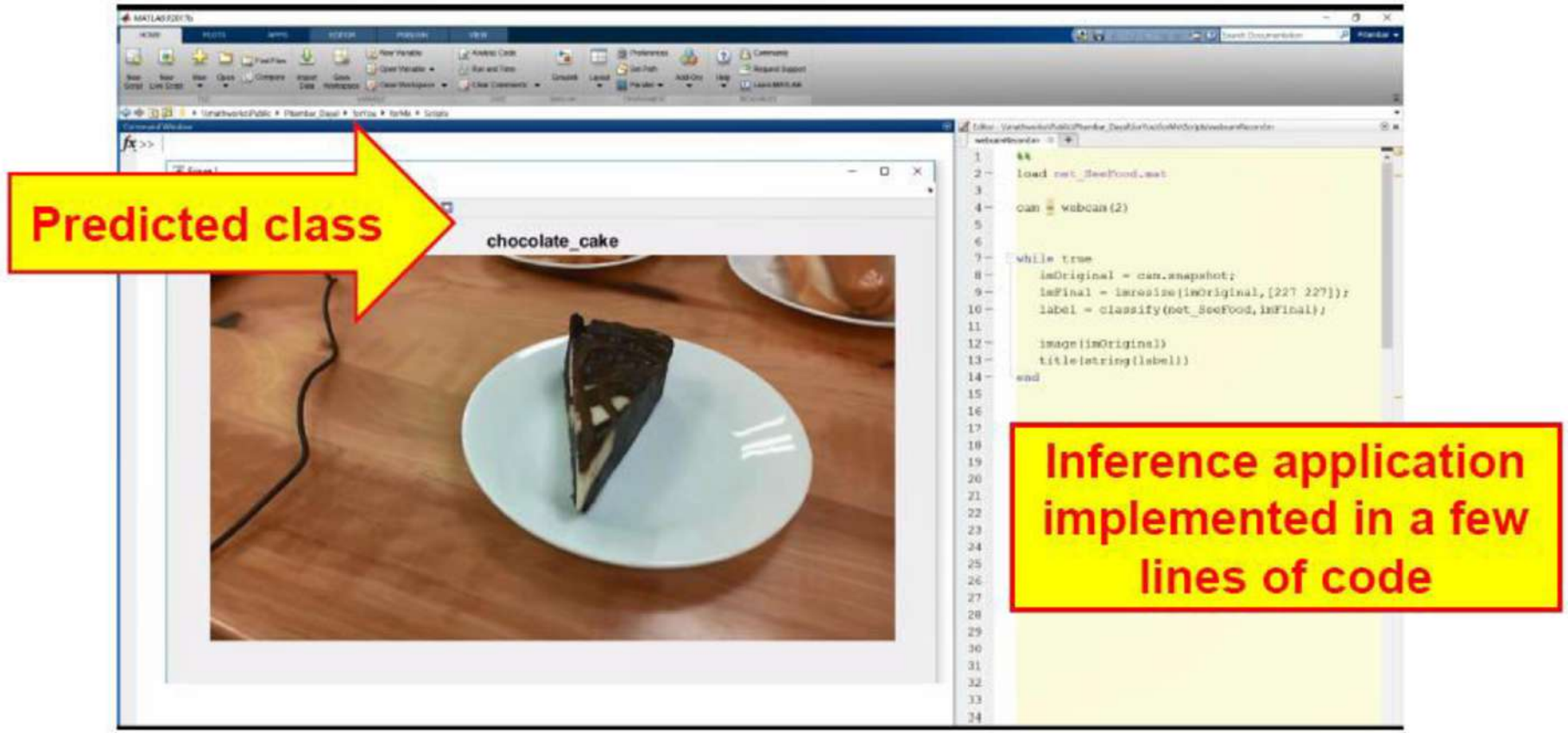
French fries →

Chocolate cake →

Pizza →

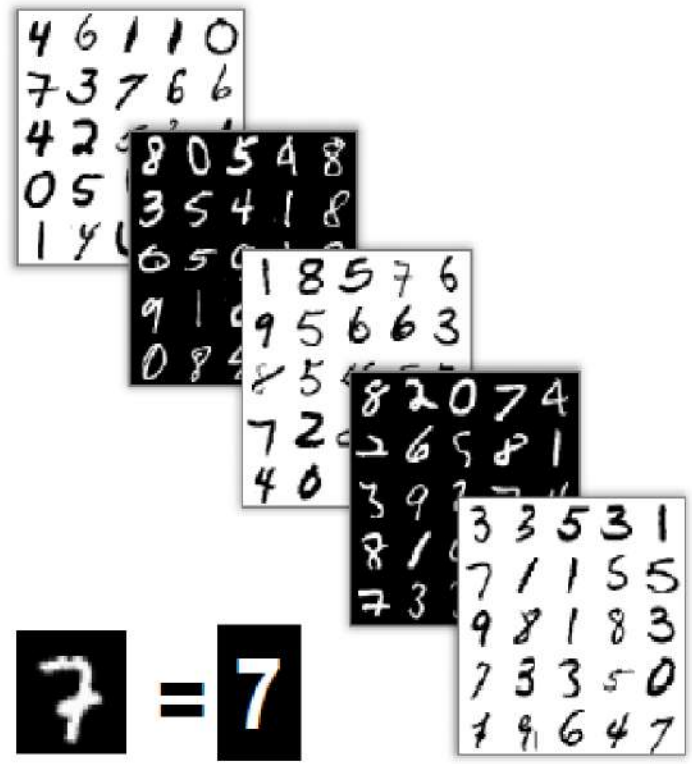
Ice cream →

5 Category Classifier


Predicted class
chocolate_cake
load net_FoodFood.mat
cam = webcam(2)
while true
imOriginal = cam.snapshot;
imFinal = imresize(imOriginal,[227 227]);
label = classify(net_FoodFood,imFinal);
image(imOriginal)
title(string(label))
end
Inference application implemented in a few lines of code

MNIST: The “Hello, World!” of computer vision

What?	A set of handwritten digits from 0-9
Why?	An easy task for machine learning beginners
How many?	60,000 training images 10,000 test images
Best results?	99.79% accuracy



Sources: http://yann.lecun.com/exdb/mnist/
https://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results

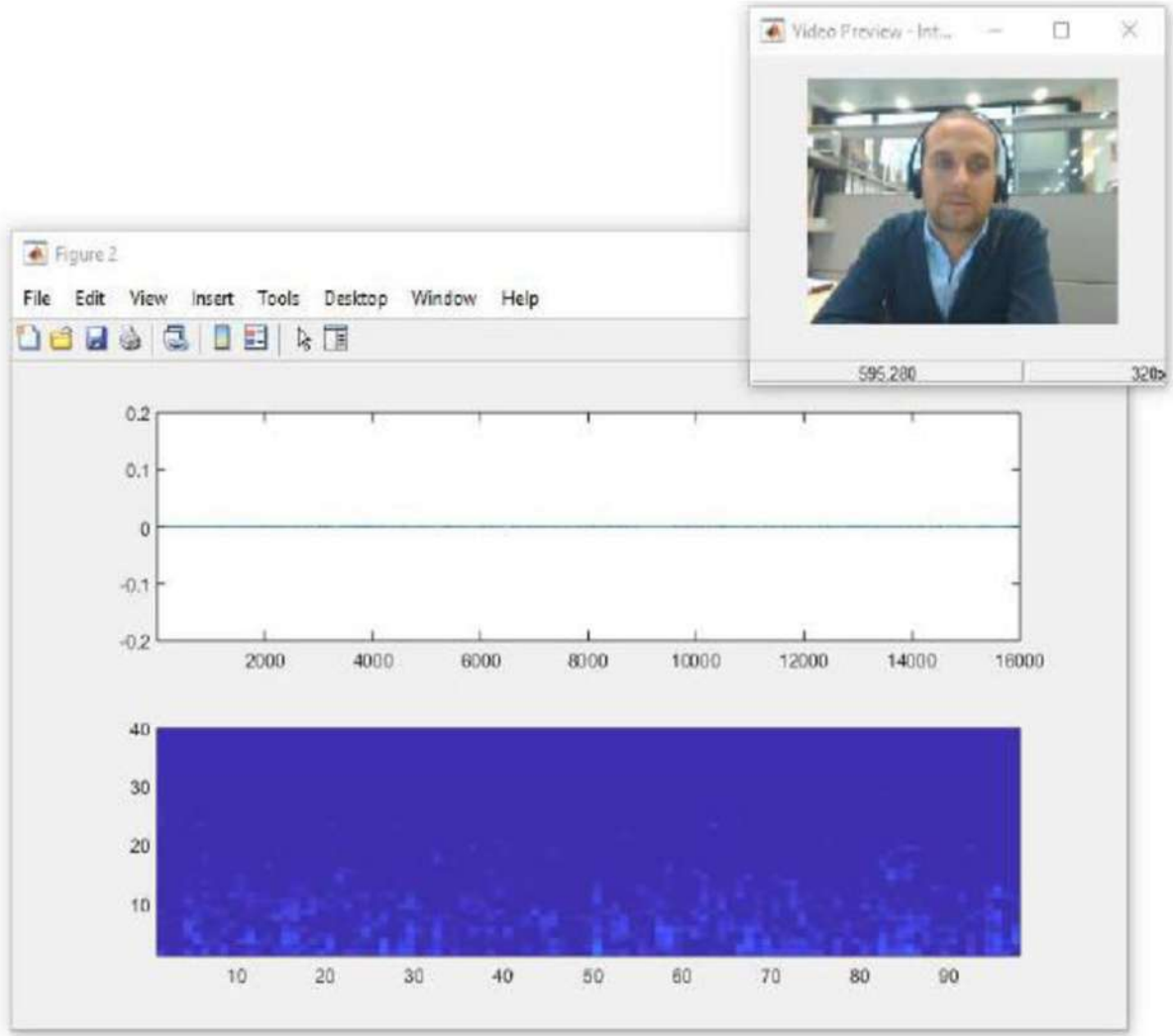
Demo: **Speech Command Recognition Using Deep Learning**

Commands

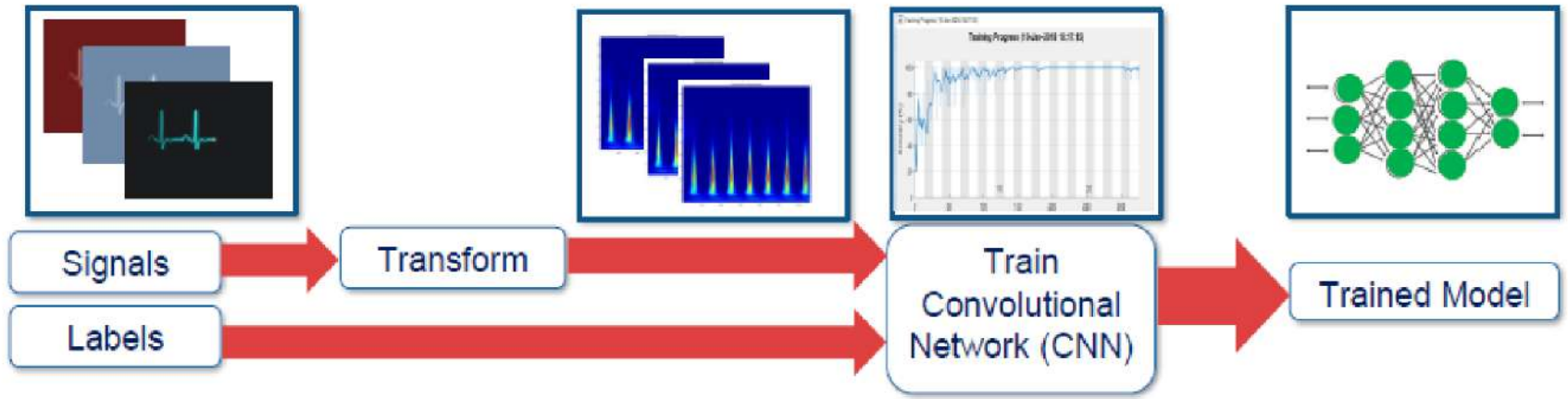
- Yes
- No
- Up
- Down
- Left
- Right
- On
- Off
- Stop
- Go

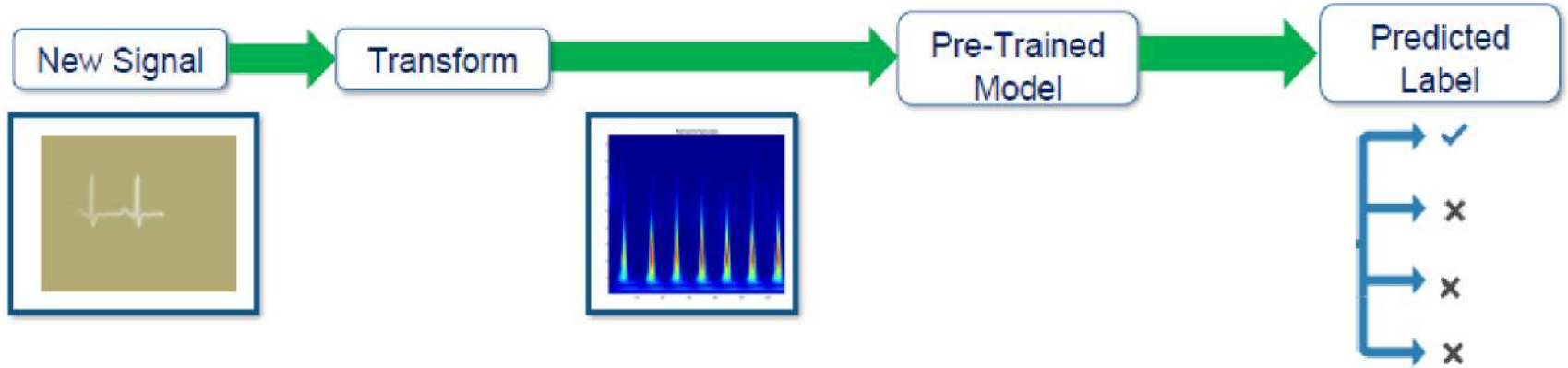
Non-Commands
(= Unknown)

- Bed
- Bird
- Cat
- Dog
- Happy
- House
- Marvin
- Sheila
- Tree
- Wow
- Zero
- One
- Two
- Three
- Four
- Five
- Six
- Seven
- Eight
- Nine

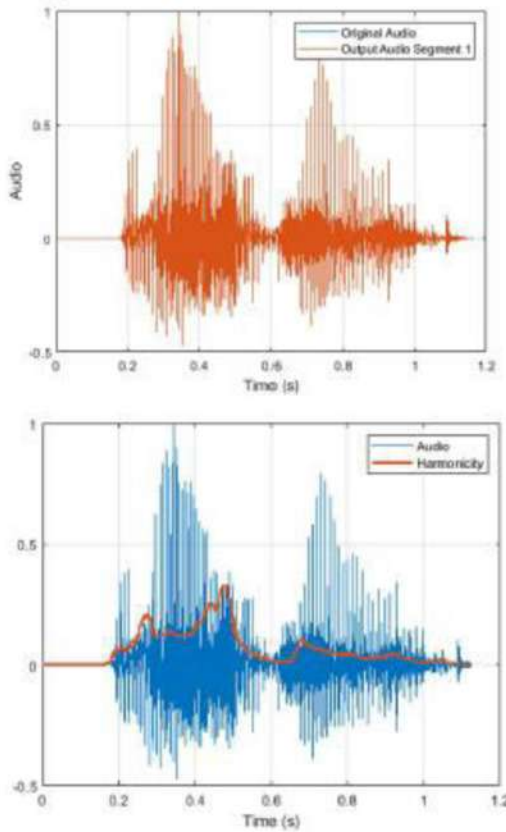


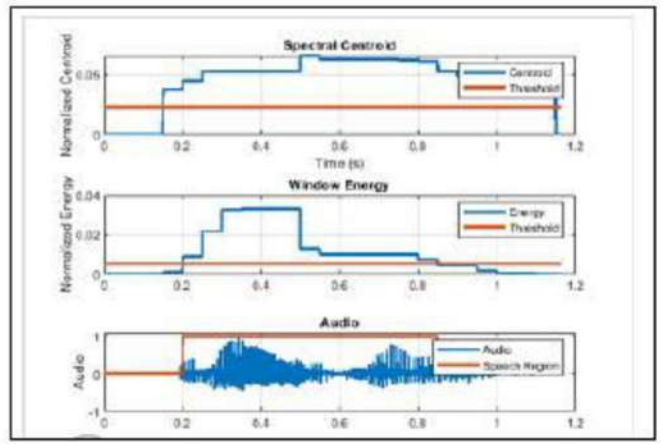
How can things work with Audio?



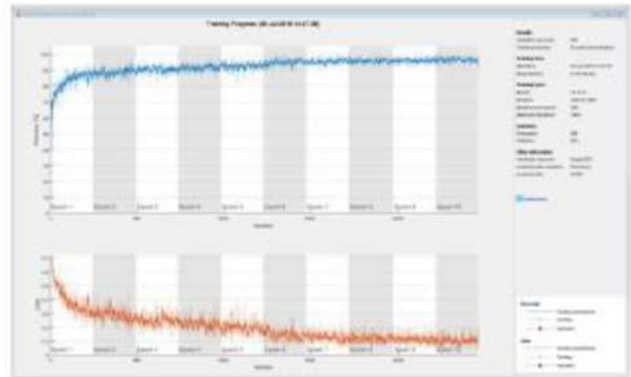


Classify Speaker Using LSTMs

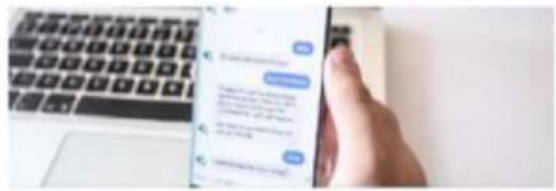





```
layers = [ ...
    sequenceInputLayer(NumFeatures)
    bilstmLayer(100,"OutputMode","sequence")
    reluLayer
    bilstmLayer(100,"OutputMode","last")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];
```



Deep Learning for Text Generation



More from Chatbots Life

5 Incredible Ways in Which Chatbots Can Enhance Custome...

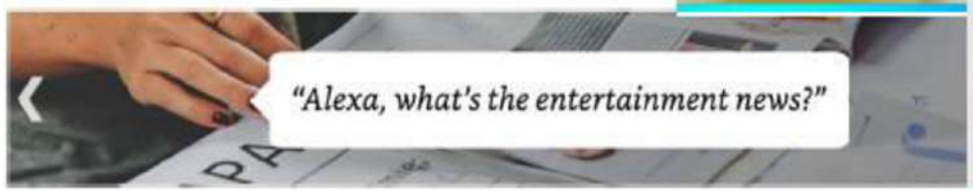
THE ALGORITHMS BEHIND THE HEADLINES

How machine-written news redefines the core skills of human journalists

Arjen van Dalen

Pages 648-658 | Published online: 30 Mar 2012

Download citation https://doi.org/10.1080/17512786.2012.667268



Typically, integrations have focused on what a user can say to Alexa, not what she can say back. With Wordsmith, we get human-sounding responses that naturally use different language each time.

Narrative Science Brings Natural Language to Qlik Visualization Software

Word-By-Word Text Generation Using Deep Learning

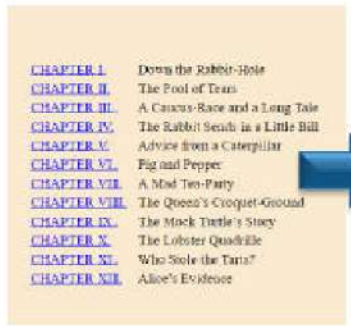
Access and Explore Data

Preprocess Data

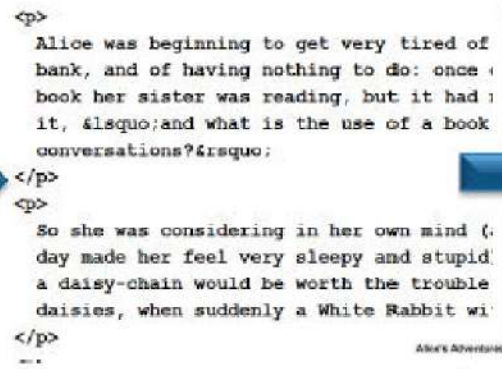
Clean-up Text

Convert to Numeric

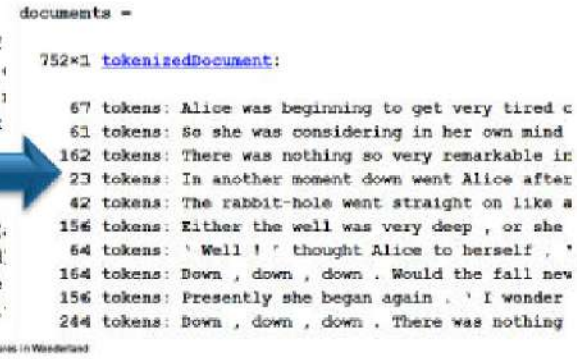
Develop Predictive Models



CHAPTER I. Down the Rabbit-Hole
CHAPTER II. The Pool of Tears
CHAPTER III. A Caucus-Race and a Long Tale
CHAPTER IV. The Rabbit Sends in a Little Bill
CHAPTER V. Advice from a Caterpillar
CHAPTER VI. Pig and Pepper
CHAPTER VII. A Mad Tea-Party
CHAPTER VIII. The Queen's Croquet-Ground
CHAPTER IX. The Mock Turtle's Story
CHAPTER X. The Lobster Quadrille
CHAPTER XI. Who Stole the Tarts?
CHAPTER XII. Alice's Evidence



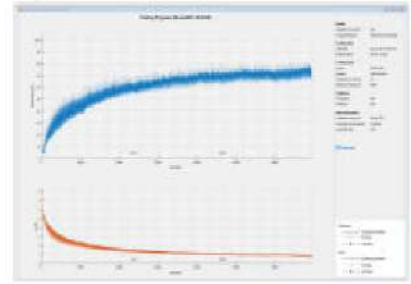
```
<p>
  Alice was beginning to get very tired of
  bank, and of having nothing to do: once
  book her sister was reading, but it had
  it, &lsquo;and what is the use of a book
  conversations?&rsquo;
</p>
<p>
  So she was considering in her own mind (
  day made her feel very sleepy and stupid
  a daisy-chain would be worth the trouble
  daisies, when suddenly a White Rabbit wi
</p>
```



```
documents =

  752×1 tokenizedDocument:

    67 tokens: Alice was beginning to get very tired o
    61 tokens: So she was considering in her own mind
   162 tokens: There was nothing so very remarkable in
    23 tokens: In another moment down went Alice after
    42 tokens: The rabbit-hole went straight on like a
   156 tokens: Either the well was very deep , or she
    64 tokens: ' Well ! ' thought Alice to herself , '
   164 tokens: Down , down , down . Would the fall new
   156 tokens: Presently she began again . ' I wonder
   244 tokens: Down , down , down . There was nothing
```



- Word Docs
- PDF's
- Text Files

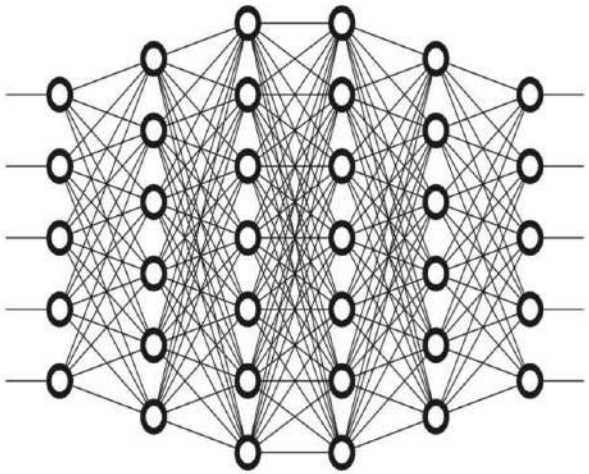
- Stop Words
- Stemming
- Tokenization

- Bag of Words
- TF-IDF
- Word Embeddings

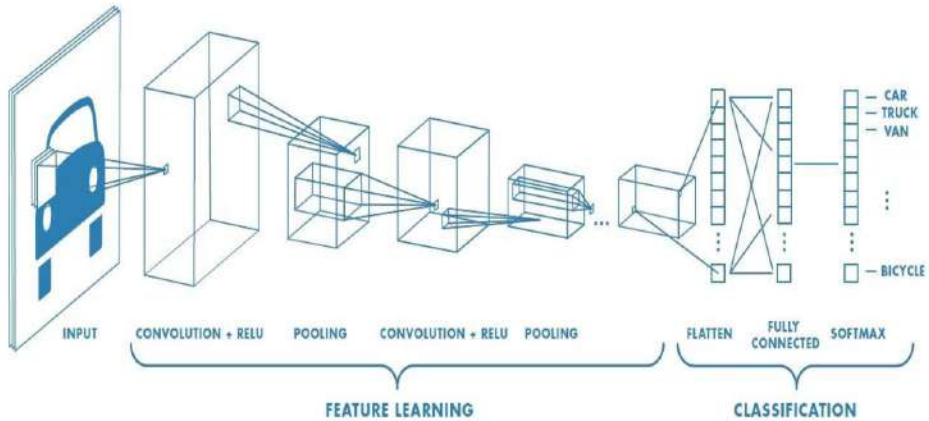
Deep Learning Architecture

https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306

Convolution neural network



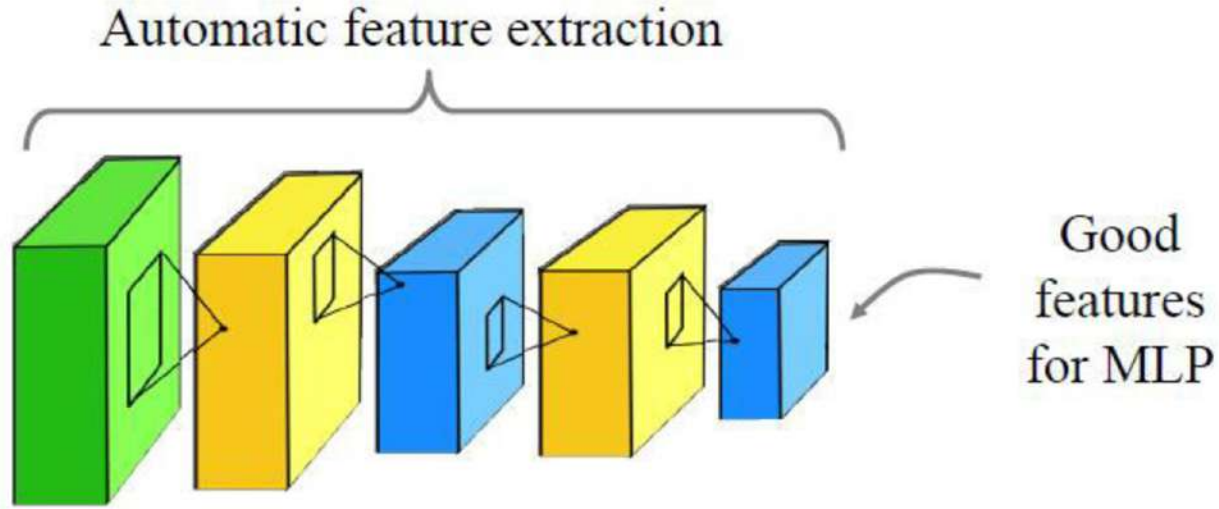
VS



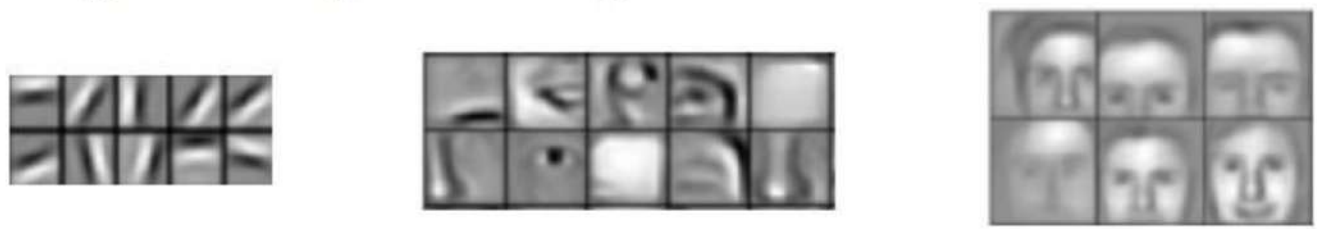
- Why do I see boxes every time , where are neurons?
- Why is that I see only a part of image being taken ?
- Why Convolution, pooling , flattening, dense layer?

Learning deep representations

Neurons of deep convolutional layers learn complex representations that can be used as features for classification with MLP.

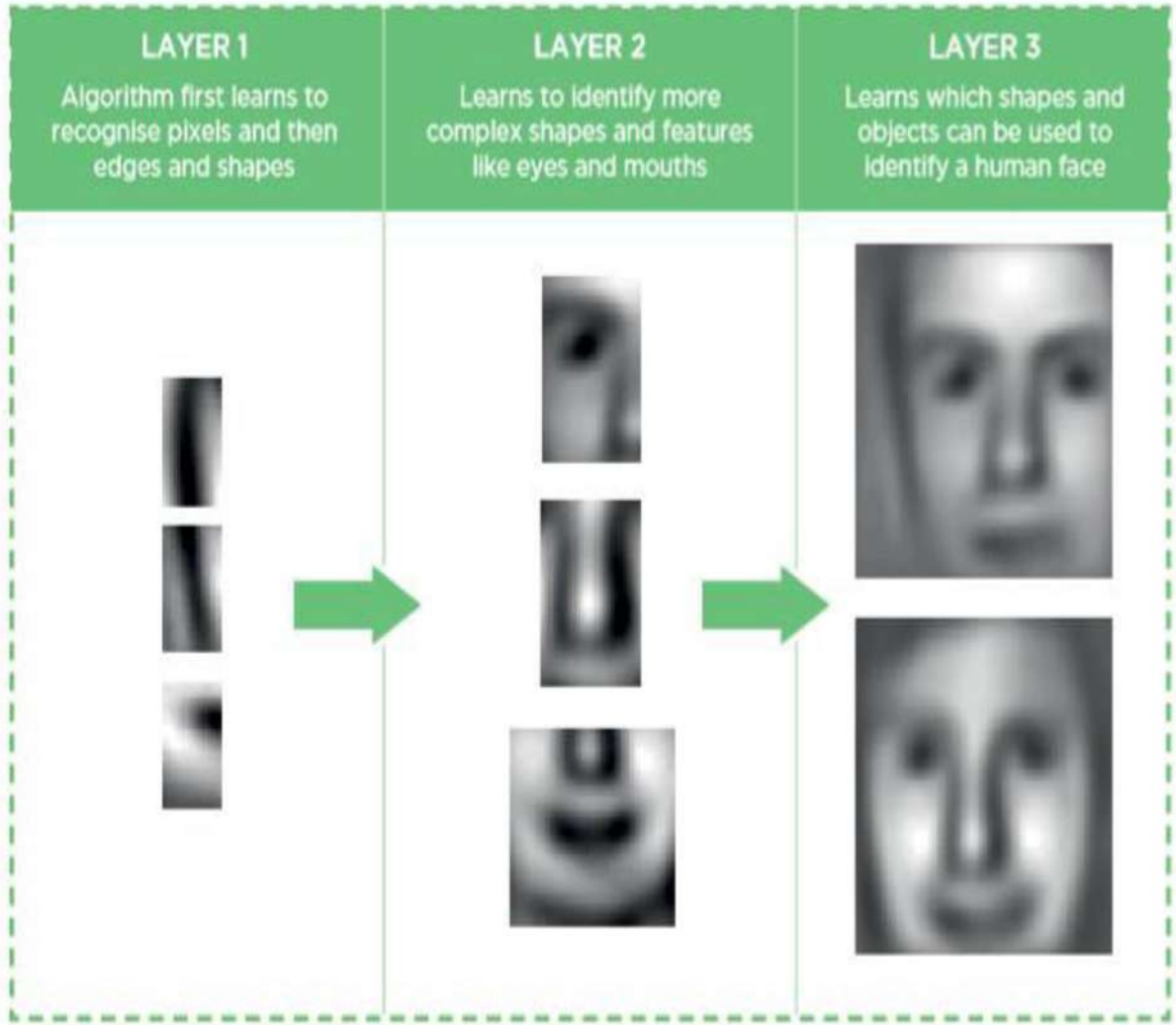


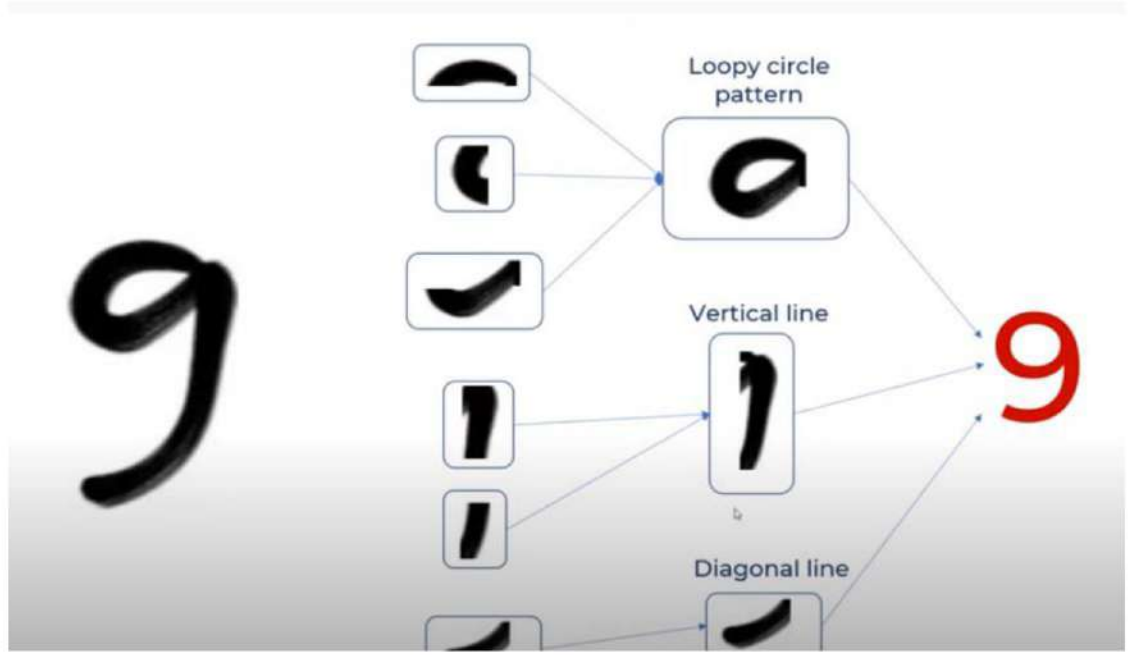
Inputs that provide highest activations:



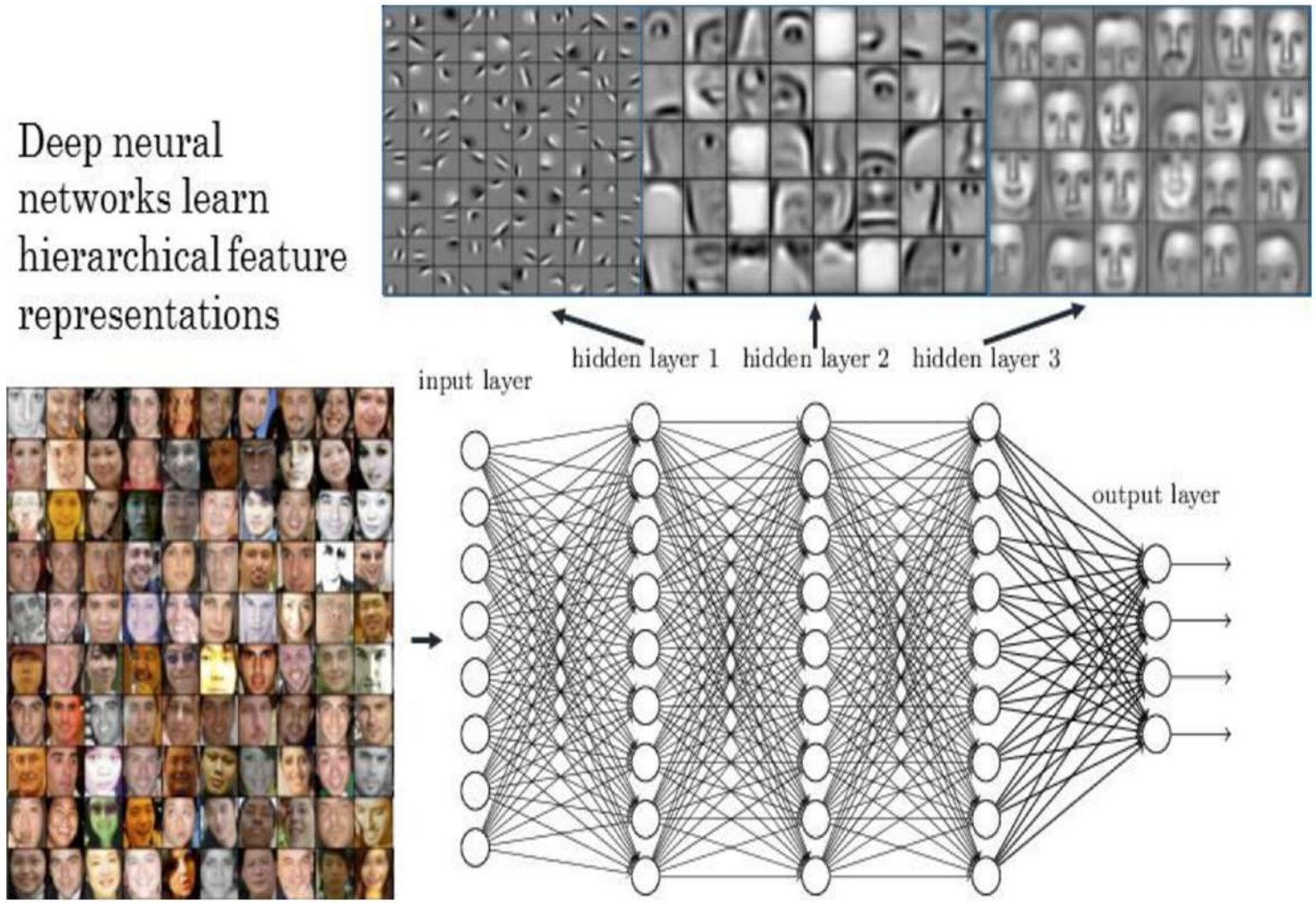
http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

What are layers and what happens inside?

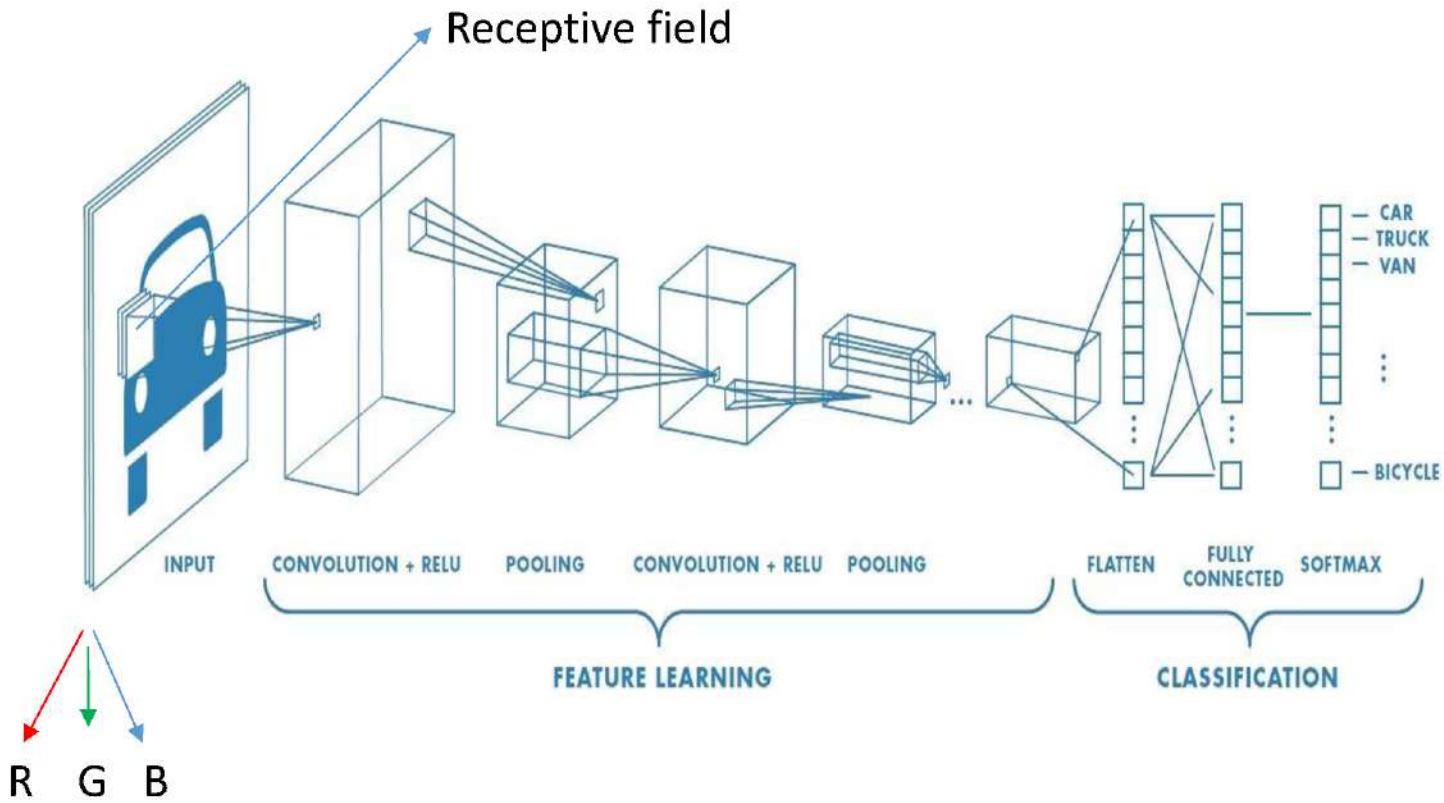



Loopy circle pattern
Vertical line
Diagonal line
9

What are layers and what happens inside?

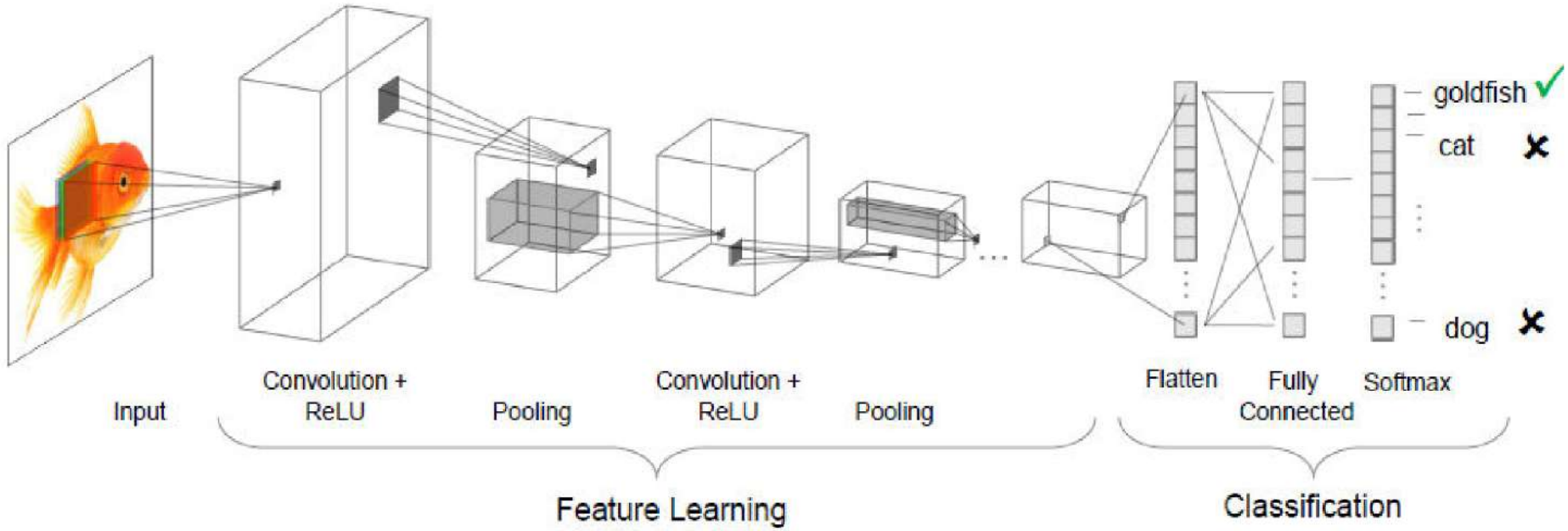


CNN –simplified



- Why do I see boxes every time , where are neurons?
- Why is that I see only a part of image being taken
- Why Convolution, pooling , flattening, dense layer

Convolutional Neural Network (CNN)/ Deep Neural Network (DNN)



- The process of building a Convolutional Neural Network always involves four major steps.
- **Step - 1 : Convolution**
- **Step - 2 : Pooling**
- **Step - 3 : Flattening**
- **Step - 4 : Full connection**

Step1: Image data

- Great training datasets are CIFAR and CoCo. We'll use CIFAR.

255	144	...	120
57	100	...	32
...	...	...	...
251	41	...	23
57	209	...	118
27	59	...	19
...	...	...	...
4	203	...	69
35	170	...	199
17	11	...	97
...	...	...	...
145	165	...	210

Step 2: Convolution

- Convolution is an element-wise multiplication of two matrices (sub-matrix) followed by a sum

Input Image

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

5 (rows) × 5 (columns)

Kernel / Filter

1	-1	-1
-1	1	-1
-1	-1	1

3 × 3

Feature Map

3	-3	-2
-3	1	-3
-2	-3	3

3 × 3

$$\text{output size} = [\text{input size} - \text{kernel size}\ ] + 1$$

$$= [5 - 3] + 1$$

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

1	-1	-1
-1	1	-1
-1	-1	1

Stride = 1

3	-3	-2
-3	1	-3
-2	-3	3

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

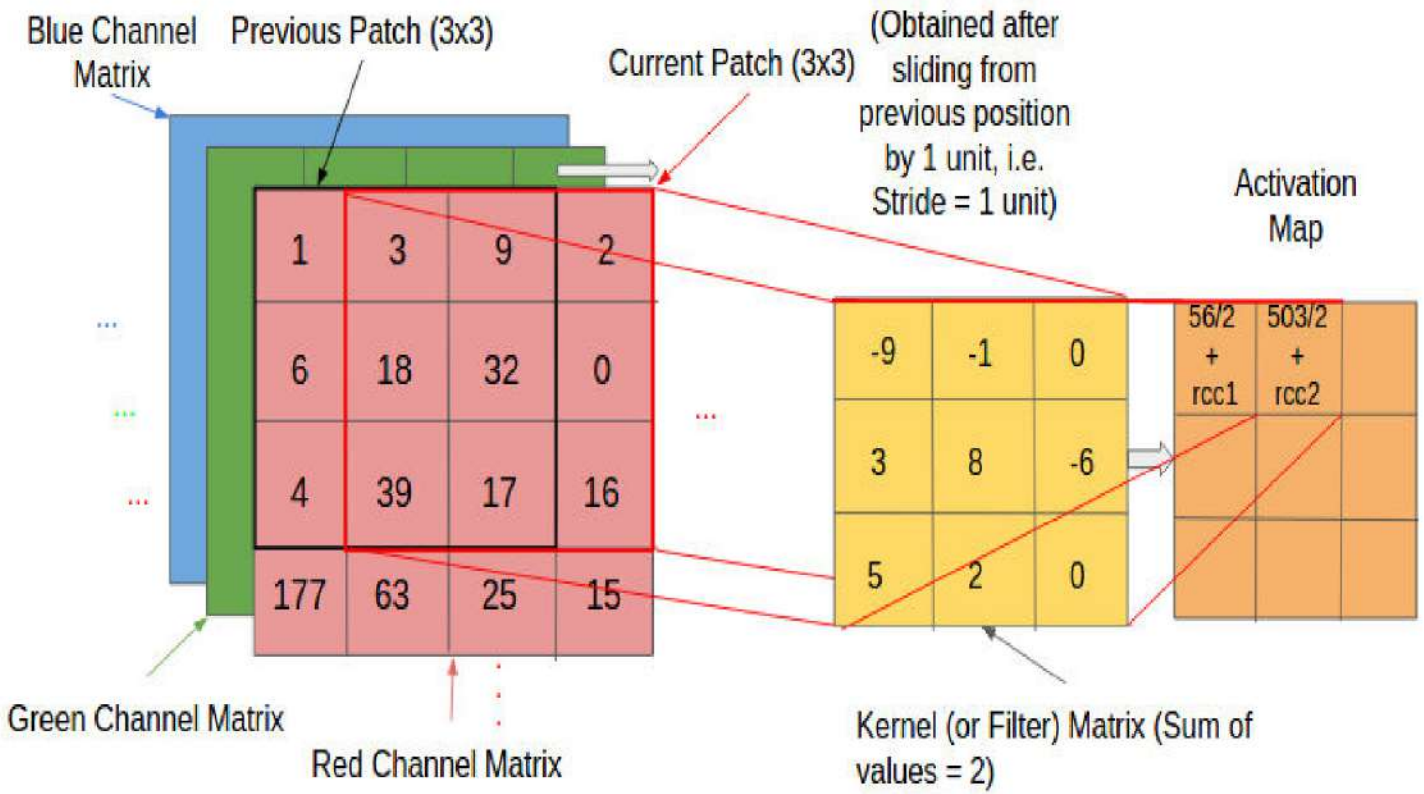
1	-1	-1
-1	1	-1
-1	-1	1

Stride = 2

3	-2
-2	3

$$\text{Output size(without padding)} = \frac{[\text{input size} - \text{kernel size}]}{\text{stride}} + 1$$

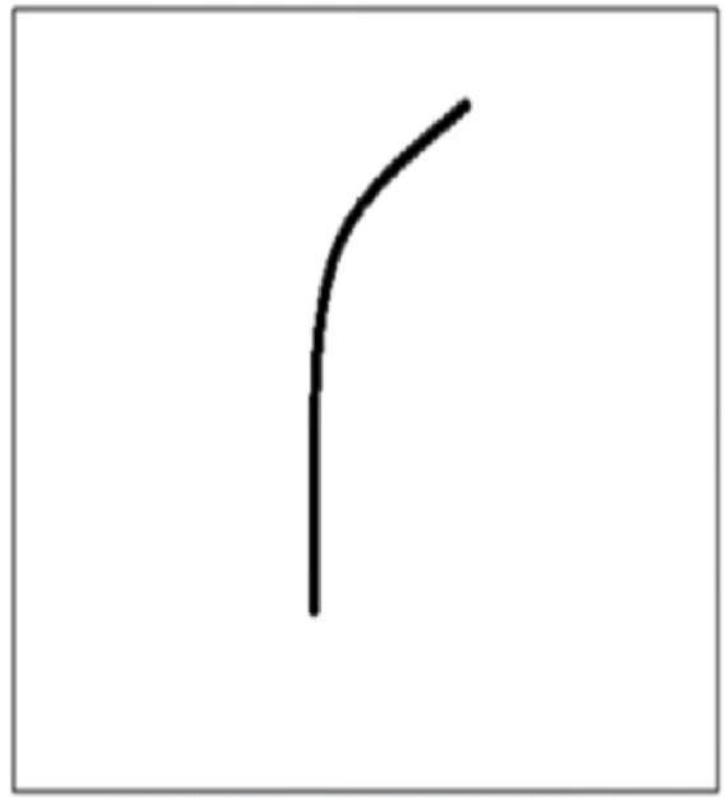
Step 2 : Convolution



Step 2 : Convolution

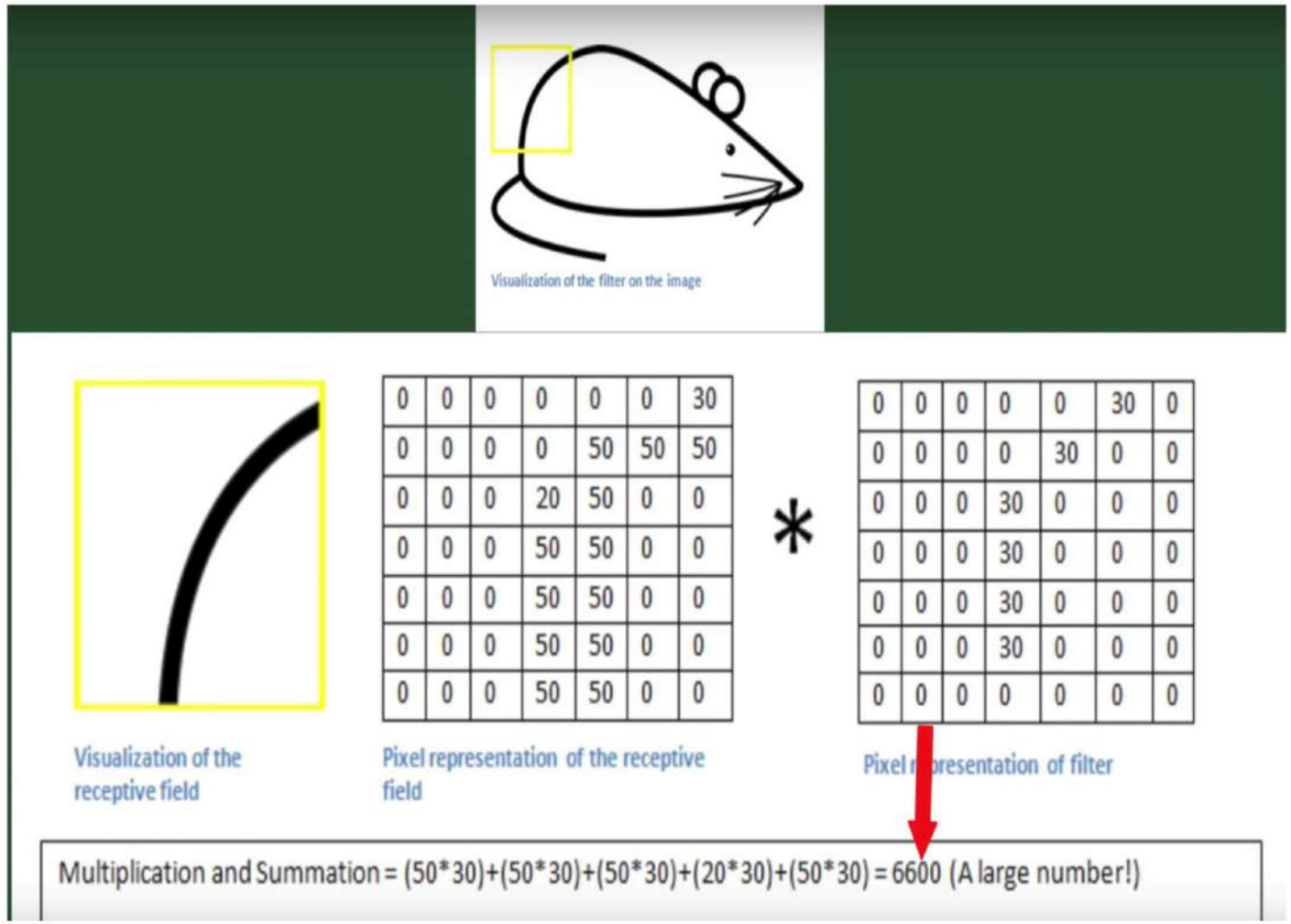
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

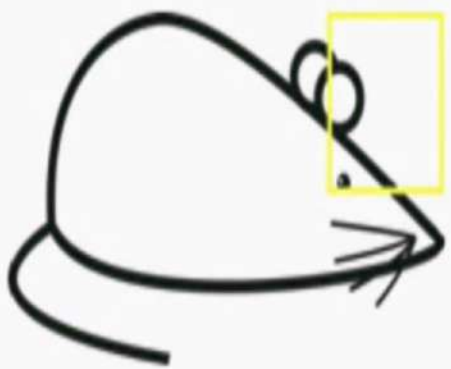


Visualization of a curve detector filter

Step 2 : Convolution



Step 2 : Convolution



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

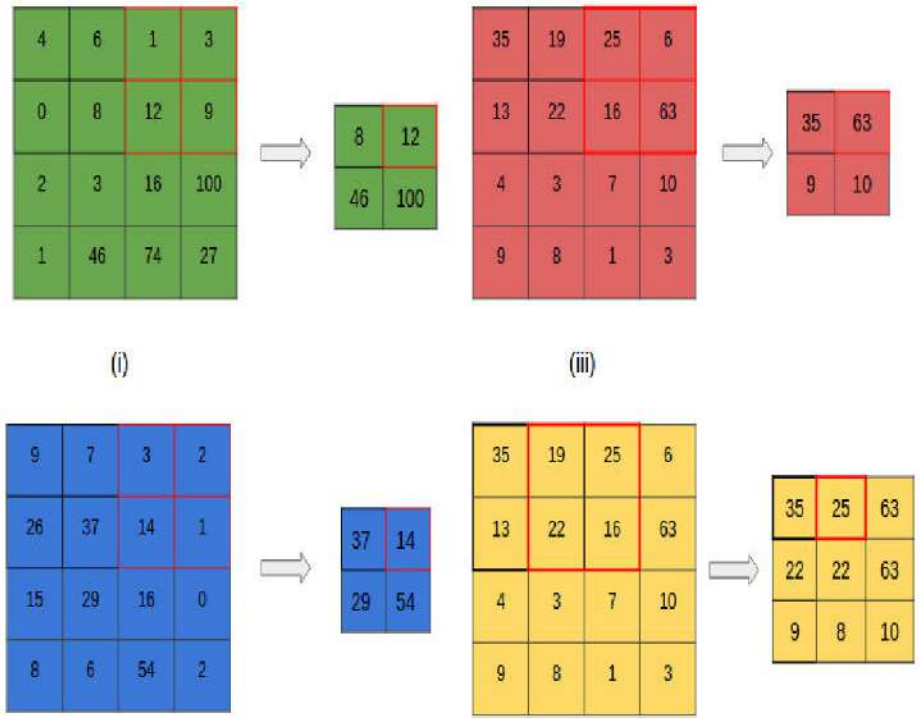
*

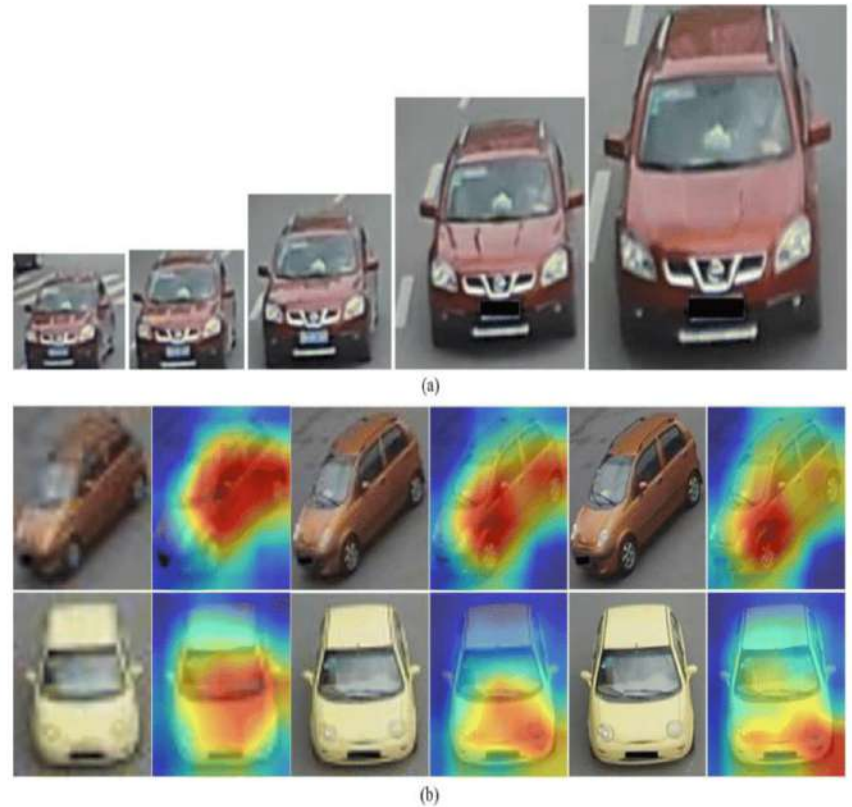
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

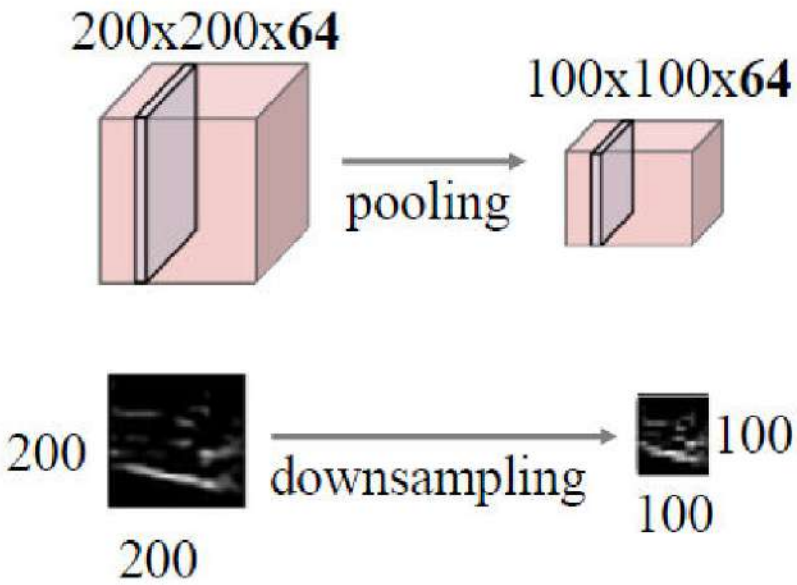
Step3 : Pooling





Pooling layer will help!

This layer works like a convolutional layer but doesn't have kernel, instead it calculates **maximum** or **average** of input patch values.



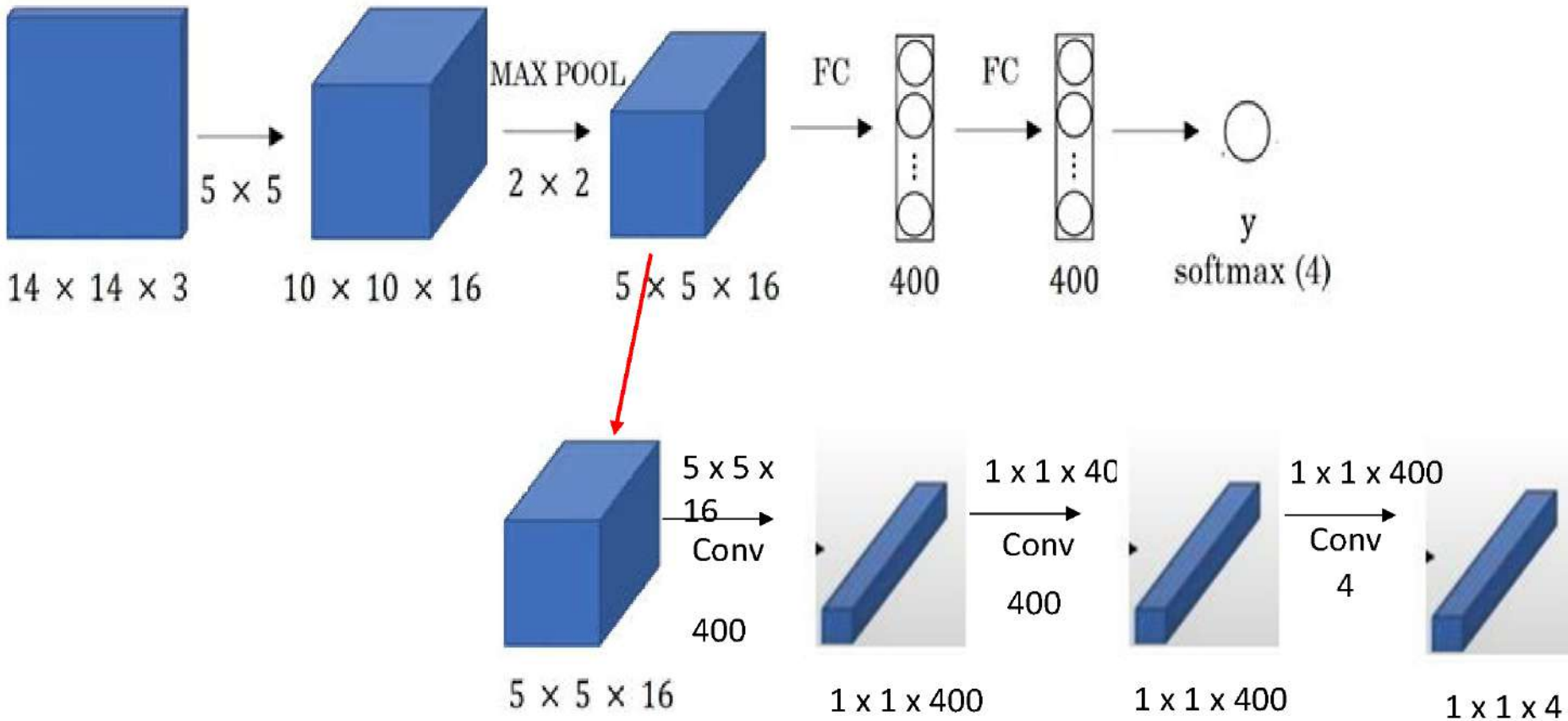
Single depth slice

1	1	1	4
2	**6**	5	**8**
3	2	1	0
1	1	3	**5**

6	8
3	5

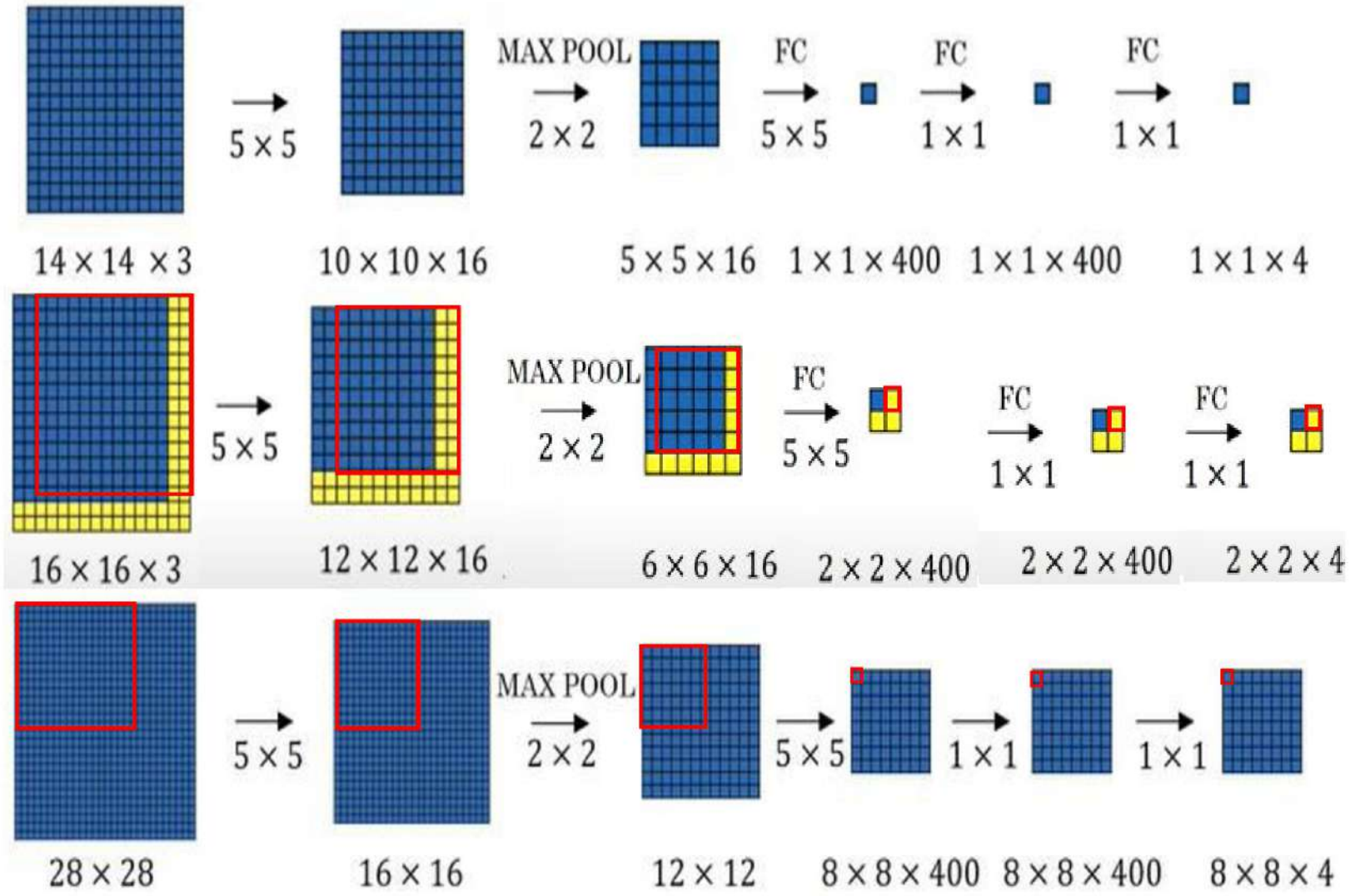
2x2 **max pooling** with stride 2

Turning FC Layer into Convolutional Layer



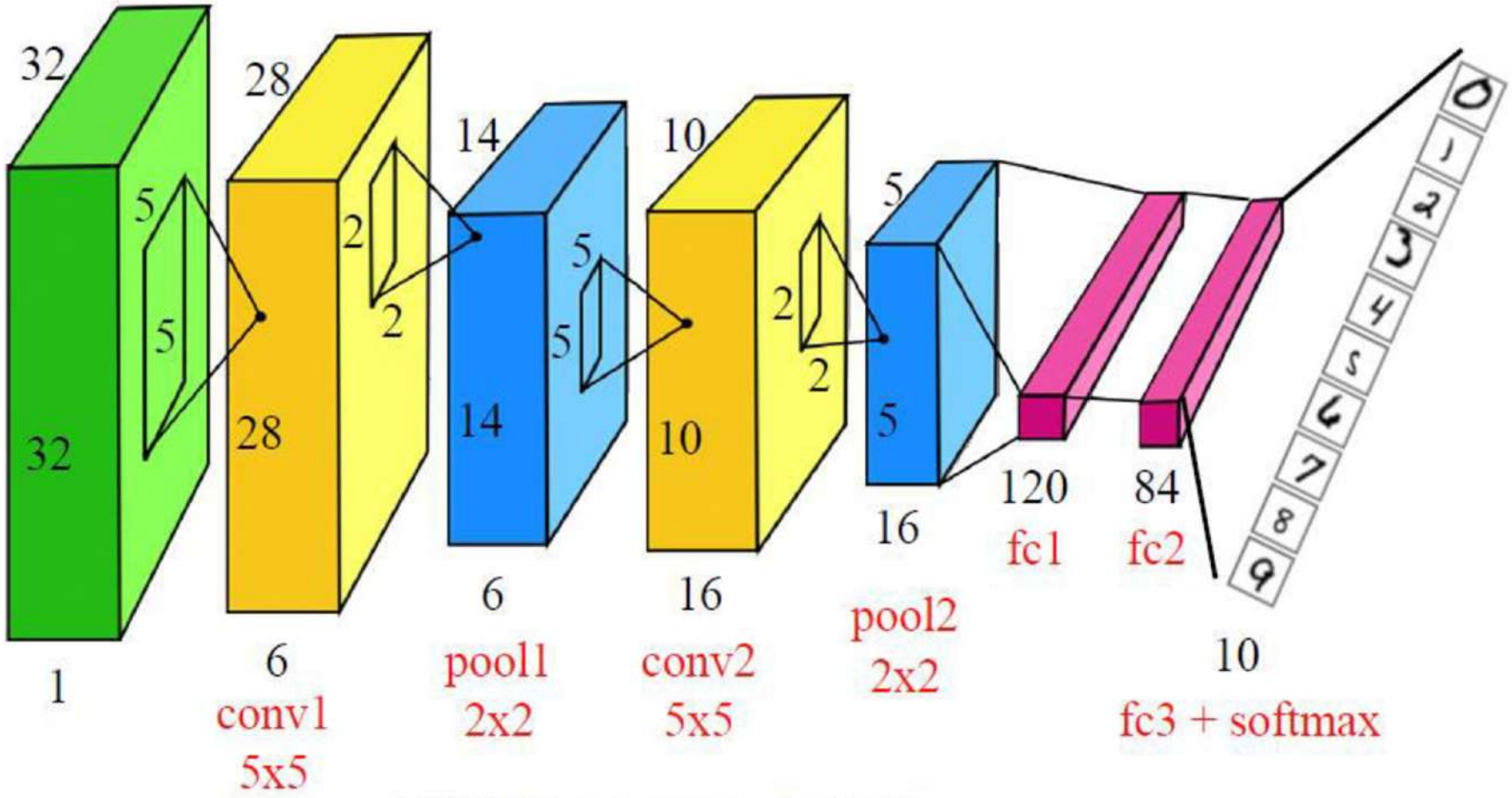
https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d

Convolution Implementation of Sliding Windows



Putting it all together into a simple CNN

LeNet-5 architecture (1998) for handwritten digits recognition on MNIST dataset:



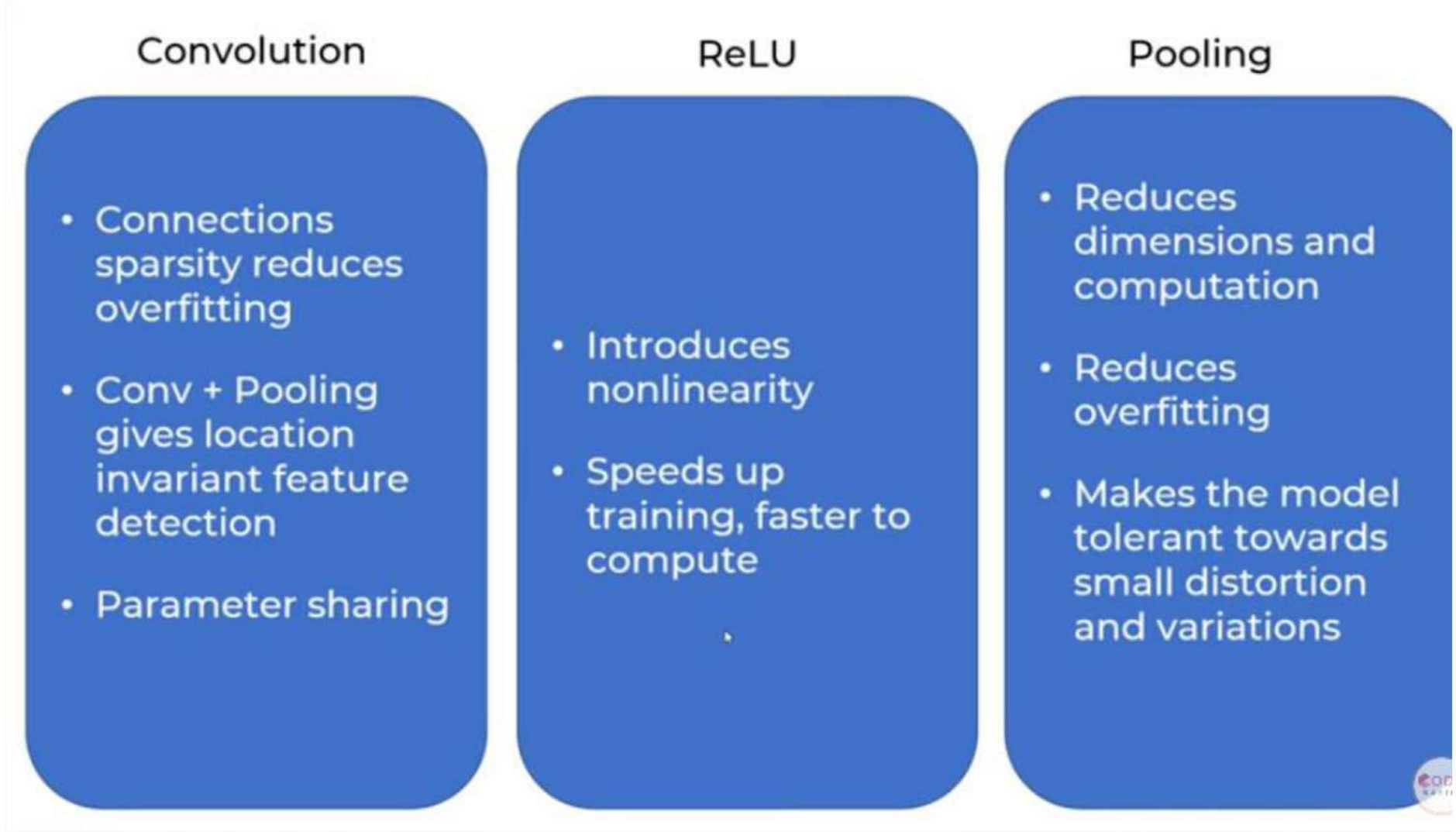
http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Benefits of pooling

Reduces dimensions & computation

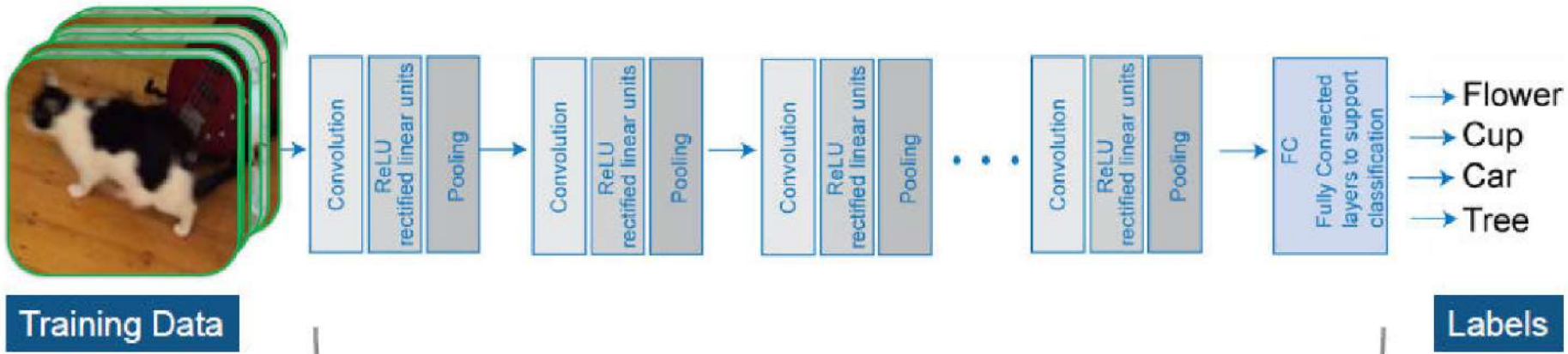
Reduce overfitting as there are less parameters

Model is tolerant towards variations, distortions


Convolution
• Connections sparsity reduces overfitting
• Conv + Pooling gives location invariant feature detection
• Parameter sharing
ReLU
• Introduces nonlinearity
• Speeds up training, faster to compute
Pooling
• Reduces dimensions and computation
• Reduces overfitting
• Makes the model tolerant towards small distortion and variations

What Happens During Training?

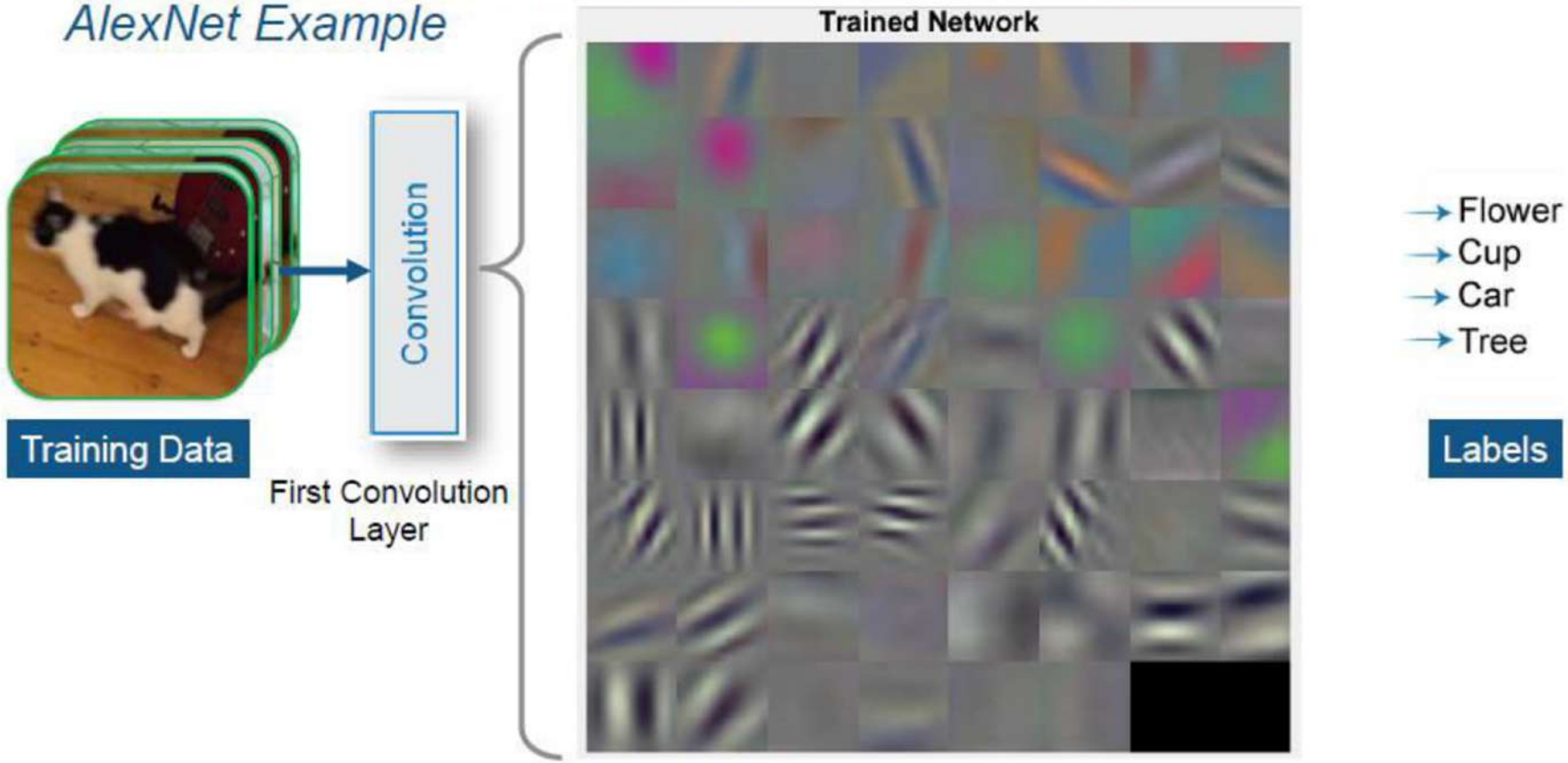
AlexNet Example



Layer weights are learned during training

What Network Learns During Training

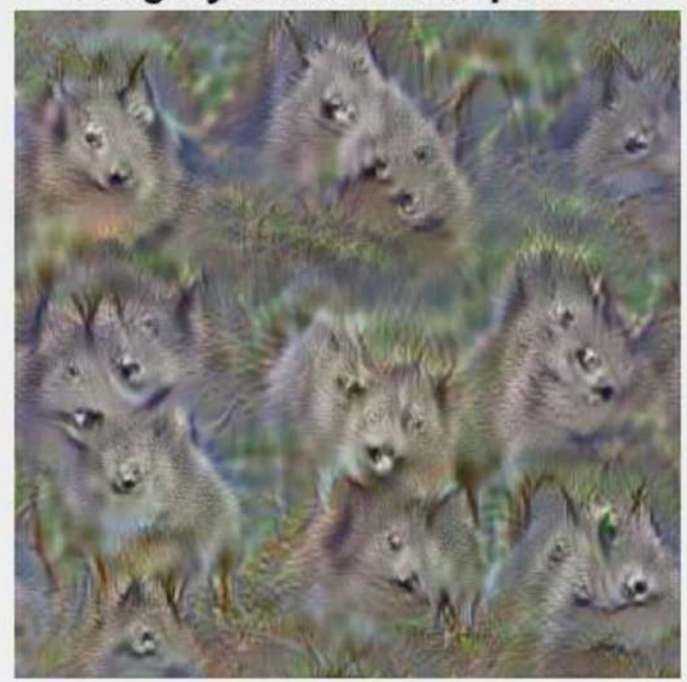
AlexNet Example



Visualize Features Learned During Training

AlexNet Example

Sample Training Data



Features Learned by Network

Visualize Features Learned During Training

AlexNet Example

Sample Training Data

Features Learned by Network

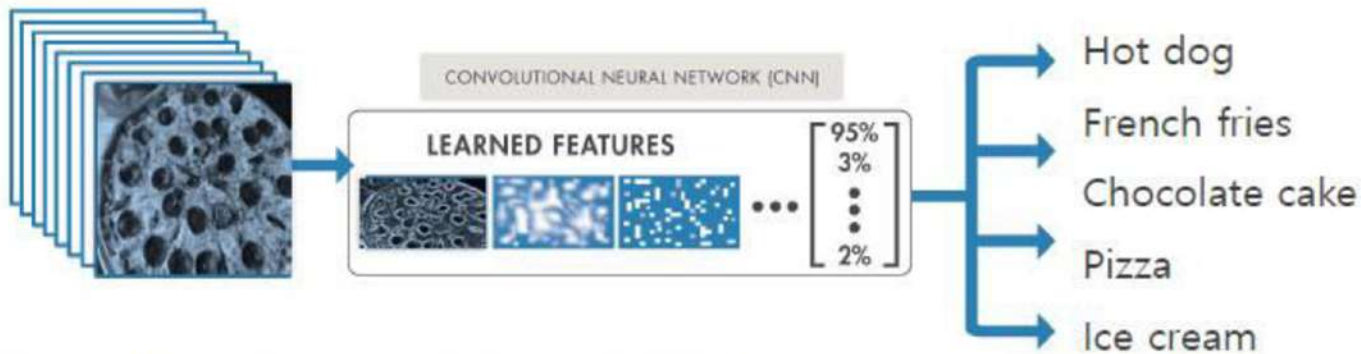
Lenet

- https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/

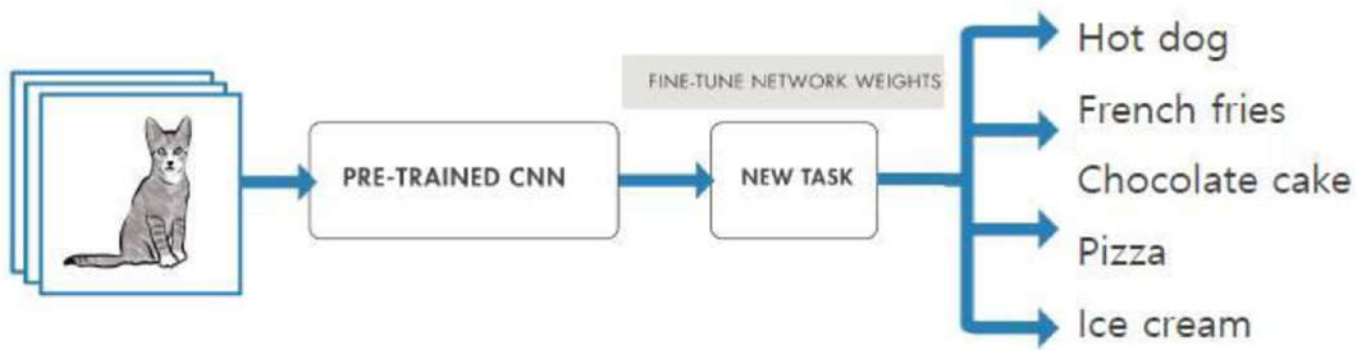
- https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306

Two Approaches for Deep Learning

1. Train a Deep Neural Network from Scratch



2. Fine-tune a pre-trained model (transfer learning)

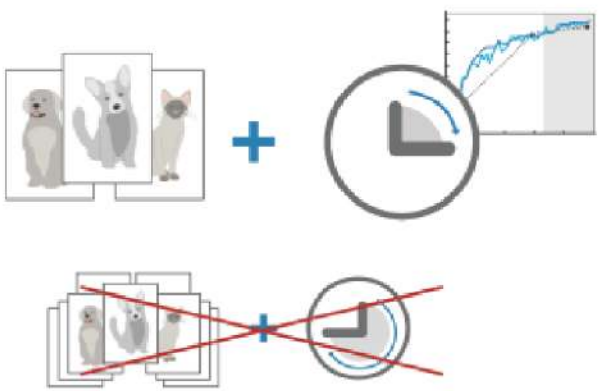


Transfer Learning

- **Transfer learning** is the process of taking a pre-trained model (the weights and parameters of a network that has been trained on a large dataset by somebody else) and “fine-tuning” the model with your own dataset.
- The idea is that this pre-trained model will act as a feature extractor.
- You will remove the last layer of the network and replace it with your own classifier (depending on what your problem space is).
- You then freeze the weights of all the other layers and train the network normally (Freezing the layers means not changing the weights during gradient descent/optimization).

Why Perform Transfer Learning ?

- Leverage best network types from top researchers
- Reference models (such as AlexNet, VGG-16, VGG-19) are great feature representations
- Requires less data and training time

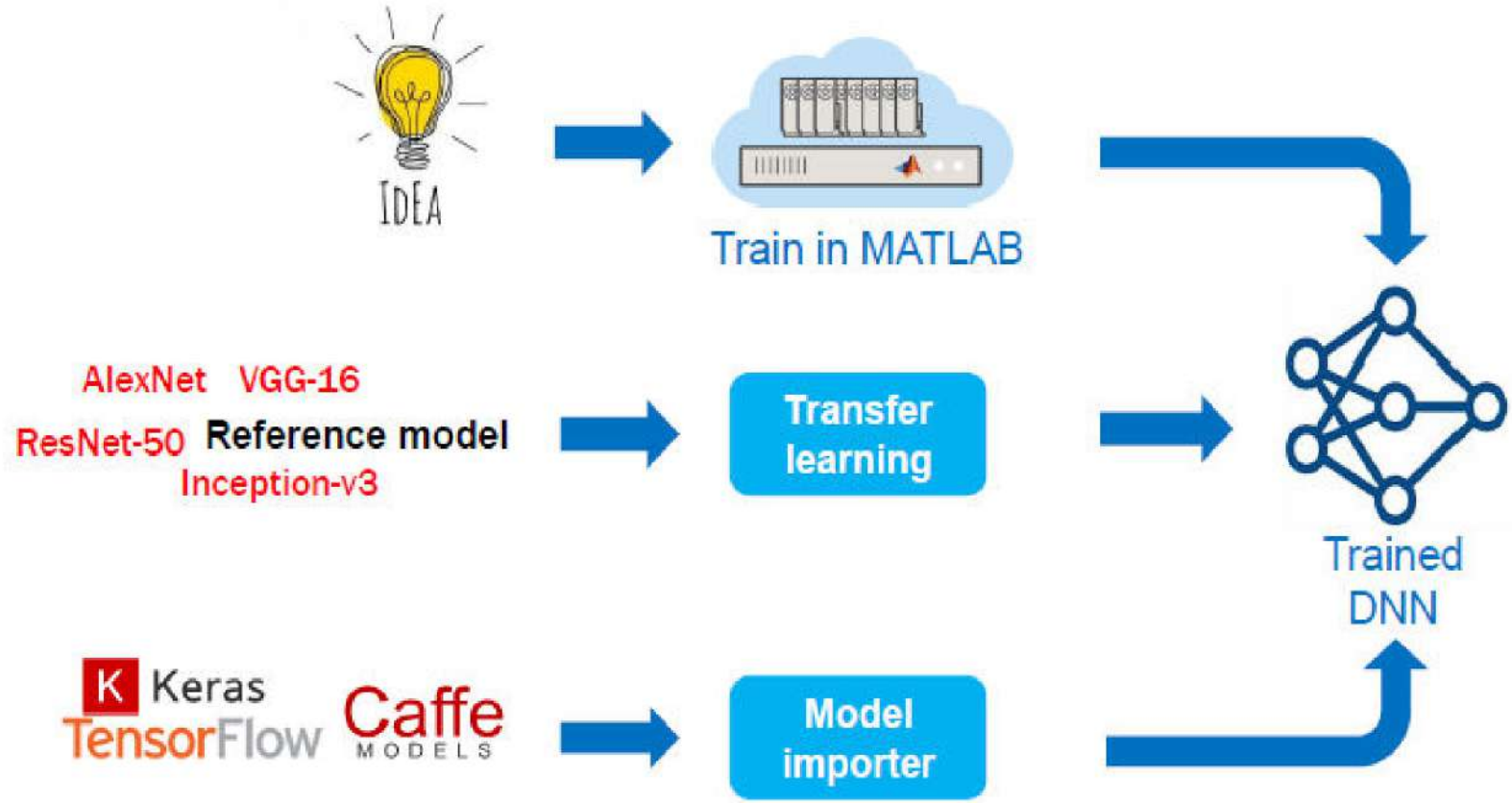


Pretrained Models in MATLAB

- AlexNet `net = alexnet;`
- VGG-16 `net = vgg16;`
- VGG-19 `net = vgg19;`
- GoogLeNet `net = googlenet;`
- Inceptionv3 `net = inceptionv3;`
- Resnet50 `net = resnet50;`
- Resnet101 `net = resnet101;`
- InceptionResnetv2 `net = inceptionresnetv2`
- Squeezenet `net = squeezenet;`

Download from within MATLAB 

How do I obtain trained DNN



Import the Latest Models

* single line of code to access model

Pretrained Models*

- `alexnet`
- `vgg16`
- `vgg19`
- `googlenet`
- `inception-v3`
- `resnet-18`
- `resnet-50`
- `resnet-101`
- `inception-resnet-v2`
- `squeezenet`
- `densenet-201`

```
net = alexnet;
net = vgg16;
net = vgg19;
net = googlenet;
net = inceptionv3;
net = resnet50;
net = resnet101;
net = inceptionresnetv2;
net = squeezenet;
net = densenet201;
```

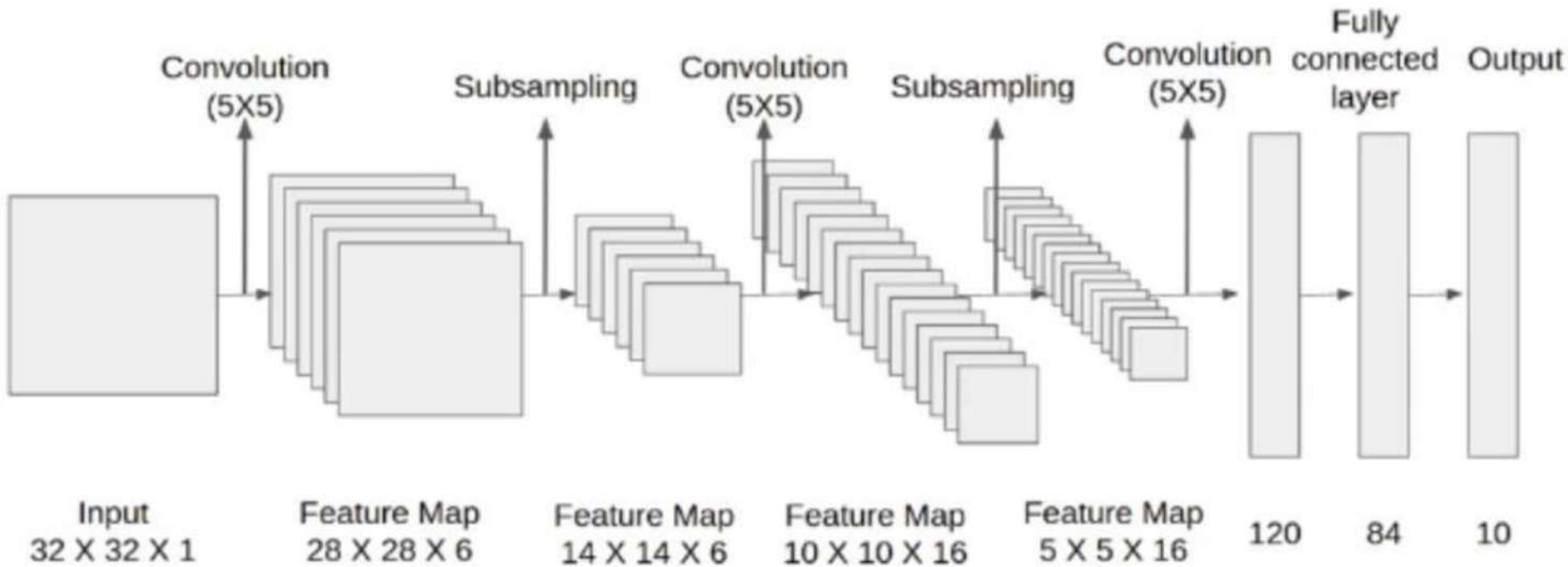
Lenet-5

- Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998
- used this architecture for recognizing the handwritten and machine-printed characters.
- The main reason behind the popularity of this model was its simple and straightforward architecture.
- It is a multi-layer convolution neural network for image classification.

Lenet-5

- The network has 5 layers with learnable parameters
- It has three sets of convolution layers with a combination of average pooling. Tanh is the activation function used in convolution layers.
- After the convolution and average pooling layers, we have two fully connected layers.
- At last, a Softmax classifier which classifies the images into respective class.

Here is the final architecture of the Lenet-5 model.



Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

Number of parameters

Alexnet

- Watch the video available in the below link
- https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

Alexnet

- proposed in 2012 by Alex Krizhevsky and his colleagues.
- won the Imagenet large-scale visual recognition challenge in 2012
- 8 layers
- five layers with a combination of max pooling followed by 3 fully connected layers
- Relu activation in each of these layers and softmax in the output layer.
- using the relu as an activation function accelerated the speed of the training process by almost six times.

Alexnet

- used the dropout layers, that prevented their model from overfitting.
- model is trained on the Imagenet dataset
- The Imagenet dataset has almost 14 million images across a thousand classes.
- The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.
- introduced padding to prevent the size of the feature maps from reducing drastically.
- The input to this model is the images of size 227X227X3.

What is the use of dropout layer?

- One of the method to avoid overfitting.
- One typical characteristic of CNNs is a Dropout layer.
- The Dropout layer is a mask that **nullifies the contribution of some neurons towards the next layer and leaves unmodified all others**.

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Fully Connected 1	-	-	-	-	4096	ReLU
Dropout 2	rate = 0.5	-	-	-	4096	-
Fully Connected 2	-	-	-	-	4096	ReLU
Fully Connected 3	-	-	-	-	1000	Softmax