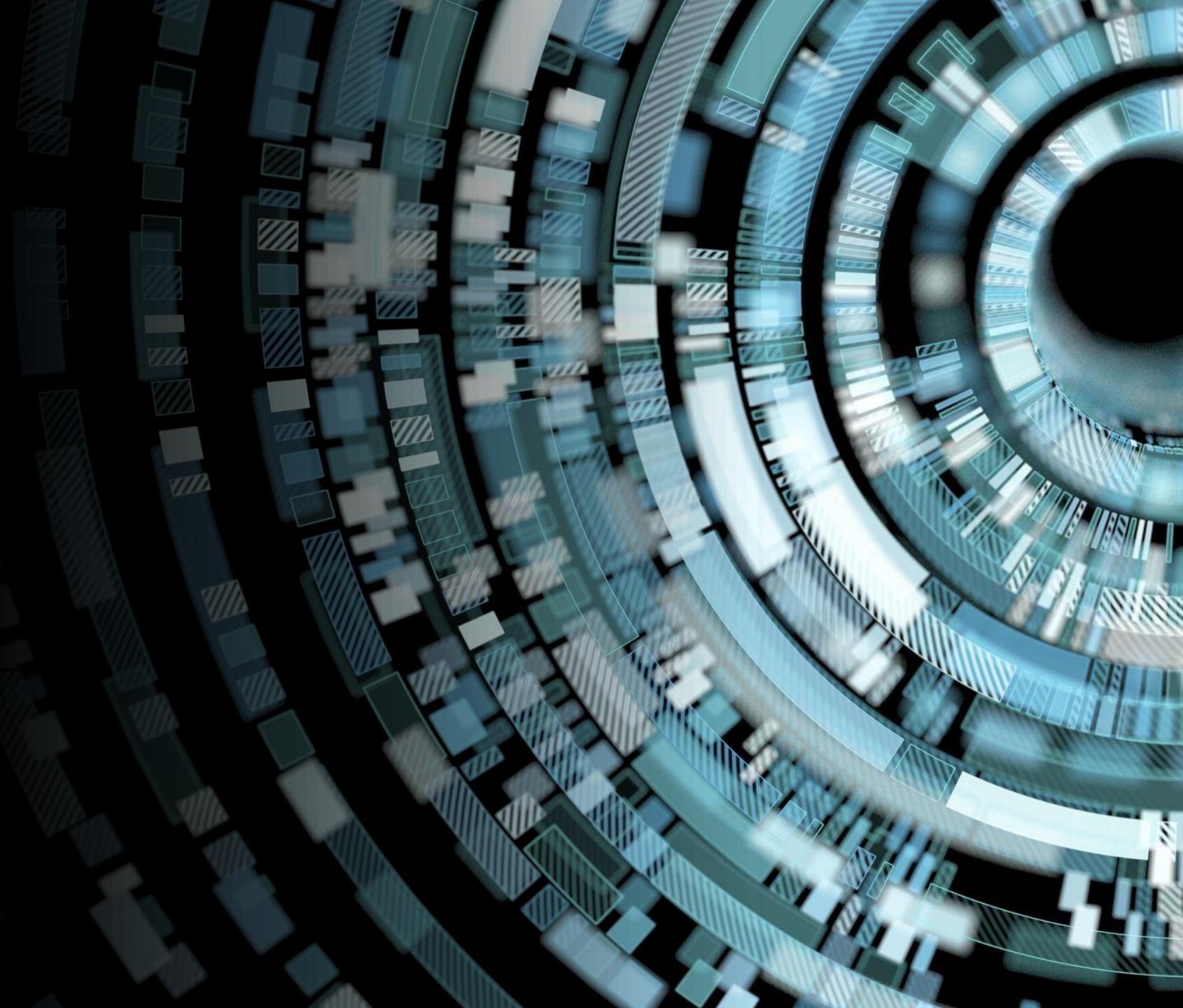


Unit 4: Machine learning and Deep Learning for Time Series





Generating and Selecting Features for a Time Series

Concerns you should keep in mind when generating time series features.

- Nature of the Time Series : Stationarity, Length of Time series
- Domain Knowledge
- External Considerations

Some Simple and often used time series features are:

- Mean and variance, Maximum and minimum
- Difference between last and first values

Number of local maxima and minima

- Smoothness of the time series, Periodicity and autocorrelation of the time series



Open Source Time Series Feature Generation Libraries

- The ***tsfresh*** Python module: It implement large and general set of features which includes
 - *Descriptive statistics*
 - *Physics-inspired indicators of nonlinearity and complexity*
 - *History-compressing counts*
- The ***Cesium*** time series analysis platform
 - Features that describe the overall distribution of the data values, without regard to its temporal relationships.
 - Features that describe the distribution of the timing of data:
 - Features that describe measures of the periodicity of the behavior within the time series.

Time Series Classification: Machine learning Methods

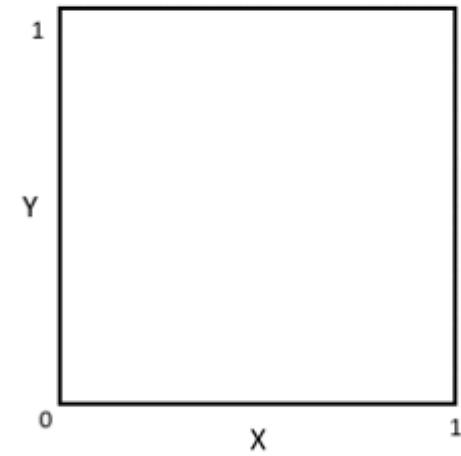
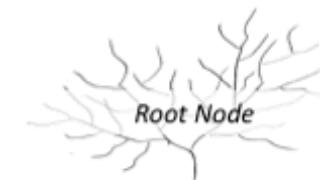
Decision Tree Methods

- Tree-based methods mirror the way humans make decisions: one step at a time, and in a highly nonlinear fashion.

For example : Stock marketing, Medical signals like EEG and ECG



Decision Tree

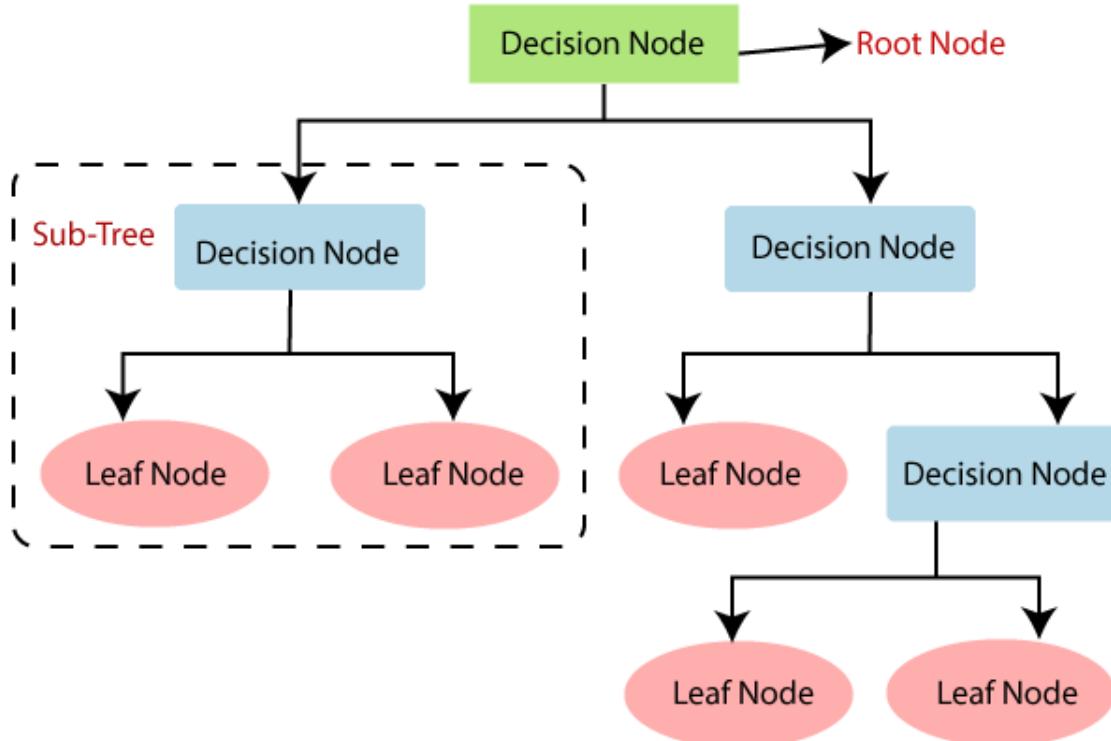


For more tutorials: annalyzin.wordpress.com





Terminologies in Decision Tree



Root Node: represents entire population and further divided in two/ more homogeneous sets.

Splitting: Process of dividing node in two or more sub-nodes.

Decision Node: When a sub-node splits into further sub-nodes

Leaf / Terminal Node: Nodes do not split.

Pruning: When we remove sub-nodes of a decision node.

Branch/Sub-Tree: A subsection of the entire tree.

Parent and Child Node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

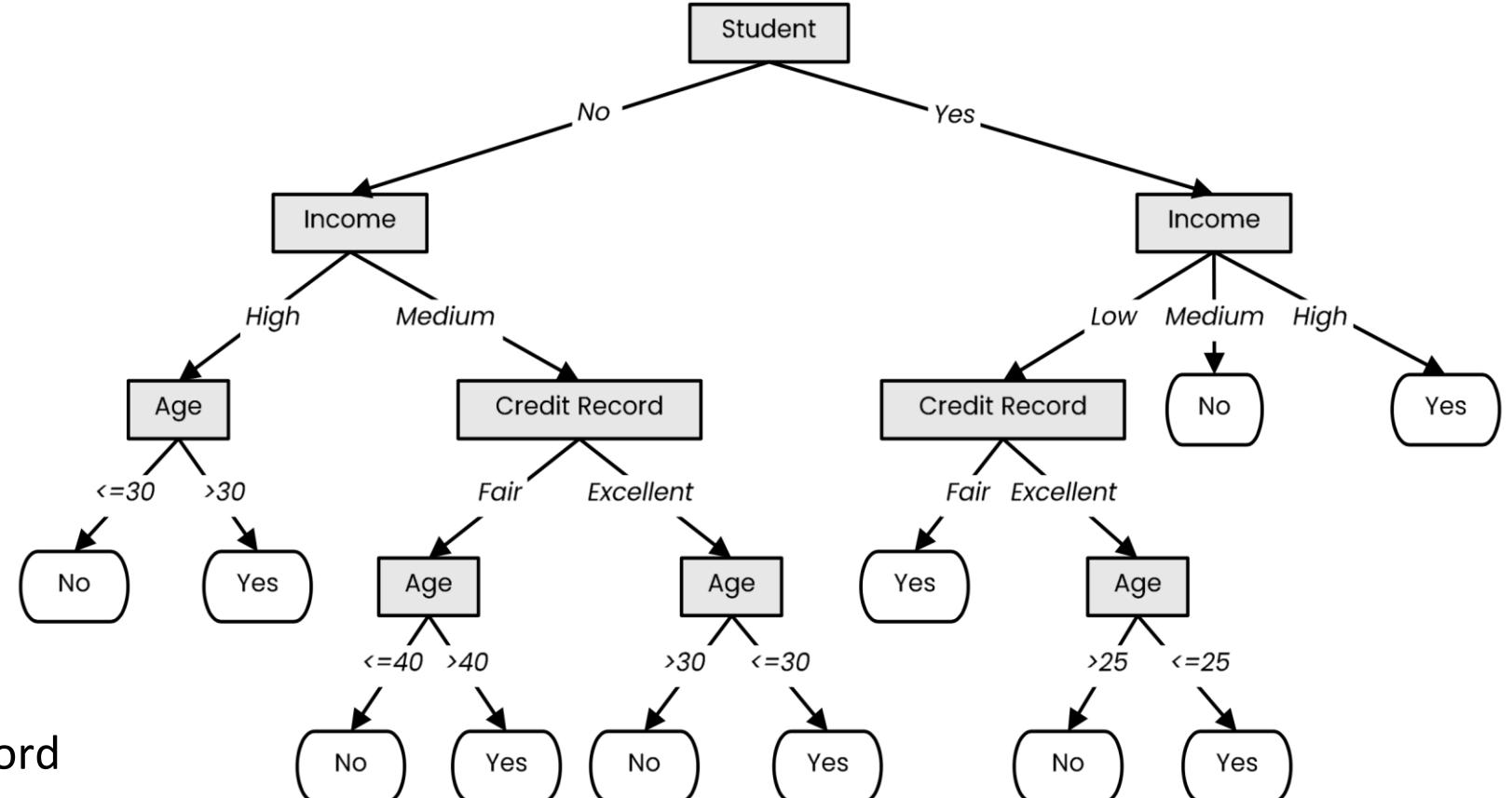
Who to loan?



- Not a student
- 45 years old
- Medium income
- Fair credit record



- Student
- 27 years old
- Low income
- Excellent credit record





Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

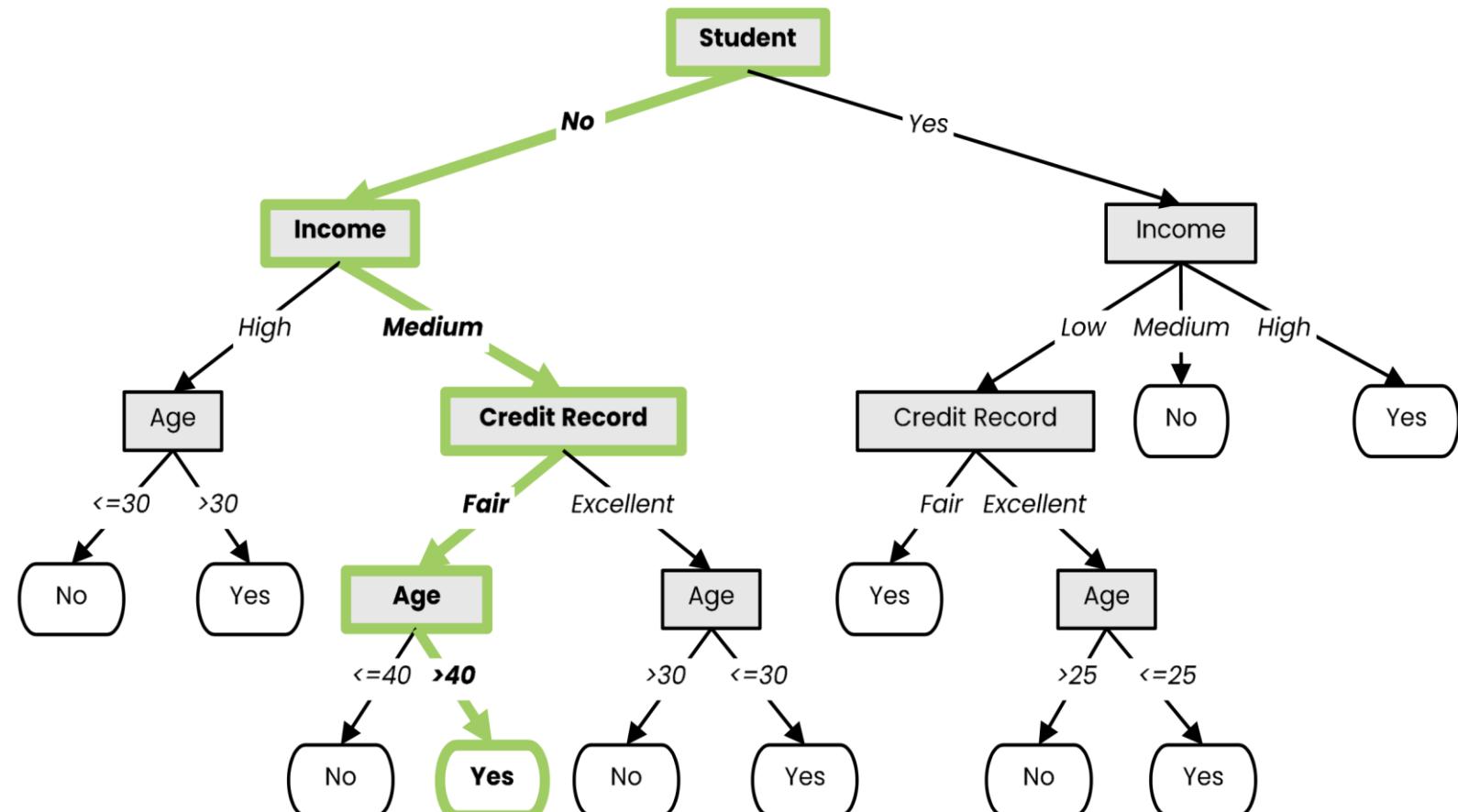
Who to loan?



- Not a student
- 45 years old
- Medium income
- Fair credit record
- Yes



- Student
- 27 years old
- Low income
- Excellent credit record





Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

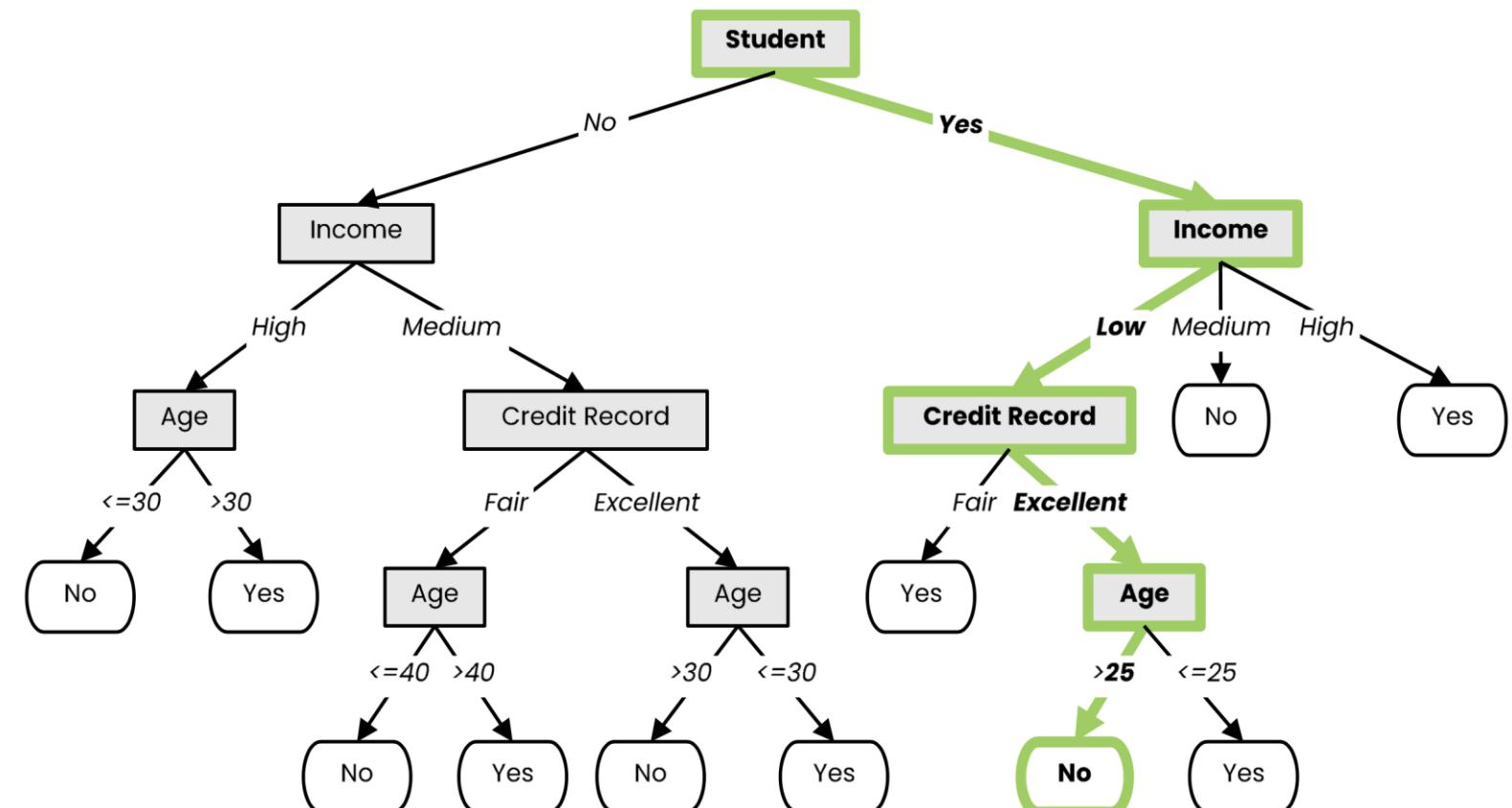
Who to loan?



- Not a student
- 45 years old
- Medium income
- Fair credit record
- Yes



- Student
- 27 years old
- Low income
- Excellent credit record





Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

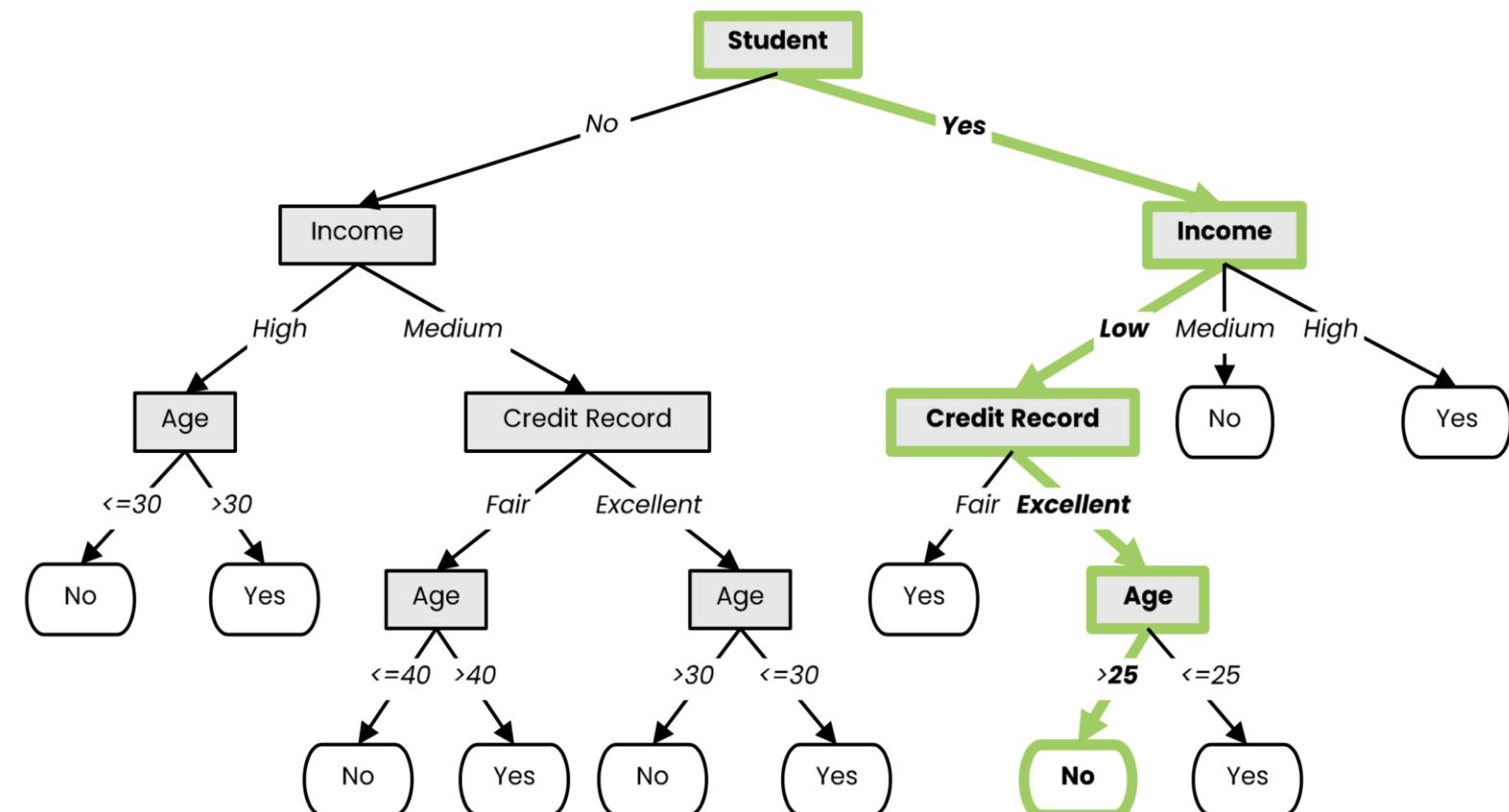
Who to loan?



- Not a student
- 45 years old
- Medium income
- Fair credit record
- Yes



- Student
- 27 years old
- Low income
- Excellent credit record
- No

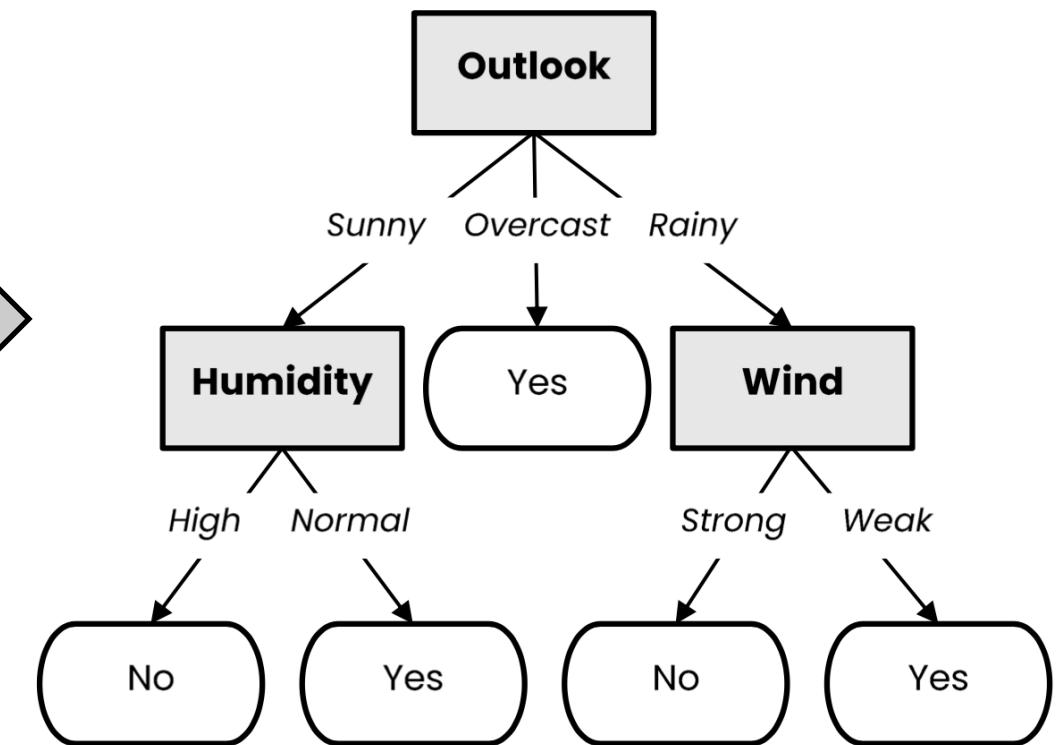
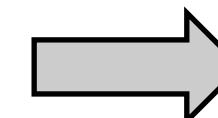




Decision Tree Learning

- We use labeled data to obtain a suitable decision tree for future predictions
- We want a decision tree that works well on unseen data, while asking as few questions as possible

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rainy	Mild	High	Strong	No





Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
 - Recursively repeat this step until we can surely decide the label

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rainy	Mild	High	Strong	No

Outlook



Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
 - Recursively repeat this step until we can surely decide the label

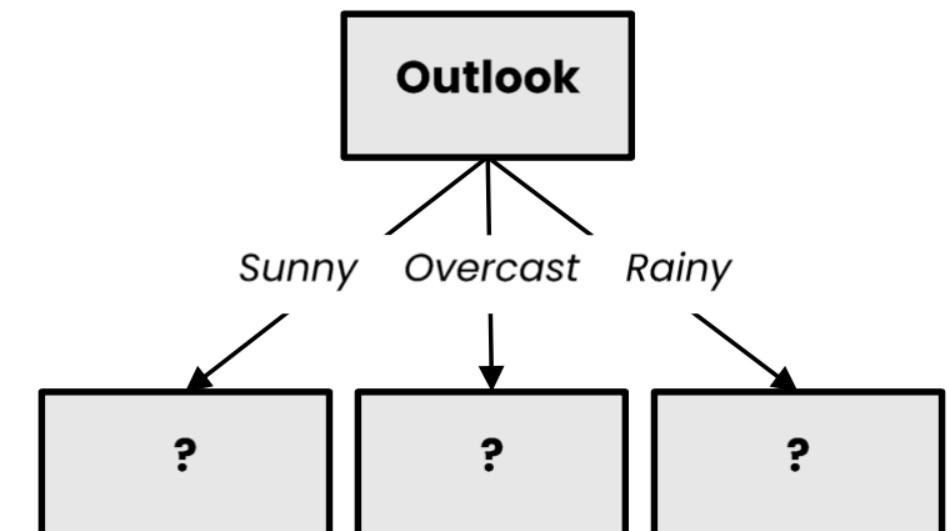
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Outlook = Overcast

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rainy

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No





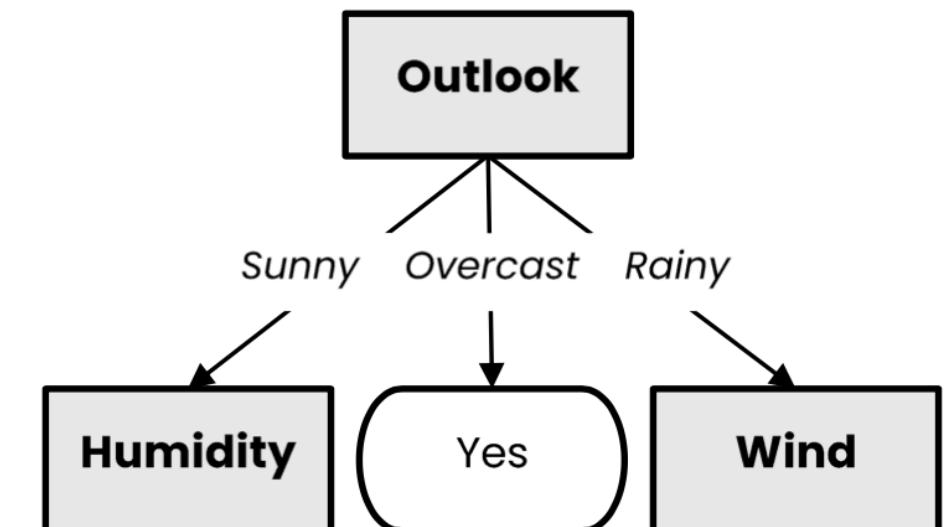
Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
 - Recursively repeat this step until we can surely decide the label

	Temperature	Humidity	Wind	Play Tennis?
Outlook = Sunny	Hot	High	Weak	No
	Hot	High	Strong	No
	Mild	High	Weak	No
	Cool	Normal	Weak	Yes
	Mild	Normal	Strong	Yes

	Temperature	Humidity	Wind	Play Tennis?
Outlook = Overcast	Hot	High	Weak	Yes
	Cool	Normal	Strong	Yes
	Mild	High	Strong	Yes
	Hot	Normal	Weak	Yes

	Temperature	Humidity	Wind	Play Tennis?
Outlook = Rainy	Mild	High	Weak	Yes
	Cool	Normal	Weak	Yes
	Cool	Normal	Strong	No
	Mild	Normal	Weak	Yes
	Mild	High	Strong	No





Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
 - Recursively repeat this step until we can surely decide the label

Outlook = Sunny

Humidity = High		
Temperature	Wind	Play Tennis?
Hot	Weak	No
Hot	Strong	No
Mild	Weak	No

Humidity = Normal		
Temperature	Wind	Play Tennis?
Cool	Weak	Yes
Mild	Strong	Yes

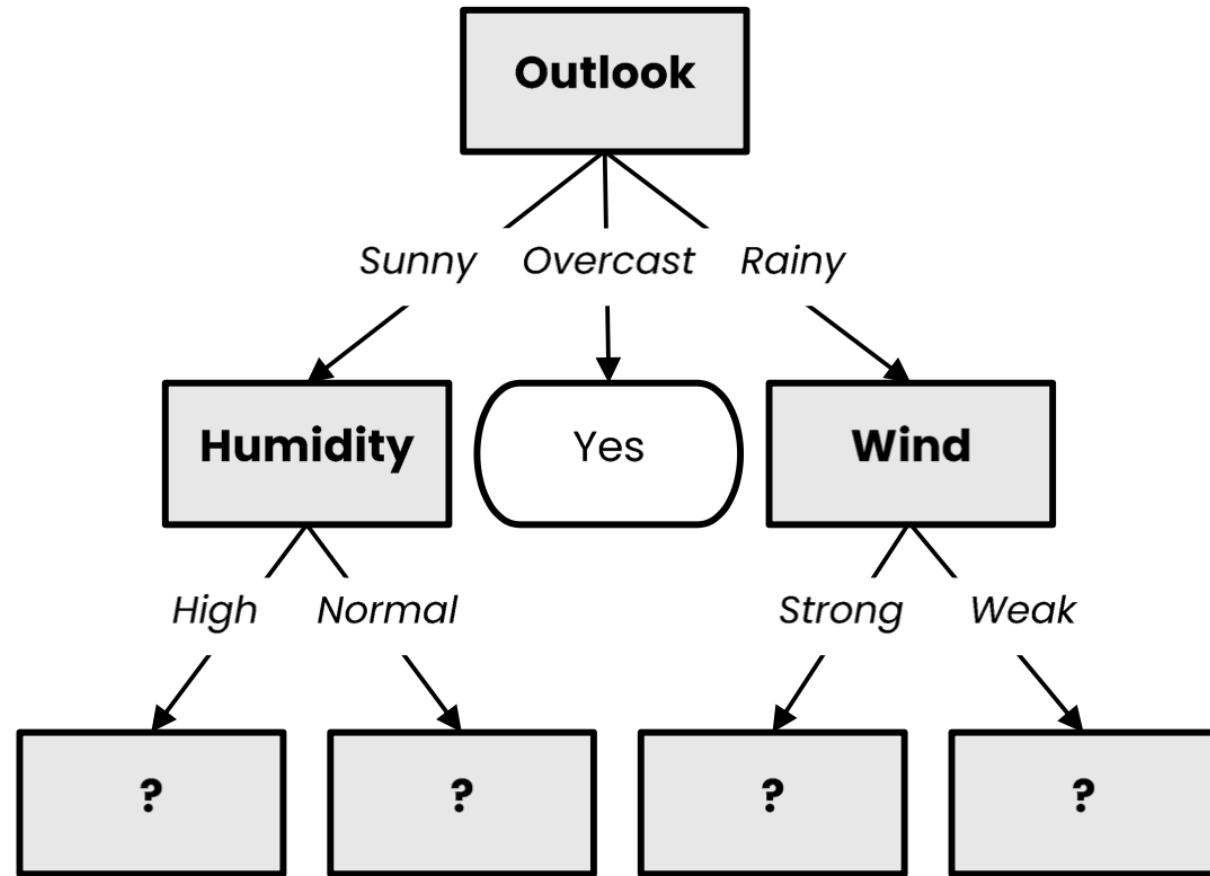
Outlook = Overcast

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rainy

Wind = Strong		
Temperature	Humidity	Play Tennis?
Cool	Normal	No
Mild	High	No

Wind = Weak		
Temperature	Humidity	Play Tennis?
Mild	High	Yes
Cool	Normal	Yes
Mild	Normal	Yes





Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
 - Recursively repeat this step until we can surely decide the label

Outlook = Sunny

Humidity = High		
Temperature	Wind	Play Tennis?
Hot	Weak	No
Hot	Strong	No
Mild	Weak	No

Humidity = Normal		
Temperature	Wind	Play Tennis?
Cool	Weak	Yes
Mild	Strong	Yes

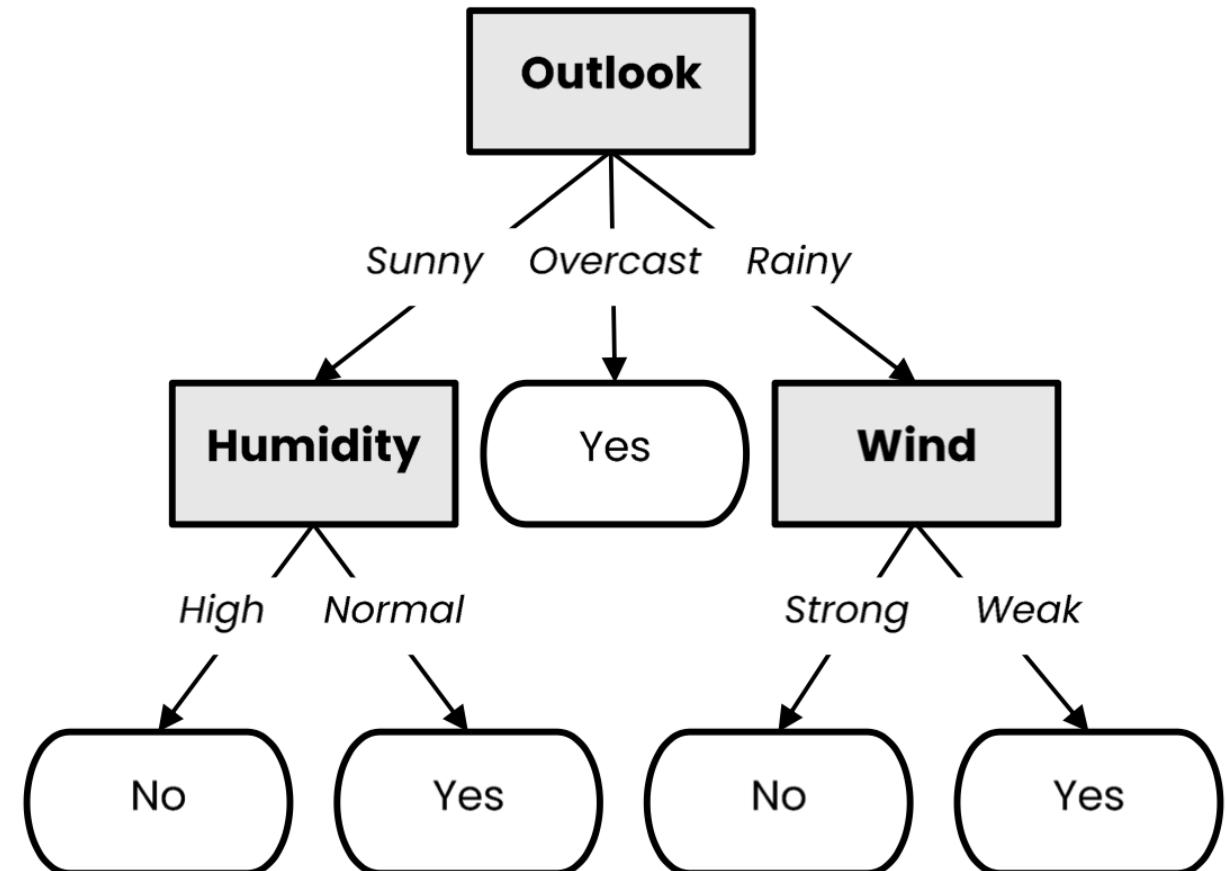
Outlook = Overcast

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rainy

Wind = Strong		
Temperature	Humidity	Play Tennis?
Cool	Normal	No
Mild	High	No

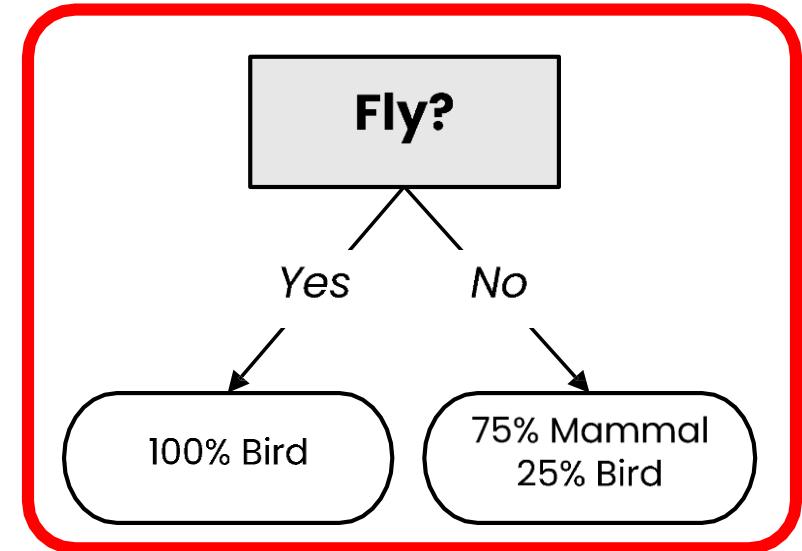
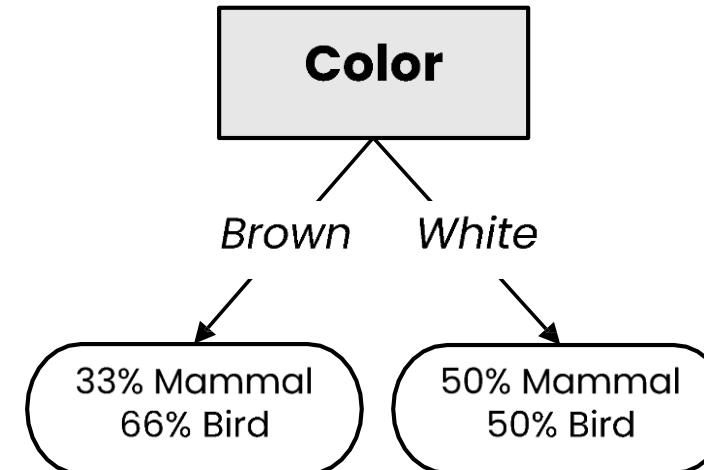
Wind = Weak		
Temperature	Humidity	Play Tennis?
Mild	High	Yes
Cool	Normal	Yes
Mild	Normal	Yes





What is a good attribute?

Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird



- Which attribute provides **better** splitting?
- Why?
 - Because the resulting subsets are more **pure**
 - Knowing the value of this attribute gives us **more information** about the label (**the entropy of the subsets is lower**)





When To Stop Building the Tree?

- When all the leaf nodes are pure (leaf nodes have data that belong to one class)
- When a certain criteria is met (Such as the height of the tree exceeds certain limit, In which case the leaf nodes might not be pure but might output probability of a class instead of the class itself)
- When all the features are used.





Summary

- Decision tree: A tree-like model that illustrates series of events leading to certain decisions
- A good attribute selection leads to the better splitting of the data
- Stopping the building of tree is also important to prevent the model complexity and overfitting





Attribute selection measure

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems Attribute selection measure methods are used.
- There are two popular techniques for ASM, which are:
 - *Entropy and Information Gain*
 - *Gini Index*



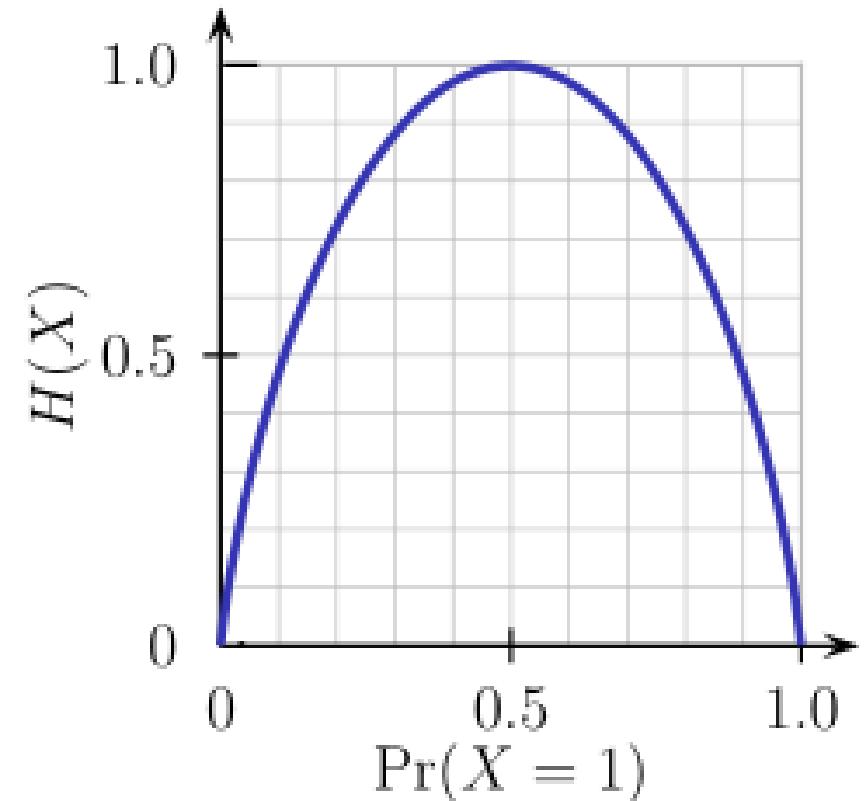


Entropy and Information Gain

- Entropy is a measure of the randomness in the information being processed



- The higher the entropy, the harder it is to draw any conclusions from that information
- Lower entropy implies greater predictability!



E.g. Flipping a coin

Note: ID3 (Iterative Dichotomiser) follows the rule — A branch with an entropy of zero is a leaf node and A branch with entropy more than zero needs further splitting.



Entropy

- To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:
 - Entropy using the frequency table of one attribute

$$E(S) = \sum_{i=1}^c -pi \log_2 p_i$$

PlayGolf	
Yes	No
9	5



$$\begin{aligned}\text{Entropy (PlayGolf)} &= \text{Entropy (5,9)} \\ &= \text{Entropy (0.36, 0.64)} \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$





Entropy

b) Entropy using the frequency table of two attributes:

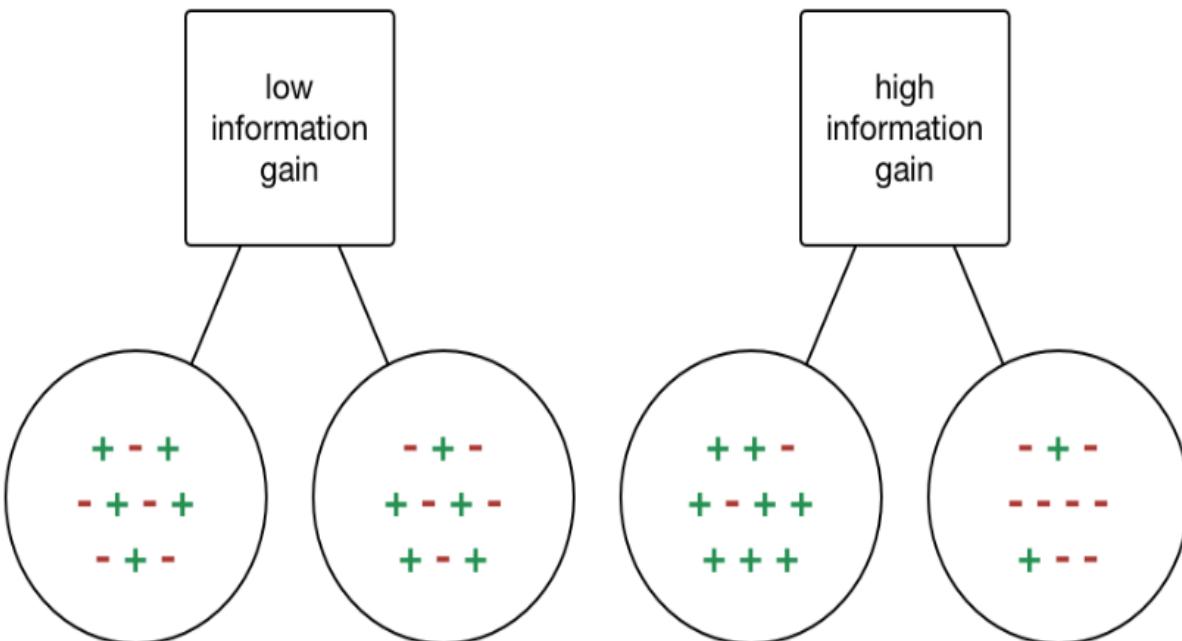
		PlayGolf		
		Yes	No	
Outlook	Sunny	2	3	5
	Overcast	4	0	4
	Rainy	3	2	5
				14

$$\begin{aligned} E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(2,3) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(3,2) \\ &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\ &= 0.693 \end{aligned}$$





Information Gain



- Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification.
- Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.
- Information gain is a decrease in entropy. It computes the *difference between entropy before split and average entropy after split of the dataset* based on given attribute values.

$$\text{Information Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$





Information Gain

- Step 1: Calculate entropy of the target.
- Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		PlayGolf	
		Yes	No
Outlook	Sunny	2	3
	Overcast	4	0
	Rainy	3	2
Gain = 0.247			

		PlayGolf	
		Yes	No
Temp	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		PlayGolf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		PlayGolf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$\text{Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

$$\text{Entropy}(\text{PlayGolf}) - \text{Entropy}(\text{PlayGolf}, \text{Outlook})$$

$$= 0.940 - 0.693$$

$$= 0.247$$





Outlook	Outlook = Sunny			
	Temperature	Humidity	Wind	Play Tennis?
	Hot	High	Weak	No
	Hot	High	Strong	No
	Mild	High	Weak	No
	Cool	Normal	Weak	Yes
	Mild	Normal	Strong	Yes

Outlook	Outlook = Overcast			
	Temperature	Humidity	Wind	Play Tennis?
	Hot	High	Weak	Yes
	Cool	Normal	Strong	Yes
	Mild	High	Strong	Yes
	Hot	Normal	Weak	Yes

Outlook	Outlook = Rainy			
	Temperature	Humidity	Wind	Play Tennis?
	Mild	High	Weak	Yes
	Cool	Normal	Weak	Yes
	Cool	Normal	Strong	No
	Mild	Normal	Weak	Yes
	Mild	High	Strong	No

- Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		PlayGolf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

Gain = 0.247

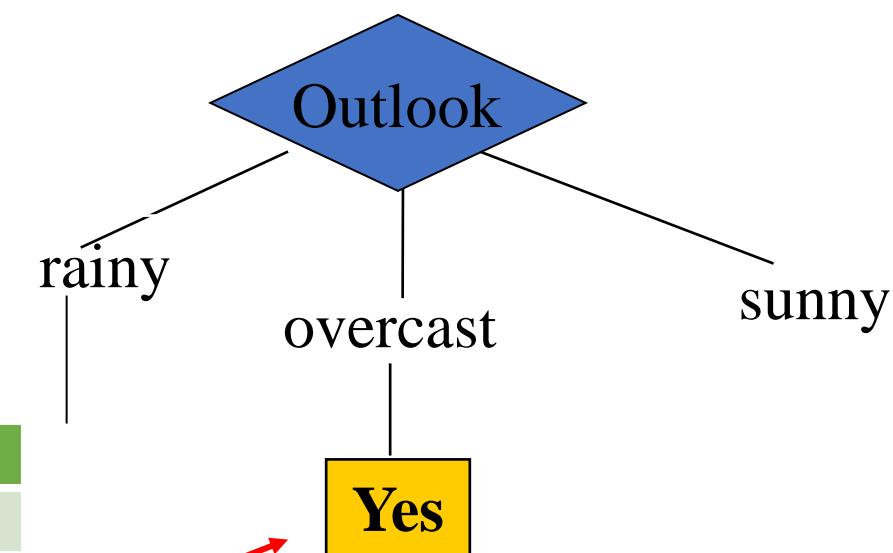




- Step 4a: A branch with entropy of 0 is a leaf node.

→ **Overcast**

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes



		PlayGolf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

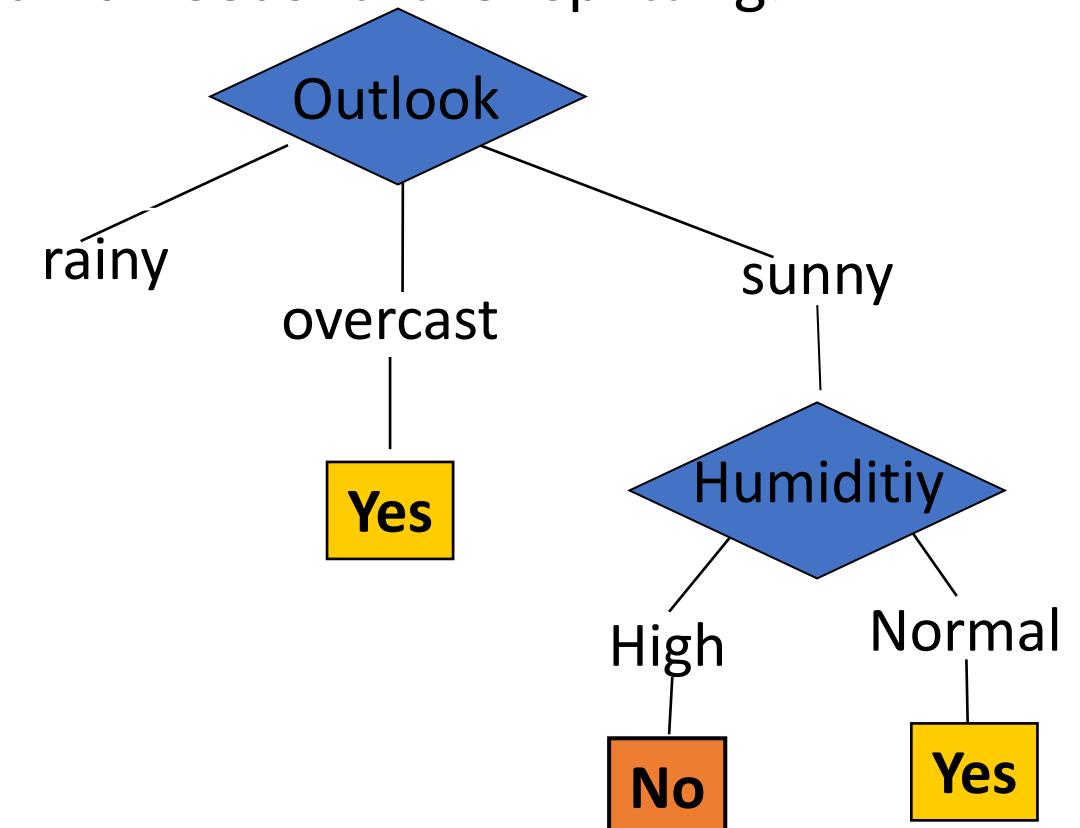
Gain = 0.247



- Step 4b: A branch with entropy more than 0 needs further splitting.

Sunny

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes



- Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

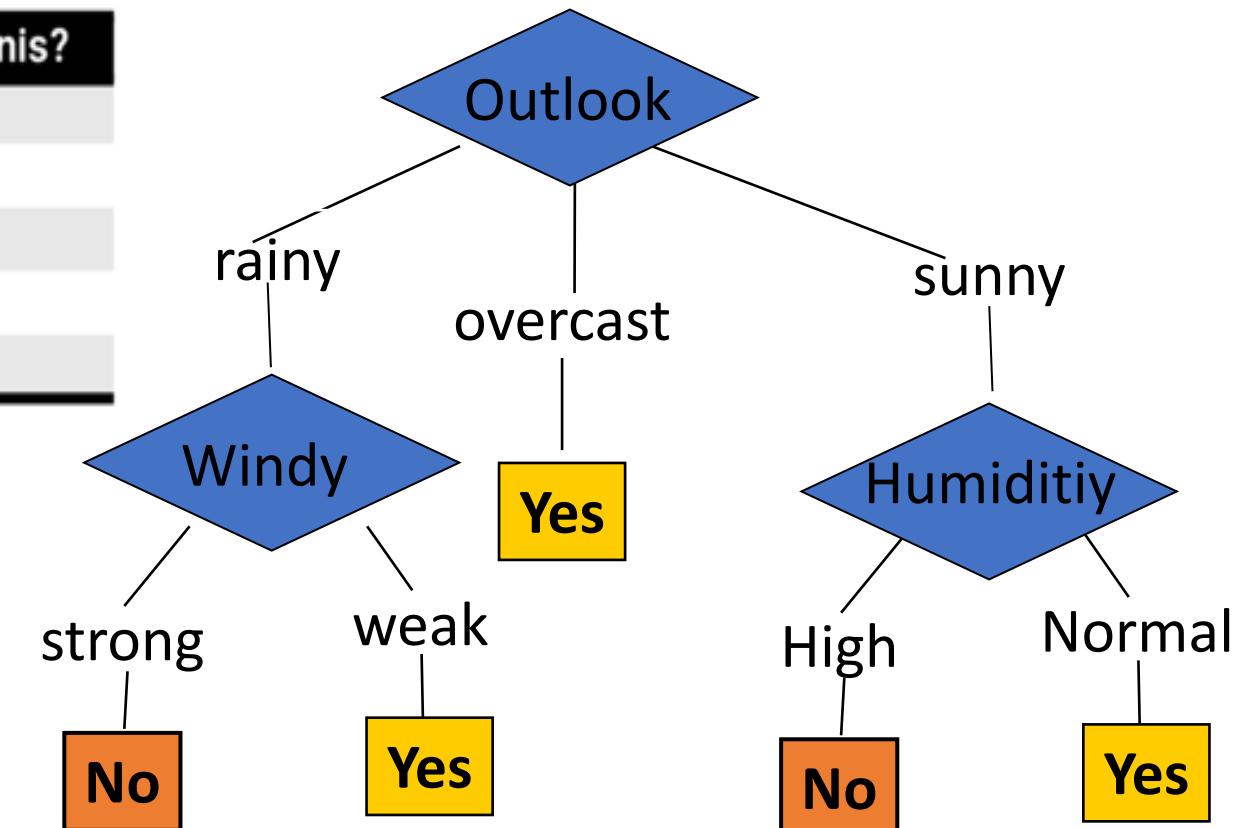


- Step 4b: A branch with entropy more than 0 needs further splitting.

→ Rainy

Outlook = Rainy

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No



- Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.



ID3 Algorithm (Python)

```
# ID = Iterative Dichotomiser
def ID3(X):
    node = TreeNode(X)
    if all_points_have_same_class(X):
        node.label = majority_label(X)
    else:
        a = select_attribute_with_highest_information_gain(X)
        if gain(X, a) == 0:
            node.label = majority_label(X)
        else:
            for v in values(a):
                Xv={x∈X | x[a] == v}
                node.children.append(ID3(Xv))
    return node
```





ID3 Decision Tree Learning - Explained

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



ID3 Decision Tree Learning - Explained

ID3(Examples, Target_attribute, Attributes)

- *Examples are the training examples.*
- *Target_attribute is the attribute whose value is to be predicted by the tree.*
- *Attributes is a list of other attributes that may be tested by the learned decision tree.*
- *Returns a decision tree that correctly classifies the given Examples.*





ID3 Decision Tree Learning - Explained

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label $\rightarrow +$
- If all *Examples* are negative, Return the single-node tree *Root*, with label $\rightarrow -$
- If *Attributes* is empty, Return the single-node *tree Root*, with label = most common value of *Target_attribute* in *Examples*





ID3 Decision Tree Learning - Explained

- Otherwise Begin
 - A \leftarrow the attribute from *Attributes* that best* classifies *Examples*
 - The decision attribute for *Root* \leftarrow A
 - For each possible value, v_i of A,
 - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
 - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
 - If $Examples_{v_i}$ is empty
 - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
 - Else
 - below this new branch add the subtree
 - $ID3(Examples_{v_i}, Target_attribute, Attributes - \{A\})$
- End
- Return *Root*



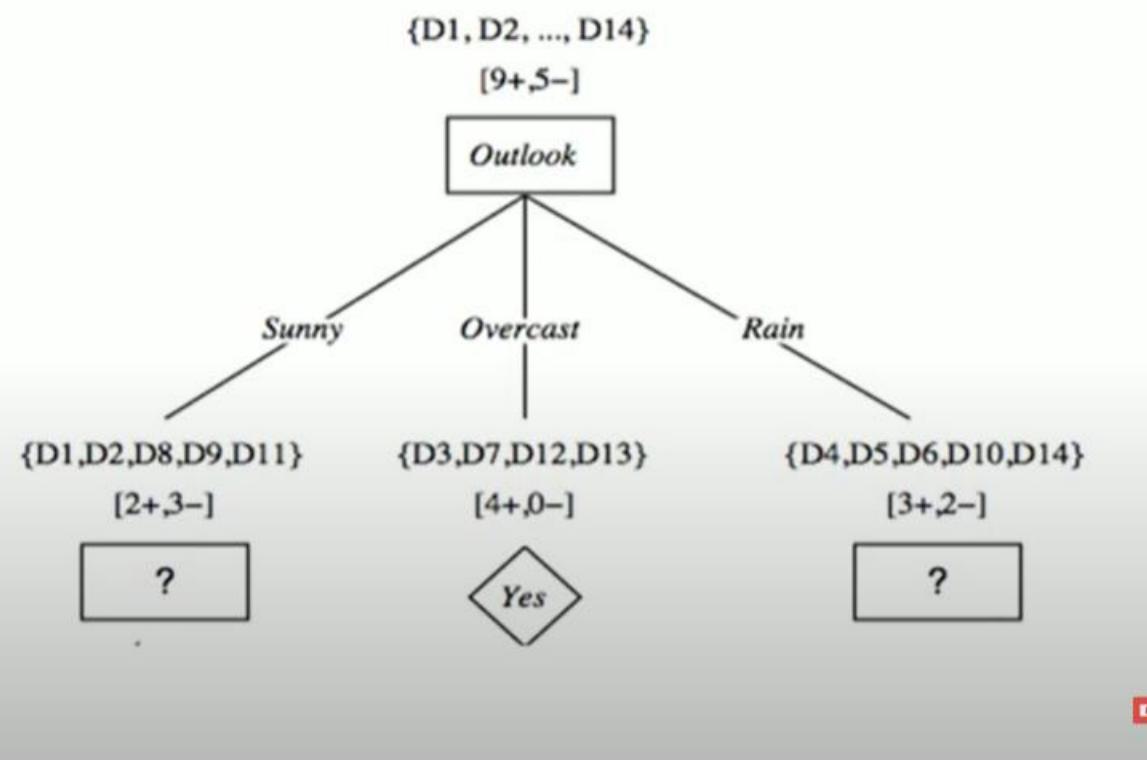


ID3 Decision Tree Learning - Explained

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Gain(S, Outlook) = 0.2464, \ Gain(S, Temp) = 0.0289$$

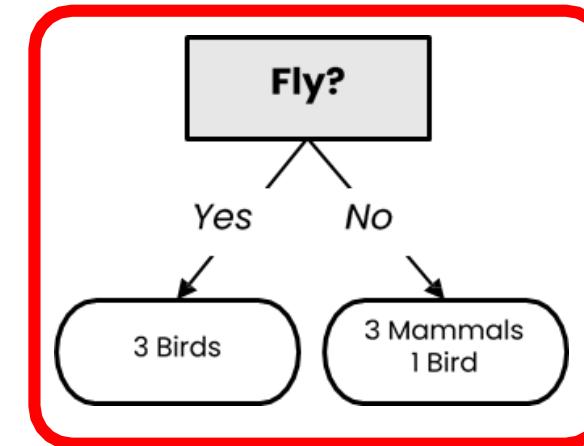
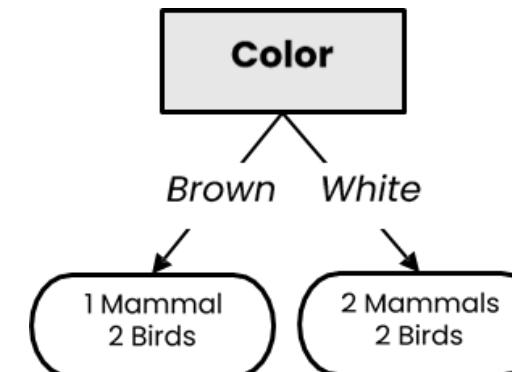
$$Gain(S, Humidity) = 0.1516, \ Gain(S, Wind) = 0.0478$$





Calculate the best attribute?

Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird



$$\text{entropy}(X) = -p_{\text{mammal}} \log_2 p_{\text{mammal}} - p_{\text{bird}} \log_2 p_{\text{bird}} = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx 0.985$$

$$\text{entropy}(X_{\text{color}=\text{brown}}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} \approx 0.918 \quad \text{entropy}(X_{\text{color}=\text{white}}) = 1$$

$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} \cdot 0.918 - \frac{4}{7} \cdot 1 \approx 0.020$$

$$\text{entropy}(X_{\text{fly}=\text{yes}}) = 0 \quad \text{entropy}(X_{\text{fly}=\text{no}}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \approx 0.811$$

$$\text{gain}(X, \text{fly}) = 0.985 - \frac{3}{7} \cdot 0 - \frac{4}{7} \cdot 0.811 \approx 0.521$$

In practice, we compute $\text{entropy}(X)$ only once!



Machine learning

- Course Code:
- Unit 4

Supervised Learning: Decision Trees and Ensemble Learning

- Lecture 3

Gini Index/Gini Impurity

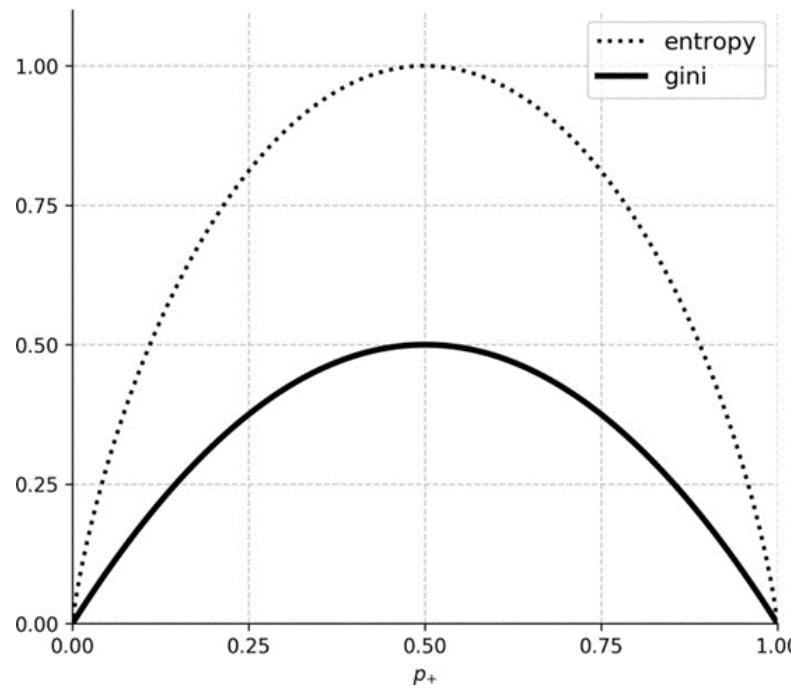




Gini Index/Gini Impurity



Error of classifying randomly picked fruit with randomly picked label

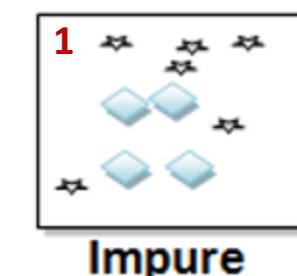
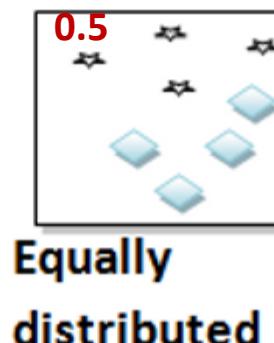
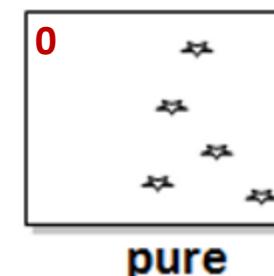


- Gini impurity measures how often a randomly chosen example would be incorrectly labeled if it was randomly labeled according to the label distribution
- Gini index as a cost function used to evaluate splits in the dataset. It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.



Gini Index/Gini Impurity

- It is calculated by subtracting the sum of the squared probabilities of each class from one.
- It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.
- Gini Index works with the categorical target variable “Success” or “Failure”. It performs only Binary splits.
- Calculate Gini index using: $\text{Gini Index} = 1 - \sum_j P_j^2$
Probability of each class
- If all the elements are linked with a single class then it is called pure. It ranges from 0-1



Note: An attribute with a lower Gini index should be preferred.



Problem: Identify the root node and calculate the gini index of each attribute?

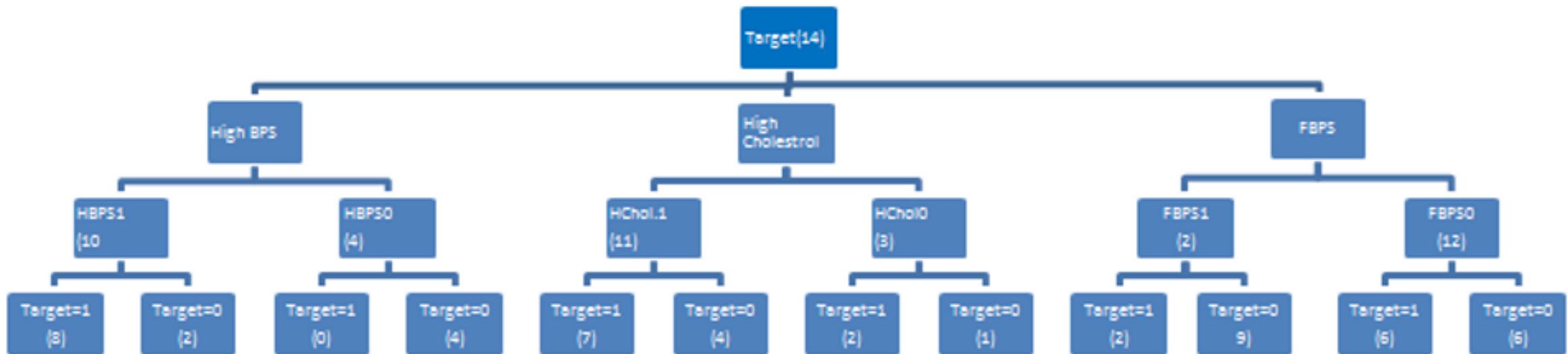
SNO	HighBps	HighChol	Fbs	target
1	1	1	1	1
2	1	1	0	1
3	1	1	0	1
4	0	1	0	0
5	0	1	0	0
6	1	0	0	0
7	1	1	0	1
8	0	1	0	0
9	1	0	1	1
10	1	0	0	0
11	1	1	0	1
12	1	1	0	1
13	1	1	0	1
14	0	1	0	0

The given table represents on which factor heart disease occurs in person(target) dependent variable depending on HighBP, Highcholesterol, FBS(fasting blood sugar).

Note: The original values are converted into 1 and 0 which depict numeric classification

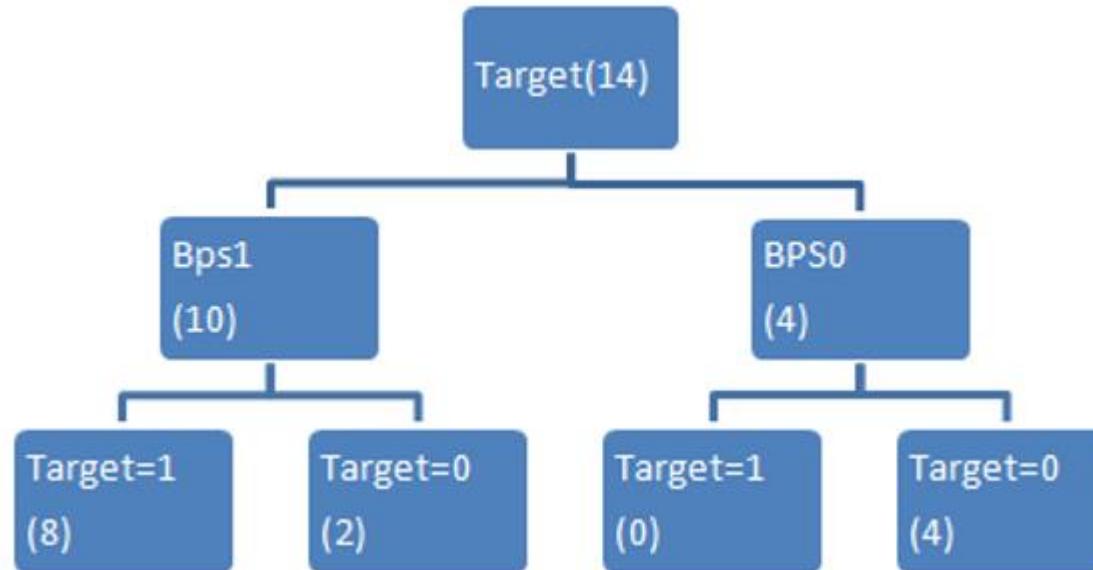


Decision tree for above table





- Gini index for High Bps



If (Bps=1 and target =1) = 8/10

if(Bps=1 and target=0)=2/10

$$\text{Gini index PBPS} = 1 - ((PBps1)^2 + (PBps0)^2)$$

$$= 1 - \{ (8/10)^2 + (2/10)^2 \}$$

$$= 0.32$$

Probability for parent node:

$$P1 = 10/14$$

$$P0 = 4/14$$

Now we calculate for child node :

i) For BPS=1

for bps =1 this is the table

HighBps	HighChol	Fbs	Target
1	1	1	1
1	1	0	1
1	1	0	1
1	0	0	0
1	1	0	1
1	0	1	1
1	0	0	0
1	1	0	1
1	1	0	1
1	1	0	1



2) if BPS=0,

HighBps	HighChol	Fbs	Target
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0

If (BPS=0 and target=0)=4/4=1

If (BPS=0 and target=1)=0

$$\text{Gini index PBPS0} = 1 - \{(1) - (0)\}$$

$$= 1 - 1$$

$$= 0$$

Weighted Gini index

$$w.g = P_0 * GBPS0 + P_1 * GBPS1$$

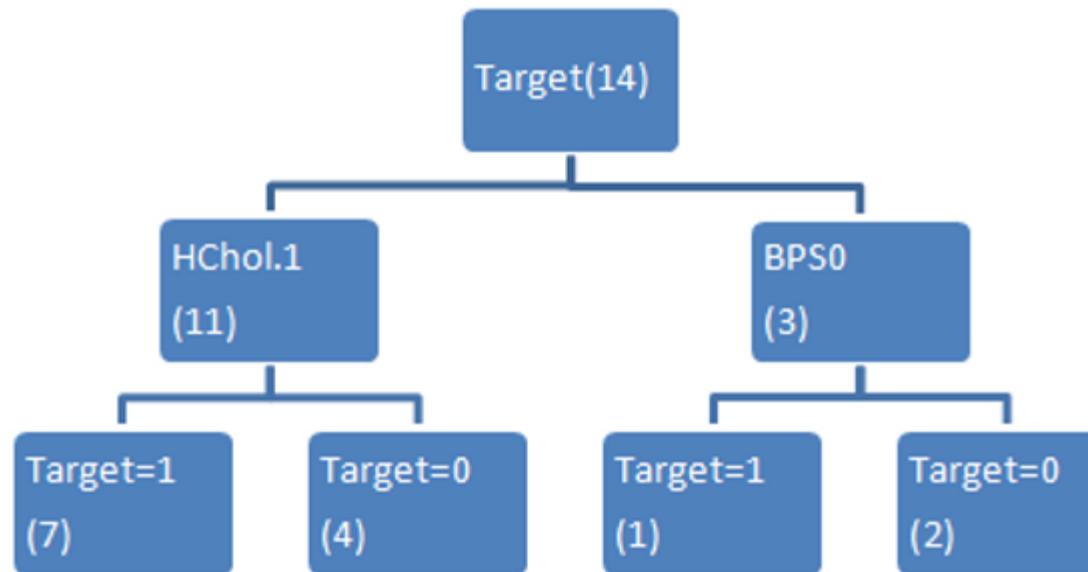
$$= 4/14 * 0 + 10/14 * 0.32$$

$$= 0.229$$





- Gini index for High Cholesterol:



If (Hchol.=1 and target=1)=7/11

If (HChol.=1 and target=0)=4/11

$$\text{Gini index} = 1 - [(7/11)^2 + (4/11)^2]$$

$$= 0.46$$

Probability of parent node

$$P1=11/14$$

$$P0=3/13$$

1) For HChol.=1

HighBps	HighChol	FBS	Target
1	1	1	1
1	1	0	1
1	1	0	1
0	1	0	0
0	1	0	0
1	1	0	1
0	1	0	0
1	1	0	1
1	1	0	1
1	1	0	1
0	1	0	0



2) If HChol.=0

HighBps	HighChol	FBS	Target
1	0	0	0
1	0	1	1
1	0	0	0

If (Hchol.=0 and target=1)=1/3

If (HChol.=0 and target=0)=2/3

$$\text{Gini index} = 1 - [(1/3)^2 + (2/3)^2]$$

$$= 0.55$$

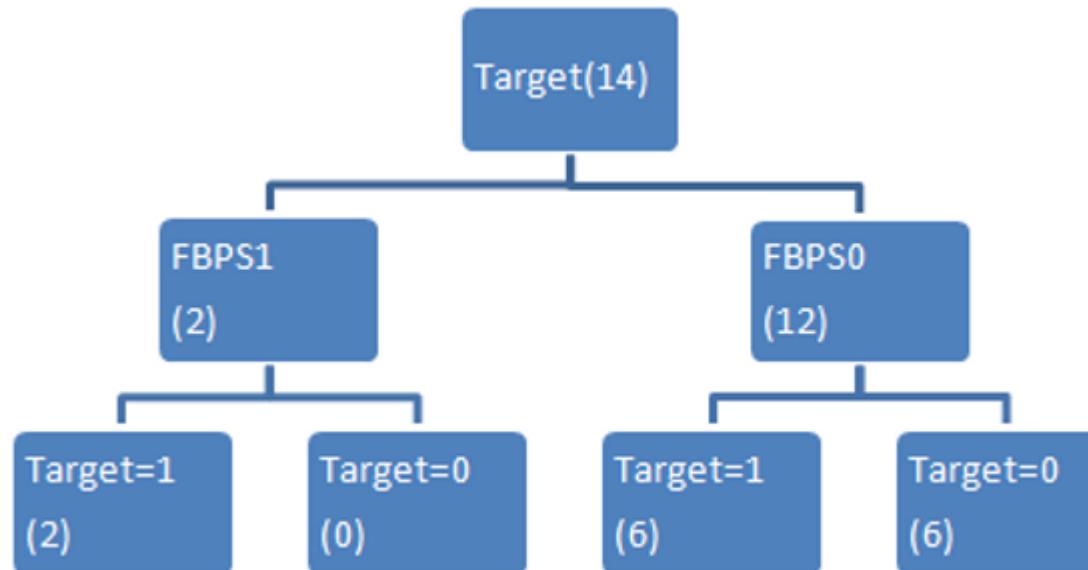
$$\begin{aligned}\text{Weighted Gini index} &= \\ P_0 * G_{HChol.0} + P_1 * G_{HChol.1} &= \end{aligned}$$

$$= 3/14 * 0.55 + 11/14 * 0.46$$

$$= 0.47$$



- Gini index for FBPS:



Probability of parent node

$$P1 = 2/14$$

$$P0 = 12/14$$

1) for FBPS=1

HighBps	HighChol	Fbs	Target
1	1	1	1
1	0	1	1

If (FBps=1 and target =1)=2/2

if(FBps=1 and target=0)=0

Gini index PFBPS1=1- $\{(PFBPS1)^2+(PFBPS0)^2\}$

$$= 1 - [(1)^2 + 0]$$

$$= 1 - 1 = 0$$



2) for FBPS=0,

HighBps	HighChol	Fbs	Target
1	1	0	1
1	1	0	1
0	1	0	0
0	1	0	0
1	0	0	0
1	1	0	1
0	1	0	0
1	0	0	0
1	1	0	1
1	1	0	1
1	1	0	1
0	1	0	0

If (FBps=0 and target =1)=6/12=0.5
if(FBps=0 and target=0)=6/12=0.5

$$\text{Gini index PFBPS0}=1-\{(PFBPS1)^2+(PFBPS0)^2\}$$
$$= 1-[(0.5)^2+(0.5)^2]$$

$$= 0.5$$

$$\text{Weighted Gini index} = P_0 \cdot G_{FBPS0} + P_1 \cdot G_{FBPS1}$$
$$= 6/7 \cdot 0.5 + 1/7 \cdot 0$$

$$= 0.42$$



Comparing Gini Index:

Attribute	HighBPS	Hchol.	FBPS
Gini index	0.22	0.47	0.42

As *HighBPS* is less it is the winner

Conclusion: *HighBPS* is used as the *root node* for constructing of Decision Tree and the further tree is built.





Bagging

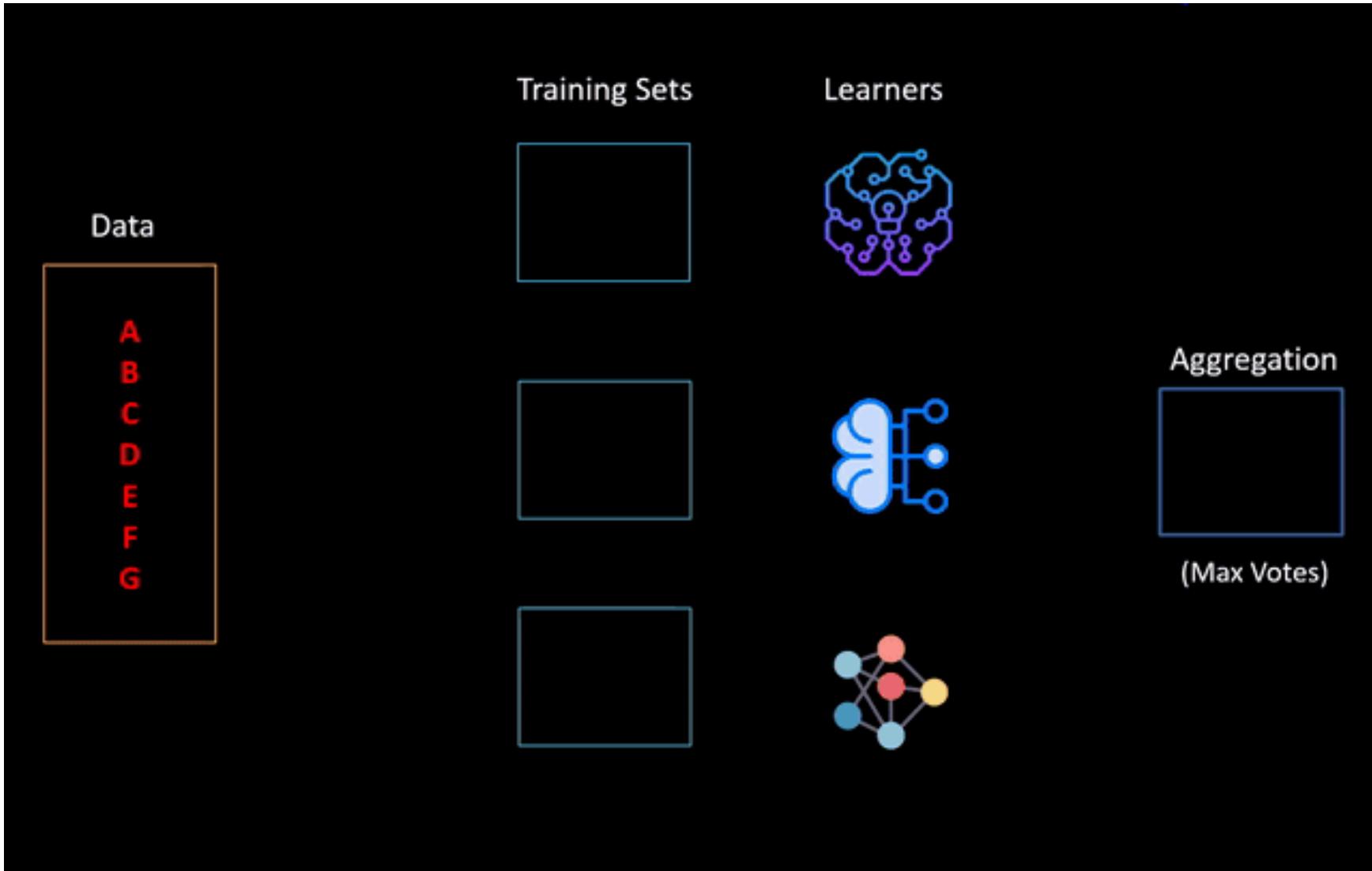
Bagging is primarily used to solve supervised machine learning problems. It helps to reduce variance and hence avoids the possibility of overfitting. It is generally completed in two steps as follows:

- **Bootstrapping:** It is a random sampling method that is used to derive samples from the data using the replacement procedure. In this method, first, random data samples are fed to the primary model, and then a base learning algorithm is run on the samples to complete the learning process.
- **Aggregation:** This is a step that involves the process of combining the output of all base models and, based on their output, predicting an aggregate result with greater accuracy and reduced variance.





Bagging





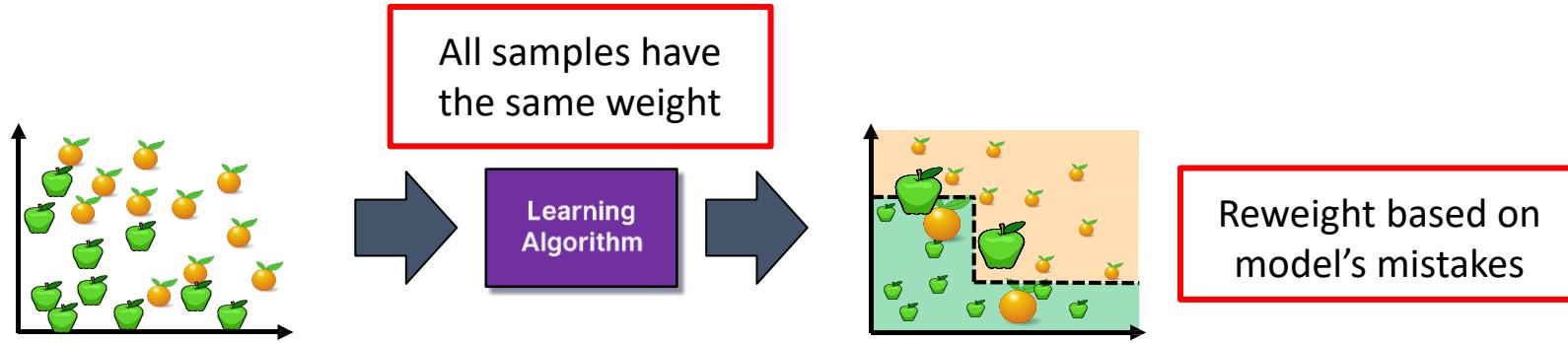
Boosting

- Boosting is an ensemble method that enables each member to learn from the preceding member's mistakes and make better predictions for the future.
- Unlike the bagging method, in boosting, all base learners (weak) are arranged in a sequential format so that they can learn from the mistakes of their preceding learner.
- Hence, in this way, all weak learners get turned into strong learners and make a better predictive model with significantly improved performance..



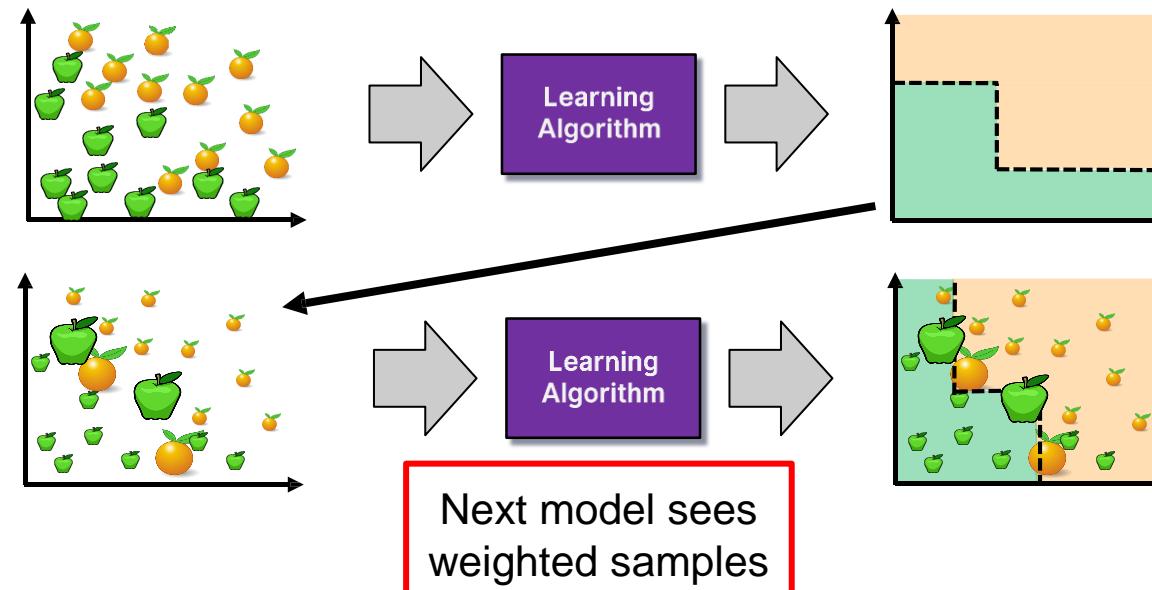


Boosting



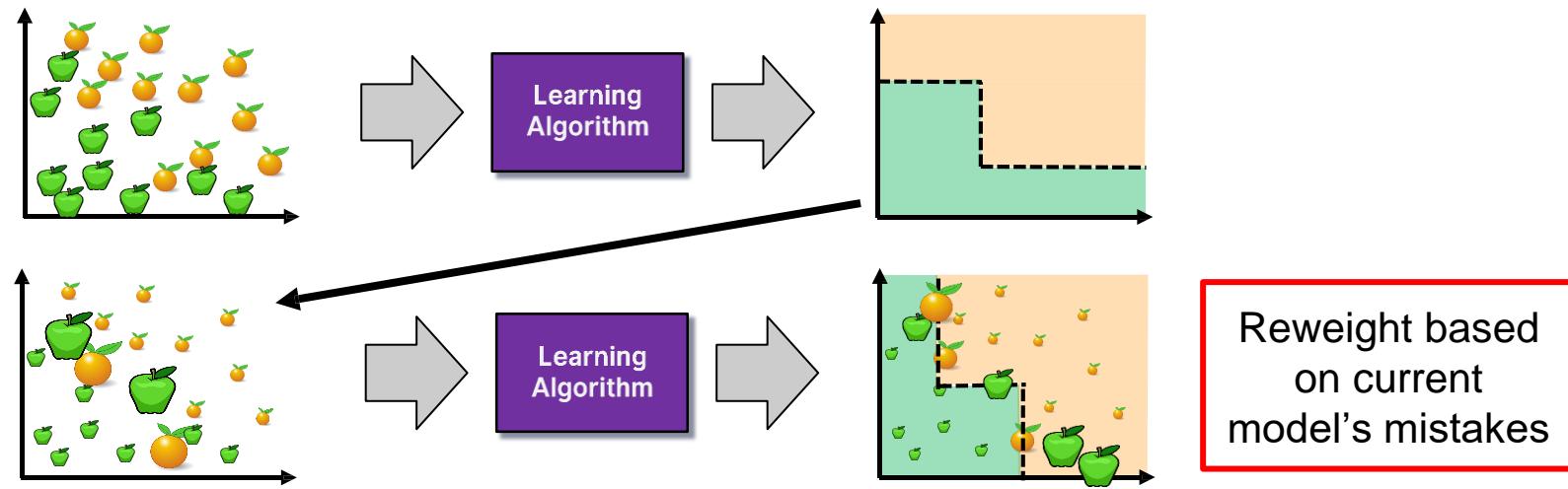


Boosting



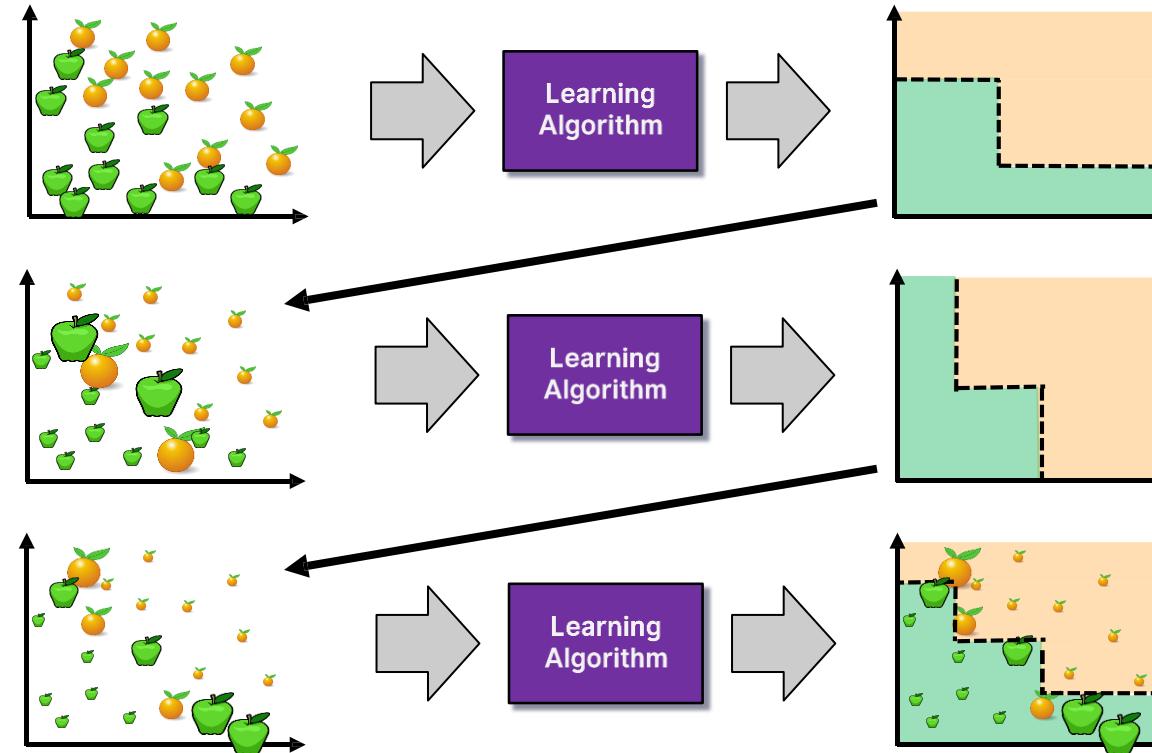


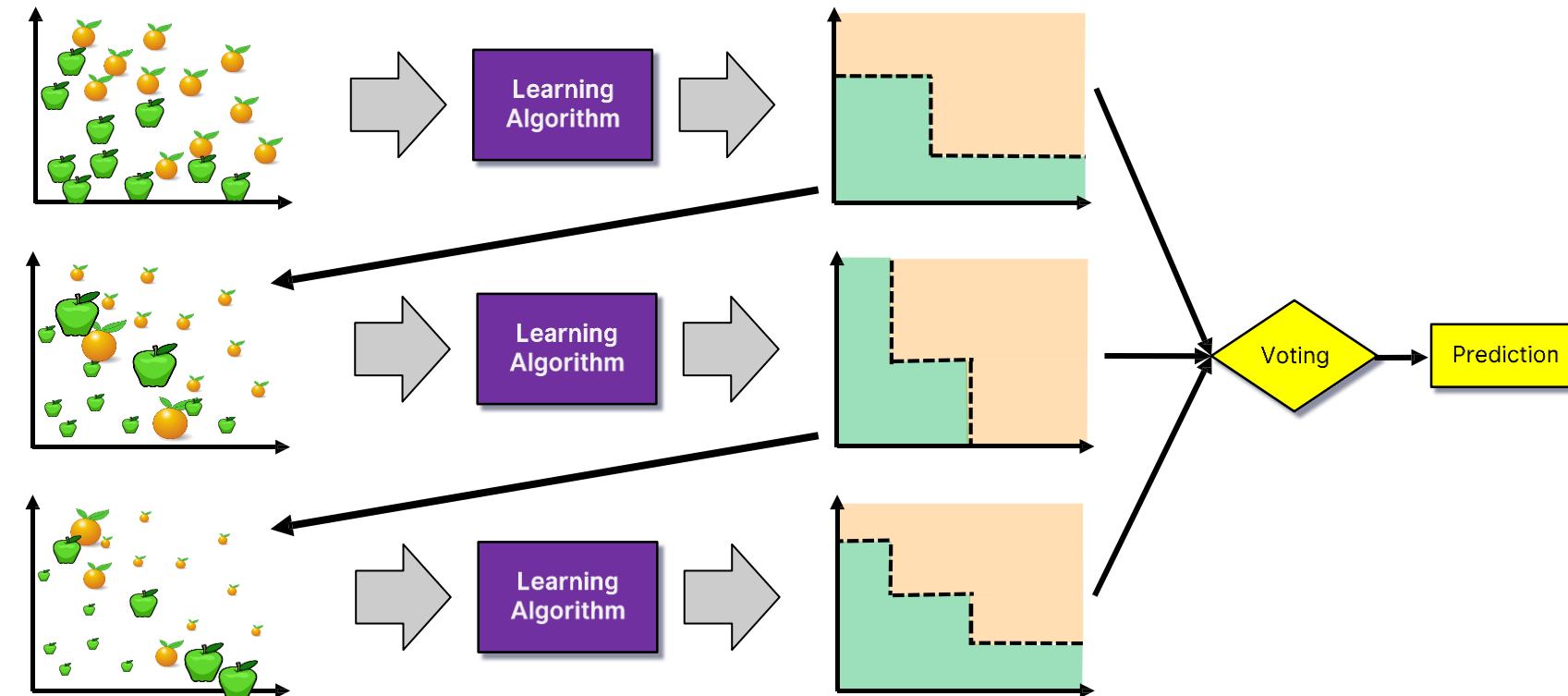
Boosting





Boosting







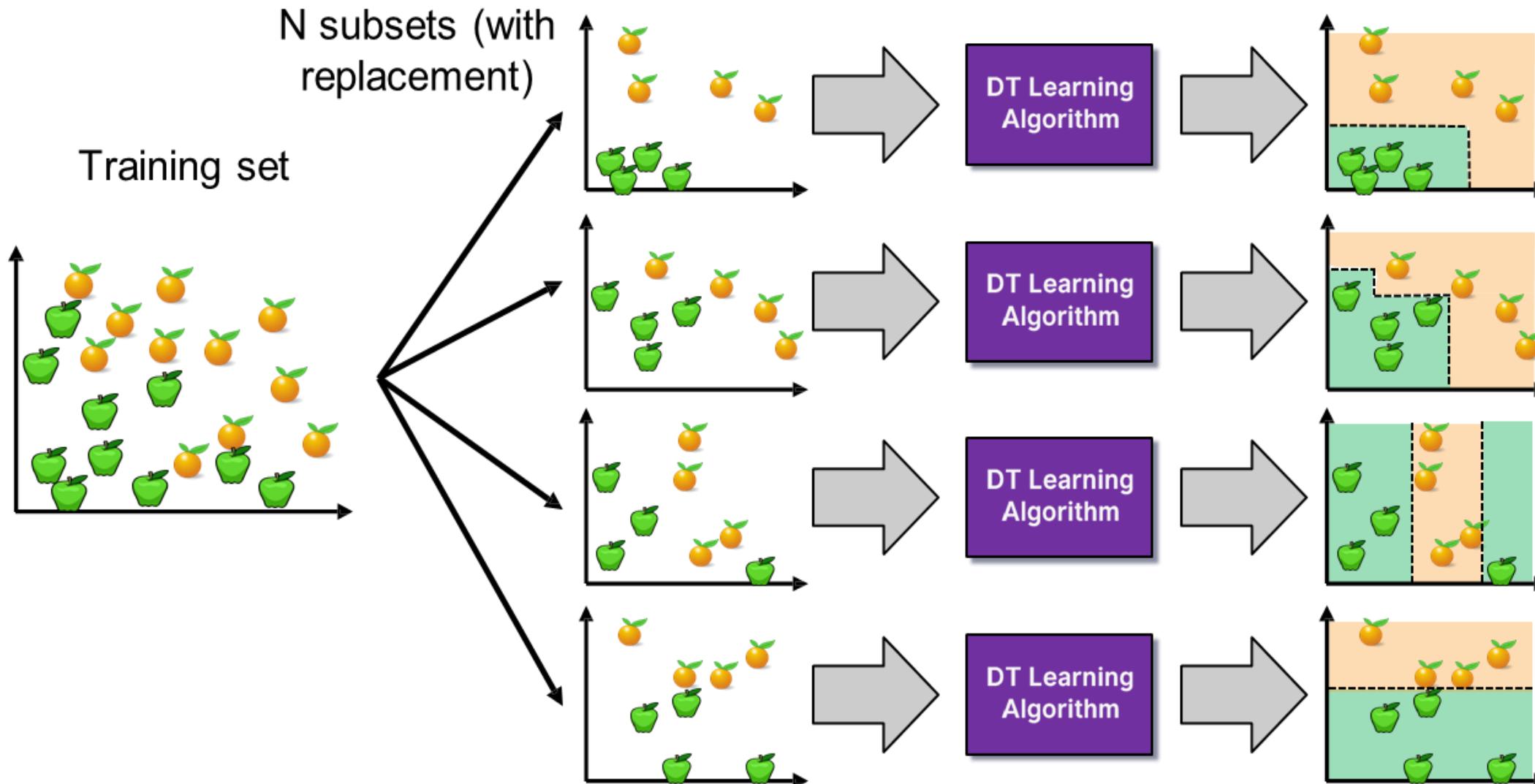
Random Forest

- Random Forests:
 - Instead of building a single decision tree and use it to make predictions, build many slightly different trees and combine their predictions
- We have a single data set, so how do we obtain slightly different trees?
 1. Bagging (**Bootstrap Aggregating**):
 - Take random subsets of data points from the training set to create N smaller data sets
 - Fit a decision tree on each subset
 2. Random Subspace Method (also known as Feature Bagging):
 - Fit N different decision trees by constraining each one to operate on a random subset of features



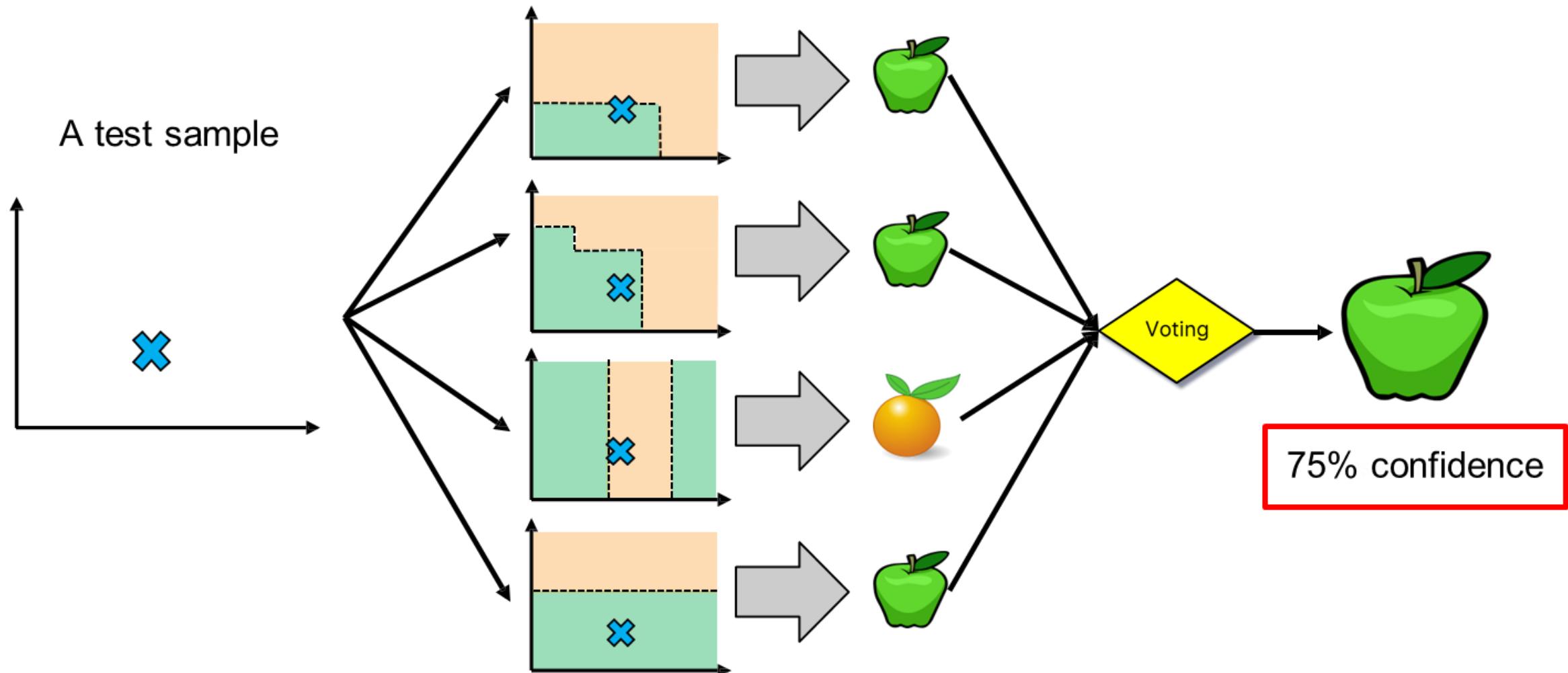


Bagging at training time





Bagging at inference time

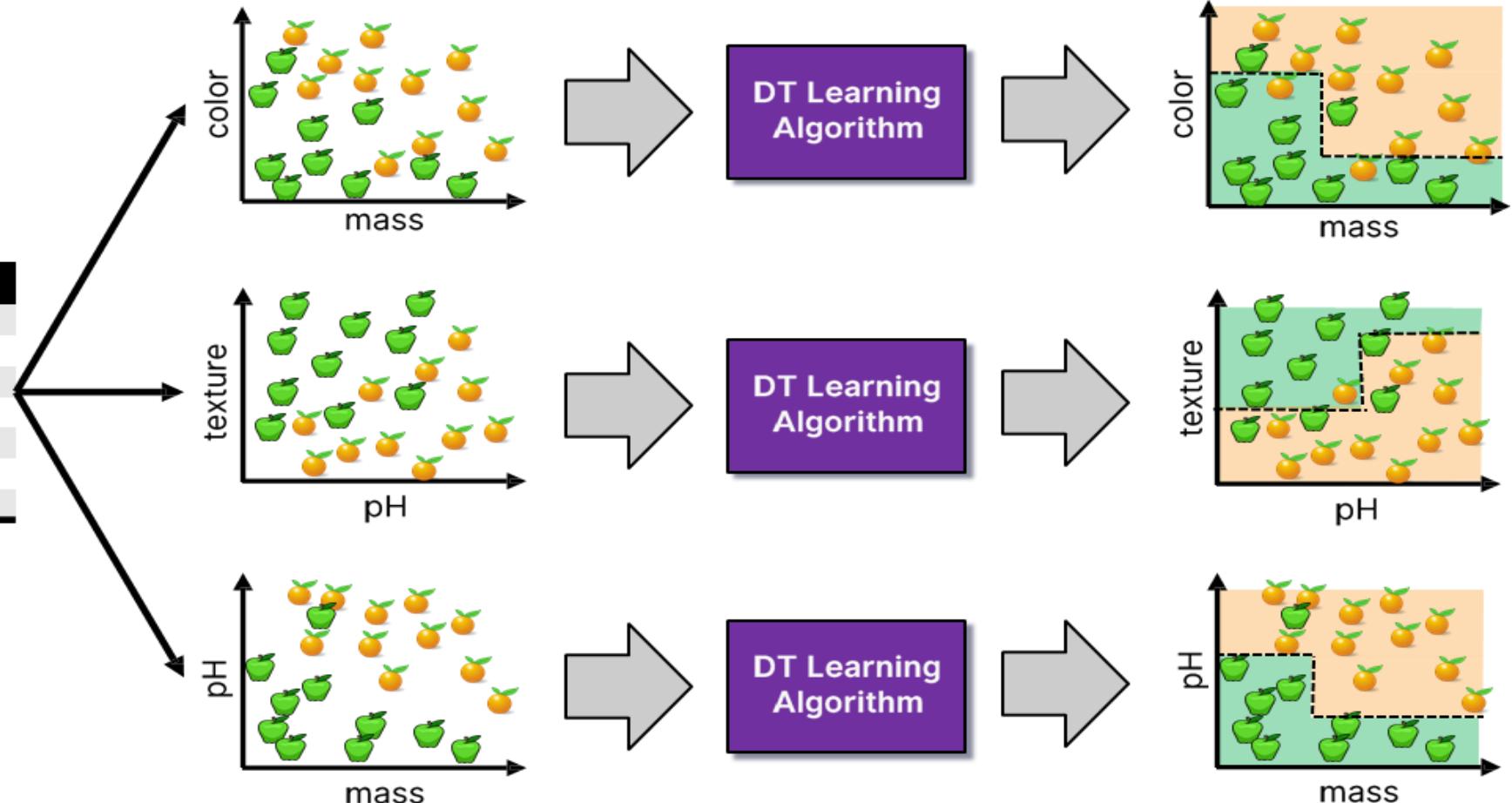




Random Subspace Method at training time

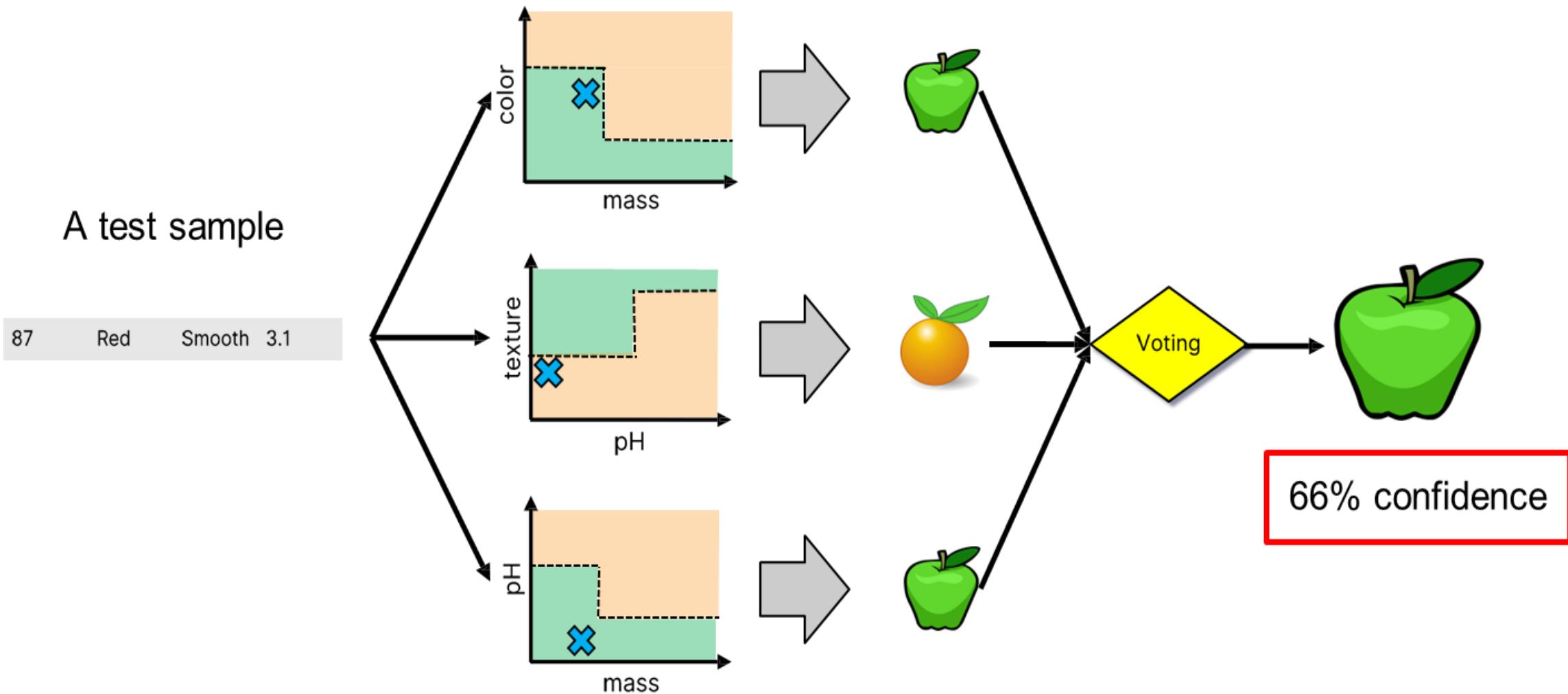
Training data

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	Apple
121	Orange	Rough	3.9	Orange
85	Red	Smooth	3.3	Apple
101	Orange	Smooth	3.7	Orange
111	Green	Rough	3.5	Apple
...				
117	Red	Rough	3.4	Orange





Random Subspace Method at inference time



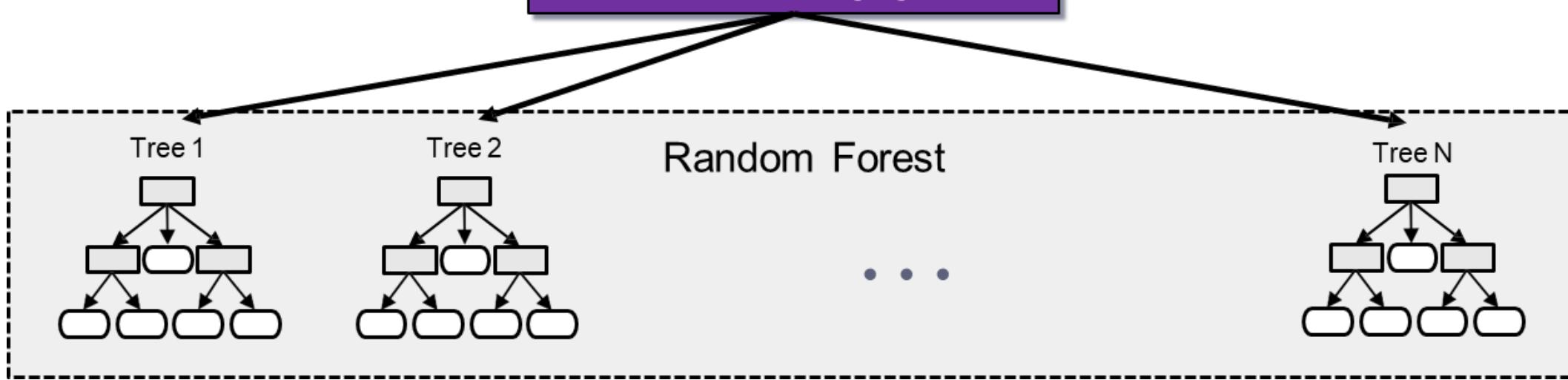


Random Forest

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	Apple
121	Orange	Rough	3.9	Orange
85	Red	Smooth	3.3	Apple
101	Orange	Smooth	3.7	Orange
111	Green	Rough	3.5	Apple
...				
117	Red	Rough	3.4	Orange



Bagging +
Random Subspace Method +
Decision Tree Learning Algorithm





Summary

Bagging	Boosting
Various training data subsets are randomly drawn with replacement from the whole training dataset.	Each new subset contains the components that were misclassified by previous models.
Bagging attempts to tackle the over-fitting issue.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then we need to apply bagging.	If the classifier is steady and straightforward (high bias), then we need to apply boosting.
Every model receives an equal weight.	Models are weighted by their performance.
Objective to decrease variance, not bias.	Objective to decrease bias, not variance.
It is the easiest way of connecting predictions that belong to the same type.	It is a way of connecting predictions that belong to the different types.
Every model is constructed independently.	New models are affected by the performance of the previously developed model.



Temporally aware distance metrics

- Temporally aware distance metrics are distance metrics that take into account the temporal relationship between the data points being compared. This is in contrast to traditional distance metrics, which do not take into account the temporal relationship between the data points.
- There are a number of different temporally aware distance metrics that have been proposed. Some of the most common ones include:
- **Dynamic Time Warping (DTW):** DTW is a dynamic programming algorithm that calculates the distance between two sequences by warping the time axis. This allows DTW to take into account the fact that the two sequences may not be aligned perfectly.
- **Elastic Distance:** Elastic Distance is a generalization of DTW that allows for more flexibility in the warping of the time axis.
- **Time Warped Euclidean Distance (TWED):** TWED is a distance metric that is based on DTW, but it uses the Euclidean distance instead of the dynamic programming algorithm.
- **Shapelet Distance:** Shapelet Distance is a distance metric that uses shapelets to represent the data points. Shapelets are small subsequences of the data points that are representative of the overall shape of the data points.

Temporally aware distance metrics can be used for a variety of tasks, such as:

- **Time series classification:** Temporally aware distance metrics can be used to classify time series data. For example, you could use DTW to classify time series data into different types of events, such as heart beats or speech patterns.
- **Time series clustering:** Temporally aware distance metrics can be used to cluster time series data. For example, you could use DTW to cluster time series data into different groups of similar time series.
- **Time series similarity search:** Temporally aware distance metrics can be used to search for similar time series. For example, you could use DTW to find all time series that are similar to a given time series.



Distance time wrapping
(DTW)

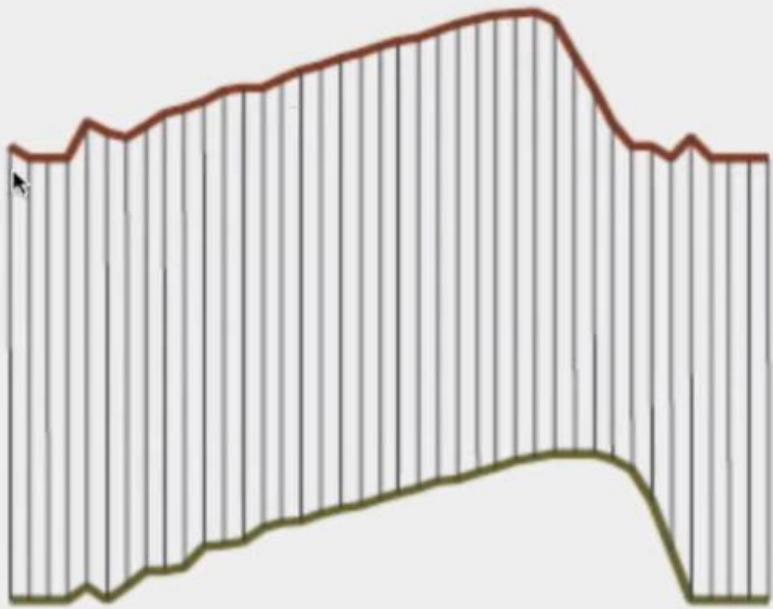
How DTW works

DTW algorithm

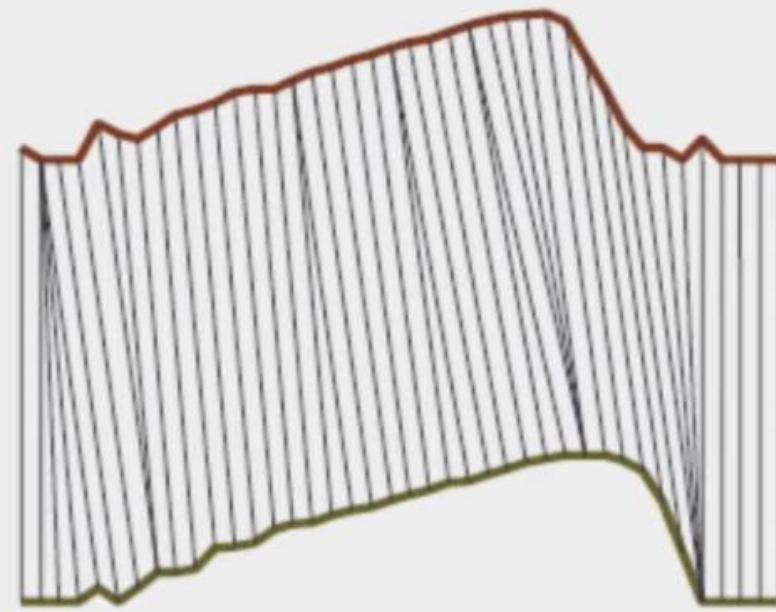
- Inputs: $x_{1:N}$ and $y_{1:M}$
- Cost matrix: $\mathbf{D} \in \mathbb{R}^{(N+1) \times M+1}$
- Initialization:
 - for $i = 1$ to N :
 - for $j = 1$ to M :
- Calculate cost matrix:
 - for $i = 1$ to N :
 - for $j = 1$ to M :
$$D_{i,j} = d(x_i, y_j) + \min \begin{cases} D_{i-1,j-1} & (\text{match}) \\ D_{i-1,j} & (\text{insertion}) \\ D_{i,j-1} & (\text{deletion}) \end{cases}$$
- Get alignment: Trace back from $D_{N,M}$ to $D_{0,0}$

DTW

- One of the most used measure of the similarity between two time series
- Originally designed to treat automatic speech recognition
- Optimal global alignment between two time series, exploiting temporal distortions between them

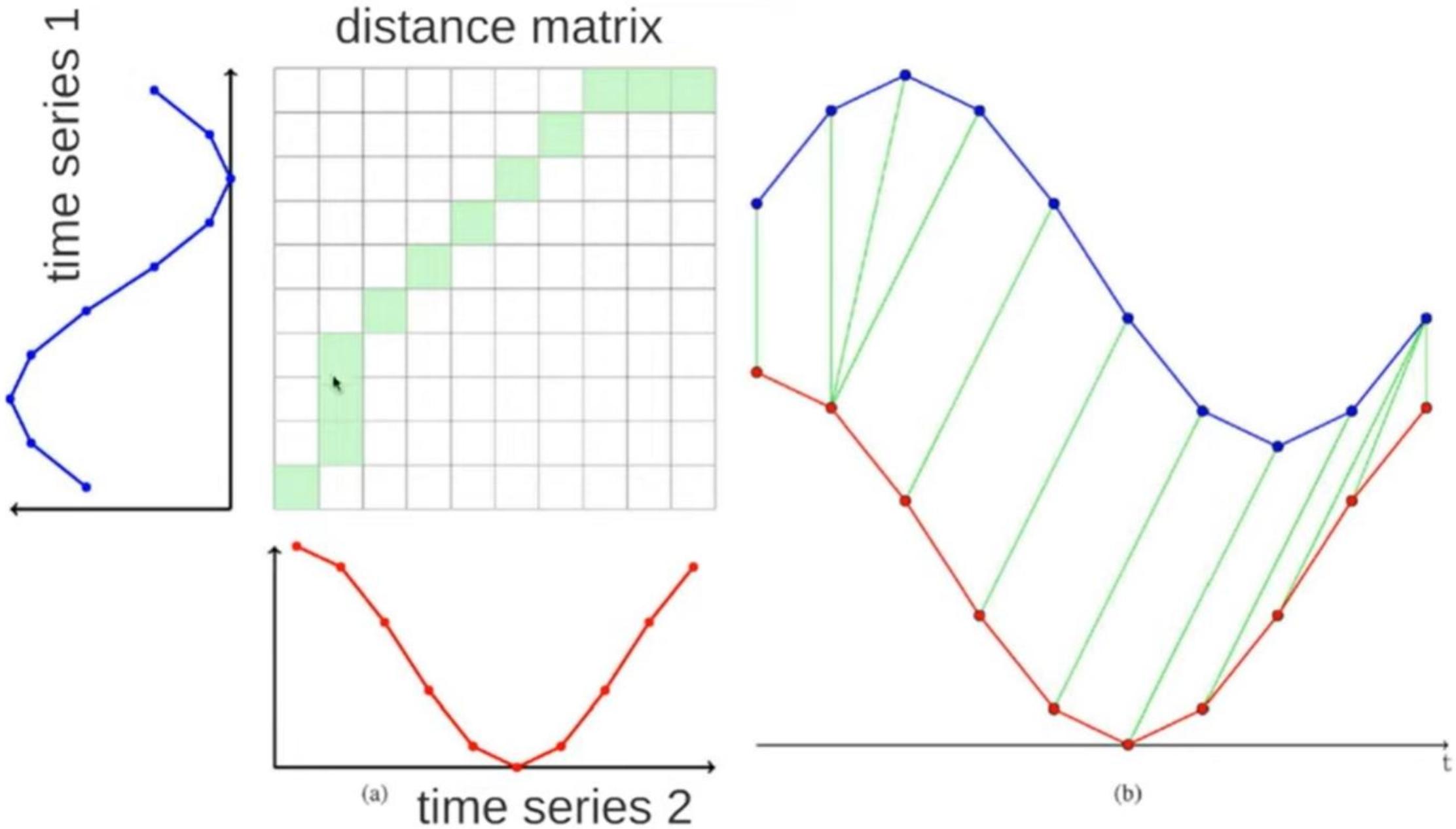


x

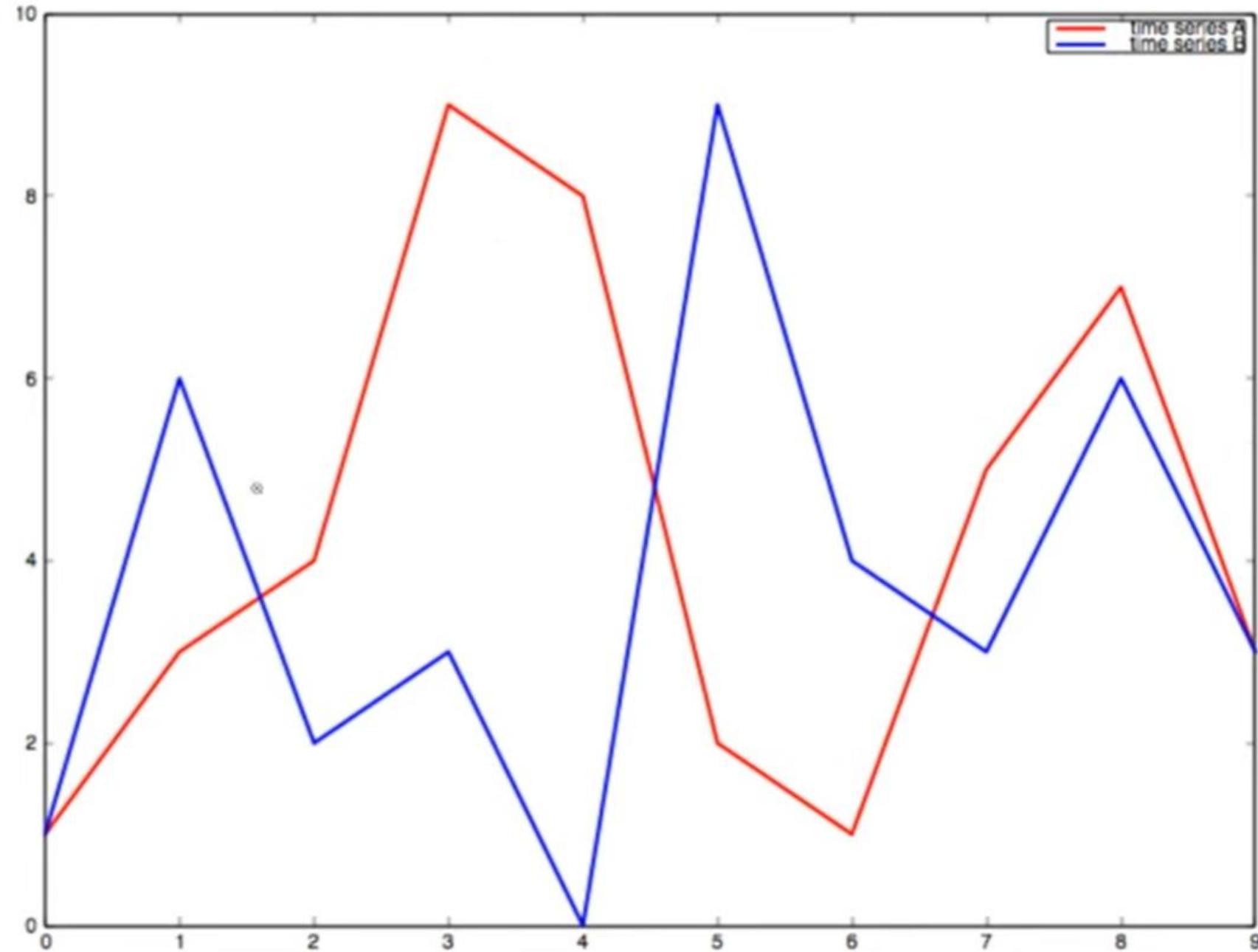


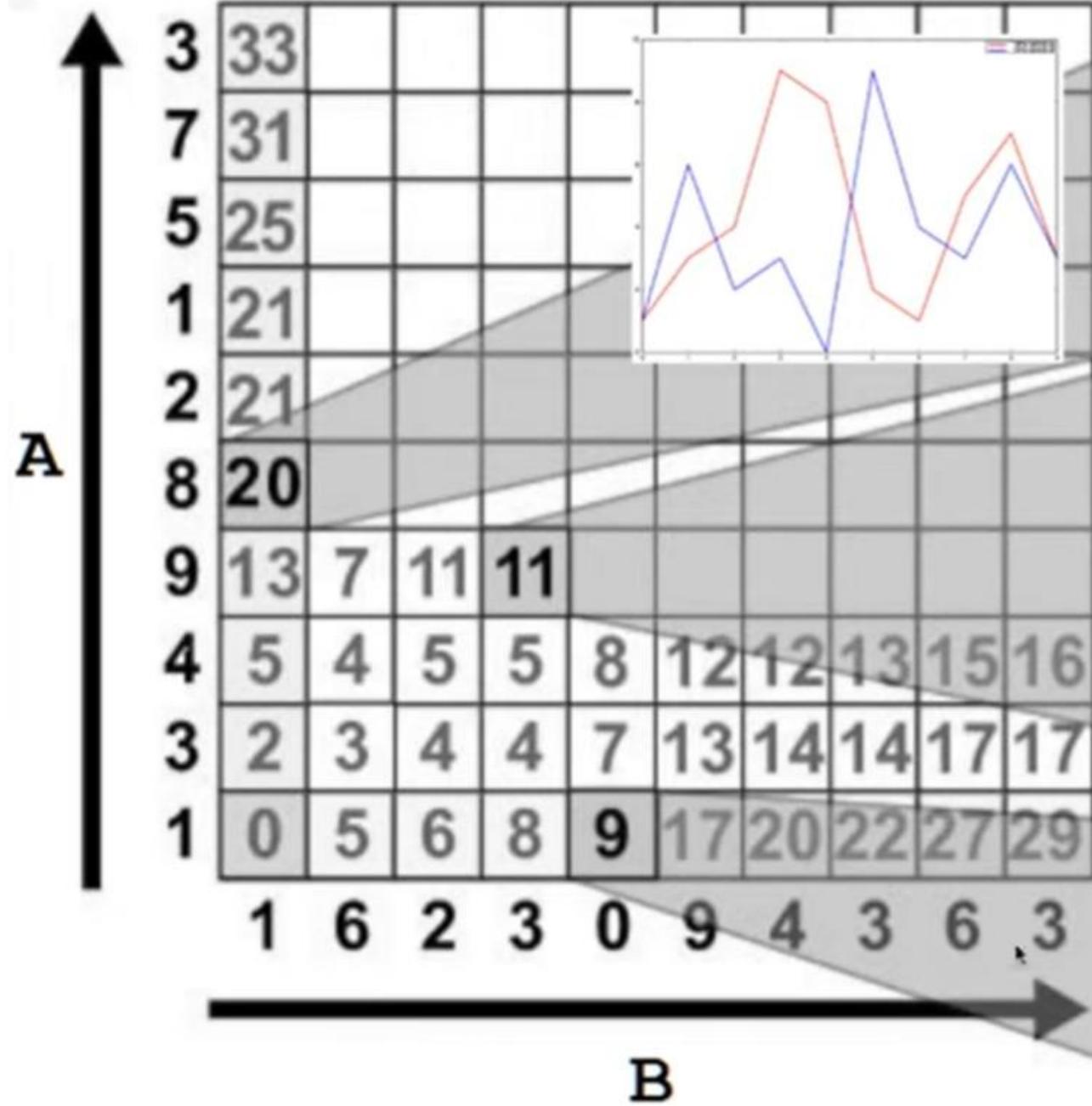
Euclidean
distance

DTW



example
DTW between
time series A
and
time series B

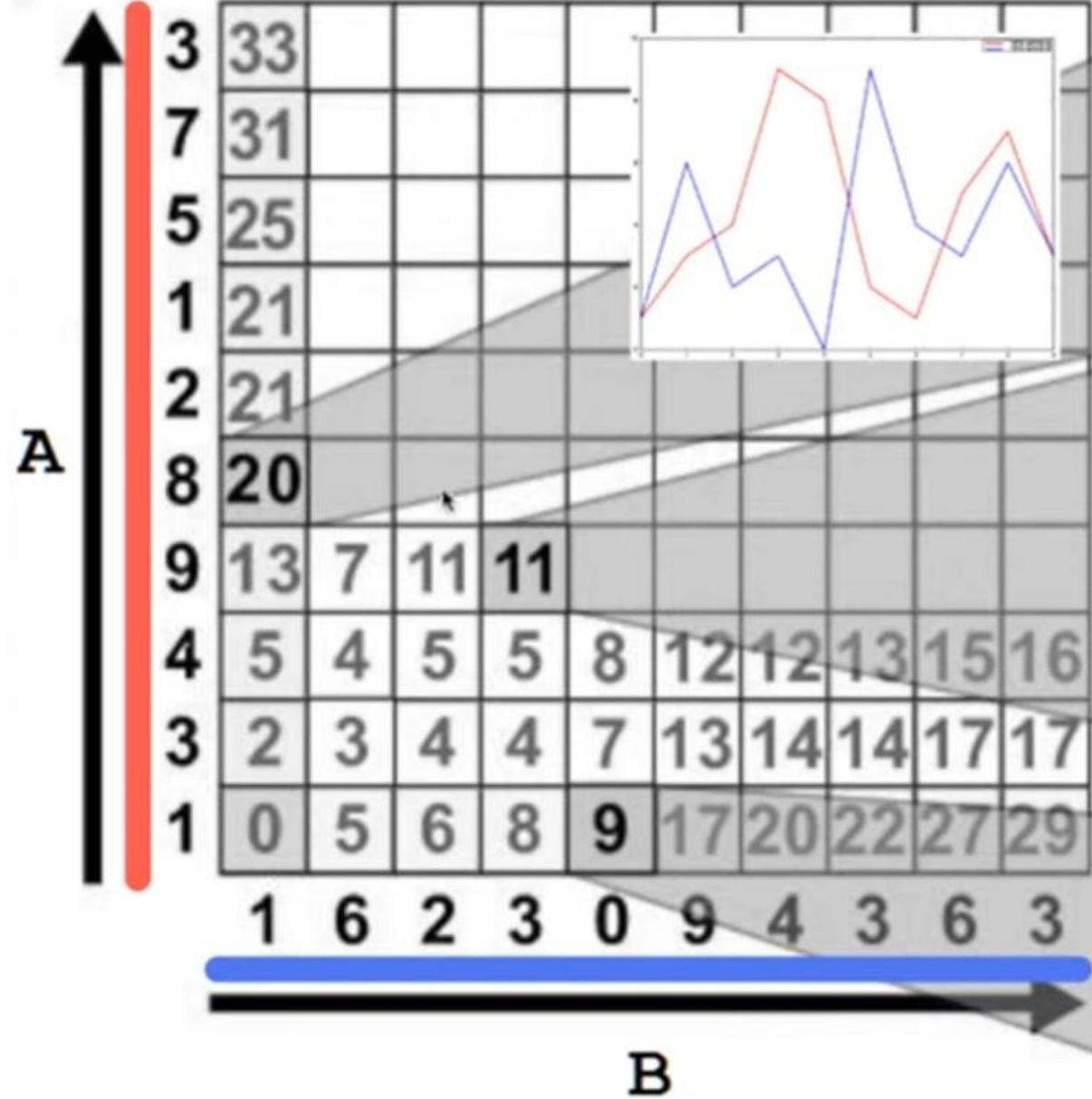




$$\begin{aligned}
 &= |A_i - B_j| + D[i-1, 0] \\
 &= |8 - 1| + 13 \\
 &= 20
 \end{aligned}$$

$$\begin{aligned}
 &= |A_i - B_j| + \min(D[i-1, j-1], \\
 &\quad D[i-1, j], \\
 &\quad D[i, j-1]) \\
 &= |9 - 3| + \min(5, 5, 11) \\
 &= 6 + 5 \\
 &= 11
 \end{aligned}$$

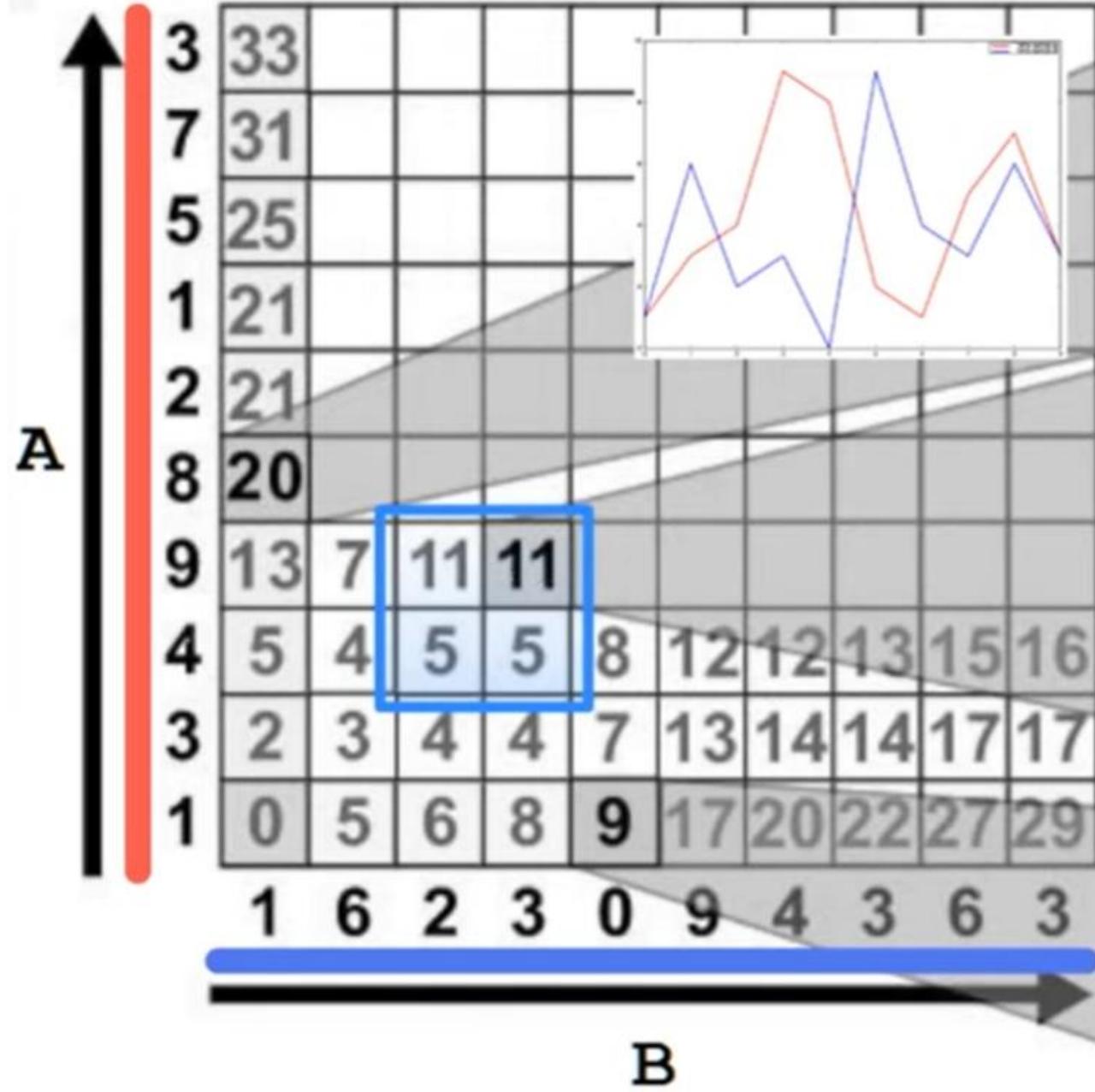
$$\begin{aligned}
 &= |A_i - B_j| + D[0, j-1] \\
 &= |1 - 0| + 8 \\
 &= 9
 \end{aligned}$$



$$\begin{aligned}
 &= |A_i - B_j| + D[i-1, 0] \\
 &= |8 - 1| + 13 \\
 &= 20
 \end{aligned}$$

$$\begin{aligned}
 &= |A_i - B_j| + \min(D[i-1, j-1], \\
 &\quad \text{red bar} \quad \text{blue bar} \\
 &\quad D[i-1, j], \\
 &\quad D[i, j-1]) \\
 &= |9 - 3| + \min(5, 5, 11) \\
 &= 6 + 5 \\
 &= 11
 \end{aligned}$$

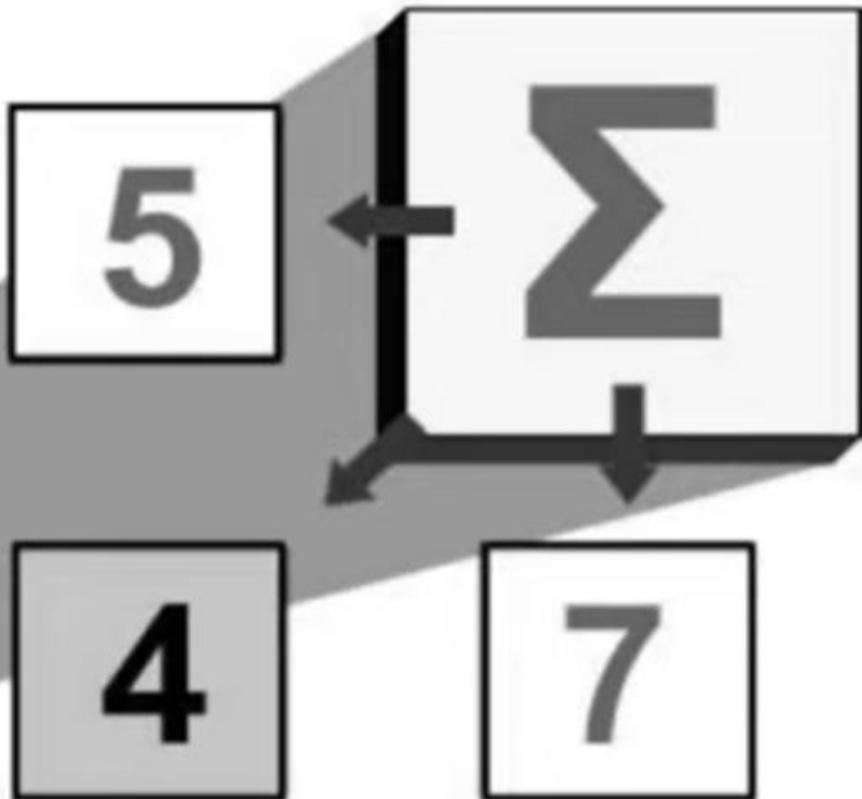
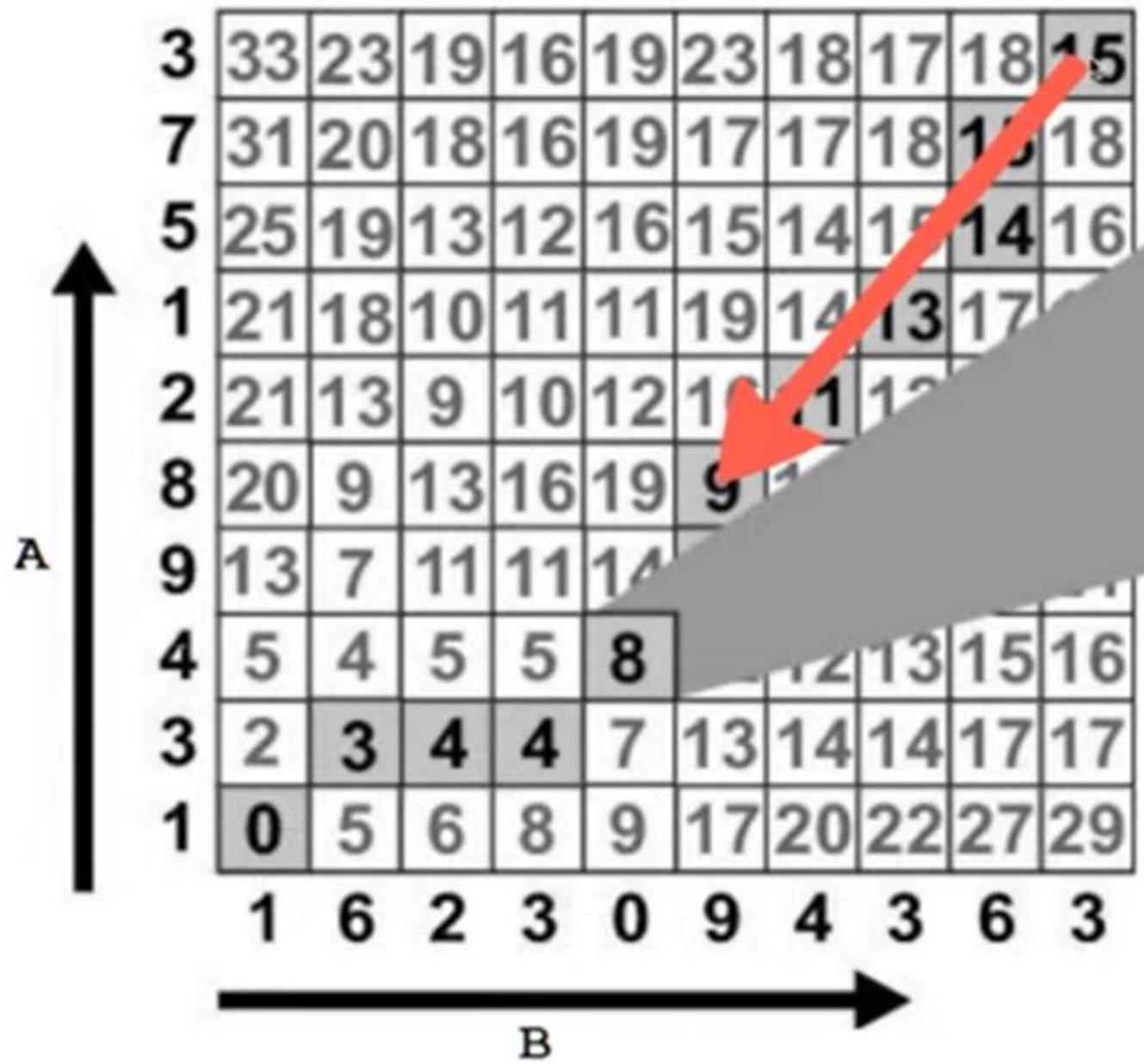
$$\begin{aligned}
 &= |A_i - B_j| + D[0, j-1] \\
 &= |1 - 0| + 8 \\
 &= 9
 \end{aligned}$$



$$\begin{aligned}
 &= |A_i - B_j| + D[i-1, 0] \\
 &= |8 - 1| + 13 \\
 &= 20
 \end{aligned}$$

$$\begin{aligned}
 &= |A_i - B_j| + \min(D[i-1, j-1], \\
 &\quad \text{red bar} \quad \text{blue bar} \\
 &\quad D[i-1, j], \\
 &\quad D[i, j-1]) \\
 &= |9 - 3| + \min(5, 5, 11) \\
 &= 6 + 5 \\
 &= 11
 \end{aligned}$$

$$\begin{aligned}
 &= |A_i - B_j| + D[0, j-1] \\
 &= |1 - 0| + 8 \\
 &= 9
 \end{aligned}$$



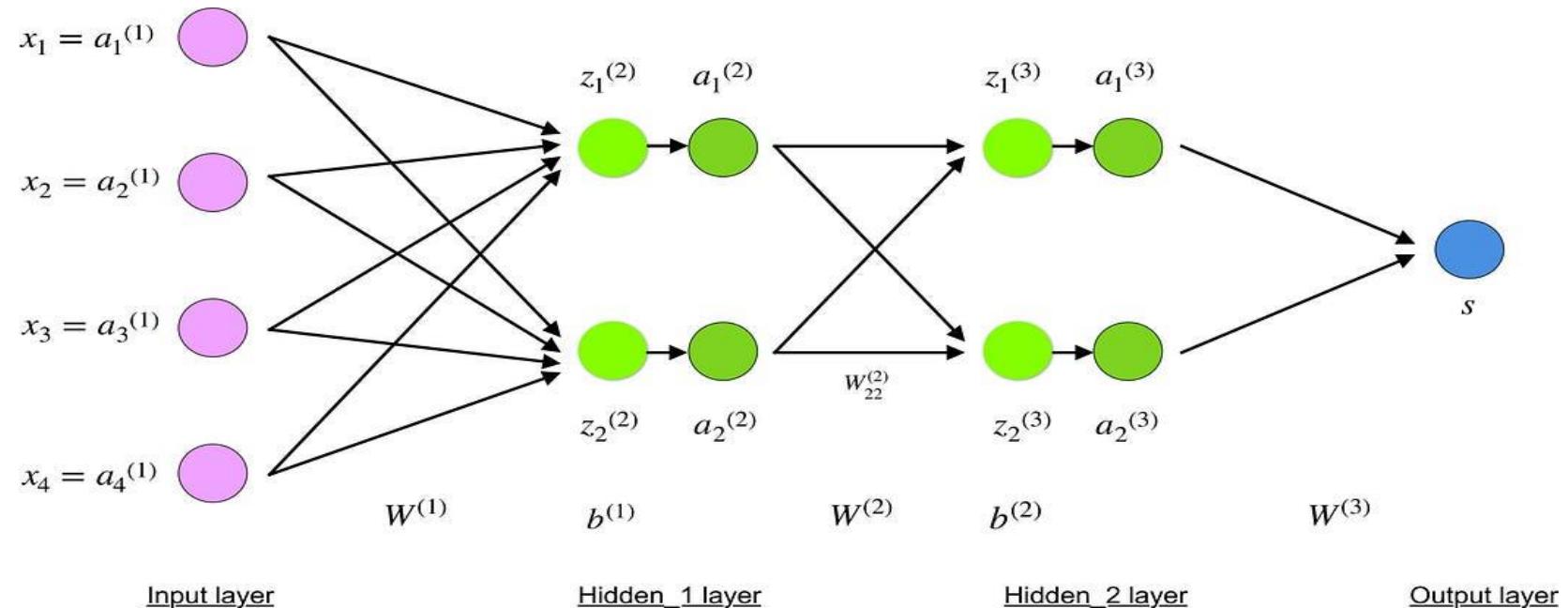


Training Neural Networks



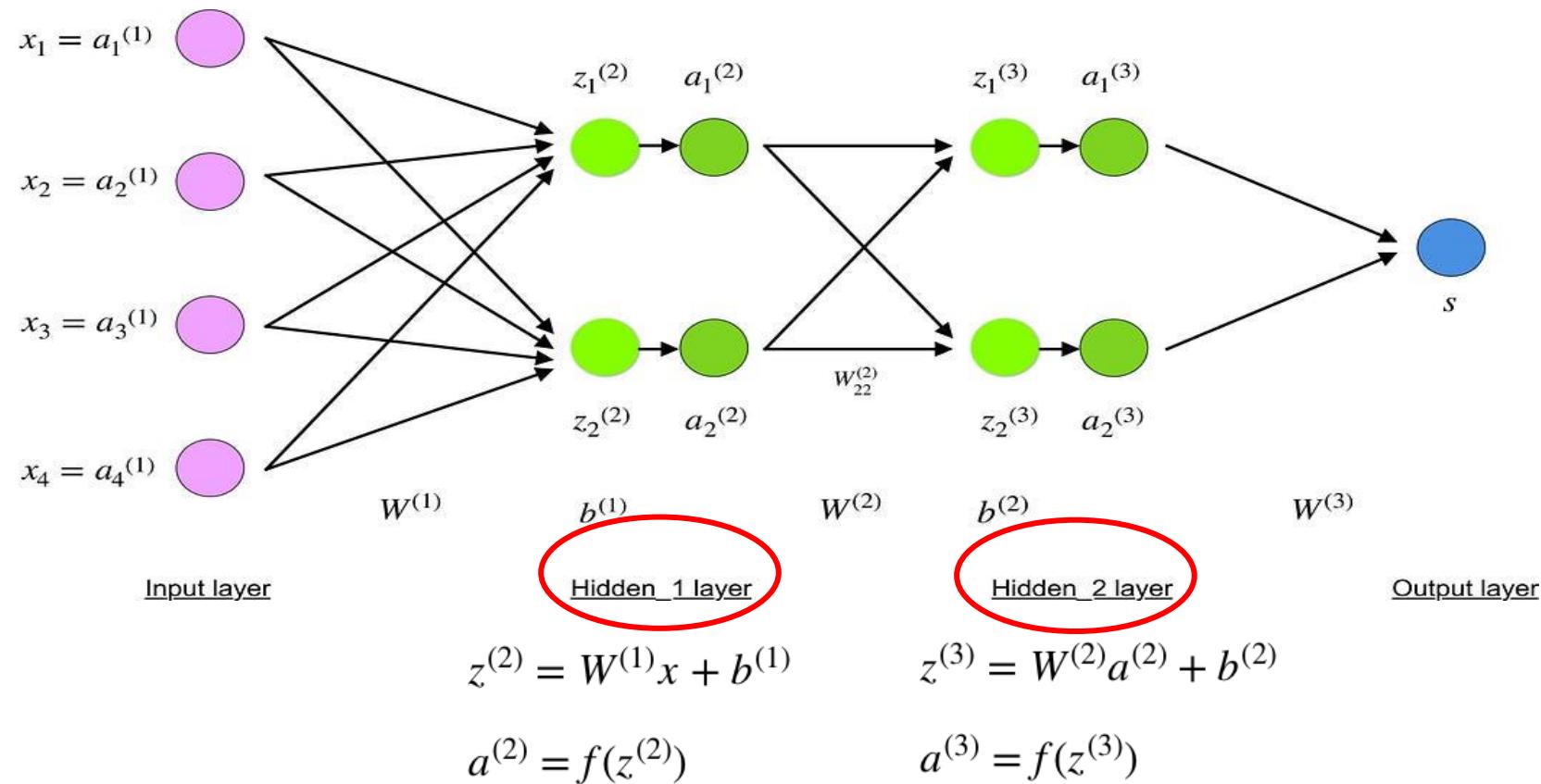


Eg. 4-layer neural network





Eg. 4-layer neural network

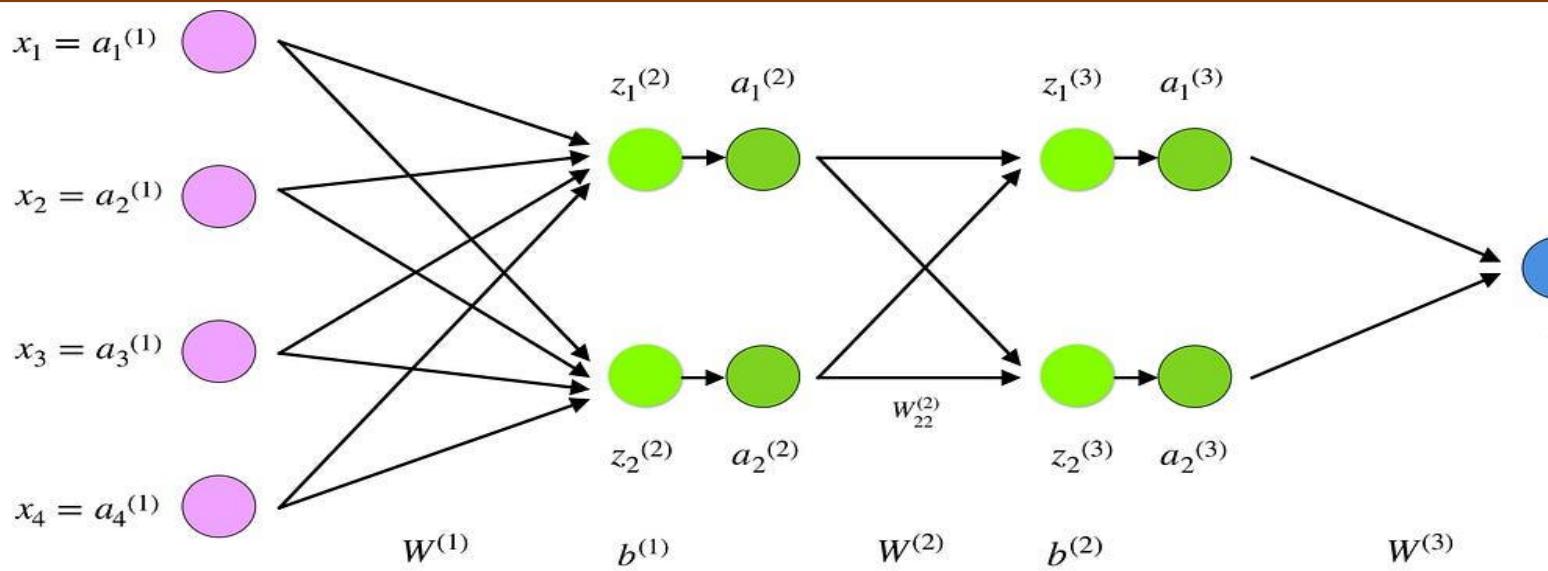


- W^2, W^3, b^2, b^3 - weights and biases in layer 2, 3
- Activations a^2, a^3 computed using activation function f . Typically, non-linear (e.g. sigmoid, ReLU, tanh)





Eg. 4-layer neural network



$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} & W_{14}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} & W_{24}^{(1)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

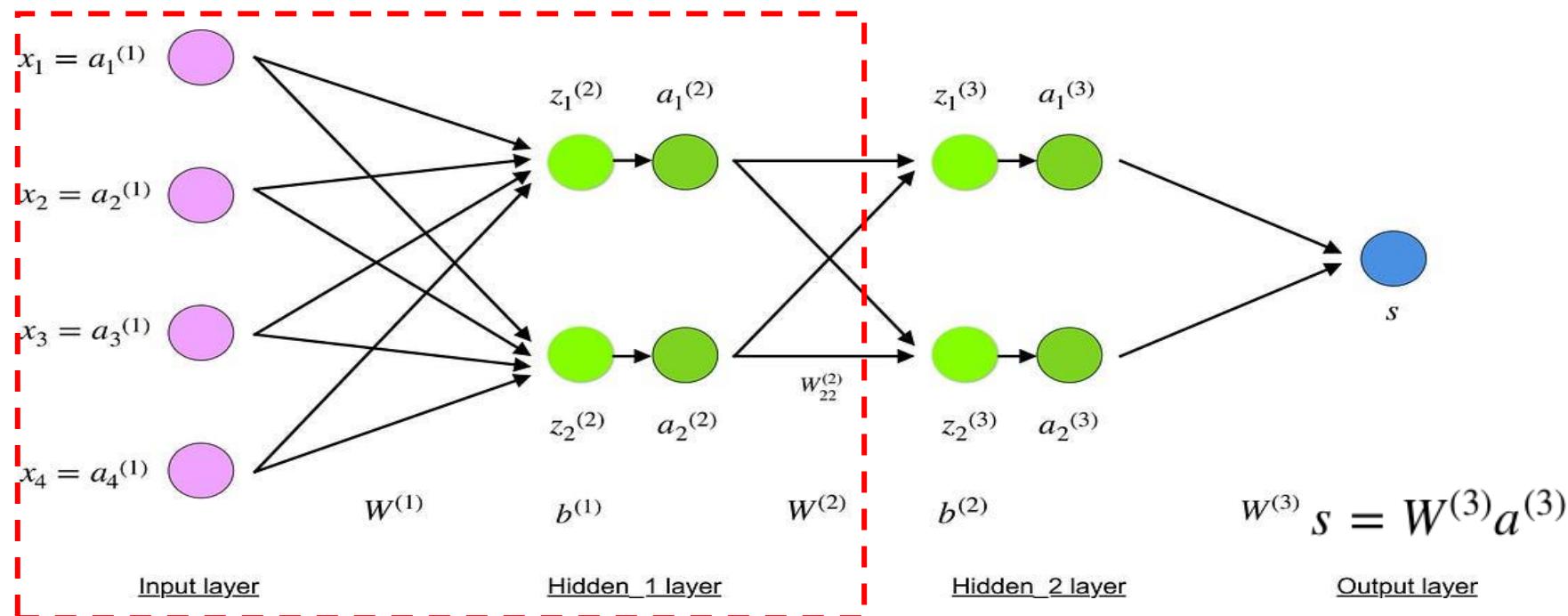
$$s = W^{(3)}a^{(3)}$$

- W^1 is a weight matrix of shape (n, m) : n, m - no. of output & input neurons (neurons of next & previous layer)
- x input vector of shape $(m, 1)$, m : no. of input neurons
- b^1 bias vector of shape $(n, 1)$, n : no. of neurons in current layer





Eg. 4-layer neural network



$$z^{(2)} = \begin{bmatrix} W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + W_{14}^{(1)}x_4 \\ W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + W_{24}^{(1)}x_4 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix}$$





Forward Propagation and evaluation

Overview of forward propagation equations colored by layer

$x = a^{(1)}$ *Input layer*

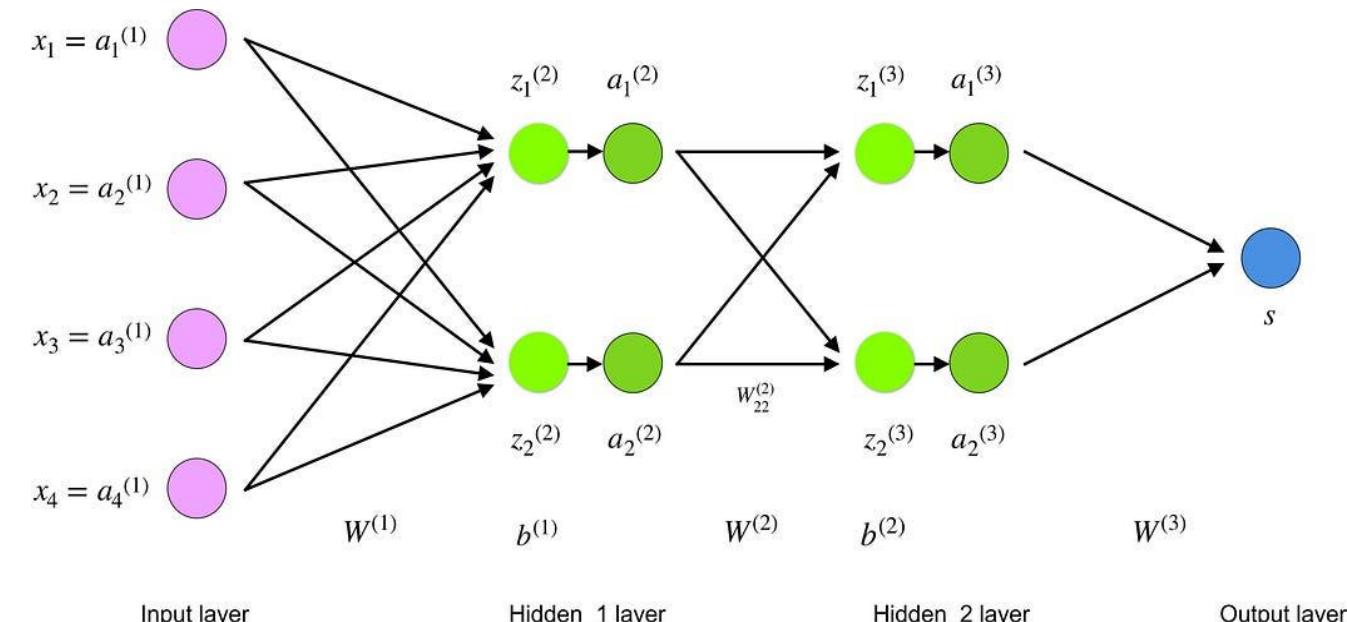
$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at Hidden₁ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at Hidden₁ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at Hidden₂ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at Hidden₂ layer*

$s = W^{(3)}a^{(3)}$ *Output layer*





Forward Propagation and evaluation

$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at Hidden₁ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at Hidden₁ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at Hidden₂ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at Hidden₂ layer*

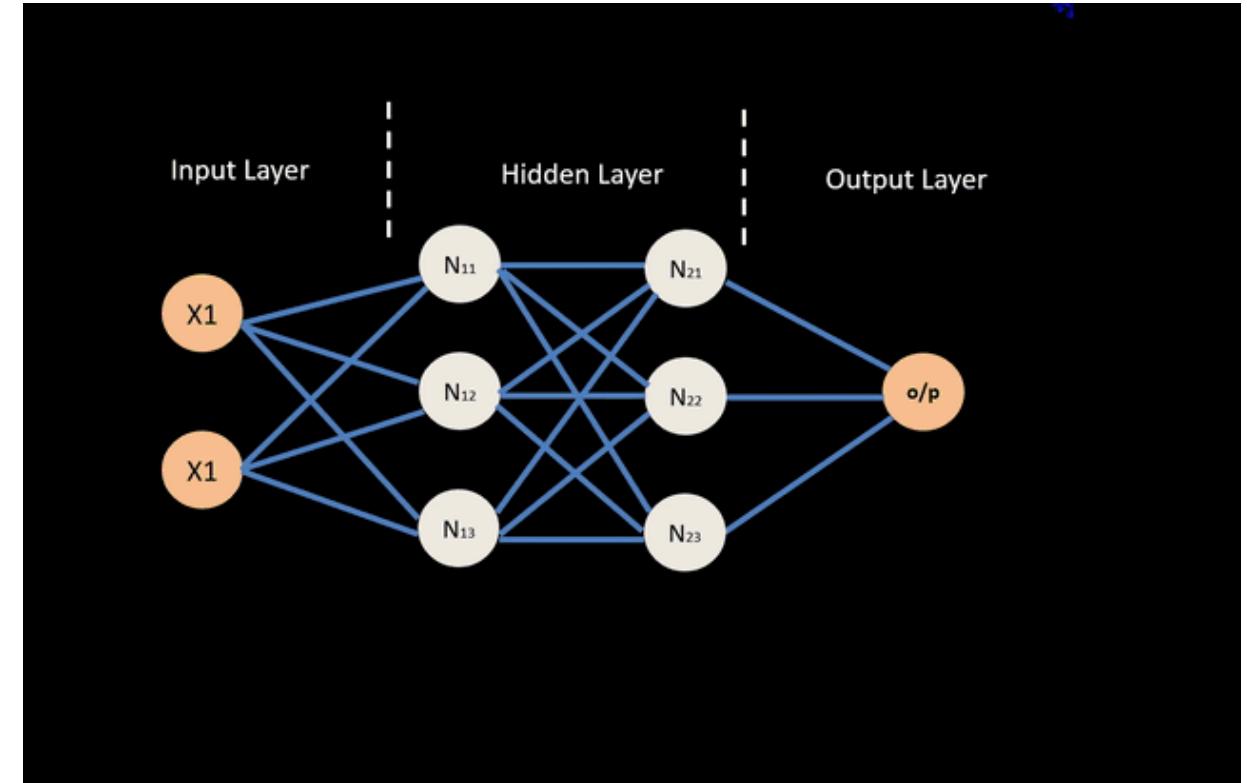
$s = W^{(3)}a^{(3)}$ *Output layer*

- Final step in forward pass is to evaluate the predicted output s against an expected output y .
- Output y is part of the training dataset (x, y) where x : input.
- Evaluation between s and y happens through **cost function or loss function**. can be MSE (mean squared error) or complex like cross-entropy.
- Equation for cost function C
$$C = \text{cost}(s, y)$$
- Based on C 's value, the model “knows” how much to adjust its parameters in order to get closer to the expected output y . This happens using the **BACKPROPAGATION ALGORITHM**.



Backpropagation Algorithm

- The algorithm is used to effectively train a neural network through a method called *chain rule*.
- After each *forward pass* through a network, backpropagation performs a *backward pass* while adjusting the model's parameters (*weights and biases*).





Backpropagation and computing gradients

- Backpropagation aims to minimize the *cost function* by adjusting network's weights and biases.
- The level of adjustment is determined by the *gradients of the cost function* with respect to those parameters.



why computing gradients?

- Gradient of a function $C(x_1, x_2, \dots, x_m)$ in point x is a vector of the partial derivatives of C in x .
- Equation for derivative of C in x $\frac{\partial C}{\partial x} = [\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \dots, \frac{\partial C}{\partial x_m}]$
- Function C derivative measures the sensitivity to change of function value (output value) with respect to a change in its argument x (input value) or the derivative tells us the direction C is going.
- The gradient shows how much the parameter x needs to change (in positive or negative direction) to minimize C .
- *Compute those gradients happens using a technique called chain rule.*



For a single weight $(W_{jk})^l$, the gradient is:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \quad \text{chain rule}$$

$$z_j^l = \sum_{k=1}^m w_{jk}^l a_k^{l-1} + b_j^l \quad \text{by definition}$$

m – number of neurons in $l - 1$ layer

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \quad \text{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \quad \text{final value} \quad \begin{matrix} \text{Equations for derivative of } C \\ \text{in a single weight } (W_{jk})^l \end{matrix}$$

Using Chain rule



Using Chain rule

Similar set of equations can be applied to $(b_j)^l$:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \quad \text{chain rule}$$

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \quad \text{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} 1 \quad \text{final value}$$

Equations for derivative of C in a single bias $(b_j)^l$

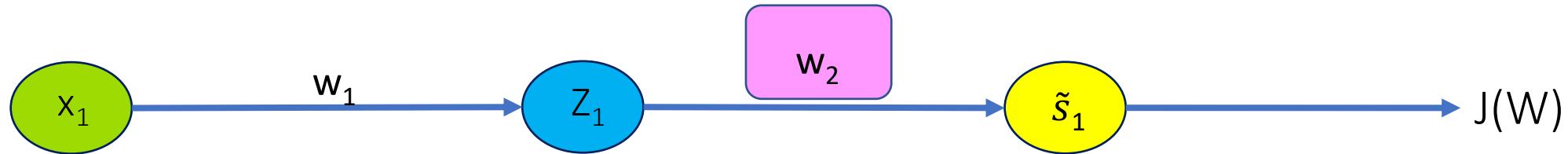
The common part in both equations is often called “local gradient” and is expressed as follows:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \quad \text{local gradient}$$

Equation for local gradient



Computing Gradients: Backpropagation



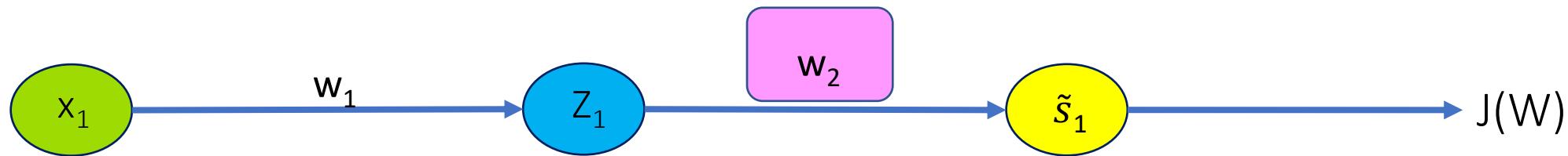
How does a small change in one weight (ex. W_2) affect the final loss $J(W)$?

- Simple network with one input layer, one hidden layer (one hidden neuron) and one output layer.
- Computing the gradient of loss of $J(W)$ with respect to w_2 , can perform lot of changes in loss value.
- How does a small change in one weight (ex. W_2) affect the final loss $J(W)$?





Computing Gradients: Backpropagation



Gradient loss of W with respect to w_2

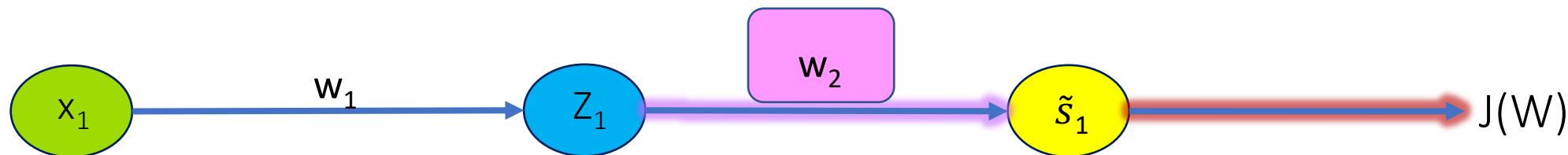
$$\frac{\partial J(W)}{\partial w_2}$$

Let's use the chain rule!





Computing Gradients: Backpropagation

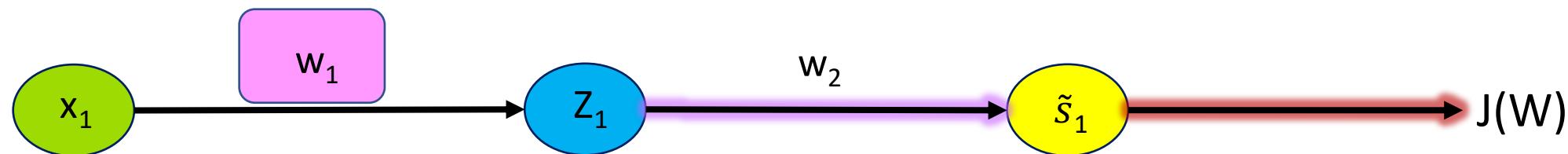


Split this into the gradient of our loss
with respect to output.

$$\frac{\partial J(W)}{\partial (w_2)} = \underline{\frac{\partial J(W)}{\partial \tilde{s}}} * \underline{\frac{\partial \tilde{s}}{\partial (w_2)}}$$



Computing Gradients: Backpropagation



- Replace w_2 with w_1 and apply chain rule but we now notice that the gradient of output s with respect to w_1 is not directly computable , apply chain rule again to evaluate.

$$\frac{\partial J(W)}{\partial (w_1)} = \frac{\partial J(W)}{\partial \tilde{s}} * \frac{\partial \tilde{s}}{\partial (w_1)}$$

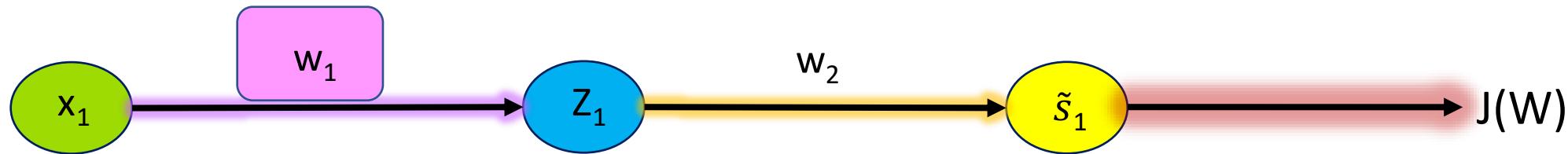
Apply chain rule!

Apply chain rule!





Computing Gradients: Backpropagation

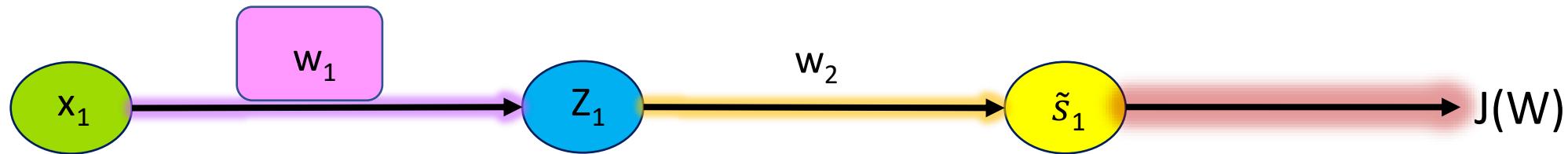


- Apply the chain rule, and split with respect to Z.
- Back propagation is done performing it, all the gradients from the output to the input that allow the error to propagate from output layer to input layer.

$$\frac{\partial J(W)}{\partial (w_1)} = \underbrace{\frac{\partial J(W)}{\partial \tilde{s}}}_{\text{red bar}} * \underbrace{\frac{\partial \tilde{s}}{\partial (Z_1)}}_{\text{yellow bar}} * \underbrace{\frac{\partial Z_1}{\partial w_1}}_{\text{purple bar}}$$



Computing Gradients: Backpropagation

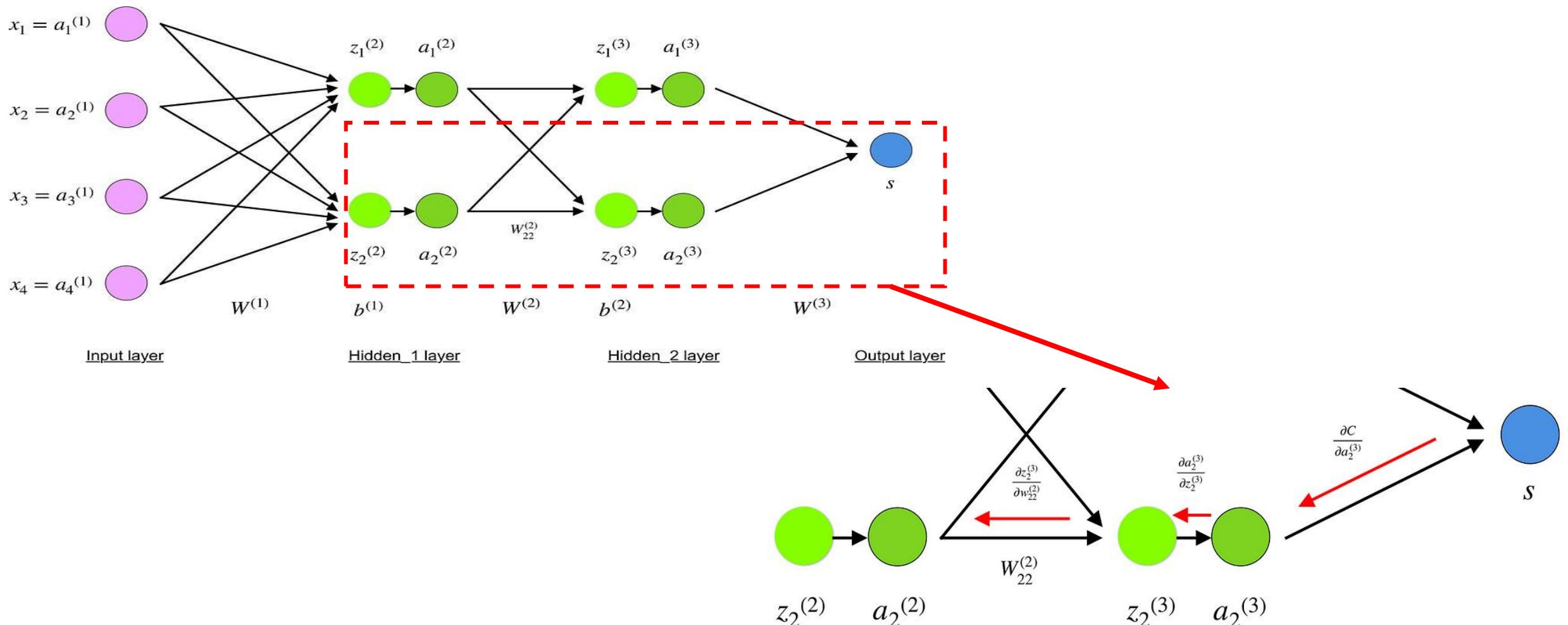


- Repeat this for every weight in the network using gradients from later layers

$$\frac{\partial J(W)}{\partial (w_1)} = \underbrace{\frac{\partial J(W)}{\partial \tilde{s}}}_{\text{red}} * \underbrace{\frac{\partial \tilde{s}}{\partial (Z_1)}}_{\text{yellow}} * \underbrace{\frac{\partial Z_1}{\partial w_1}}_{\text{purple}}$$

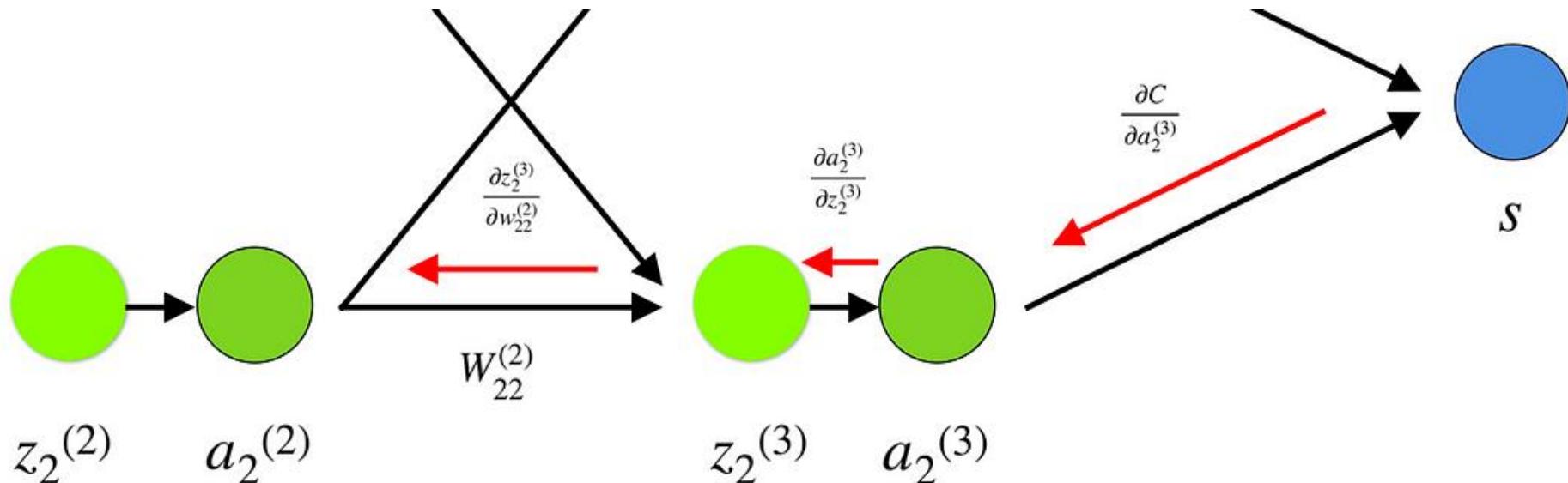


Calculate the gradient of C with respect to a single weight ($W_{22}^{(2)}$)² ?





Calculate the gradient of C with respect to a single weight $(W_{22})^2$?



- Weight $(W_{22})^2$ connects $(a_2)^2$ and $(Z_2)^2$, so computing the gradient requires applying the chain rule through $(Z_2)^3$ and $(a_2)^3$:

$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \cdot a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} \cdot f'(z_2^{(3)}) \cdot a_2^{(2)}$$

- Equation for derivative of C in $(W_{22})^2$
- Calculating the final value of derivative of C in $(a_2)^3$ requires knowledge of the function C. Since C is dependent on $(a_2)^3$,



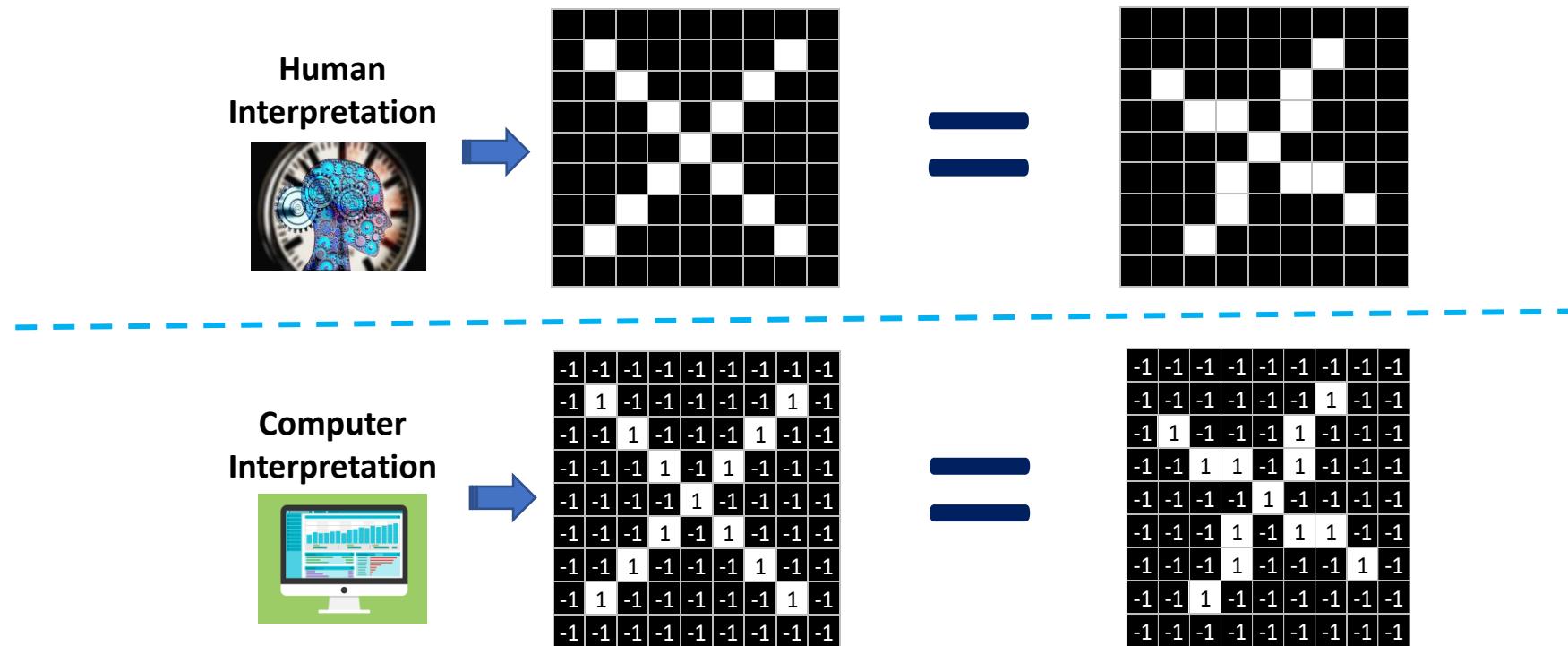
Deep learning

- Course Code:
Convolutional Neural Networks
- Lecture
Basics of convolution operation





Computer and Human Interpretation





Computer Interpretation

Pixel wise
Matching



-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1	
-1	X	X	-1	-1	X	X	-1	-1	
-1	-1	X	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	1	-1	1	X	-1	-1	
-1	-1	X	X	-1	-1	X	X	-1	
-1	X	X	-1	-1	-1	-1	X	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	

Decision



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



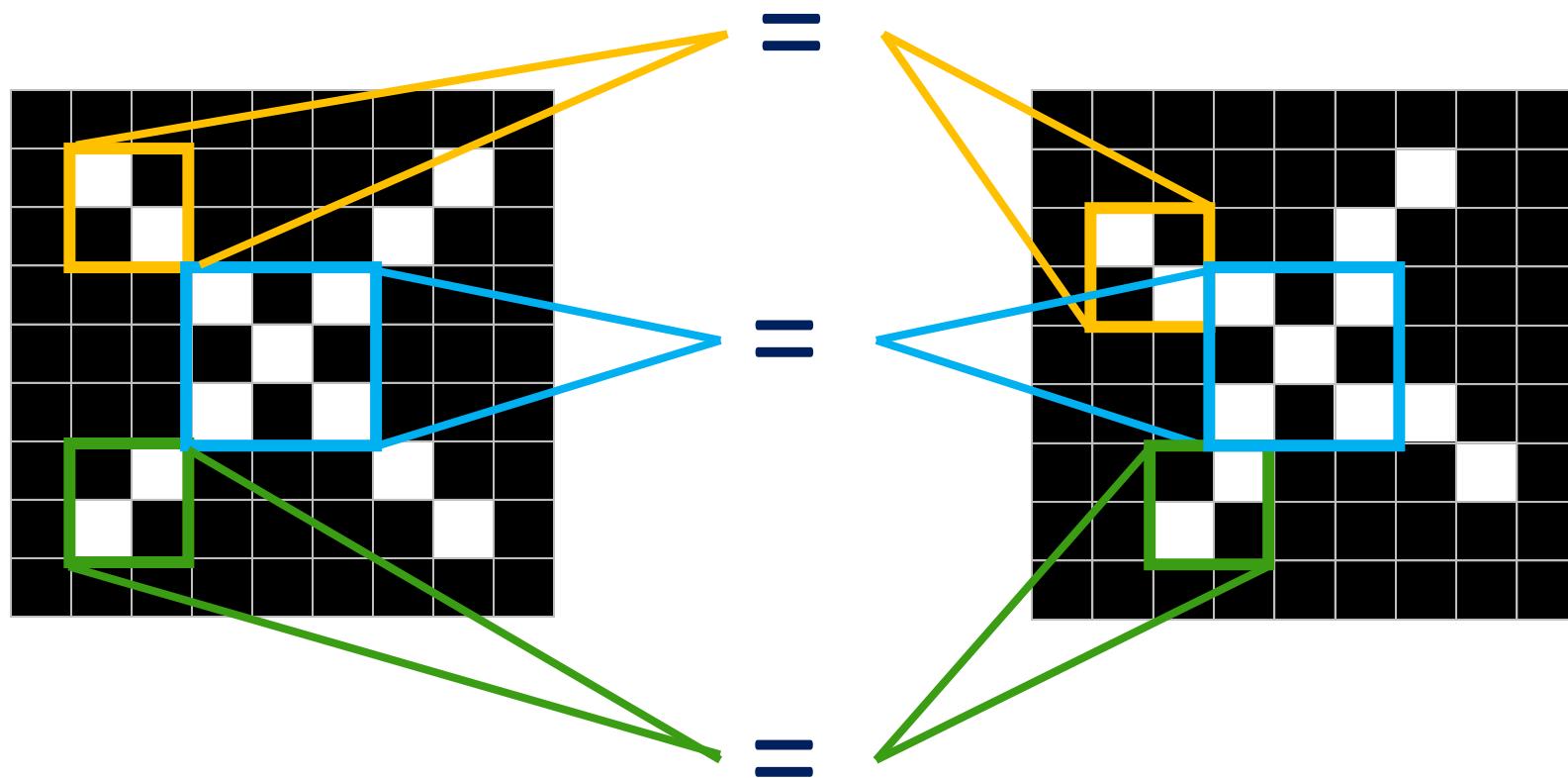
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	
-1	1	-1	-1	-1	-1	1	-1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	1	-1	-1	-1	-1	1	-1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	

Computers are Literal





Feature matching for symbol 'X'





Convolution Operation

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

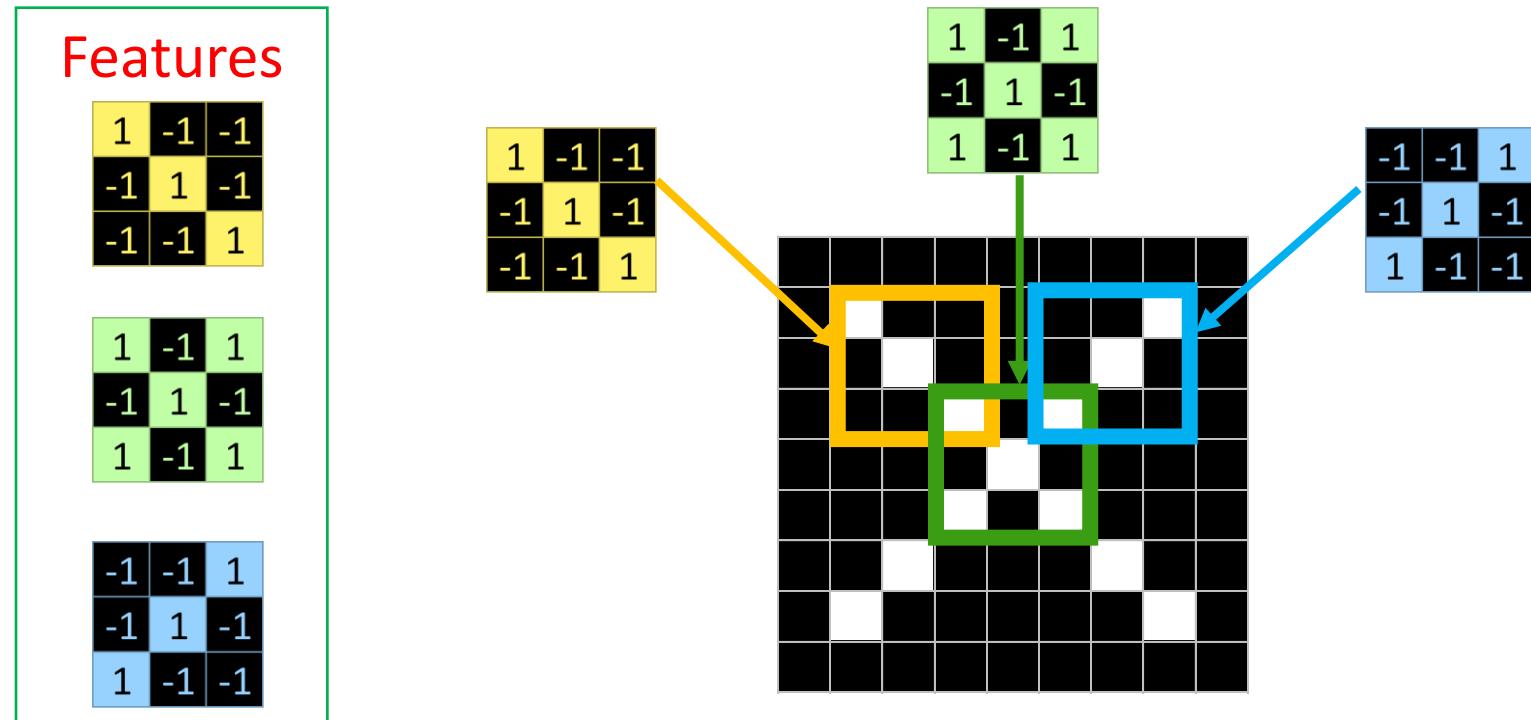
$$\boxed{1} \times \boxed{1} = 1$$

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



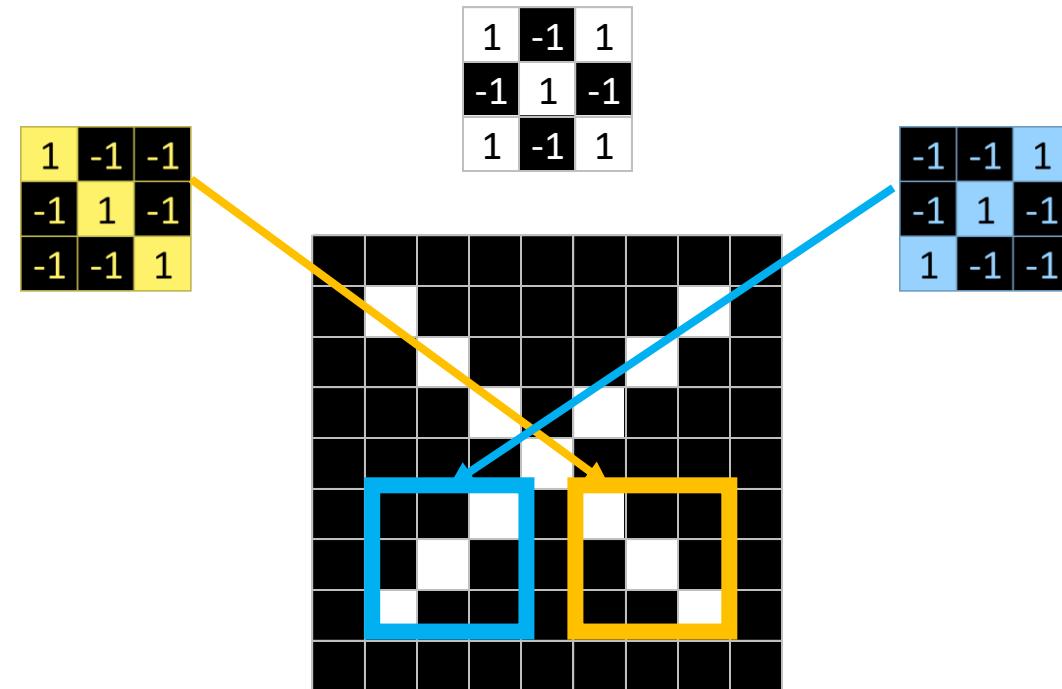


Piece Matching of Features





Piece Matching of Features





Convolution Operation

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

Image

4		

Convolved
Feature





Convolution Operation

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

$$1 \times 1 = 1$$

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

1			




Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1		





Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1	1





Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1	1
1		





Convolution Operation

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

$$\boxed{1} \times \boxed{1} = 1$$

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

1	1	1
1	1	





Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1	1
1	1	1





Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1	1
1	1	1
1		





Convolution Operation

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

$$\boxed{-1} \times \boxed{-1} = 1$$

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

1	1	1
1	1	1
1	1	





Convolution Operation

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

$$\boxed{1} \times \boxed{1} = 1$$

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

1	1	1
1	1	1
1	1	1





Convolution Operation

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

$$\boxed{1} \times \boxed{1} = 1$$

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

1	1	1
1	1	1
1	1	1





Feature Mapping

These are the network
parameters to be learned.

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

6 x 6 image

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

Filter 1

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

Filter 2

⋮

Each filter detects a small pattern (3 x 3).



Feature Mapping

stride=1

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

Dot
product



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

Filter 1

6 x 6 image



Feature Mapping

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

Filter 1

If stride=2

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



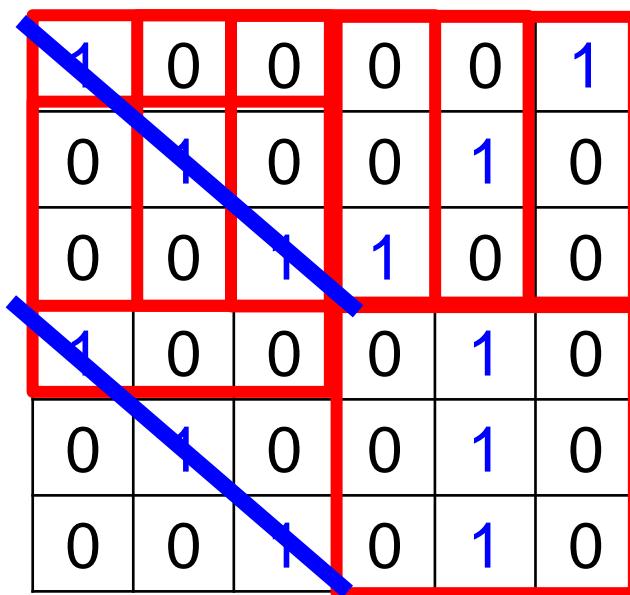
6 x 6 image





Feature Mapping

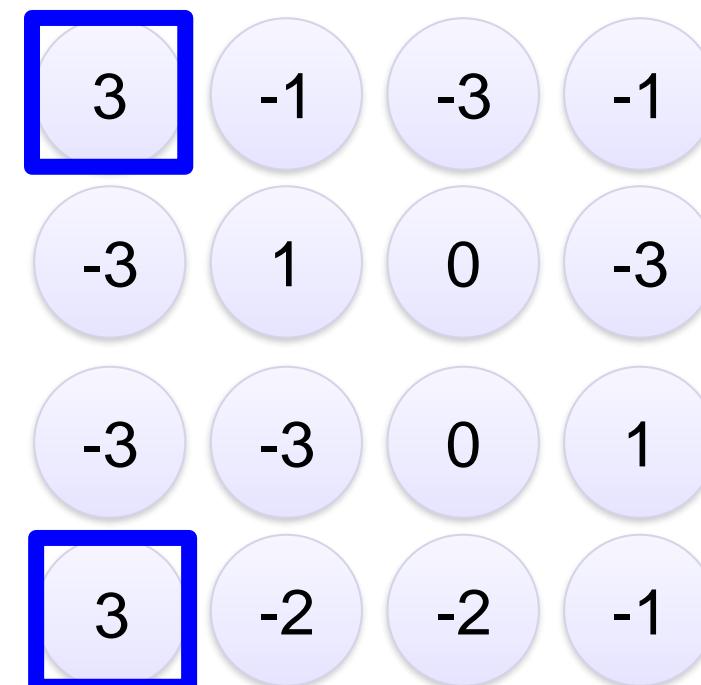
stride=1



6 x 6 image

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

Filter 1





Feature Mapping

stride=1

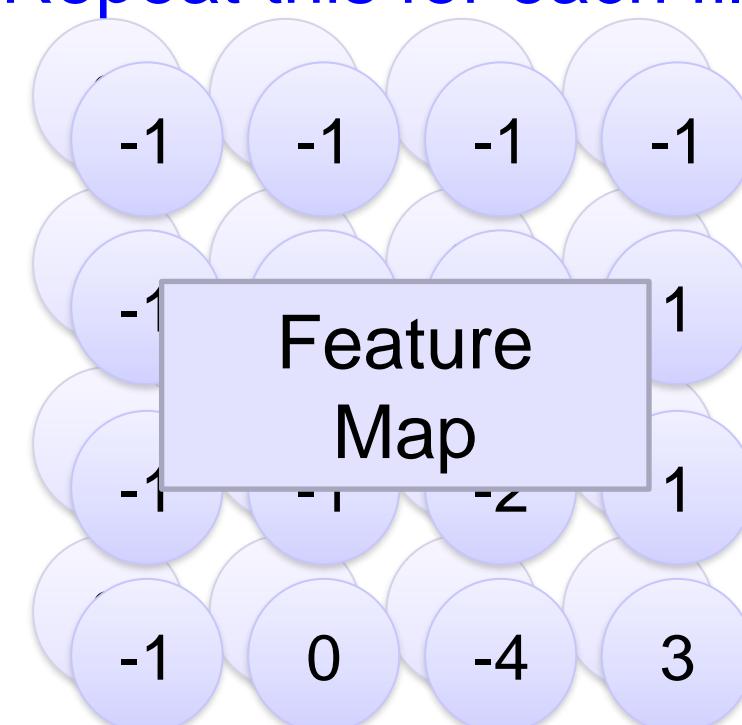
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 2

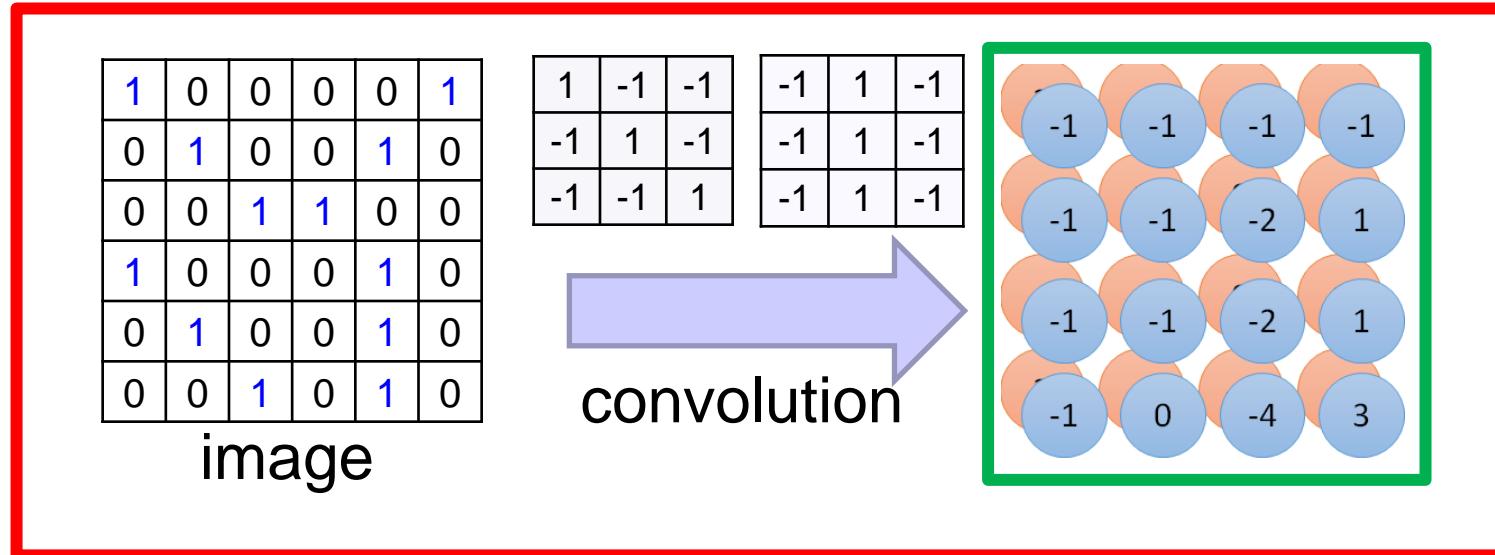
Repeat this for each filter



Two 4 x 4 images
Forming 4 x 4 x 2 matrix

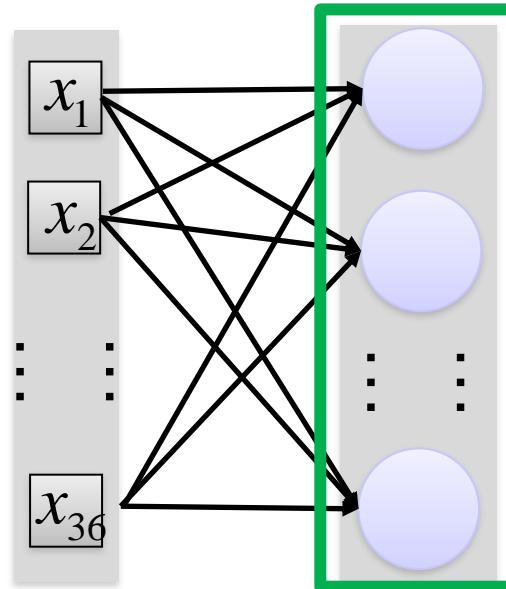


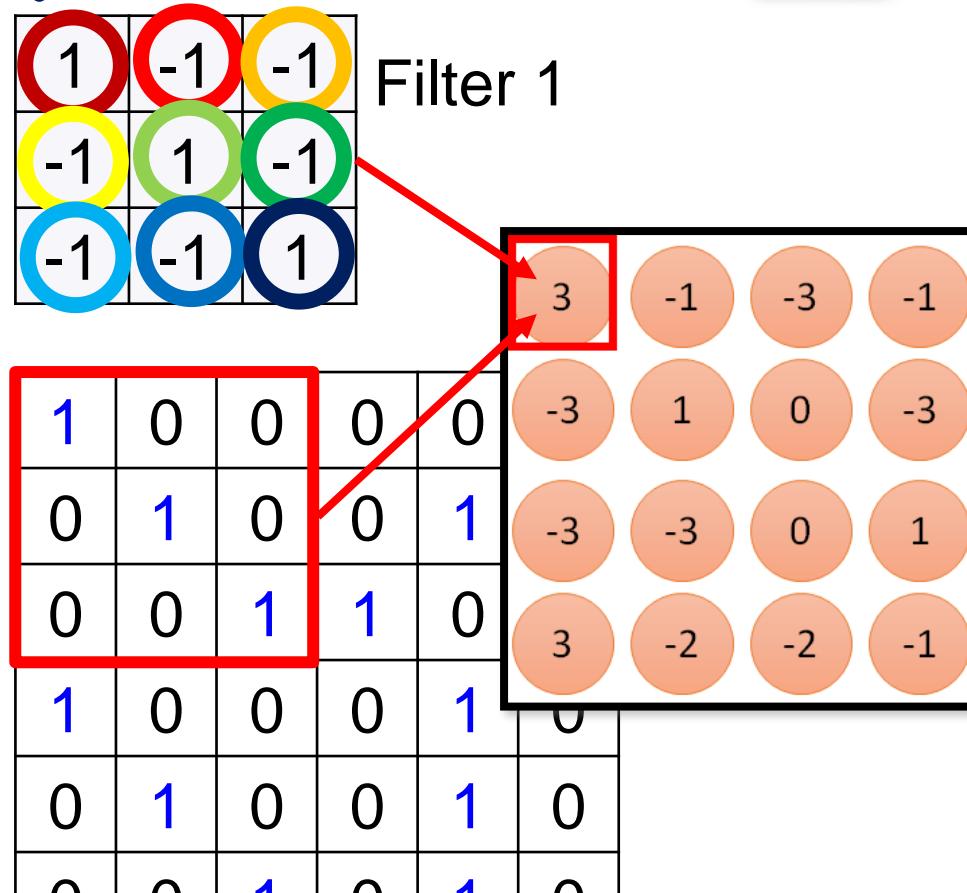
Convolution v.s. Fully Connected



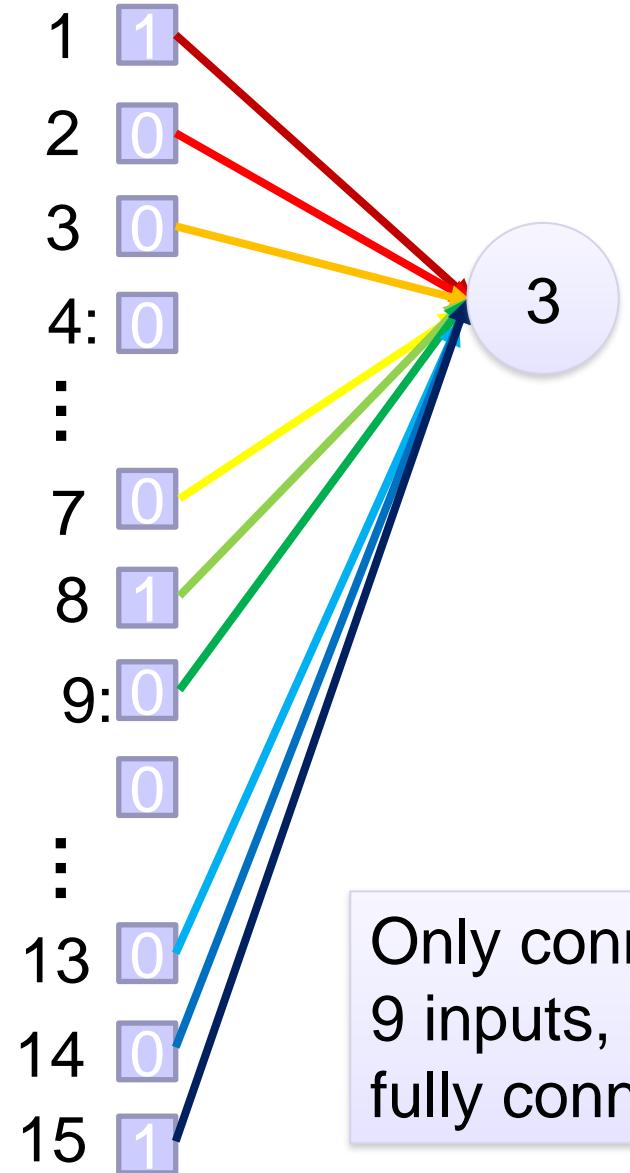
Fully-
connected

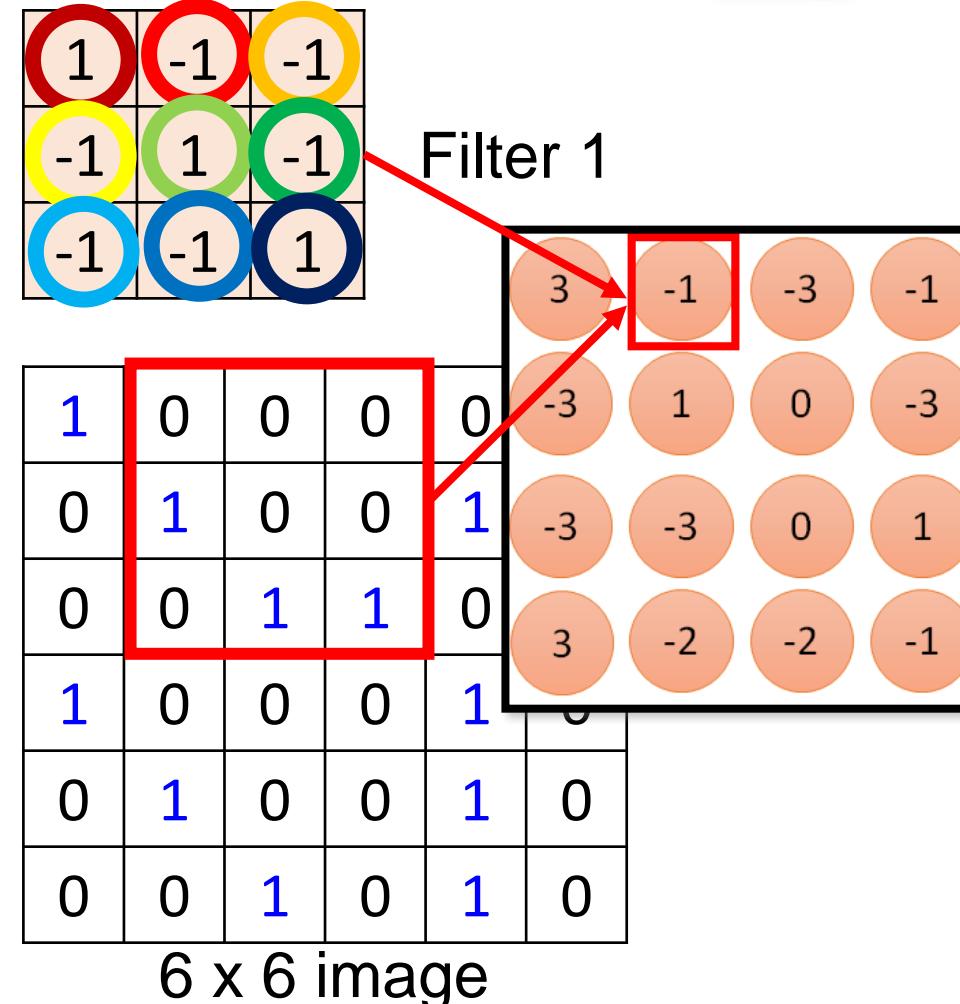
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0





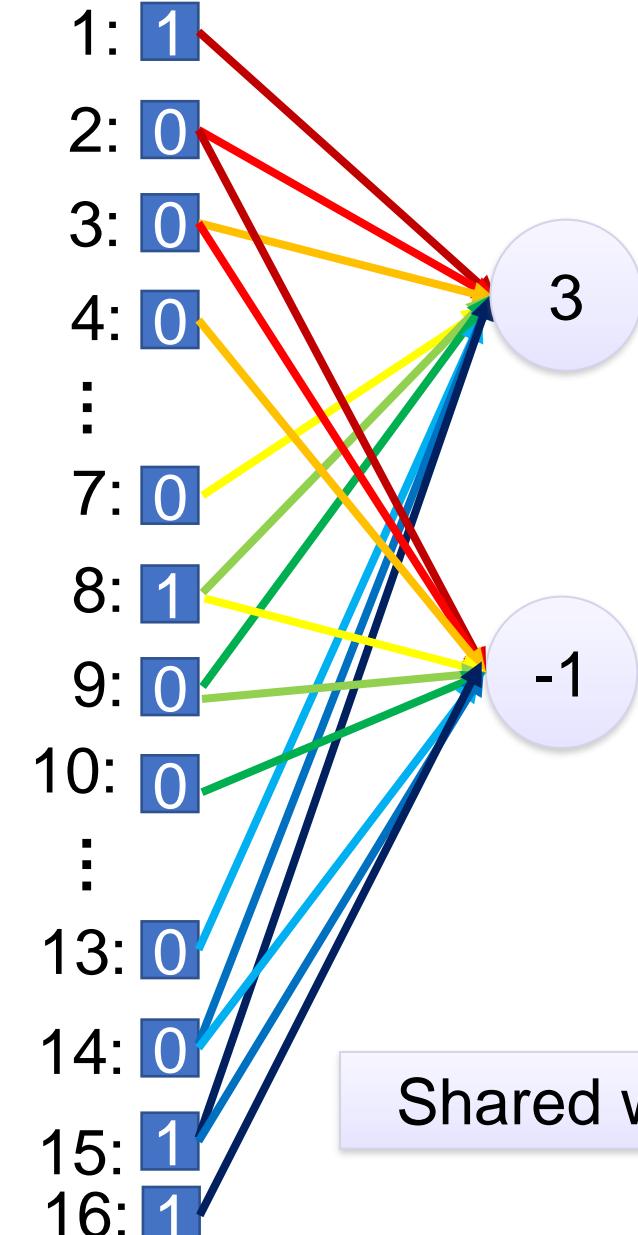
fewer parameters!





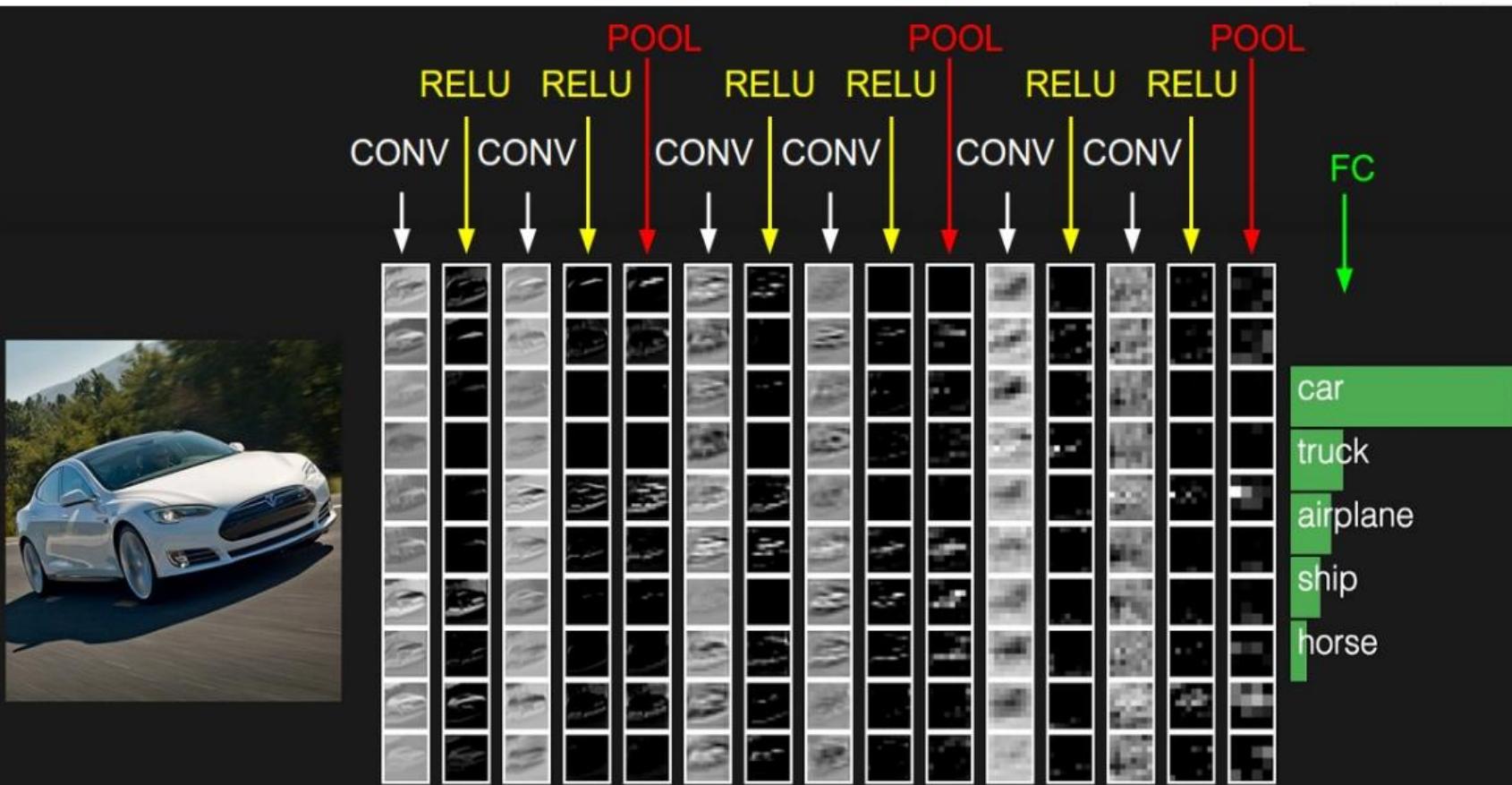
Fewer parameters

Even fewer parameters





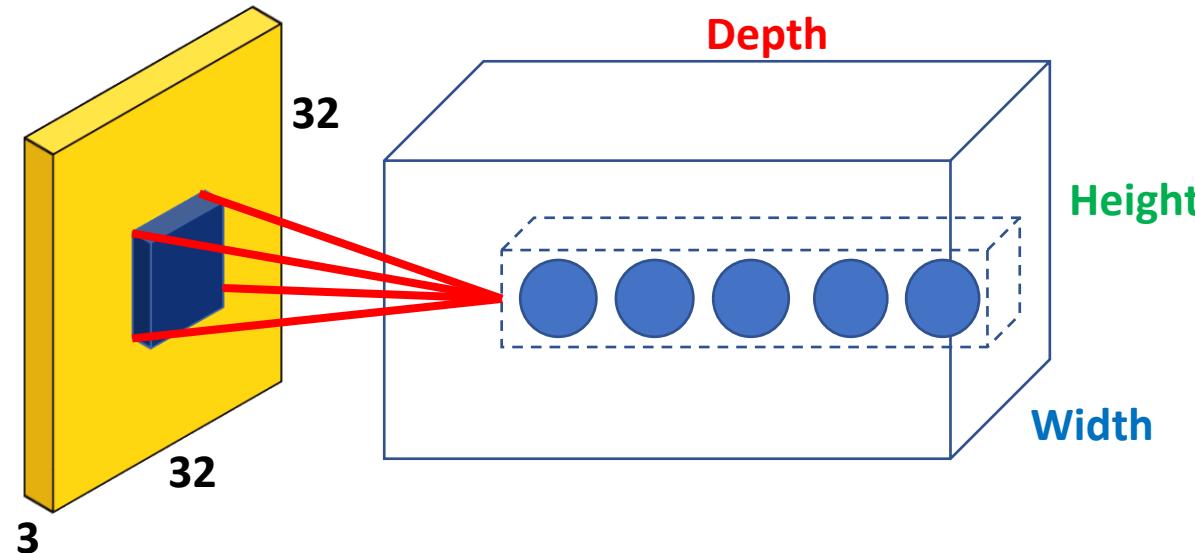
Simple CNN architecture



CONV: Convolutional kernel layer
RELU: Activation function
POOL: Dimension reduction layer
FC: Fully connection layer



Convolutional Neural Network--- Spatial View



Dimensions of Layer

$$H * W * D$$

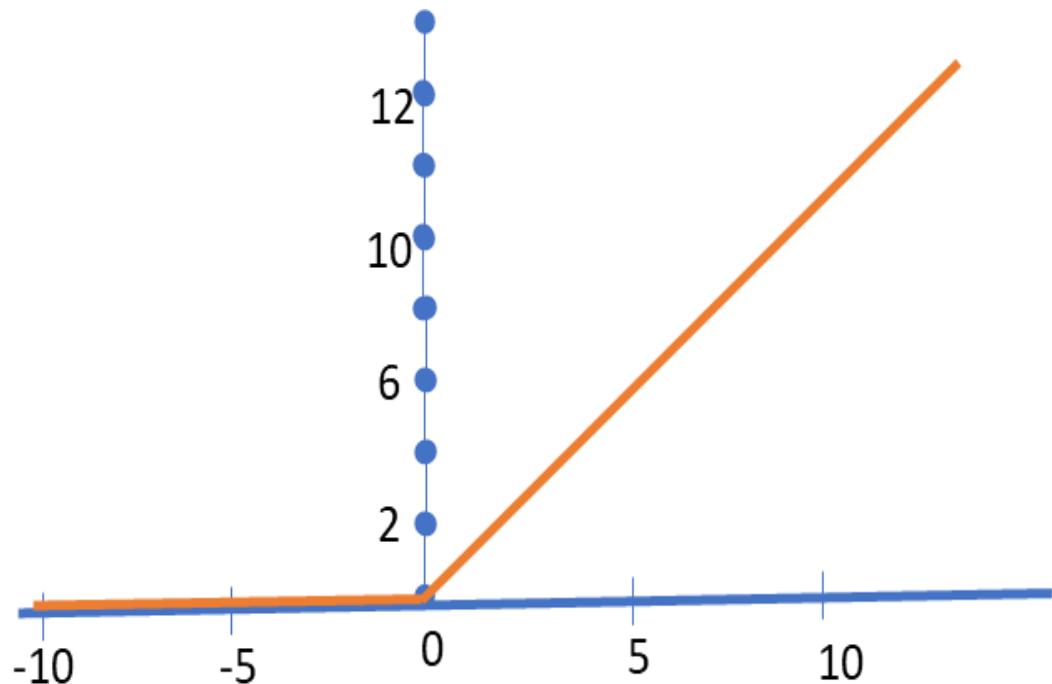
H (height) and **W (width)**
are spatial dimensions
whereas **D (depth)** is
number of filters

Stride = Step size of filter, Receptive Field = Location of connected path in an input image





Non-Linearity in Convolutional Neural Network

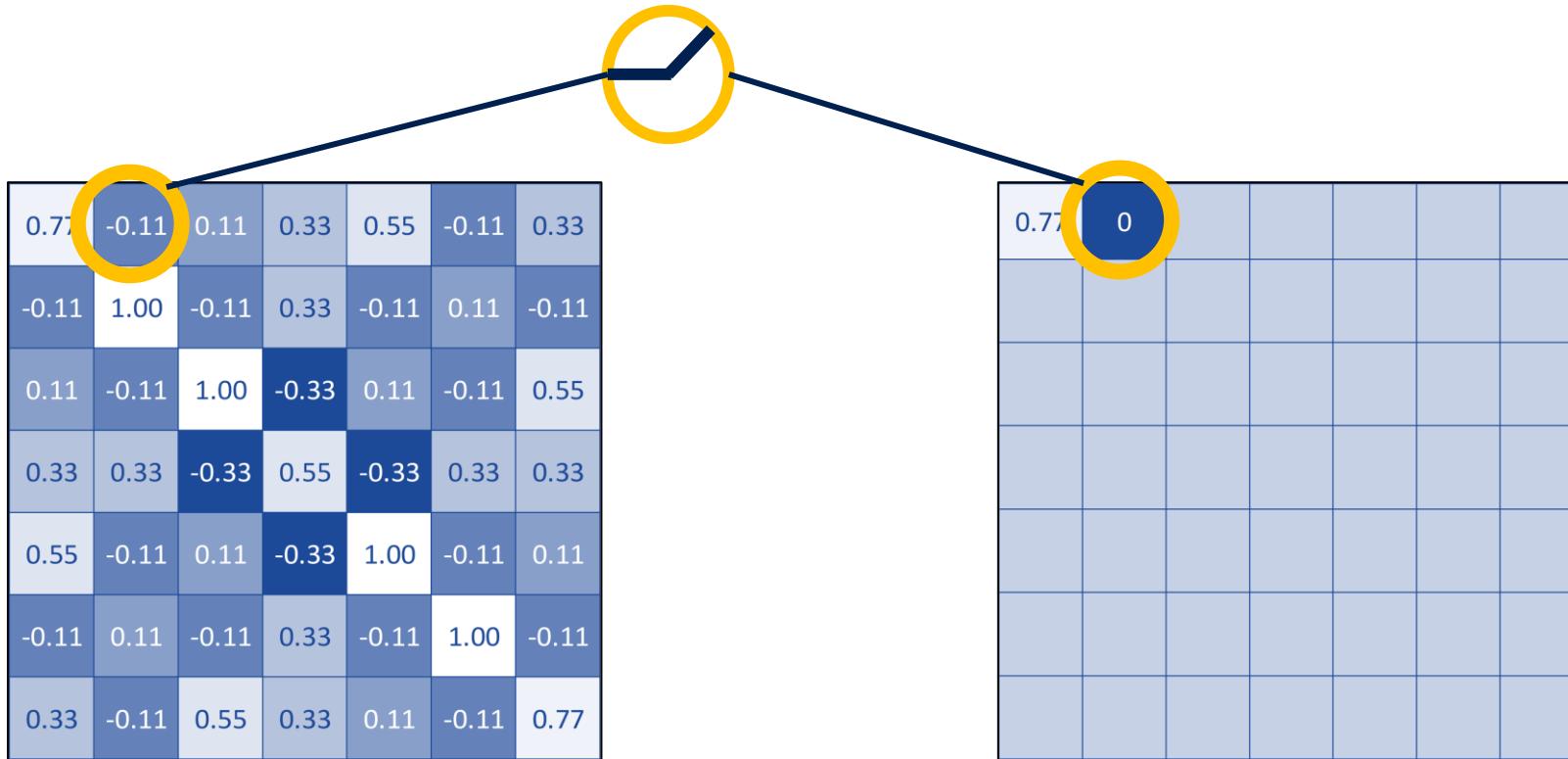


Applied after every convolutional layer. The rectified linear activation (ReLU) function is a simple calculation that returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less.

$$g(x) = \max(0, x)$$

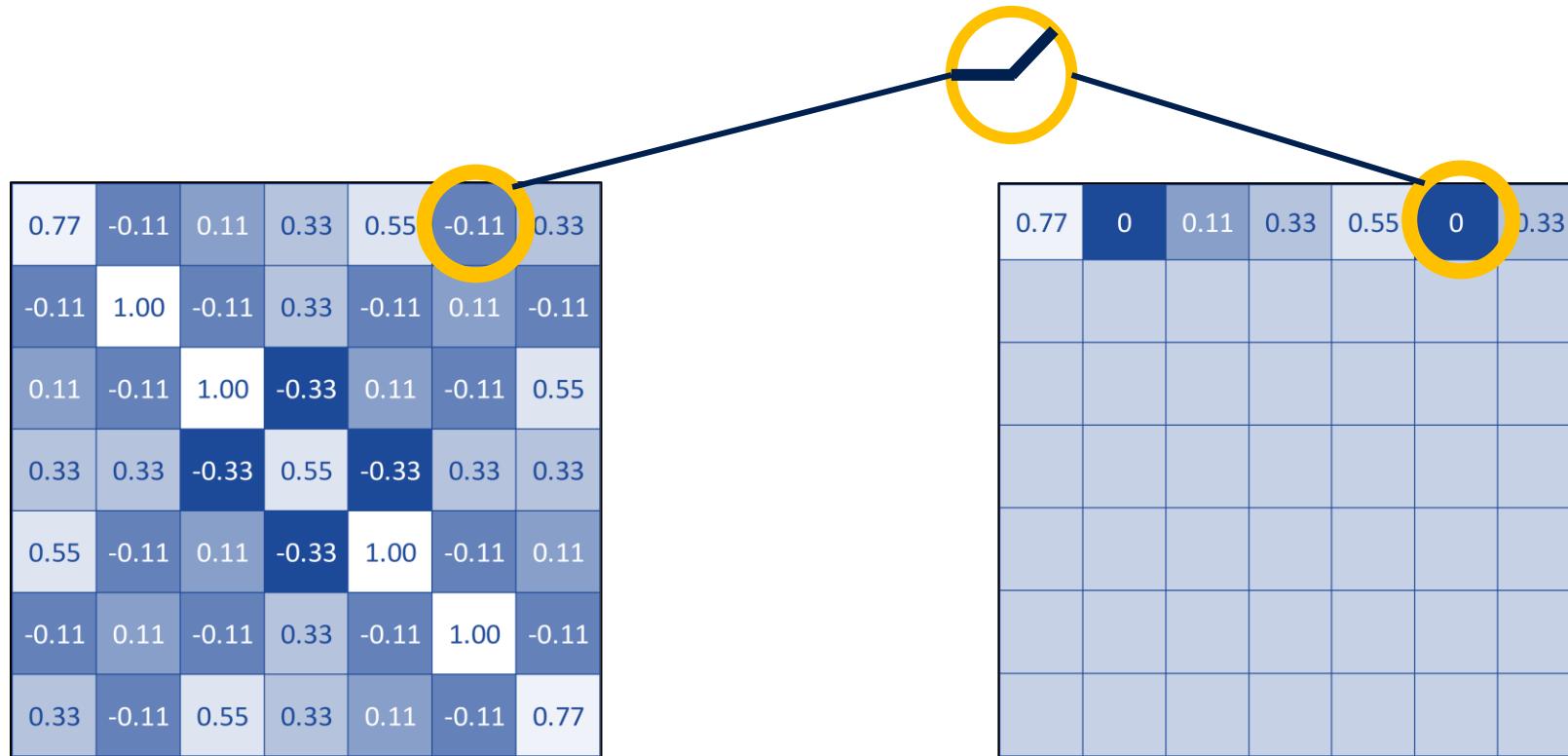


Rectified Linear Units (ReLUs)





Rectified Linear Units (ReLUs)





Rectified Linear Units (ReLUs)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77





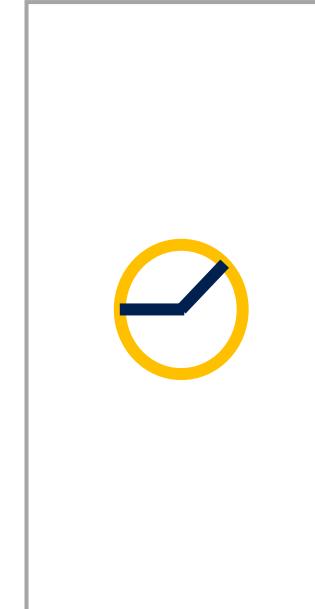
Rectified Linear Units (ReLUs)

A stack of images becomes a stack of images with no negative values.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.11	1.00	0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

0.33	0	0.11	0	0.11	0	0.33
0	0.55	0	0.33	0	0.55	0
0.11	0	0.55	0	0.55	0	0.11
0	0.33	0	1.00	0	0.33	0
0.11	0	0.55	0	0.55	0	0.11
0	0.55	0	0.33	0	0.55	0
0.33	0	0.11	0	0.11	0	0.33

0.33	0	0.55	0.33	0.11	0	0.77
0	0.11	0	0.33	0	1.00	0
0.55	0	0.11	0	1.00	0	0.11
0.33	0.33	0	0.55	0	0.33	0.33
0.11	0	1.00	0	0.11	0	0.55
0	1.00	0	0.33	0	0.11	0
0.77	0	0.11	0.33	0.55	0	0.33





Pooling

- *Dimensionality Reduction*
- *Preserve Spatial Invariance*

The types of pooling operations are:

- **Max pooling:** The maximum pixel value of the batch is selected.
- **Average pooling:** The average value of all the pixels in the batch is selected.

STEPS

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.





Pooling

max pooling

20	30
112	37

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

average pooling

13	8
79	20



Why Pooling ?

- Subsampling pixels will not change the object

bird



Subsampling

bird



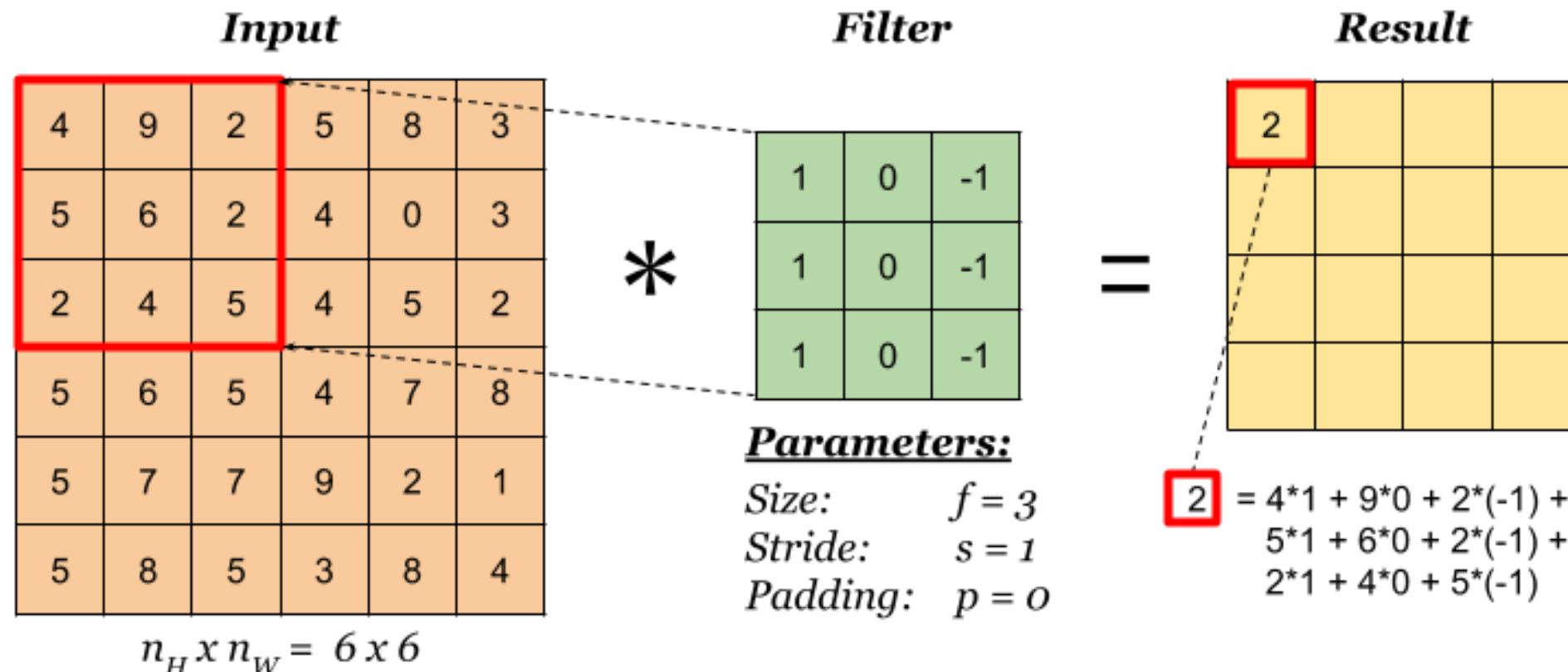
We can subsample the pixels to make image smaller fewer parameters to characterize the image





Stride

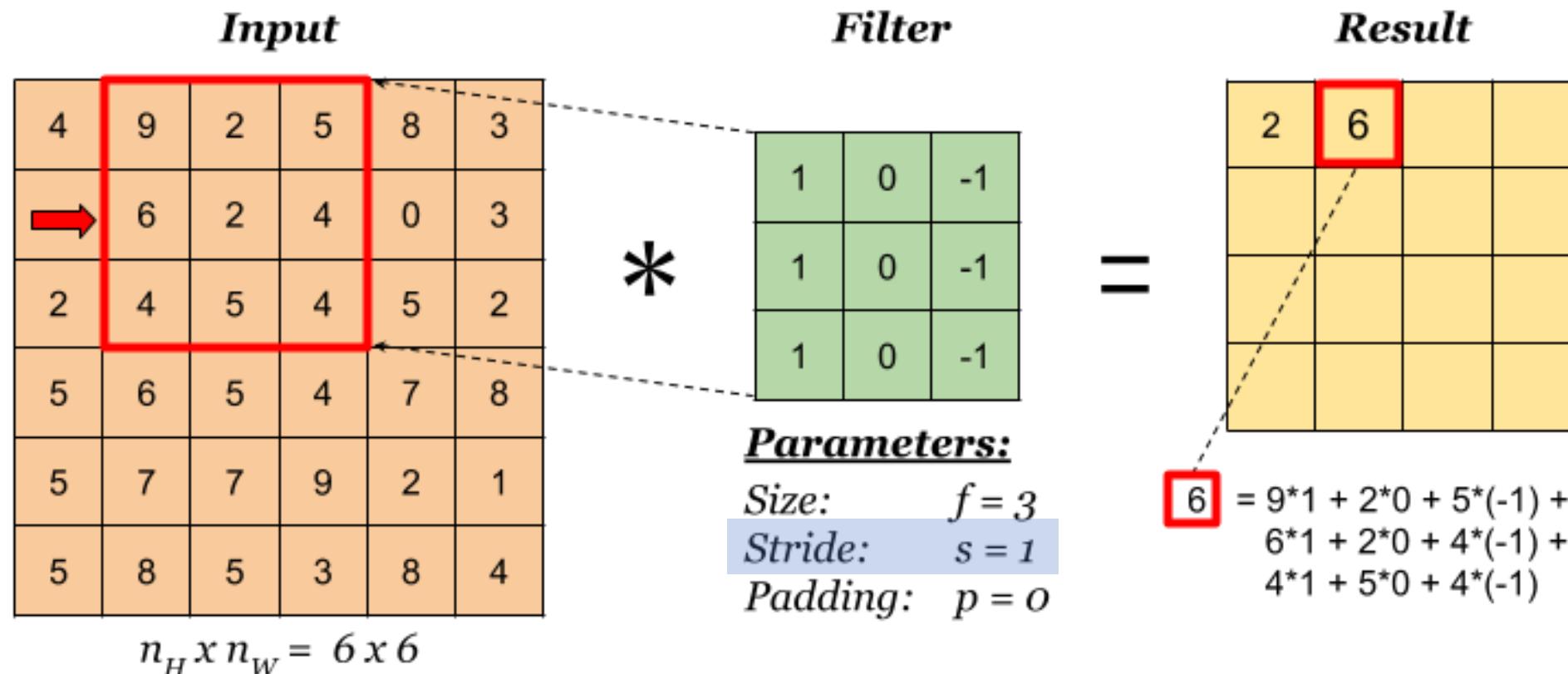
- number of cells the filter is moved to calculate the next output
- sample only every s pixels in each direction in the output





Stride

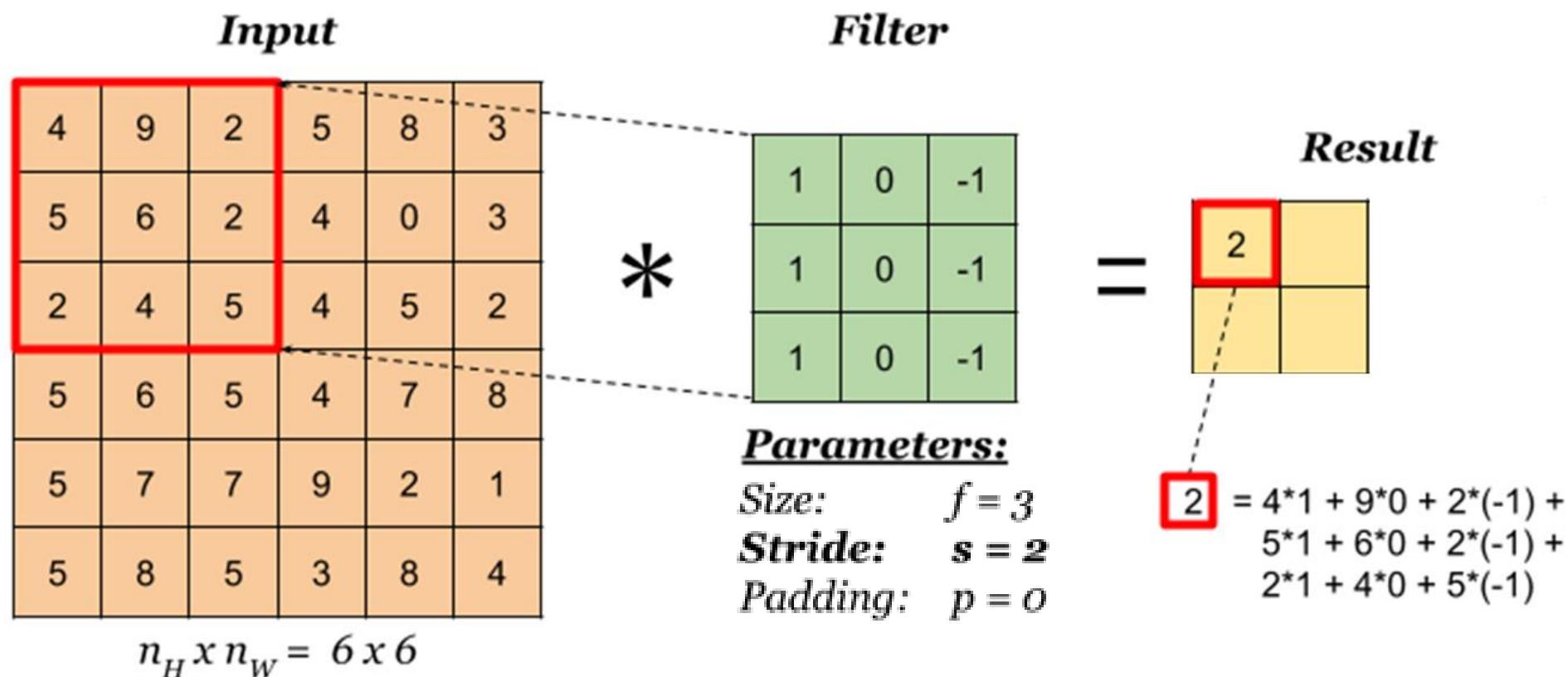
- number of cells the filter is moved to calculate the next output
- sample only every s pixels in each direction in the output





Stride

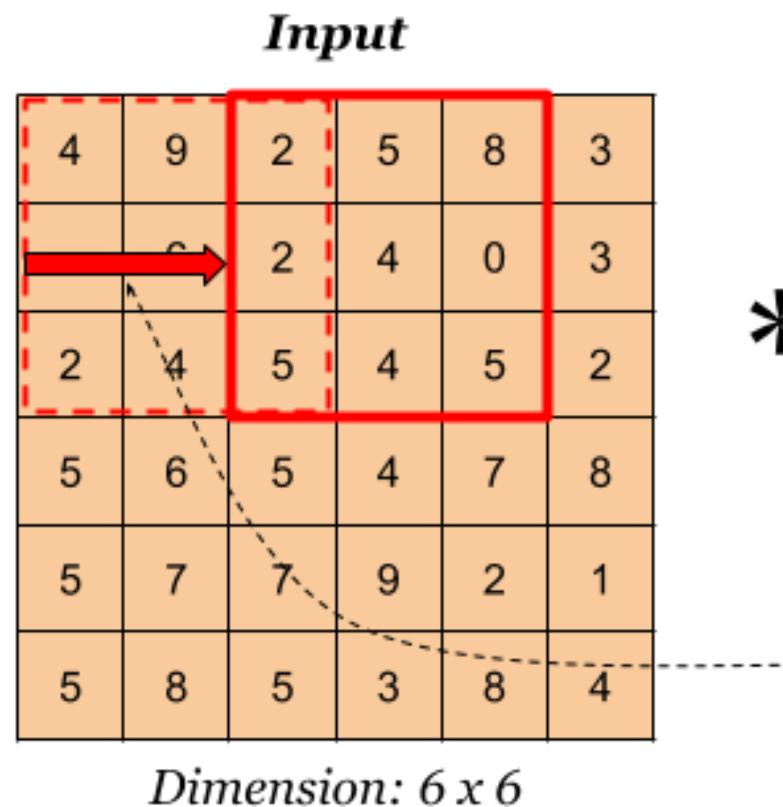
- Stride = 2
- First Value:





Stride

- Stride = 2
- Next Value:



Filter

$$\begin{matrix} * & & \\ & & \\ & & \end{matrix}$$

Filter

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$

Result

$$\begin{matrix} 2 & -4 \\ -4 & \end{matrix}$$

Parameters:

Size: $f = 3$

Stride: $s = 2$

Padding: $p = 0$

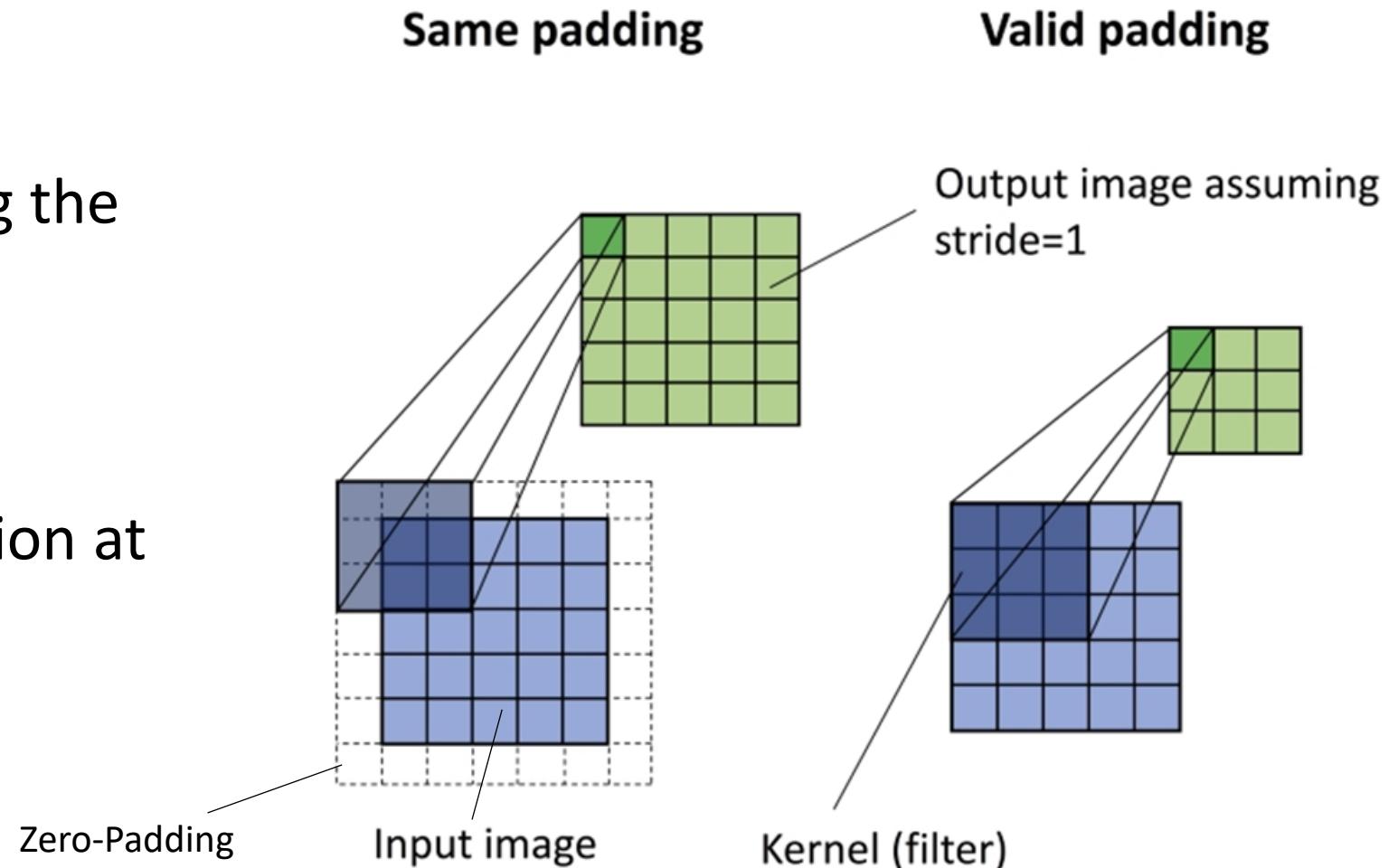
$$\begin{aligned} -4 &= 2*1 + 5*0 + 8*(-1) + \\ &2*1 + 4*0 + 0*(-1) + \\ &5*1 + 4*0 + 5*(-1) \end{aligned}$$

Size of output
feature map may
decrease



Padding

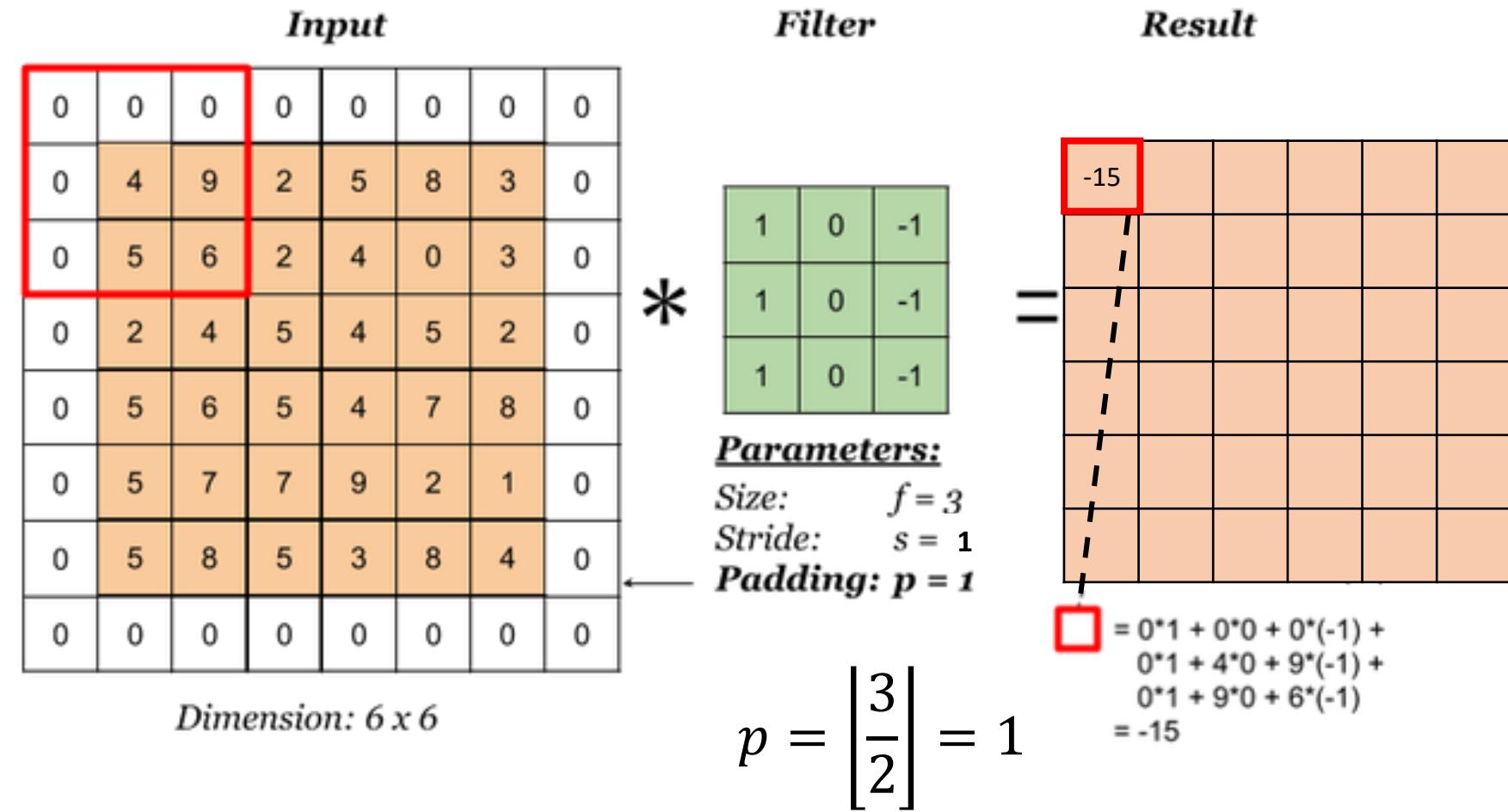
- Use Conv without shrinking the height and width
- Helpful in building deeper networks
- Keep more of the information at the border of an image





Same Padding

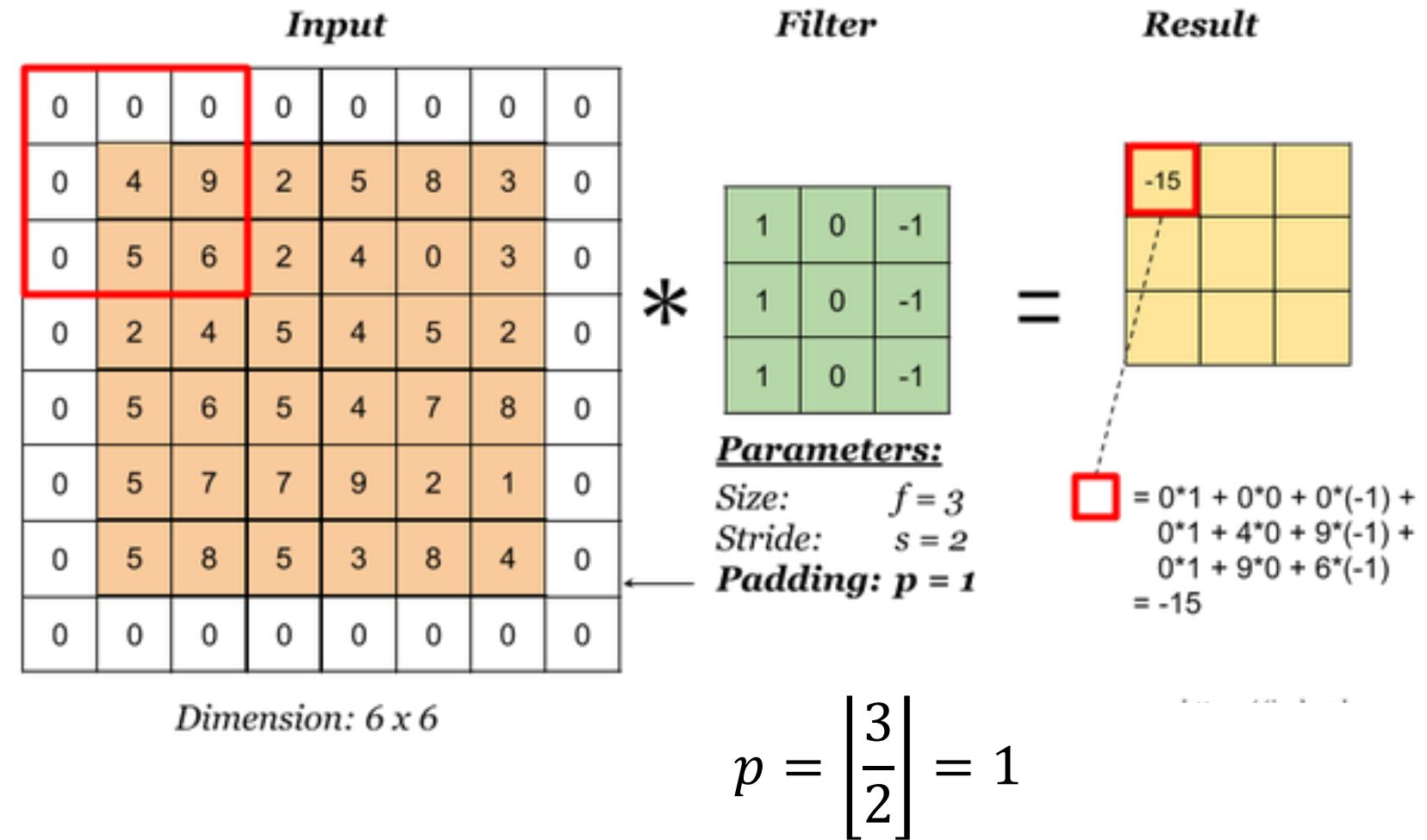
- Buffers the edge of the input with $\lfloor \text{filter_size}/2 \rfloor$ zeros (integer division)
- Output dimension is the same as the input for $s=1$





Same Padding

- Buffers the edge of the input with $\lfloor \text{filter_size}/2 \rfloor$ zeros (integer division)
- Output dimension is the same as the input for $s=1$
- Output dimension reduces less for $s>1$





HOW TO CALCULATE THE OUTPUT SIZE OF A CONVOLUTIONAL LAYER

To determine the output size of the convolution, the following equation can be applied:

$$n_{out} = \frac{n_{in}+2p-k}{s} + 1$$





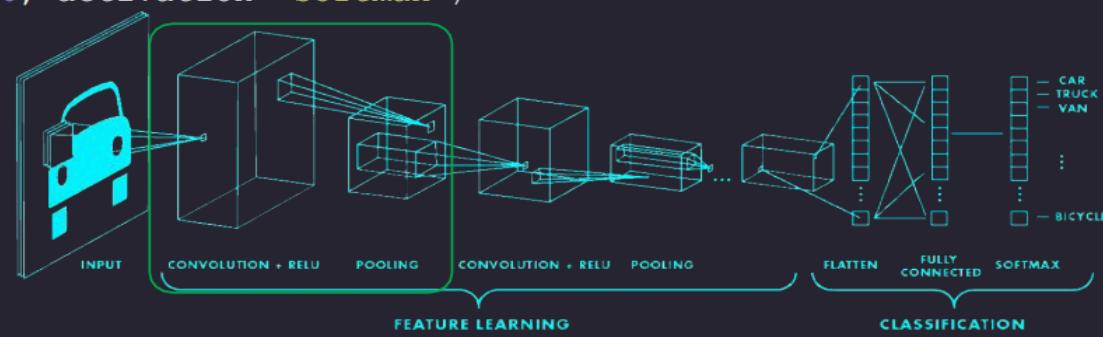
Putting it all together

```
import tensorflow as tf

def generate_model():
    model = tf.keras.Sequential([
        # first convolutional layer
        tf.keras.layers.Conv2D(32, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),  
  

        # second convolutional layer
        tf.keras.layers.Conv2D(64, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),  
  

        # fully connected classifier
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
        # 10 outputs
    ])
    return model
```





CNN Playground

Click the link below or copy paste the URL in your browser

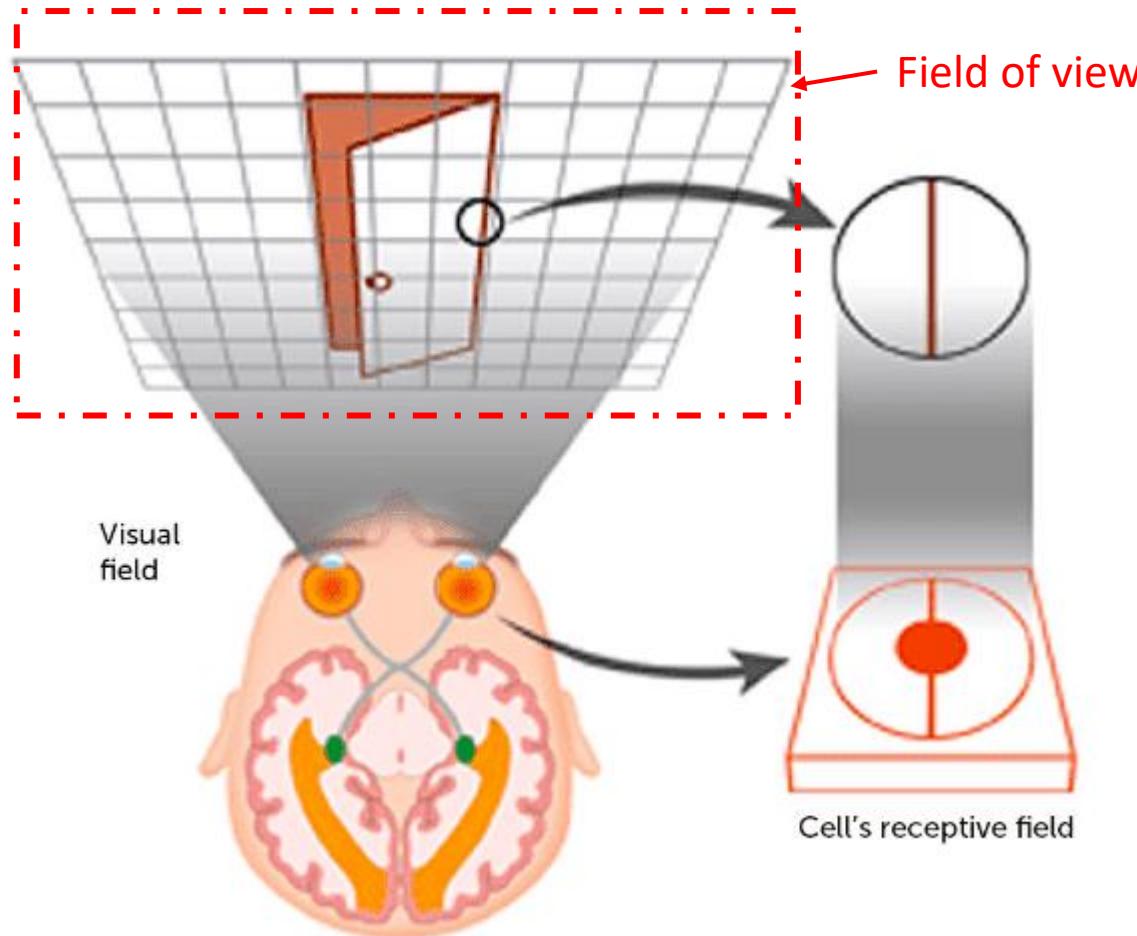
<https://poloclub.github.io/cnn-explainer/>

With the CNN Explainer you can Learn and implement Convolutional Neural Network (CNN) in your browser! With real sample image dataset





Understanding Receptive field

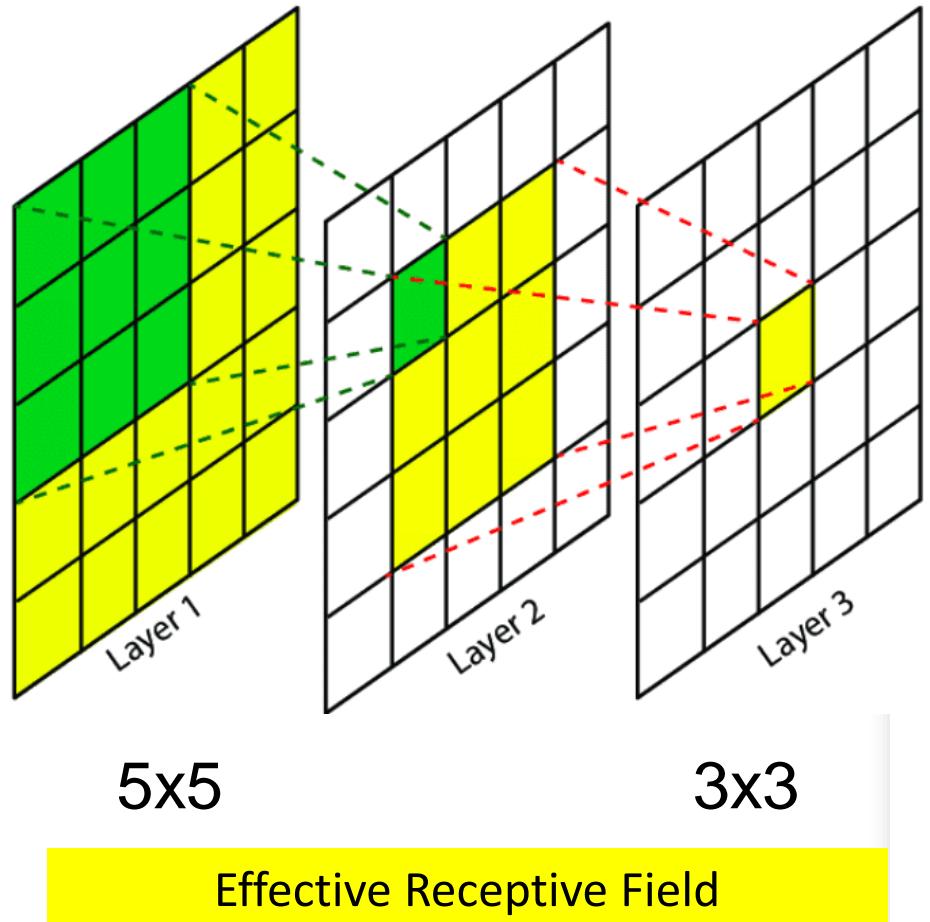


- The human visual system consists of millions of neurons, where each one captures different information.
- Defined as neuron's receptive field as the patch of the total field of view Or what information a single neuron has access to..

ImageSource: <https://www.brainhq.com/brain-resources/brain-connection>



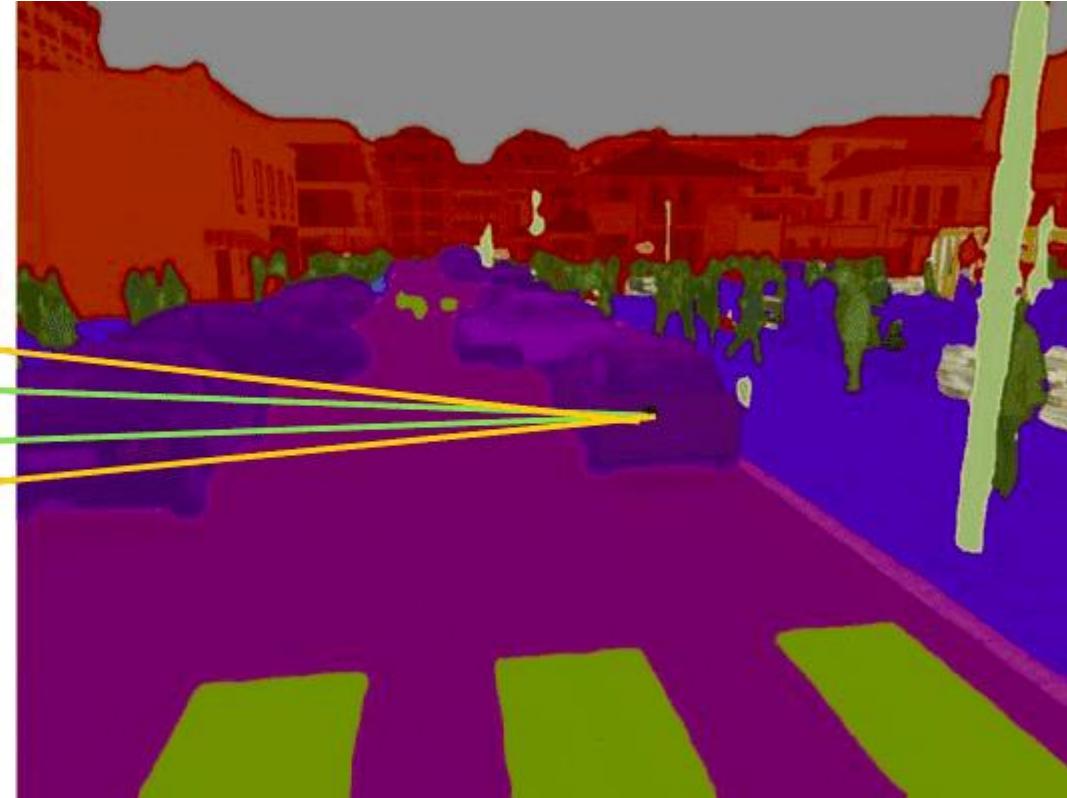
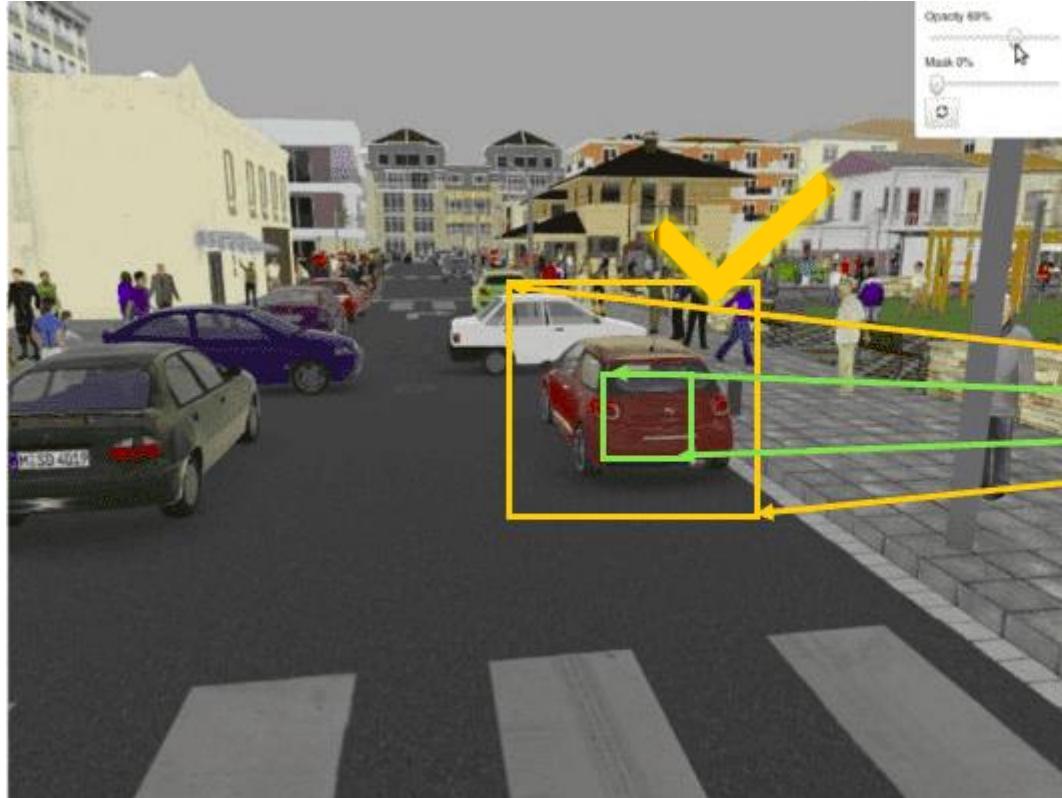
Receptive Field in Deep Learning



- Defined as a size of region in input that produces features. Basically, it is a measure of association of an output feature (of any layer) to the input region (patch).
- The idea of receptive fields applies to local operations (i.e. convolution, pooling).
- A convolutional unit only depends on a local region (patch) of the input.
- That's why RF never referred on fully connected layers since each unit has access to all the input region.



Why do we care for Receptive Field?

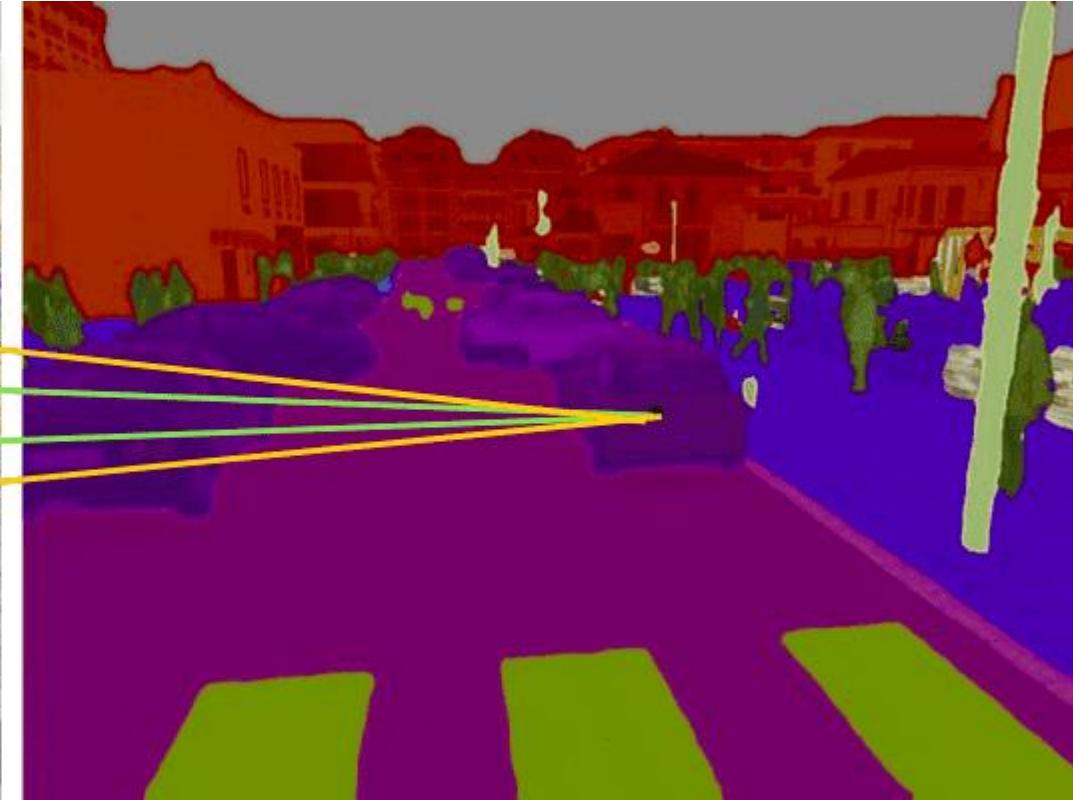
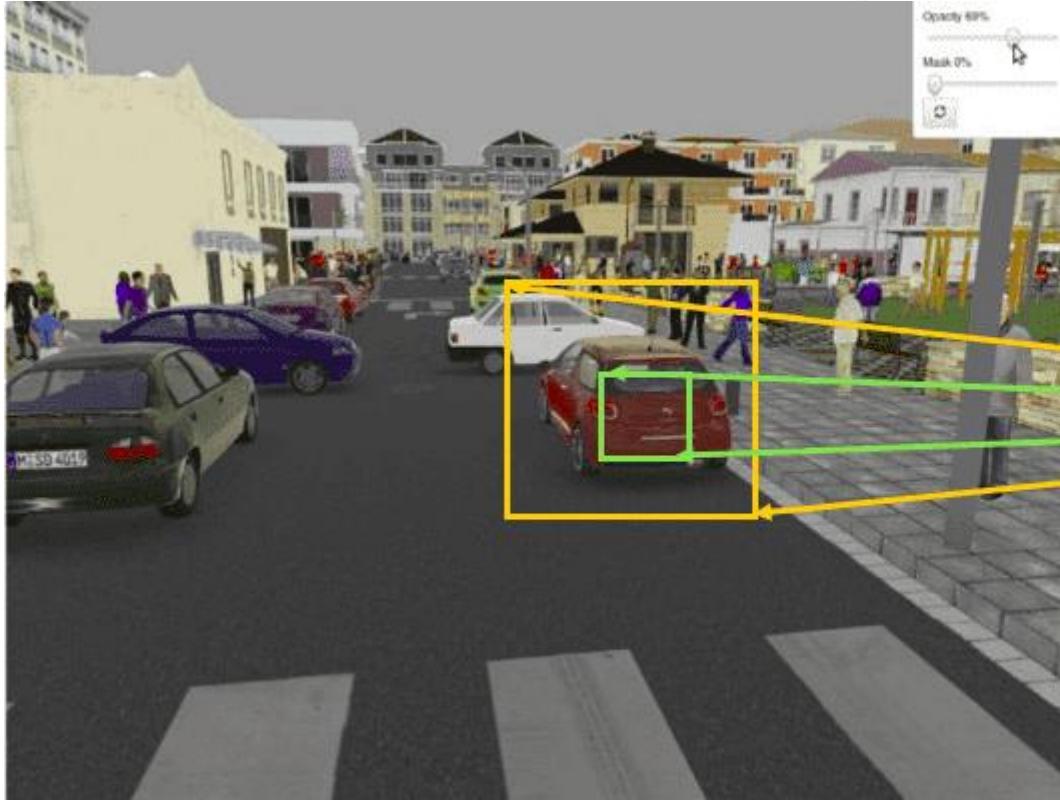


The green and the orange one. Which one would you like to have in your architecture?

Image Source: <https://developer.nvidia.com/blog/image-segmentation-using-digits-5/>



Why do we care for Receptive Field?



Therefore, our goal is to design a convolutional model so that we ensure that its RF covers the entire relevant input image region.

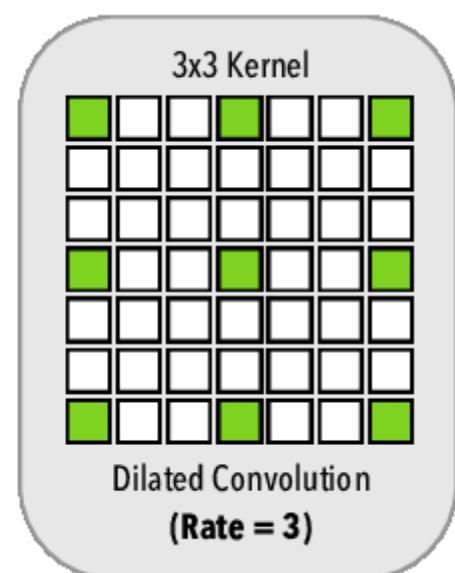
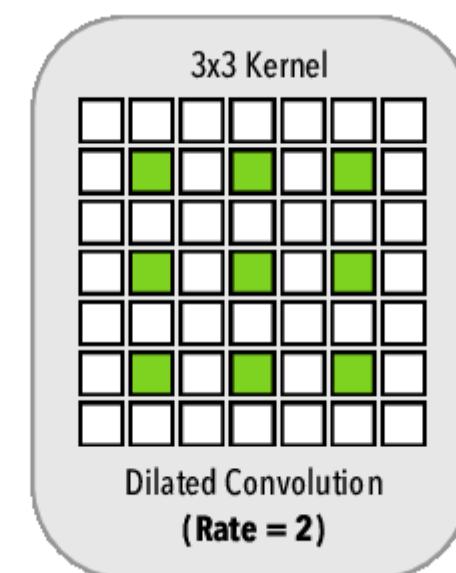
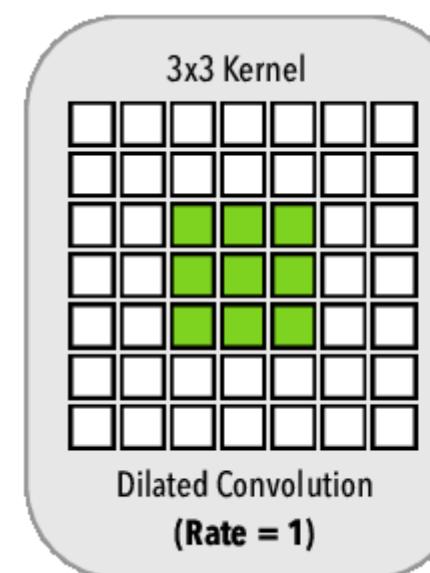
Image Source: <https://developer.nvidia.com/blog/image-segmentation-using-digits-5/>



How to increase receptive field in a convolutional network?

- Add more convolutional layers (make the network deeper)
- Add pooling layers or higher stride convolutions (sub-sampling)
- Use dilated convolutions

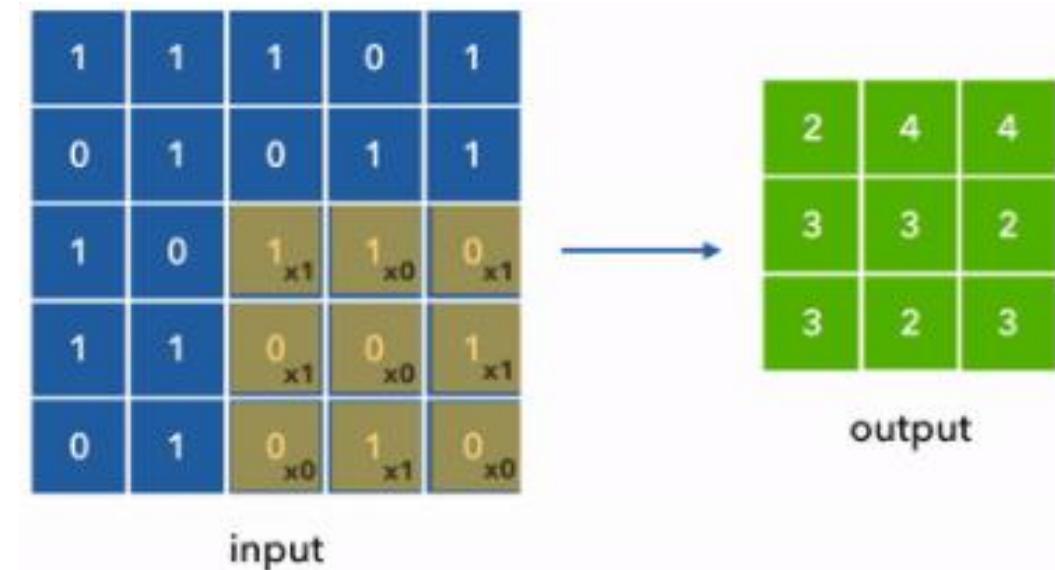
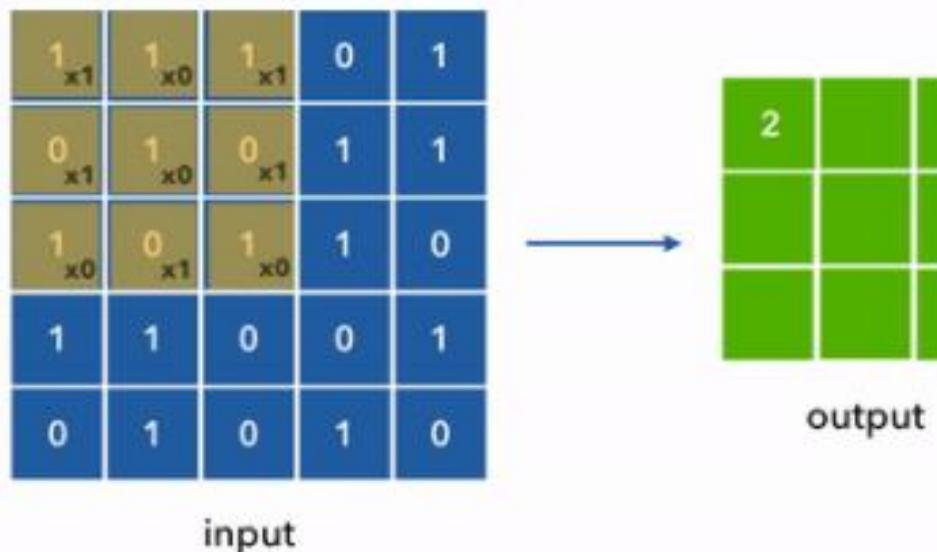
It is a technique that expands the kernel (input) by inserting holes between its consecutive elements.





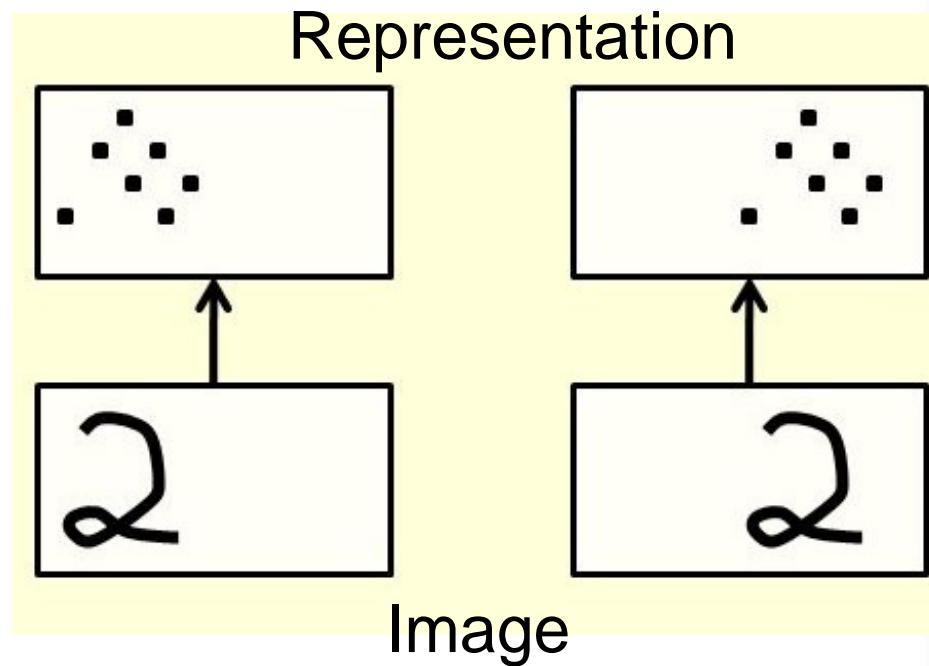
Parameter Sharing

- Parameter sharing refers to using the same parameter for more than one function in a model
- Kernel is reused (by sliding) when calculating the layer o/p
- Less weights to store & train





Equivalent Representation



- Parameter sharing causes the layer to have a property called **equivariance to translation**
- Convolution creates 2-D map of where certain features appear in the input
- **If we move the object in the input, its representation will move the same amount in the output**
- Eg: Same kernel for Edge Detection wherever the edge occurs in the image



1D, 2D Convolution

- 2D Convolution

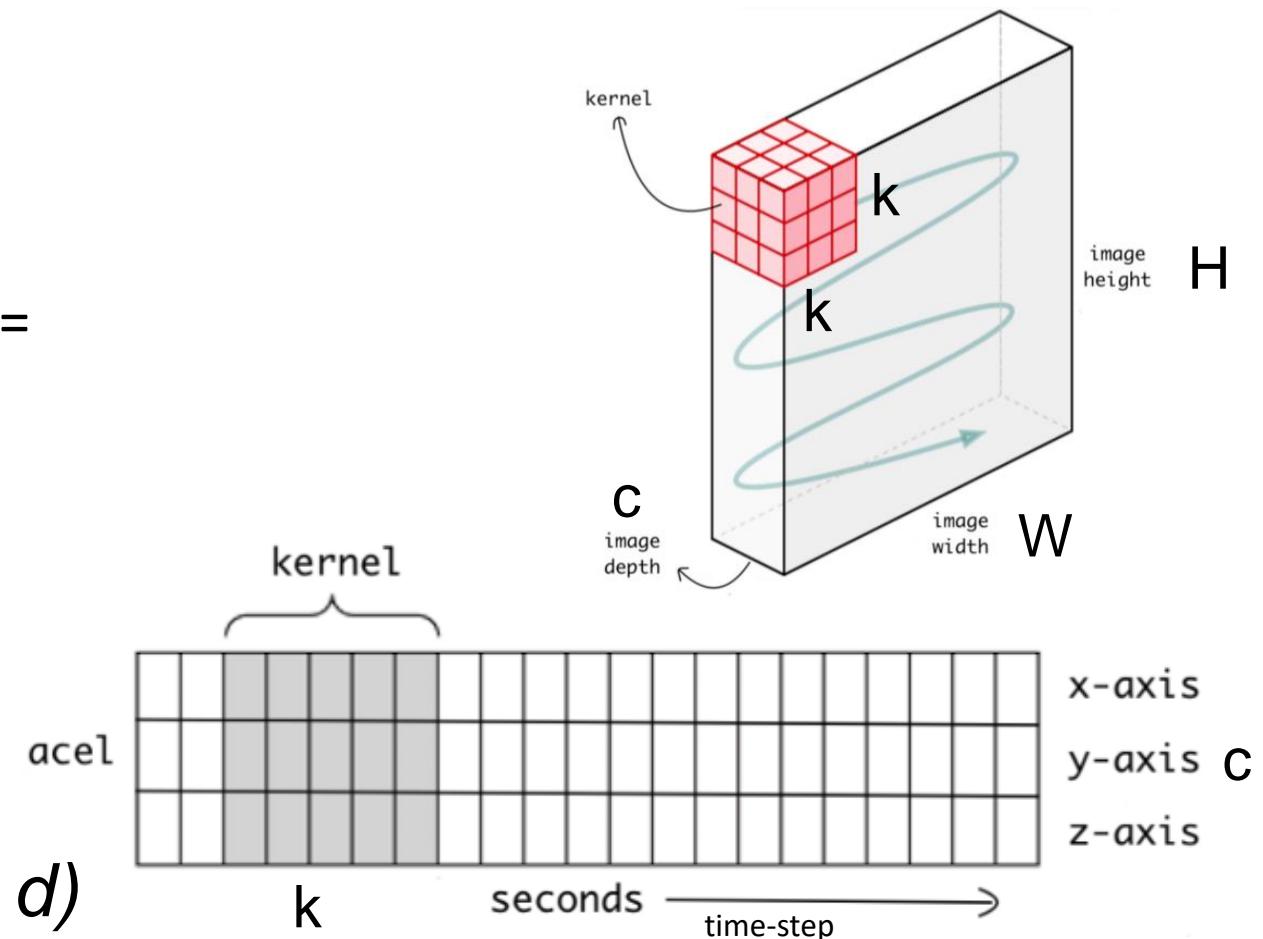
- 2-directions (x,y) to calculate conv
- input = $(W \times H \times c)$, d filters ($k \times k \times c$) output = $(W_1 \times H_1 \times d)$
- Eg: Image data (gray or color)

- 1D Convolution

- 1-direction (time) to calculate conv
- input = (time-step x c),

d filters ($k \times c$), output (time-step1 x d)

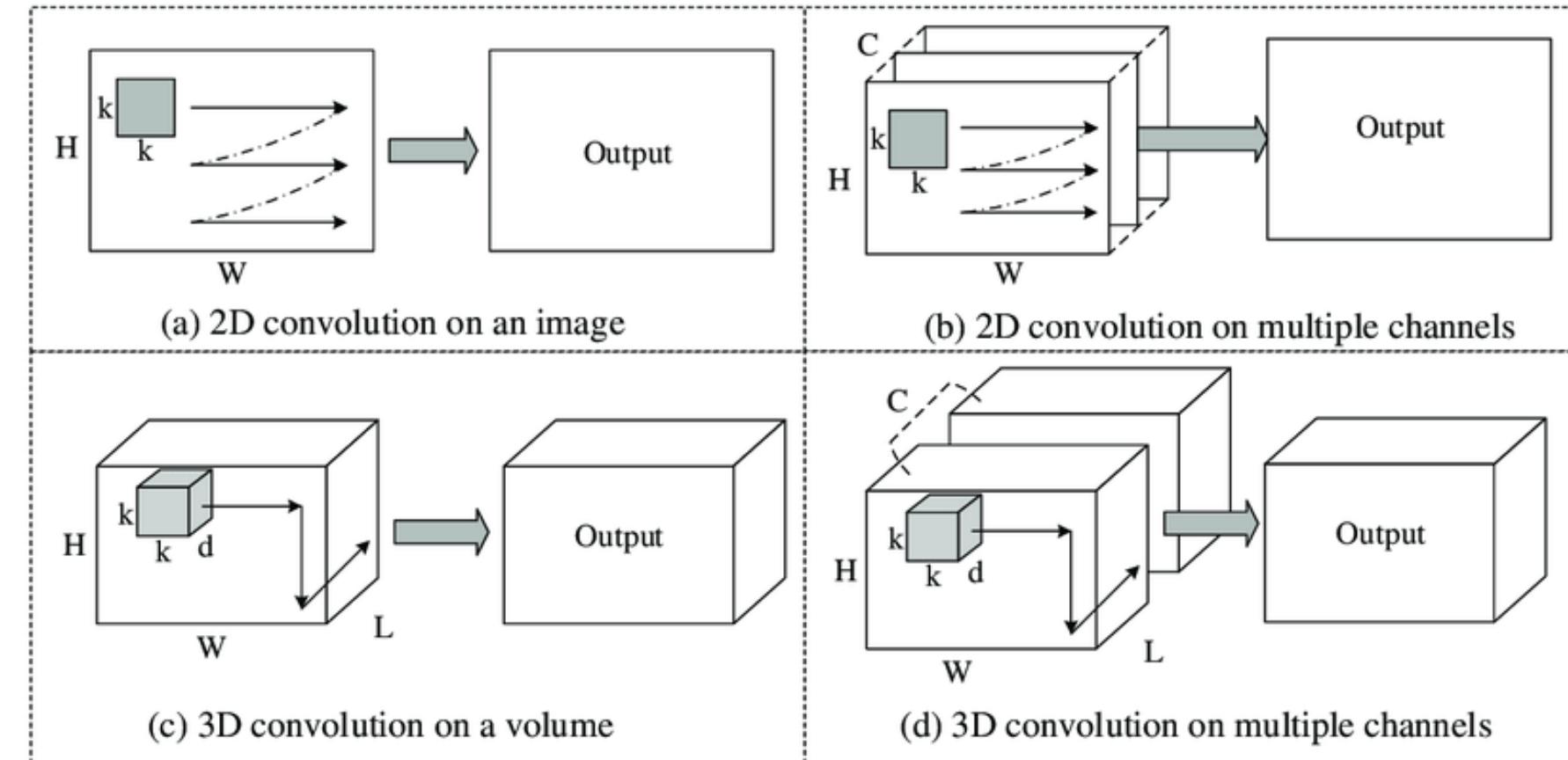
Eg: Time-series data, text analysis

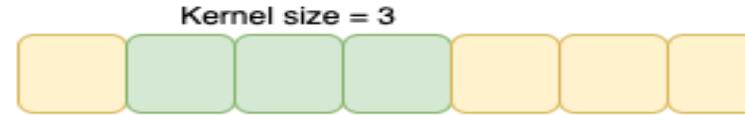




2D, 3D Convolution

- 3D Convolution
- 3-directions (x,y,z) to calculate conv
- input ($W \times H \times L \times C$), m filters ($k \times k \times d$) output ($W_1 \times H_1 \times L_1 \times m$)
- Eg: MRI data, Videos

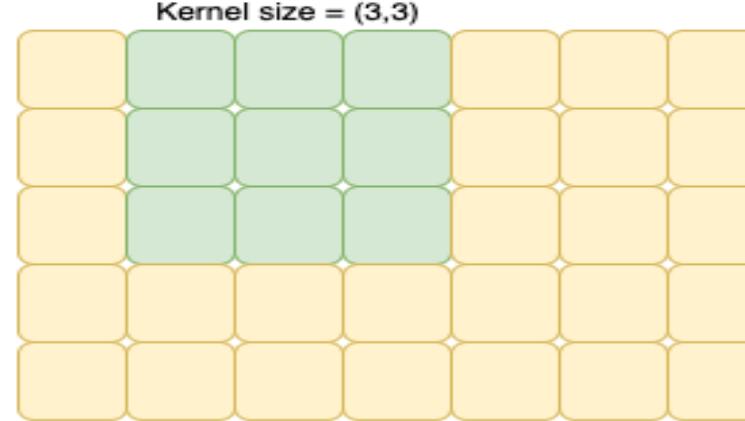


**1 D CNN****Input shape =2D**

Batch size =None

Width = Time axis =7

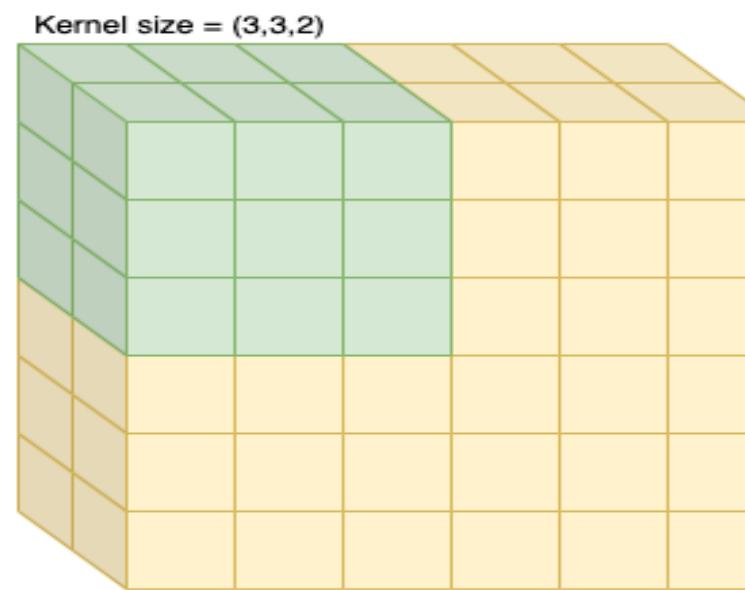
Feature map/ channels =1

2 D CNN**Input shape = 3D**

Height=5

Width = 7

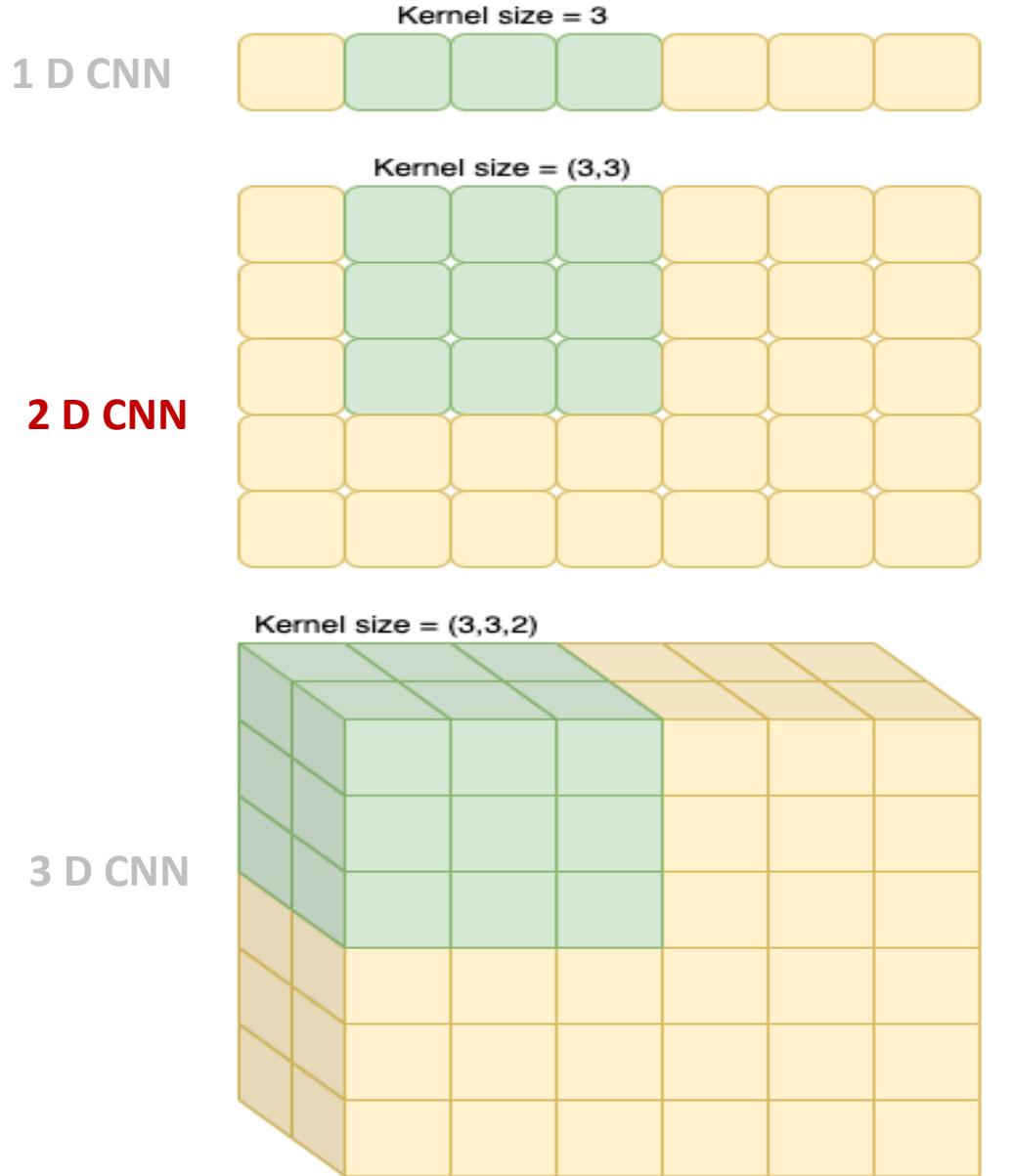
Feature map/ channels =1

3 D CNN**Input shape = 4D**

Height=6

Width = 6

Feature map/ channels
=depth=1



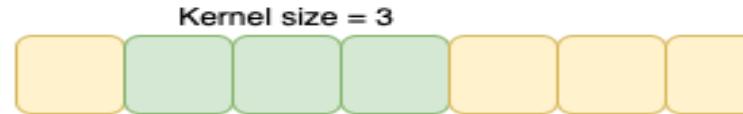
Input shape =2D
Batch size =None
Width = Time axis =7
Feature map/ channels =1

Input shape = 3D
Height=5
Width = 7
Feature map/ channels =1

Input shape = 4D
Height=6
Width = 6
Feature map/ channels =depth=1



1 D CNN



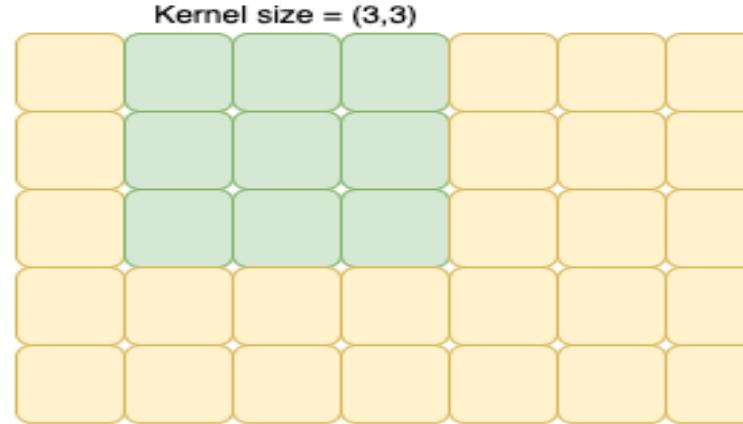
Input shape = 2D

Batch size = None

Width = Time axis = 7

Feature map/ channels = 1

2 D CNN



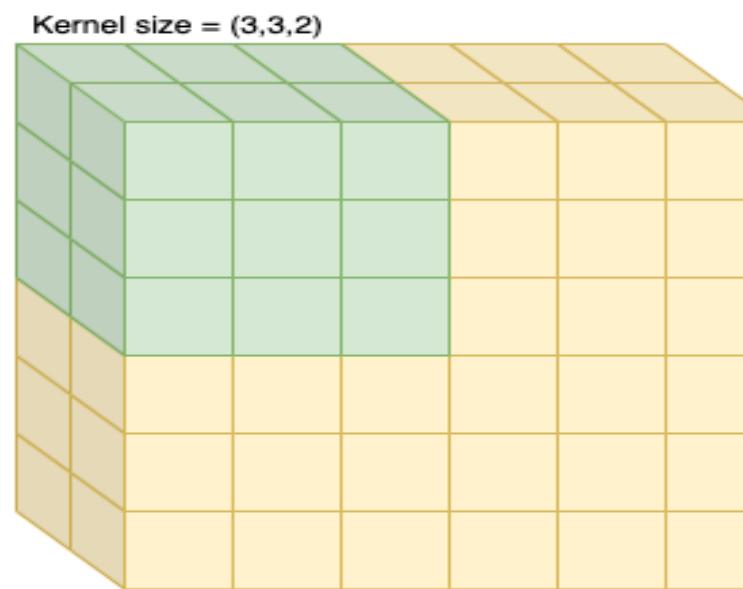
Input shape = 3D

Height=5

Width = 7

Feature map/ channels = 1

3 D CNN



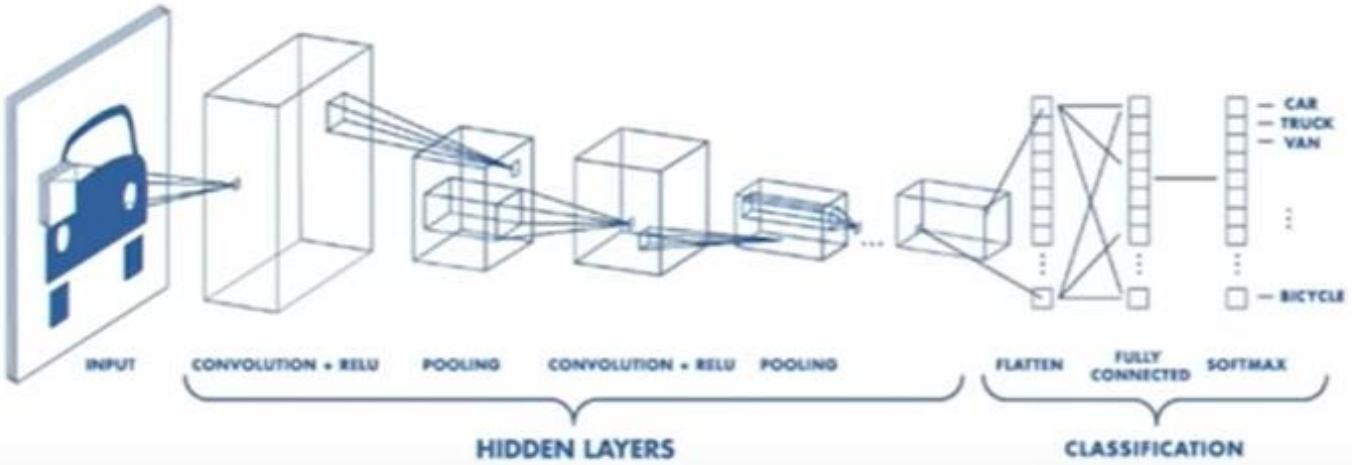
Input shape = 4D

Height=6

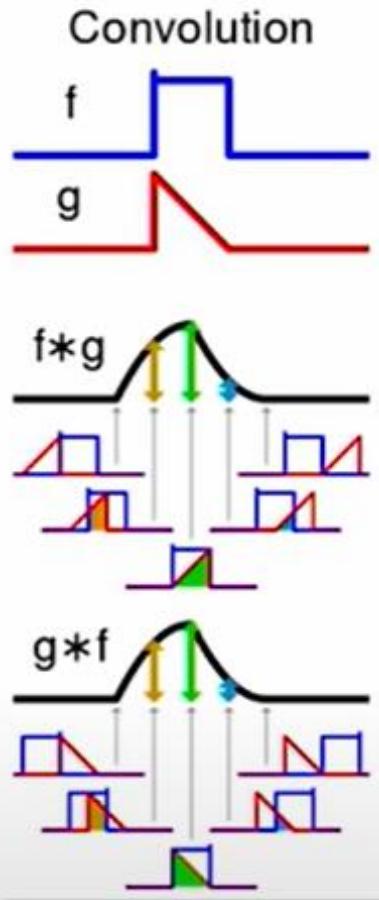
Width = 6

Feature map/ channels =depth=1

From 2D to 1D

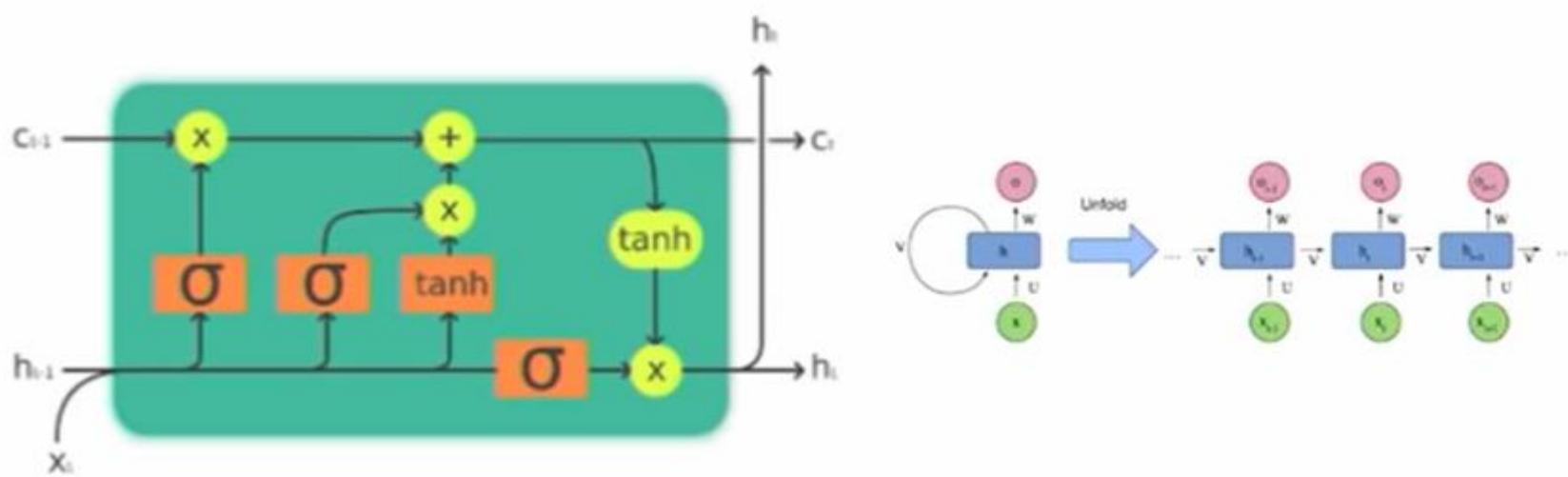


???



Motivation

- Deep learning and the new wave of neural networks are increasingly popular
- Focus is in the visual space for classification
- We are interested in time series forecasting
- Couldn't find as much modern work in this area
 - Sequence classification in language, text, audio
 - LSTM (long short-term memory), GRU (gated recurrent unit), RNN (recurrent NN)

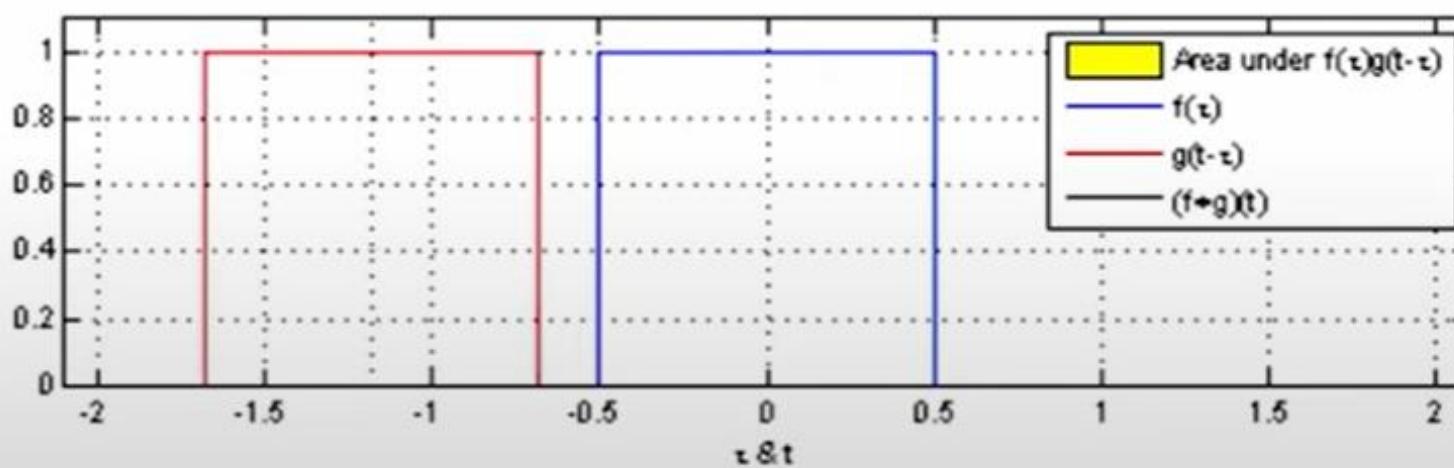


Discrete Time Signal Processing

- What about combining DSP with NNs?
 - Used in domains such as speech processing, sonar, radar, biomedical engineering, seismology
- Why not treat our hourly data as samples like one of these signals?
- 2D convolution works well for image classification
- Can 1D convolution work for time series forecasting?
- Had the idea to apply classic discrete time convolution techniques to 1D data...

Convolution

- Inspired by the convolution used in visual NNs (cross correlation)
- But instead use the definition of convolution used in signal processing
- It's the integral of the product of two functions after one is reversed and shifted

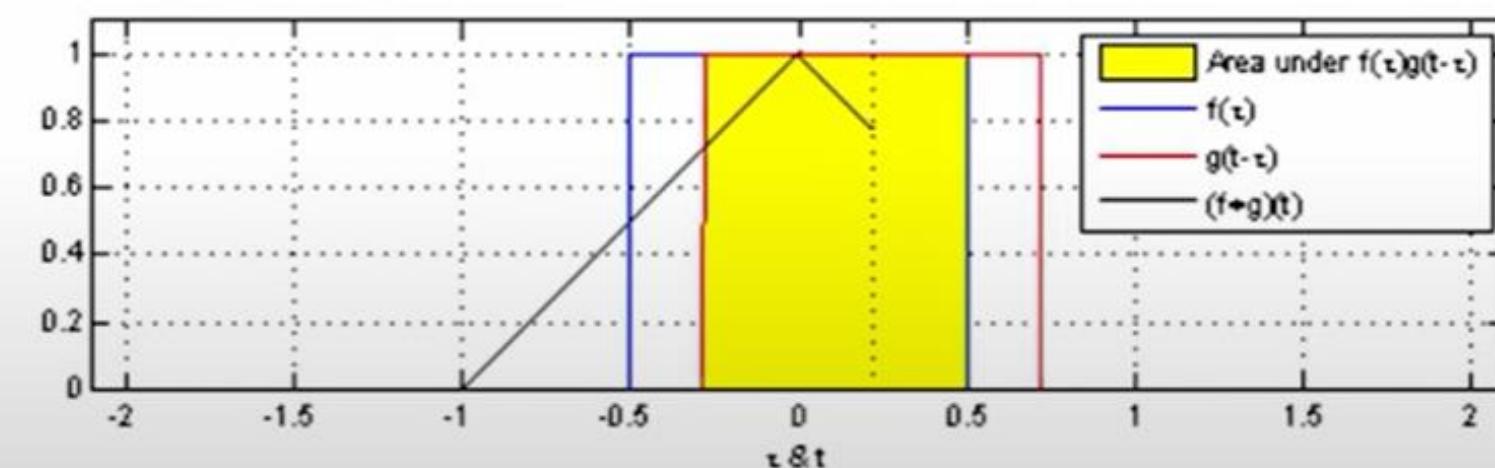


$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

Convolution

- Inspired by the convolution used in visual NNs (cross correlation)
- But instead use the definition of convolution used in signal processing
- It's the integral of the product of two functions after one is reversed and shifted

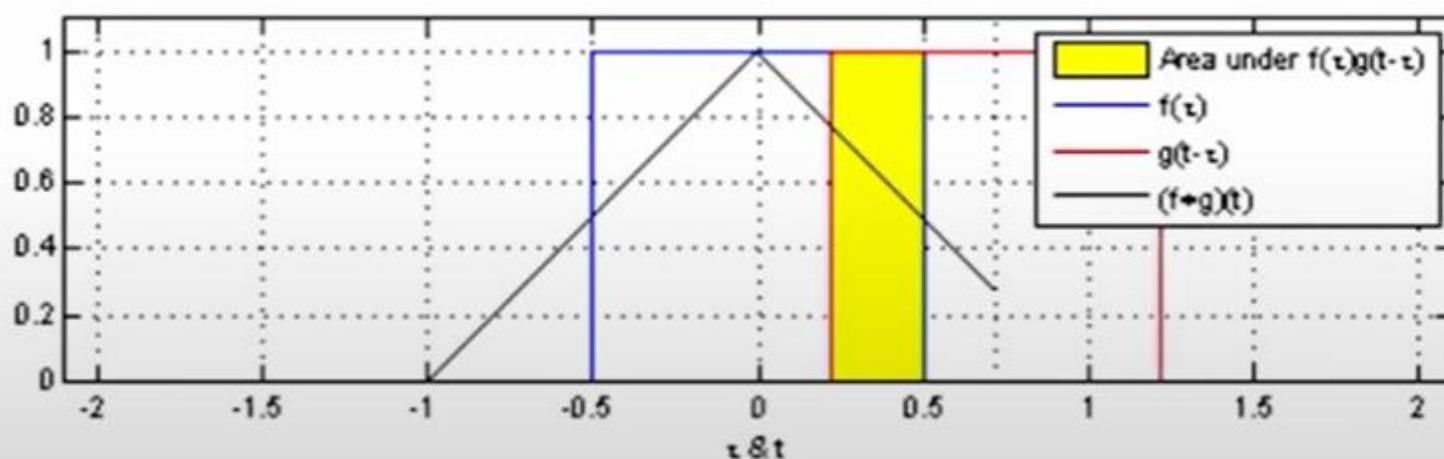


$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

Convolution

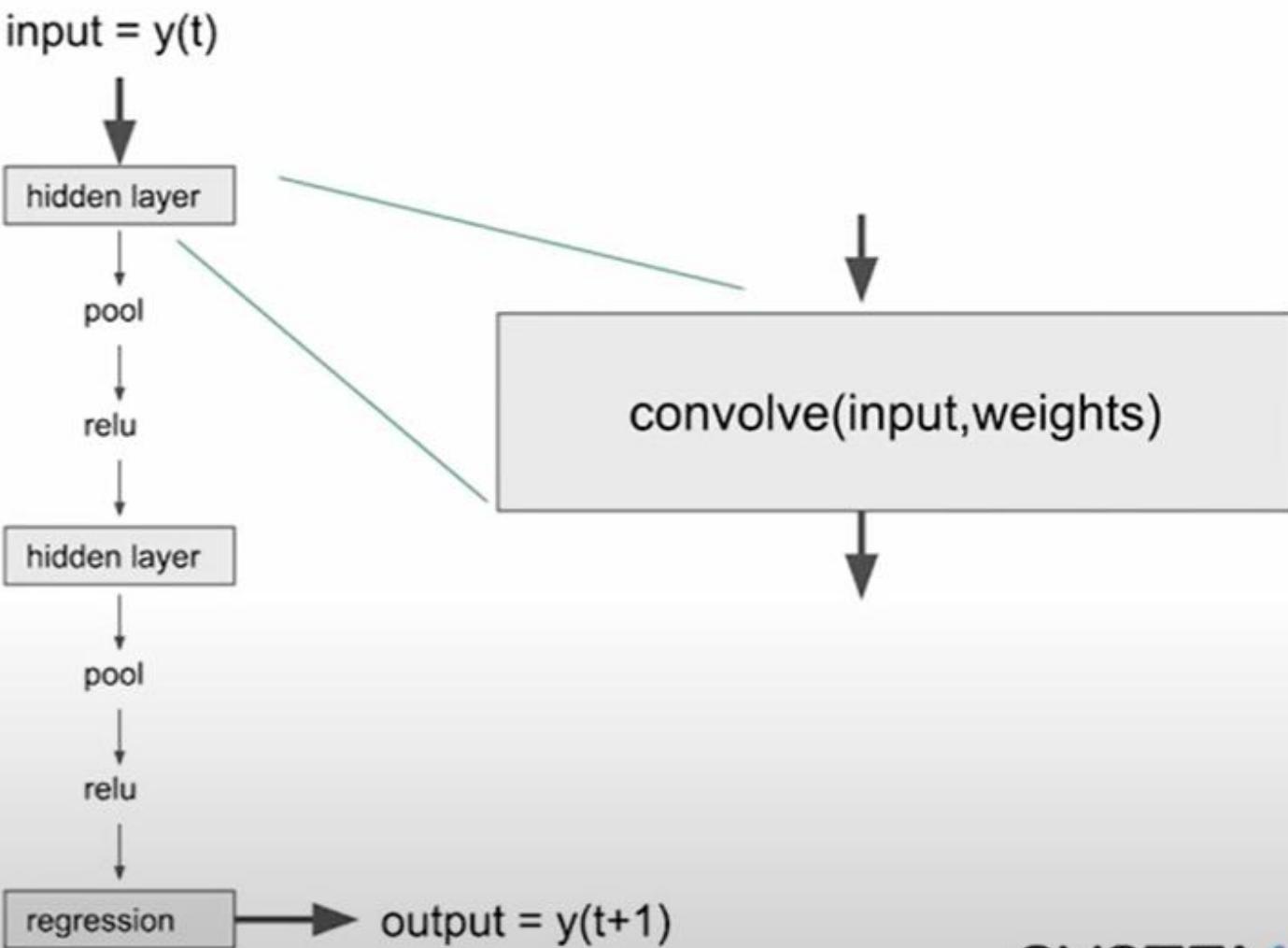
- Inspired by the convolution used in visual NNs (cross correlation)
- But instead use the definition of convolution used in signal processing
- It's the integral of the product of two functions after one is reversed and shifted



$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

Basic Architecture



Parameter Space Example

- One layer of filters has $n*(n + 1)/2 = 6(6-1)/2 = 15$ weights
- Two layers = $15 * 2 = \mathbf{30 \text{ filter weight parameters}}$
- Two layers deep and a window of 24 hours = each bottom filter output has $24/2/2 = 6$ values
- $6 \text{ filters} * 6 \text{ output values} = \mathbf{36 \text{ regression weight parameters}}$
- **66 total parameters**

- A network with 24 filters, 3 deep, running on week of hourly of data = **1404**

- <https://home.iitk.ac.in/~shalab/sptimeseries>
- <https://www.youtube.com/watch?v=0vtRKLVNhQ8> forecast error measures
- <https://otexts.com/fpp2/accuracy.html> forecast error measures
- <https://www.studocu.com/en-us/document/university-of-north-florida/operations-management/chapter-8-notes-part-3/17920362>