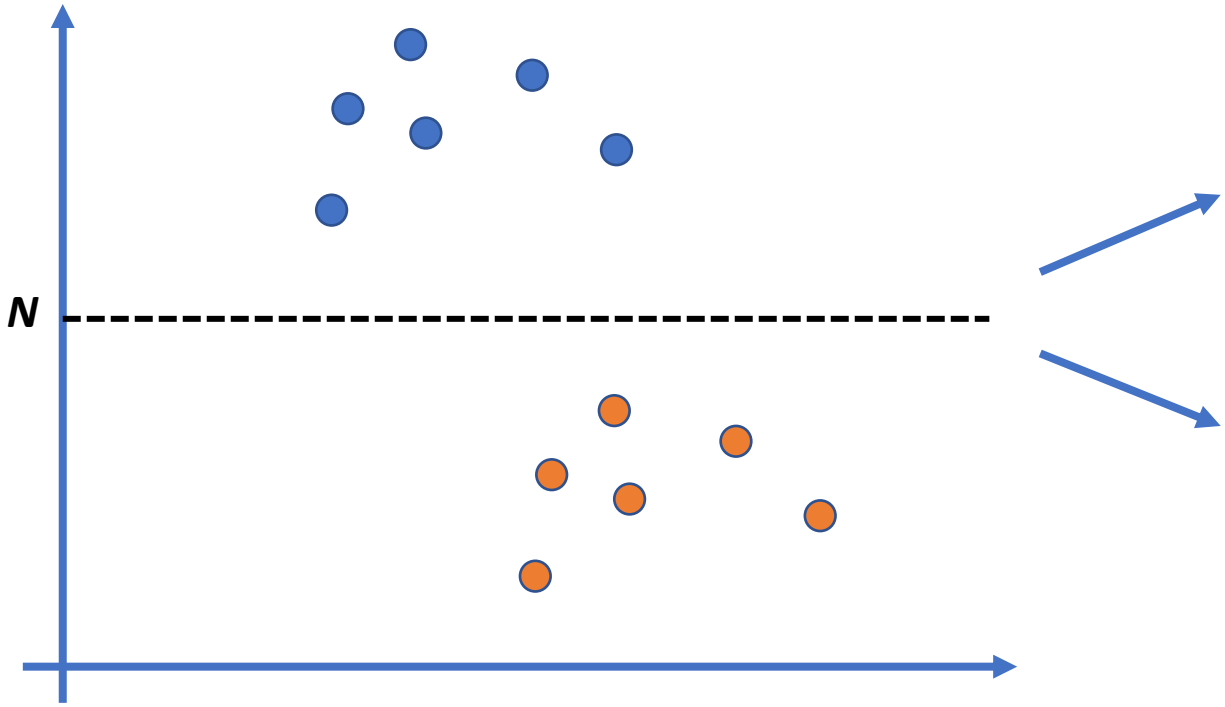
Introduction to Artificial Intelligence/Machine Learning

- Make machines think/work like humans
- Combination of Algorithms and Math
- This course is not a substitute for an ML class

Machine Learning

- **Computer algorithms that improve automatically through experience and using data**

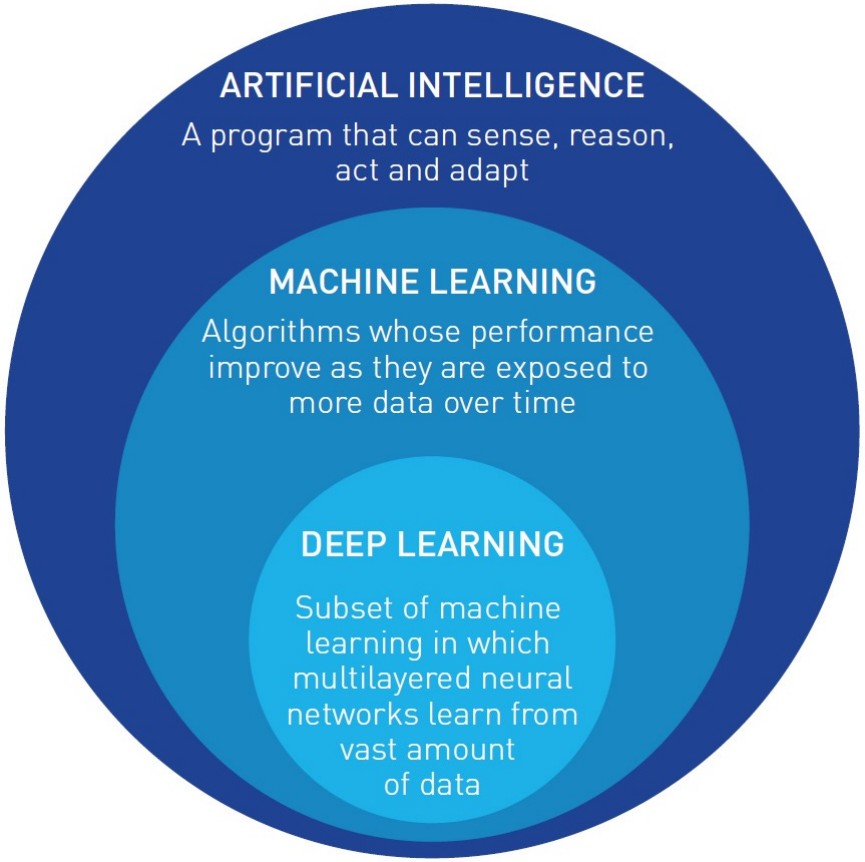


If y > *N*:
then blue
Else: orange

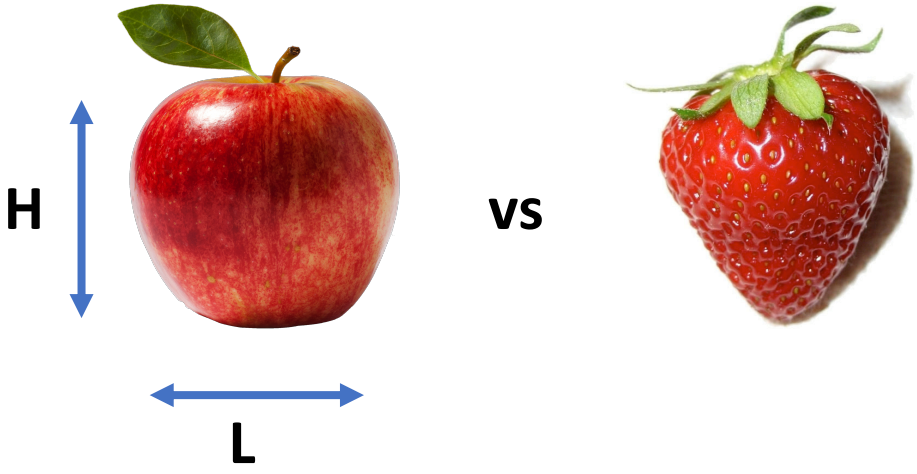
What is N?

ML vs AI vs DL?

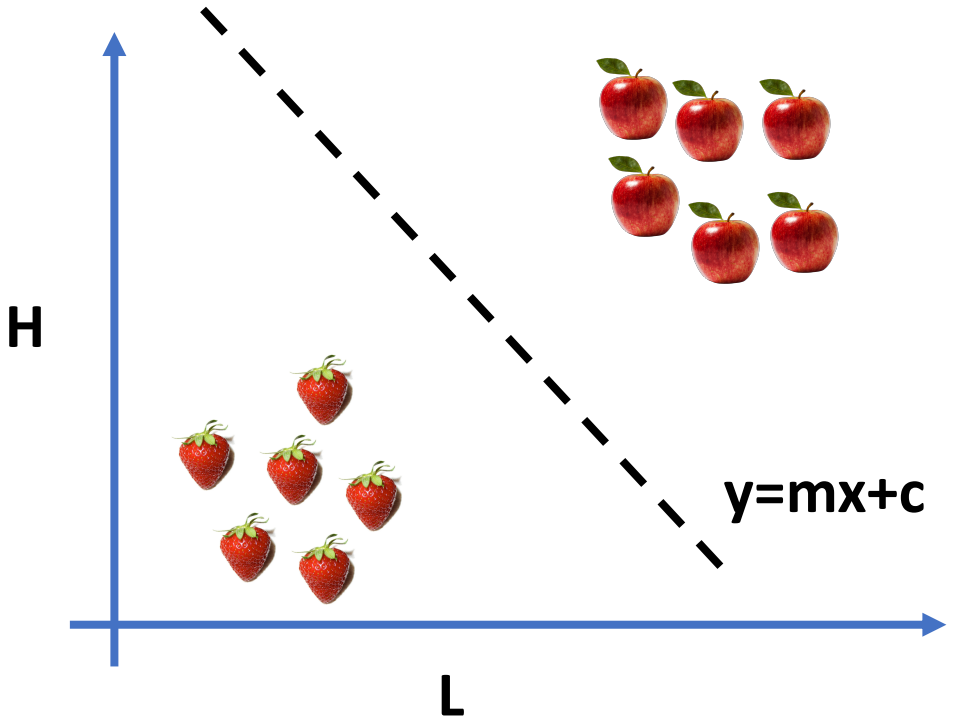
- Not the same as Artificial Intelligence (AI)
  - All ML are AI but not all AI are ML
  - Deep learning is a small subset of ML
- Chess
  - AI: Choose move based on position
  - ML: Learn to find the best move



Human intelligence

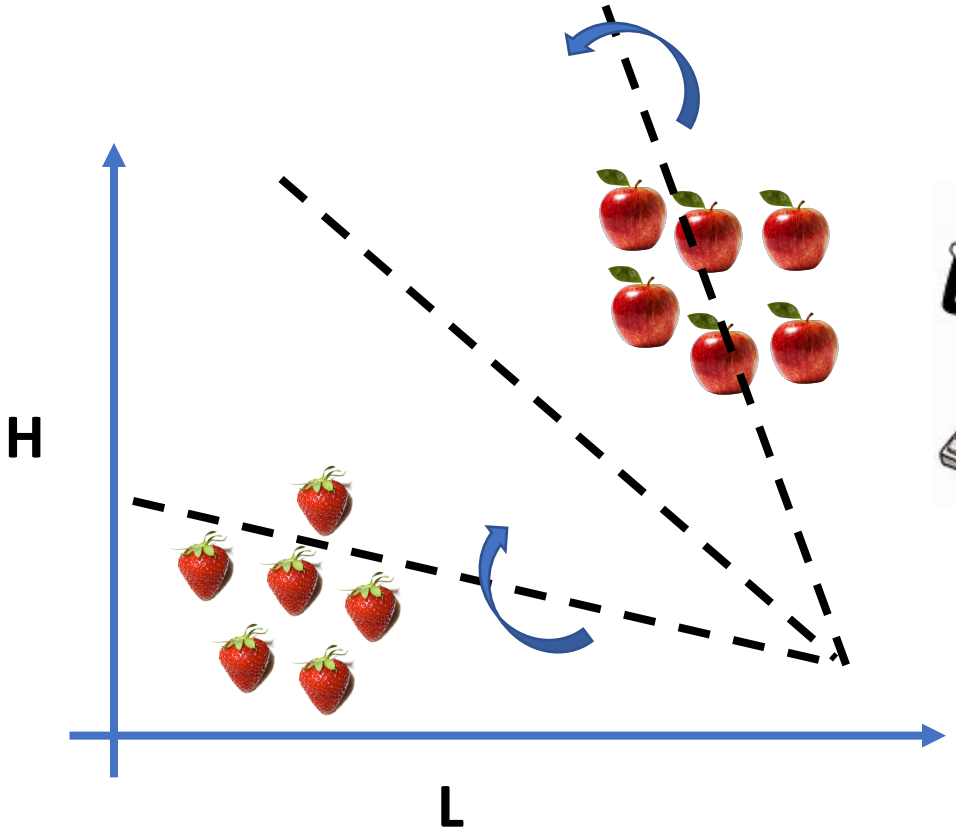


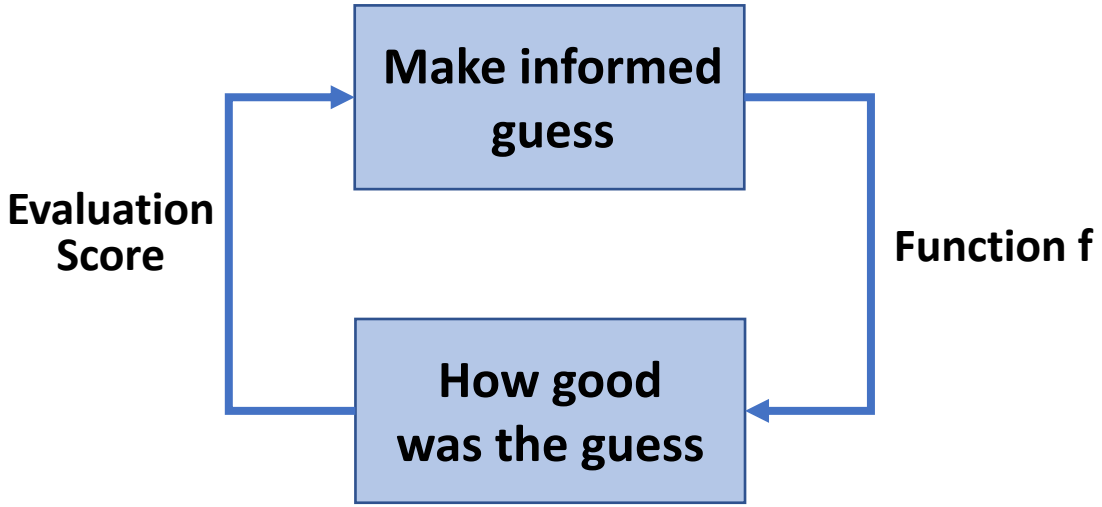
vs



- Classify apples vs strawberries
  - Given length and width only
- Easy task for humans
  - Derive a function using knowledge learnt from books, internet, etc.

Machine intelligence

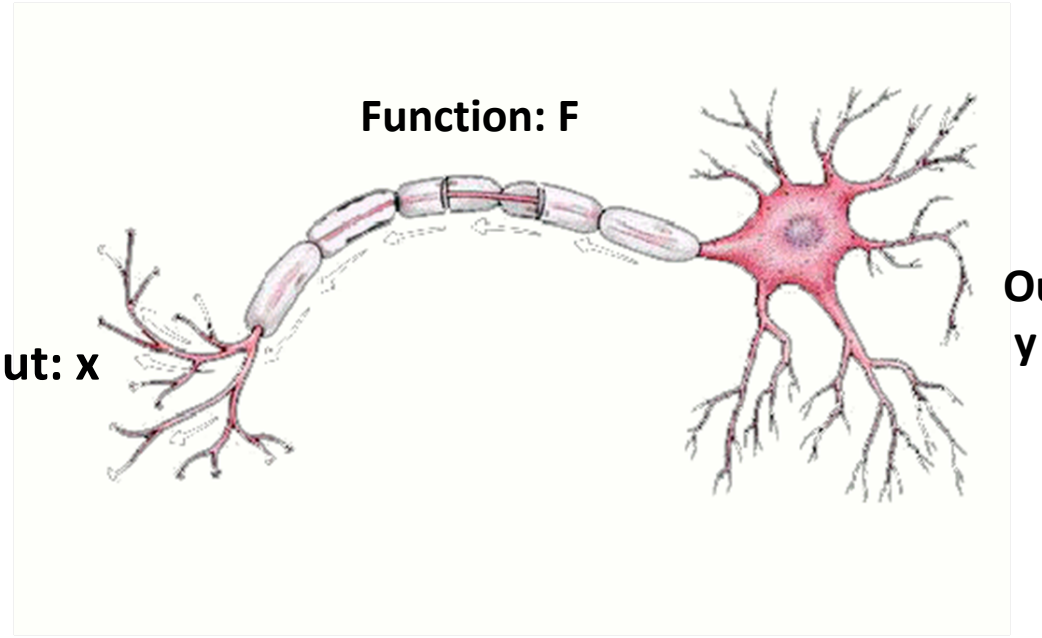




- Operates in a feedback loop
- Improve your guess using prior knowledge
- Loop till you get everything correct (or till convergence)

Human nerve cell (Neuron)

Input: x

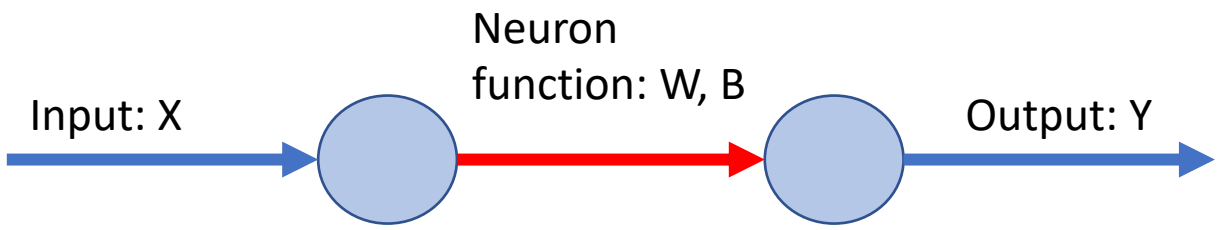


Output: y
y = F(x)

- An extremely simplistic model of nerve function
- Nerve modulates the input to produce output
- ML model should mimic this behavior

Rosenblatt, F.. “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review* 65 6 (1958): 386-408 .

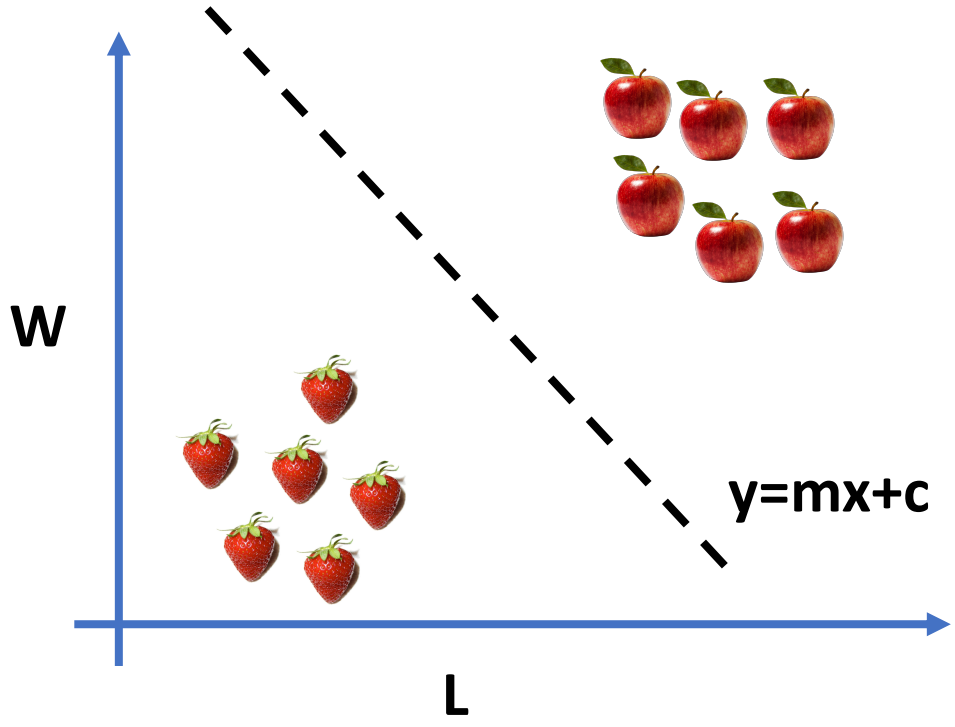
ML neuron: Perceptron



Perceptron: *Y = W.X + B* **(A straight line!!)**

W: weights, B: bias

- **Perceptron: Simplest ML model**
  - **It is a linear classifier**
  - **In 2D, it is a straight line**
  - **In 3D, it is a plane**
- **Can solve many simple problems**

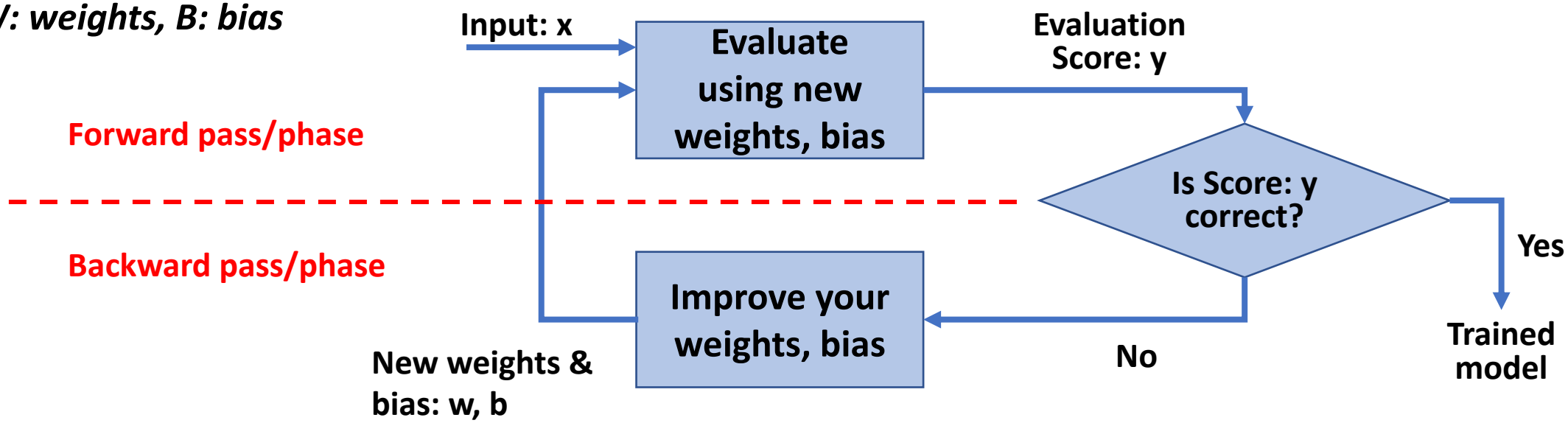


Rosenblatt, F.. “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review* 65 6 (1958): 386-408 .

Training Perceptron

Perceptron: *Y = W.X + B*

W: weights, B: bias

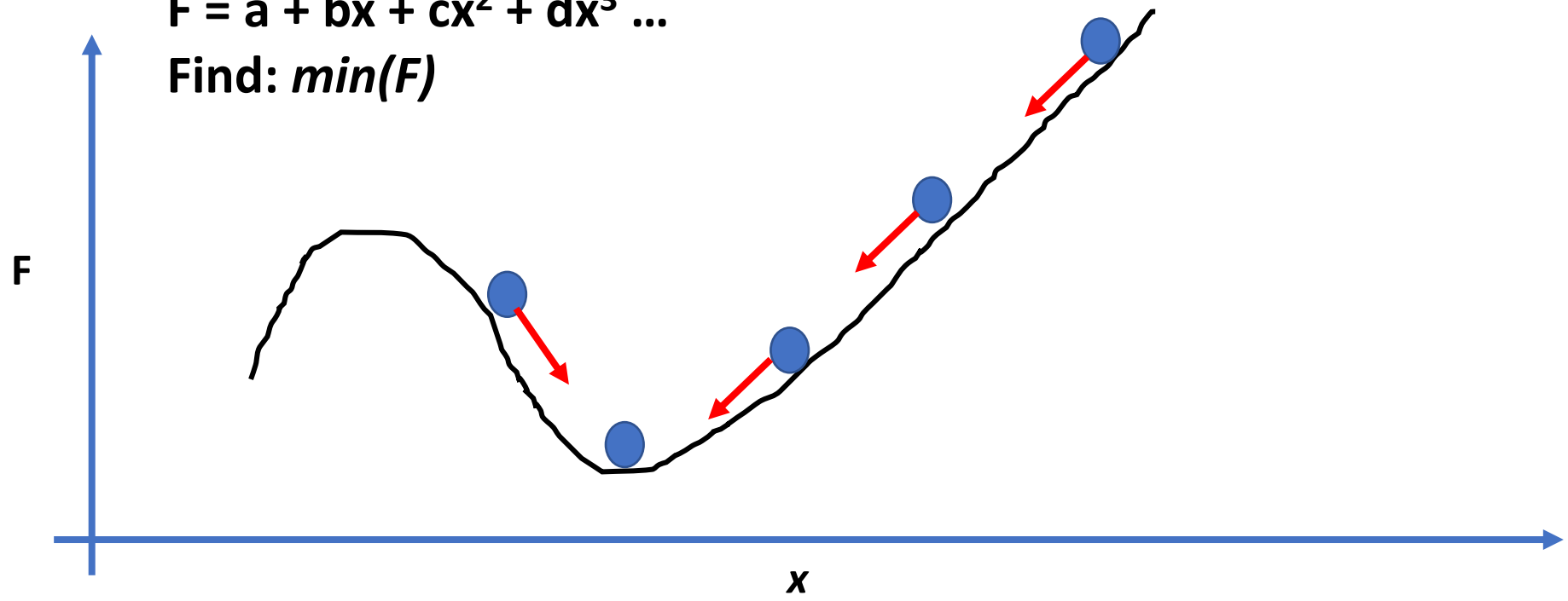


- Training has 2 phases: Forward and Backward
- Forward: Evaluate your function/model
- Backward: Improve your function/model if evaluation scores are unsatisfactory
- Repeat in loop till you get satisfactory evaluation scores

Gradient Descent Algorithm

$F = a + bx + cx^2 + dx^3 \ldots$

Find: *min(F)*

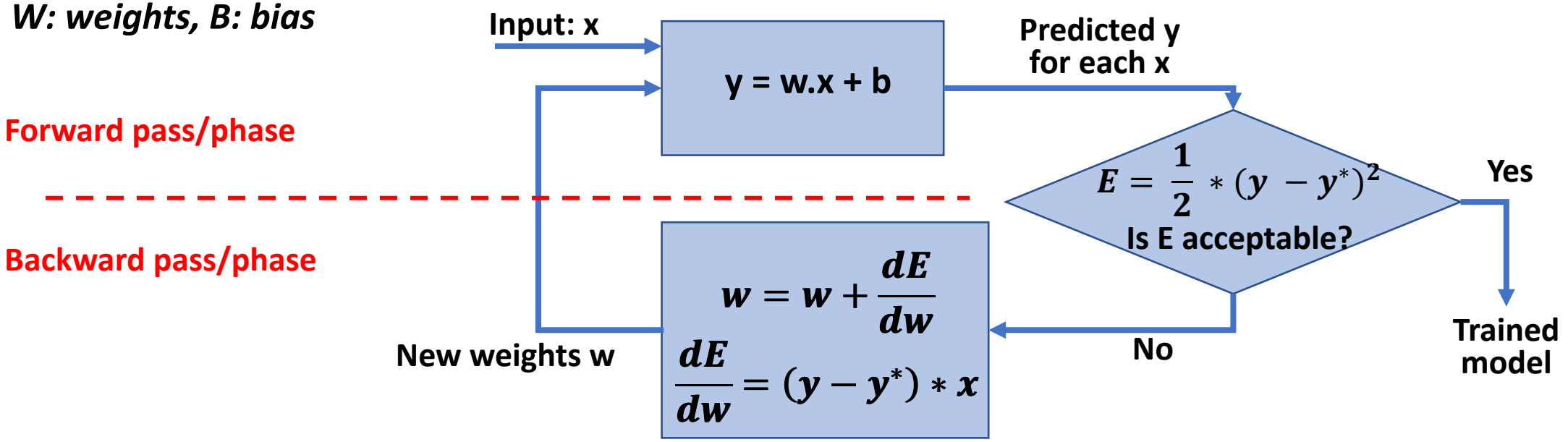


- To find minima, we can use differentiate F:
  - At minima, $\frac{dF}{dx} = 0$
  - Hard to solve
- Use “gradient descent” to move towards the minima

Gradient Descent: Perceptron

Perceptron: *Y = W.X + B*

W: weights, B: bias

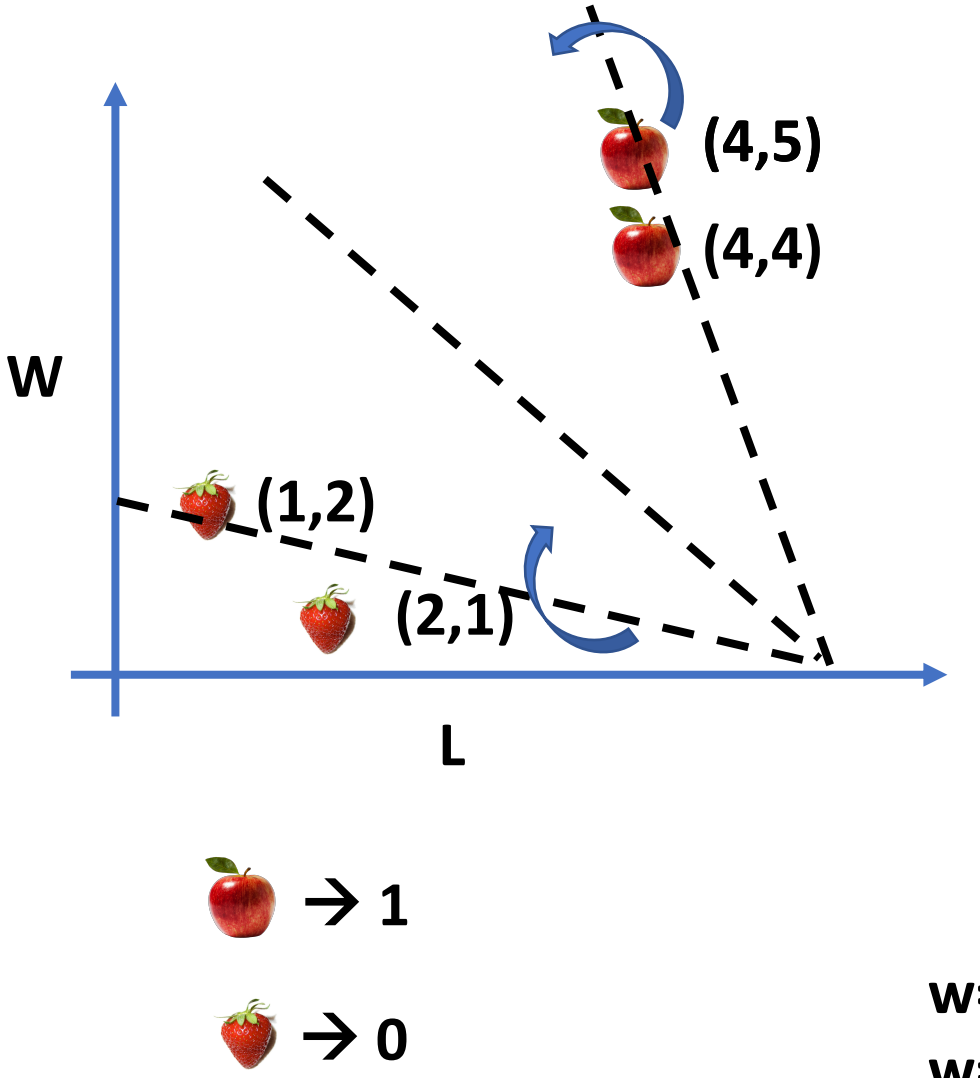


"Chain rule" of differentiation

$$\frac{dE}{dw} = \frac{dE}{dy} * \frac{dy}{dw} = (y - y^*) * x$$

- Repeat this till 100% accuracy or convergence

Perceptron example: classification



x=[4,5]

w=[1,0], b=0
y=w.x + b
y=0 if y<5, else 1

y=0, y*=1

$$E = \frac{1}{2} * (y - y^*)^2$$

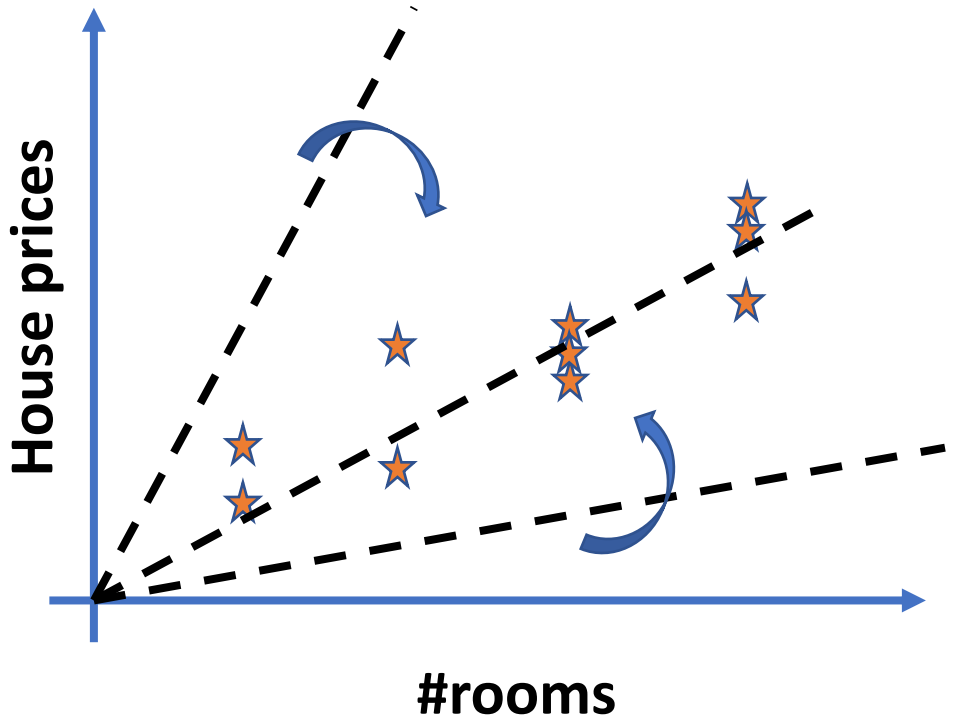
E=0.5

$$\frac{dE}{dw1} = -1 * 4$$

$$\frac{dE}{dw2} = -1 * 5$$

w=[1,0] +η *[-4,-5]
w= [-3, -5] (if η=1)

Perceptron example: regression

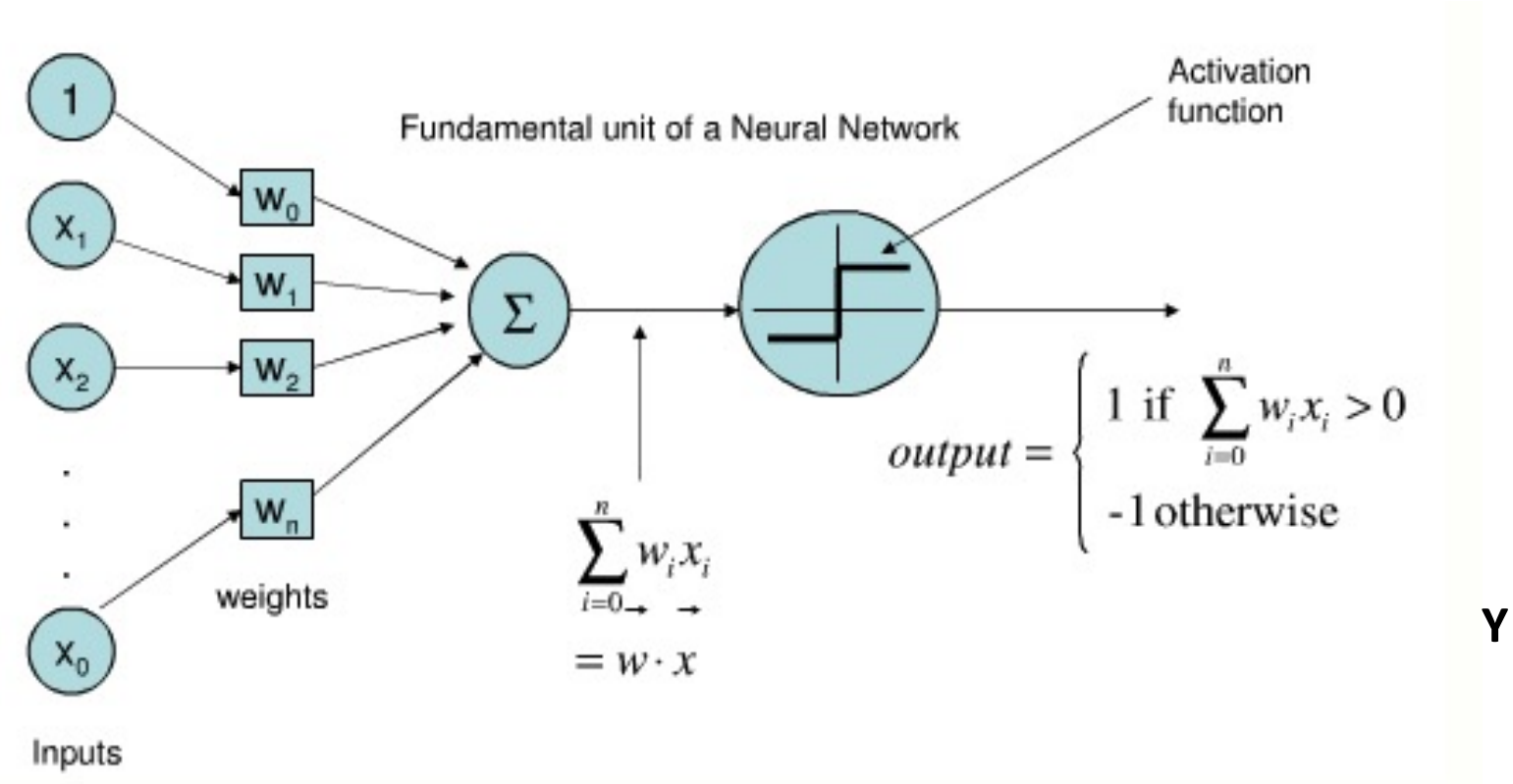


- Regression:
  - Fit a function given the data
  - *Predict the value for new data*
- Classification:
  - Learn a function that separates the data
  - *Predict the class for new data*

More Resources

- https://www.youtube.com/watch?v=PPLop4L2eGk&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN (Dr. Andrew Ng)
- https://www.youtube.com/watch?v=Gv9_4yMHFhI&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF
  - *Or just search for any machine learning tutorial on youtube*
- The elements of statistical learning (Book)

Perceptron: Summary



$$Y = \begin{bmatrix} X1 \\ X2 \\ \dots \\ Xn \end{bmatrix} * \begin{bmatrix} W1 \\ W2 \\ \dots \\ Wn \end{bmatrix} + B$$

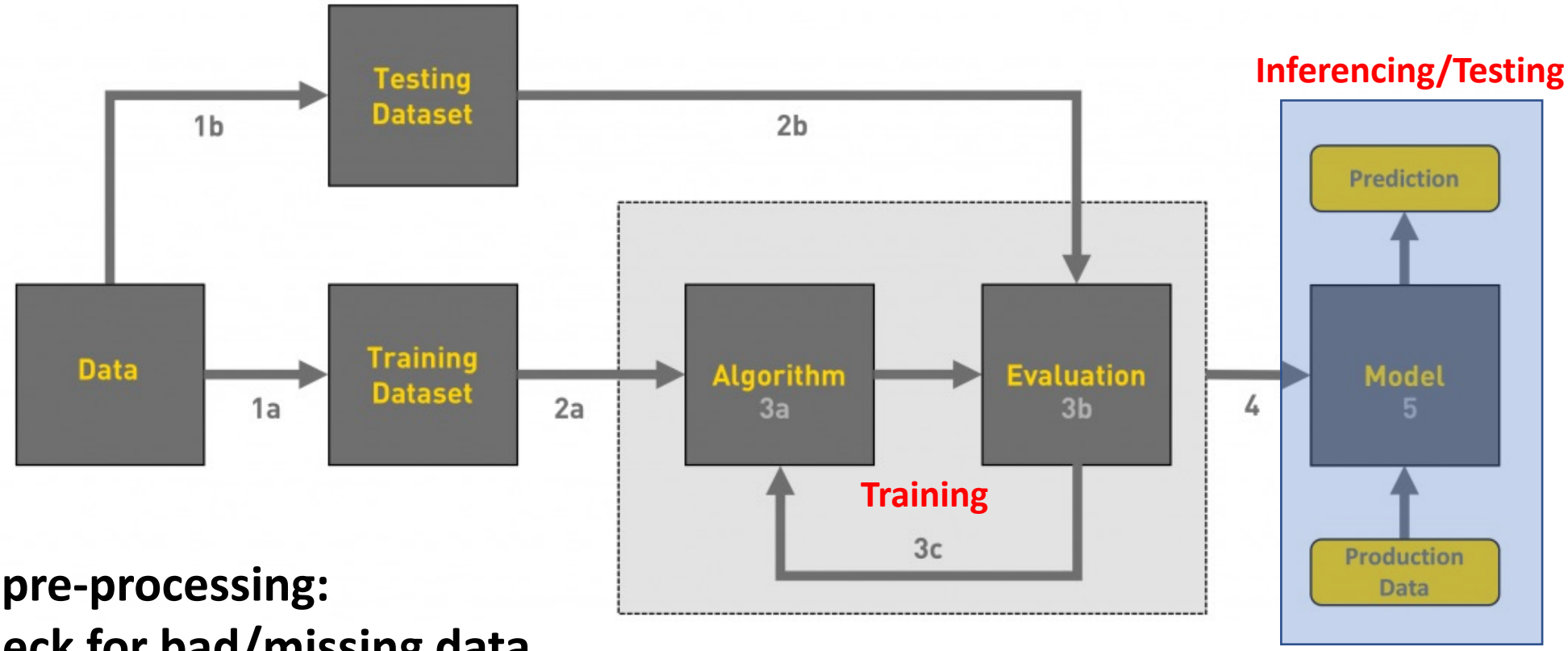
Y = W1*X1 + W2*X2 + … + Wn*Xn + B

$$\vec{w}^{t+1} \leftarrow \vec{w}^{t} + \sum_{\vec{x} \in \mathcal{Y}(\vec{x},\vec{w})} \eta^{t}(y - \hat{y})\vec{x}$$

Learning rate

$$\frac{dE}{dw}$$

Training ML model: Overall

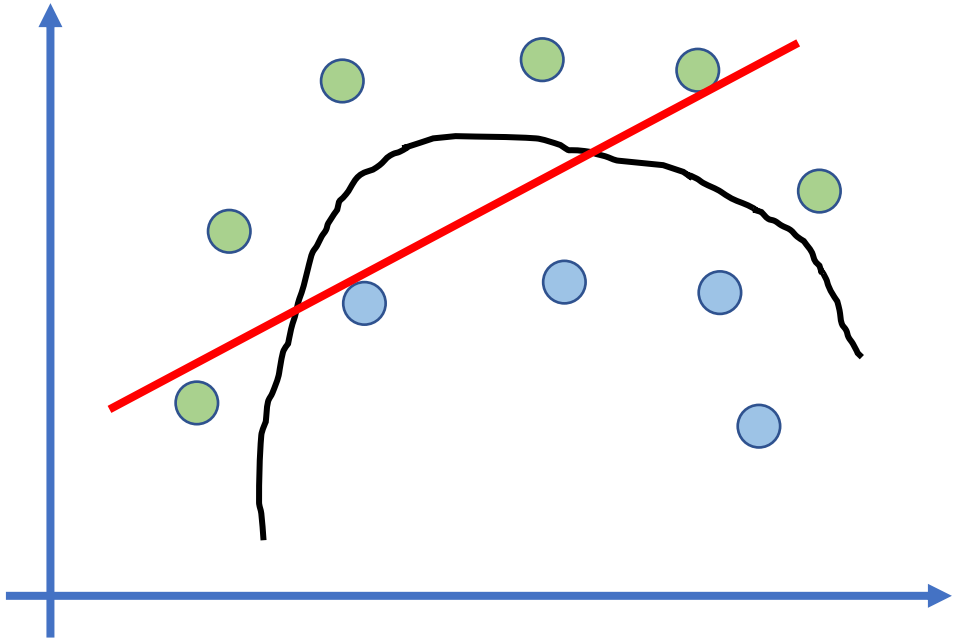


Data pre-processing:

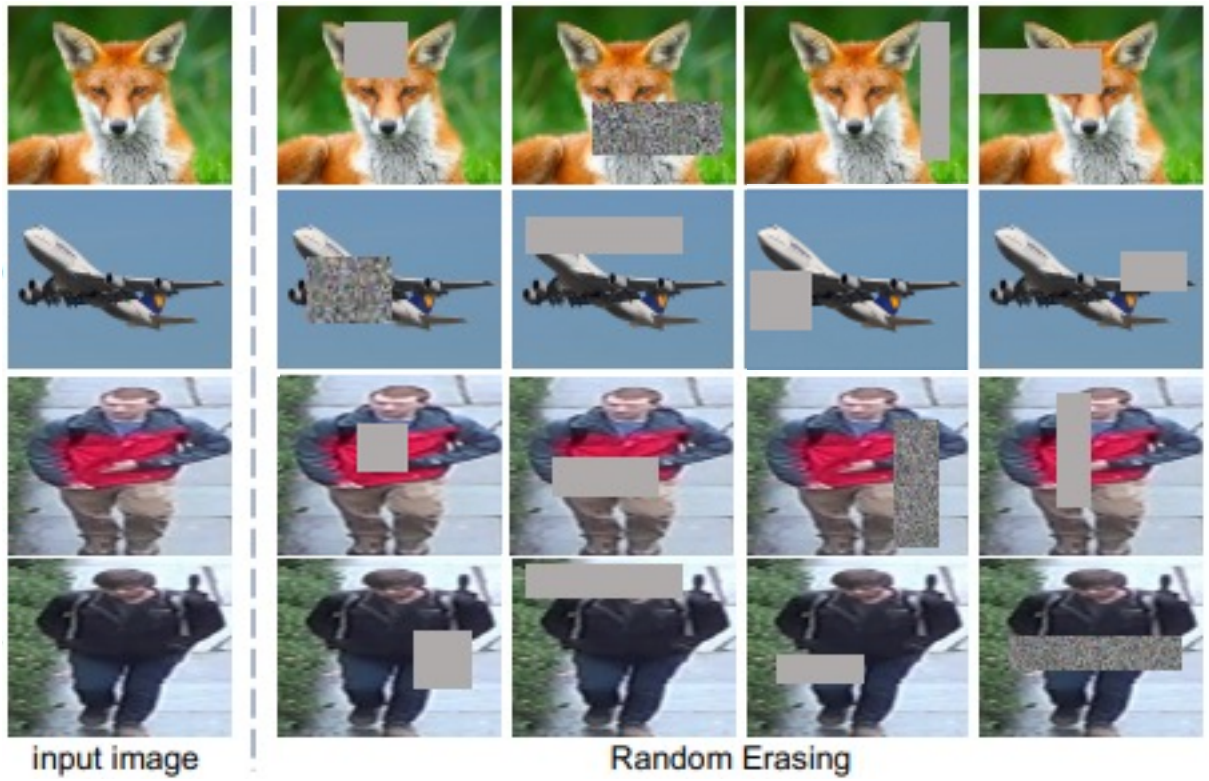
- **Check for bad/missing data**
- **Divide into training, testing, validation data**

Choosing the right model

- Data come in all shape/form
- No one size fits all in ML
- If data is non-linear, perceptron will fail
- Choose other models in that case
  - Better accuracy



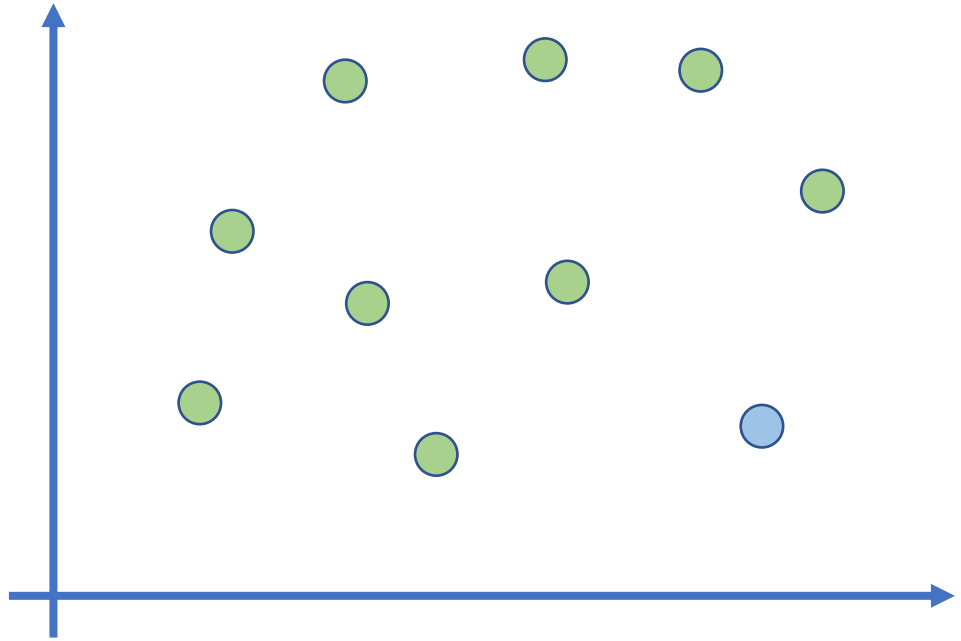
Data pre-processing



- **Model gets biased towards one type of image**
- **Important to expose it to different types**

Class imbalance

- Class imbalance
  - 90% data of type-1
  - 10% data of type-2
  - Model can cheat by always predicting type-1
    - Accuracy: 90%
- Spam/Fraud detection

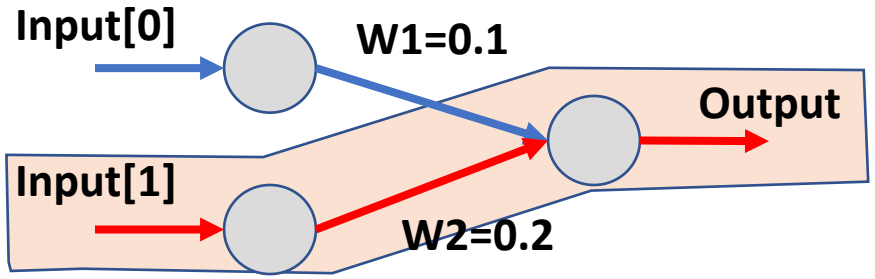


ML model can classify everything as green (9/10 correct, 90% accuracy)

Normalization

- Data can have wide range of values
- Large data can influence scores than smaller data
- Scale to uniform range

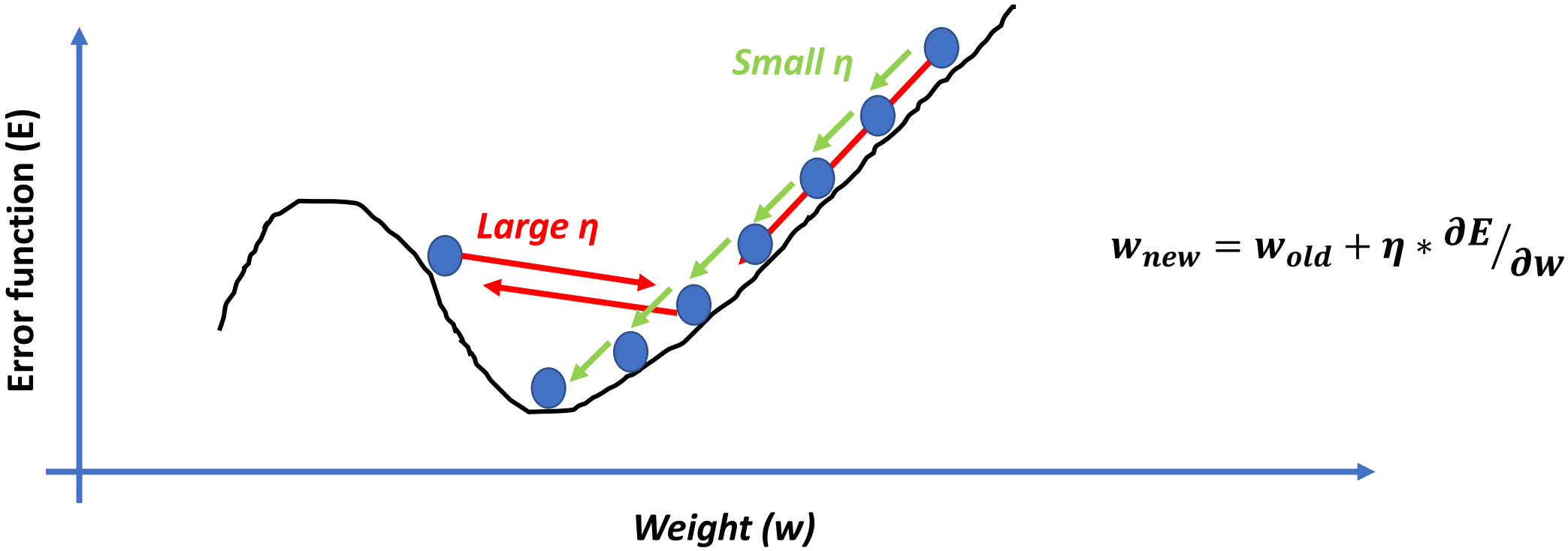
Input = [0.1, 10000]



Dominating input

Normalize/Scale data between 0-1

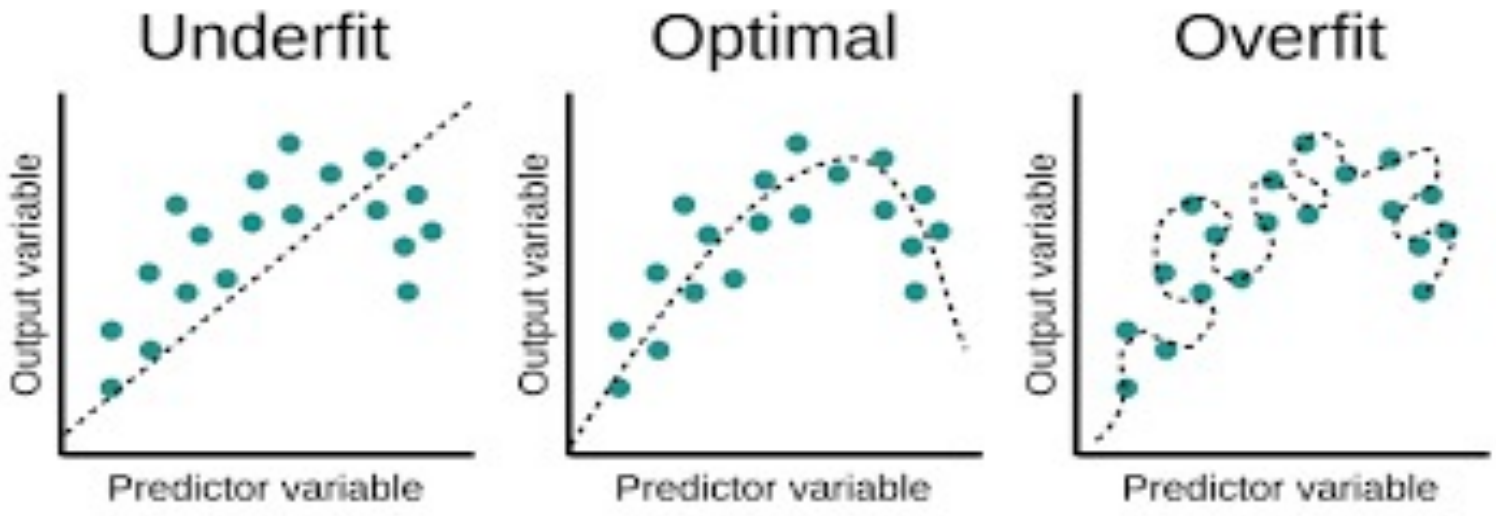
Learning rate



$$w_{new} = w_{old} + \eta * {\partial E}/{\partial w}$$

- Learning rate controls rate of convergence
  - Large: Chance of overshooting
  - Small: Takes too much time
  - *Hybrid strategies can be used*

Overfitting vs Underfitting



- **Underfitting**
  - **Both training and testing accuracy are bad**
  - **Model performing below its potential**
- **Overfitting**
  - **Training accuracy is good but testing accuracy is bad**
  - **Model learns useless features, starts to memorize**

Boston Housing dataset

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

- **Predict house prices using Perceptron**
  - **13 features**
  - **Normalize the inputs**
  - **Sklearn python library**

https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Sklearn library in Python

scikit-learn

Machine Learning in Python

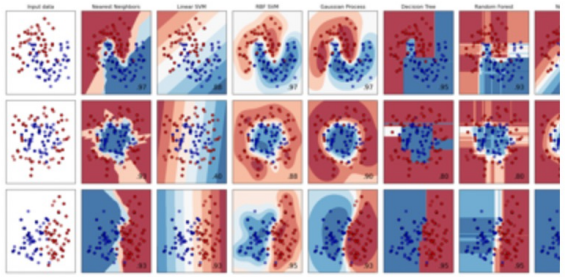
Getting Started | Release Highlights for 0.24 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

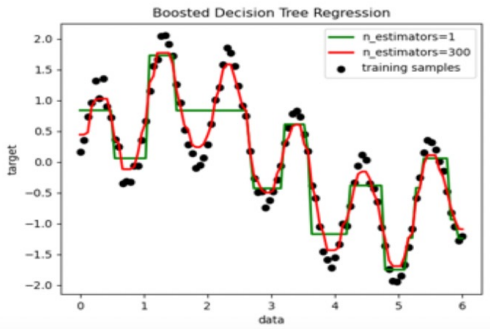
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

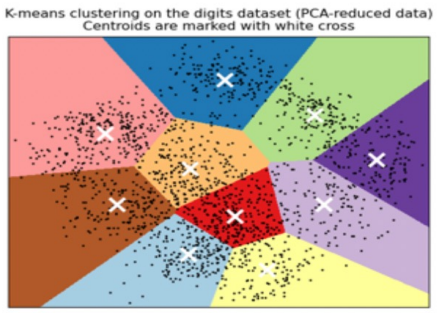
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Groupir experiment outcomes
Algorithms: k-Means, spectral clustering, mea shift, and more...



https://scikit-learn.org/stable/

Deep Learning

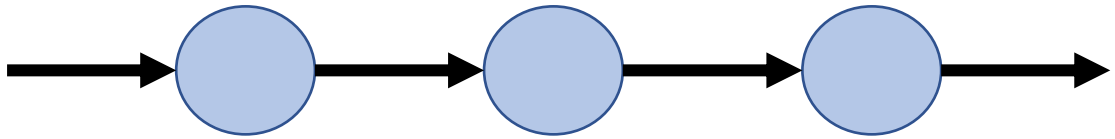
Deep Learning algorithms

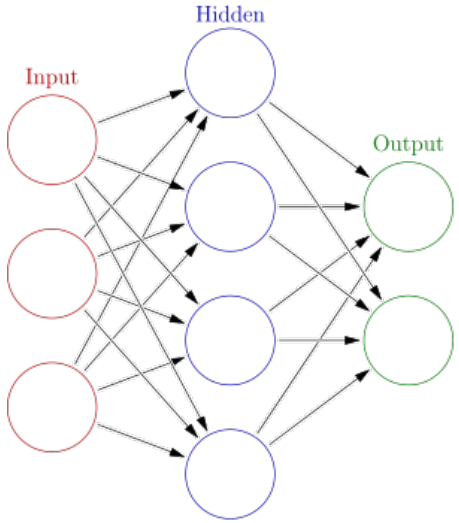
Types of Algorithms used in Deep Learning

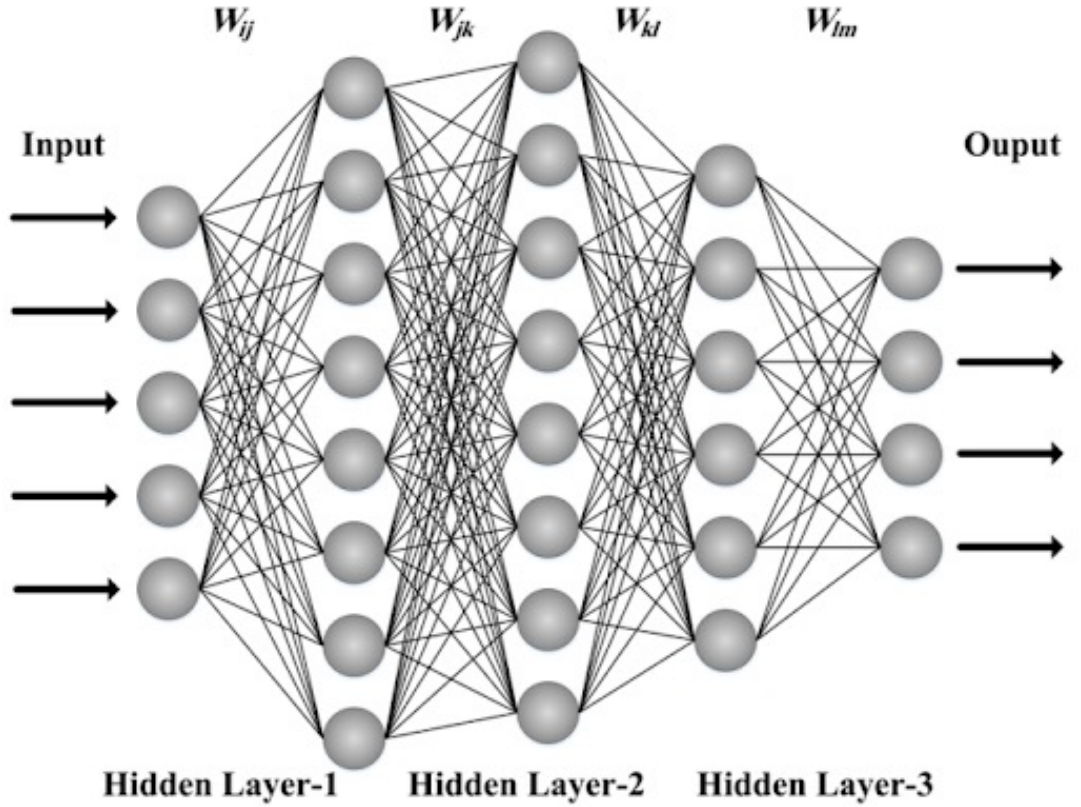
Here is the list of top 10 most popular deep learning algorithms:

1. Convolutional Neural Networks (CNNs)
2. Long Short Term Memory Networks (LSTMs)
3. Recurrent Neural Networks (RNNs)
4. Generative Adversarial Networks (GANs)
5. Radial Basis Function Networks (RBFNs)
6. Multilayer Perceptrons (MLPs)
7. Self Organizing Maps (SOMs)
8. Deep Belief Networks (DBNs)
9. Restricted Boltzmann Machines(RBMs)
10. Autoencoders

Multi-Layer Perceptron (MLP)



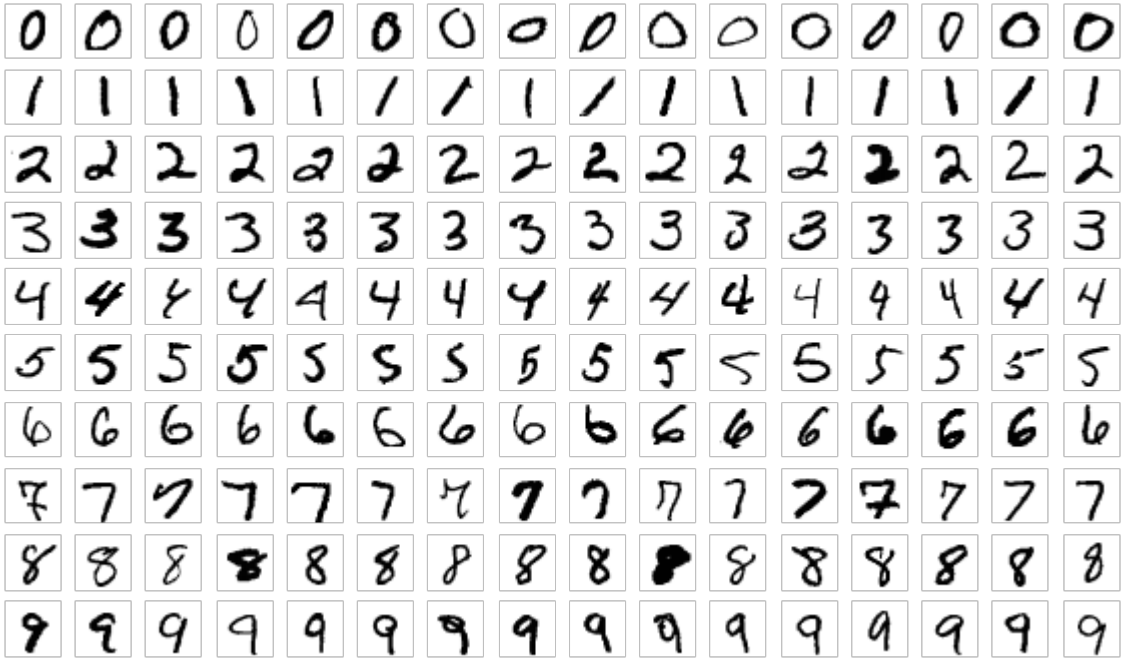




- MLPs popularly used in Deep Learning (DL)
- Can represent complex features
  - Handwritten digits recognition

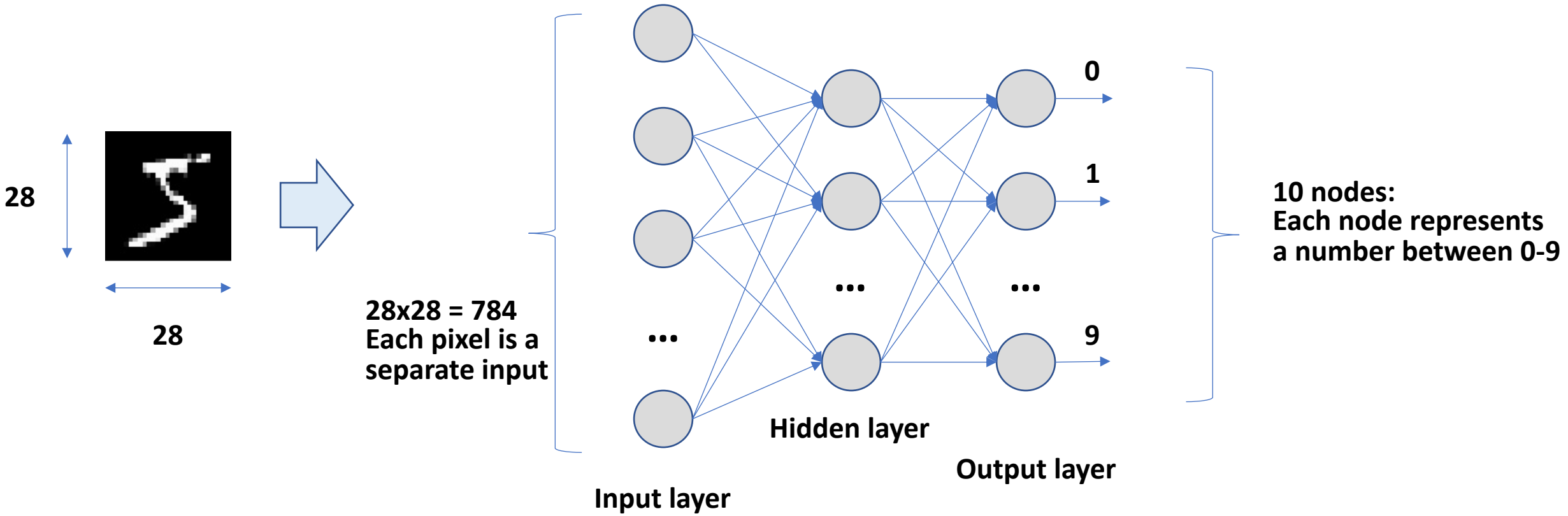
MNIST dataset: “Hello world” of DL!

- Handwritten digits dataset
  - 10 classes: Numbers 0-9
  - 60000 images for training
  - 10000 images for testing
    - *Unseen data*
- Widely used in research



http://yann.lecun.com/exdb/mnist/

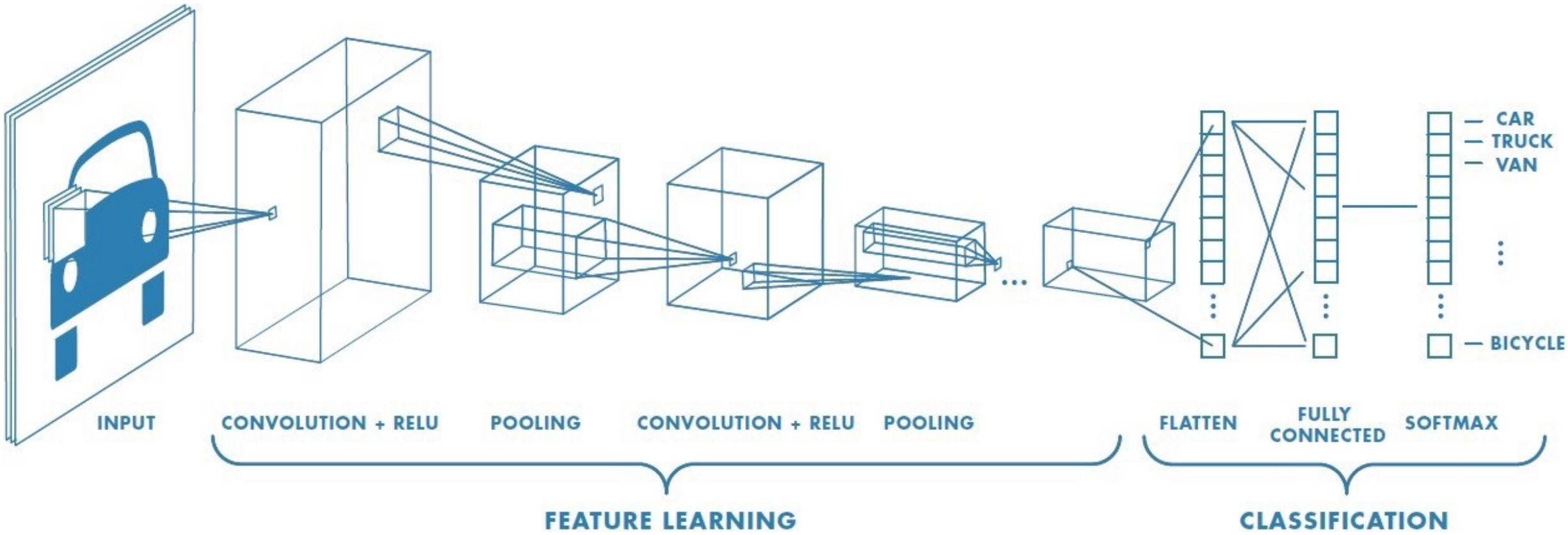
MLP using MNIST



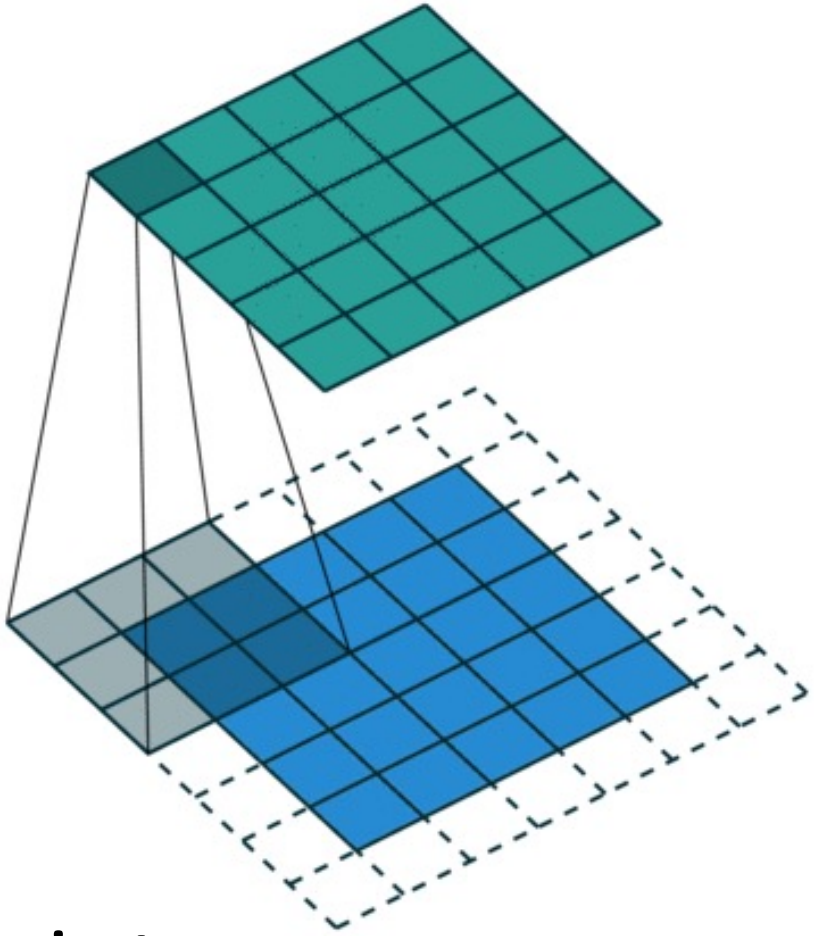
- Easily achieves >90% accuracy
  - Up to 99.9% accuracy reported

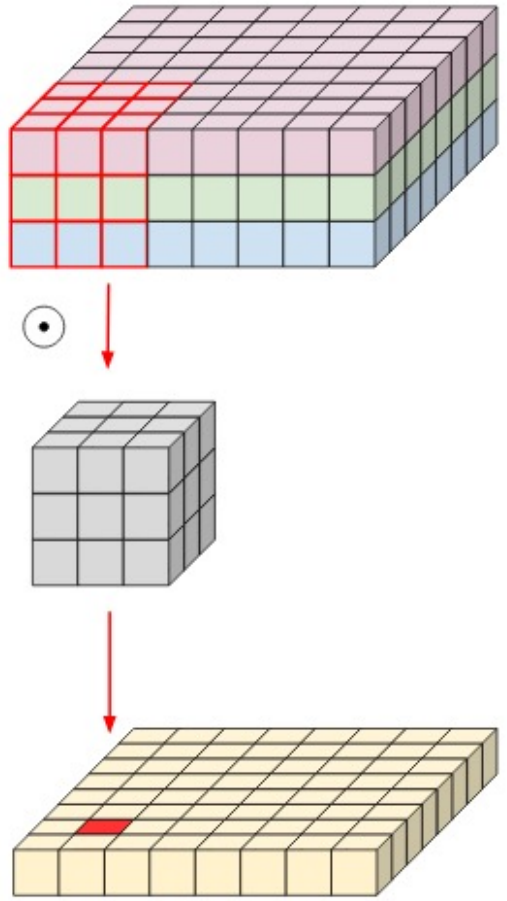
https://www.youtube.com/watch?v=aircAruvnKk
https://www.youtube.com/watch?v=IHZwWFHWa-w
https://www.youtube.com/watch?v=Ilg3gGewQ5U
https://www.youtube.com/watch?v=tIeHLnjs5U8

Convolutional Neural Networks (CNNs)



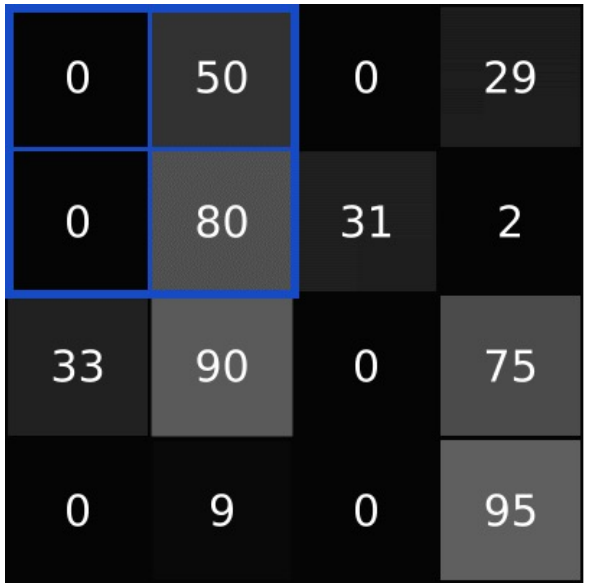
Convolution layer

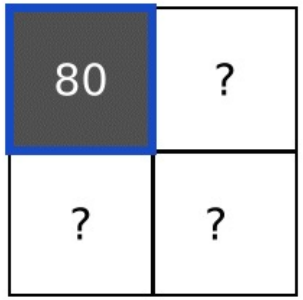




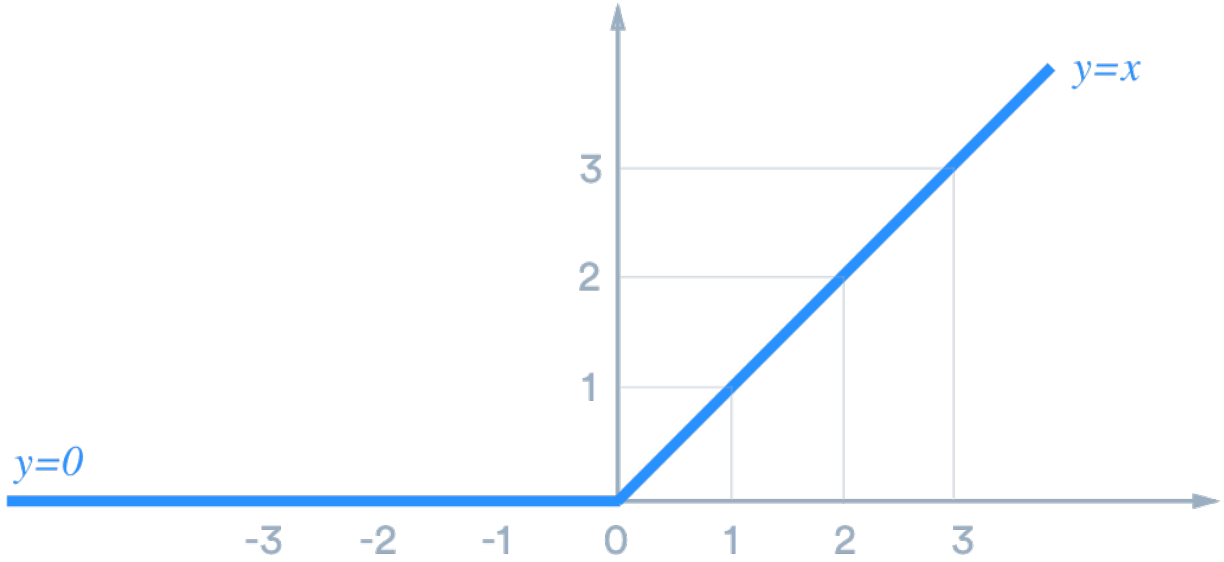
- **Input size invariant**
- **Preserves spatial information**
- **Extremely compute-intensive**

Pooling and ReLu





Max Pool

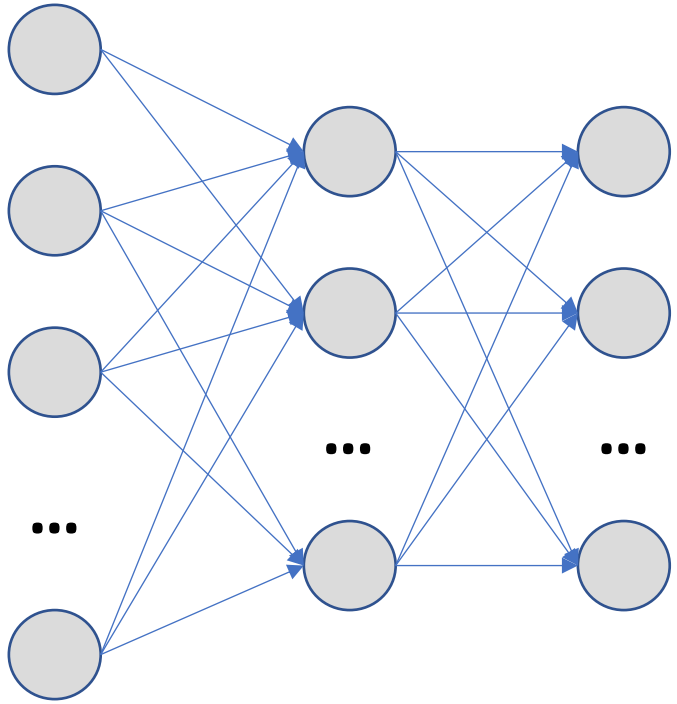


ReLU

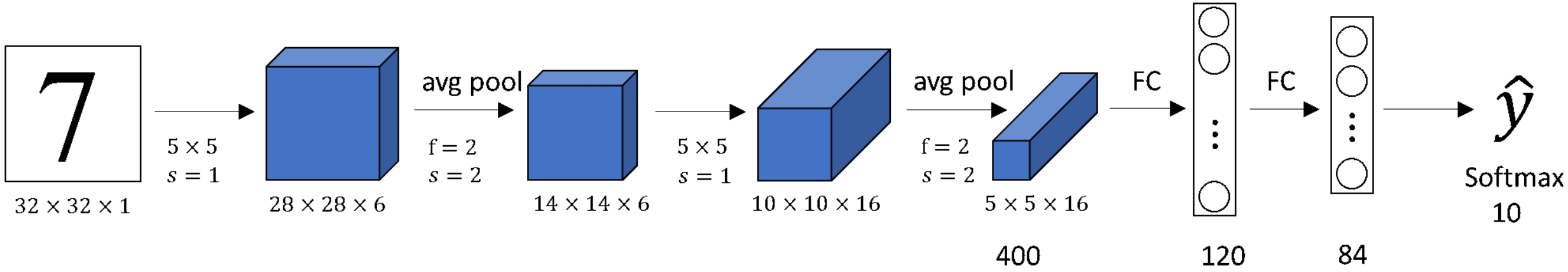
- **Smaller layers, not much computation**

Fully-Connect layer

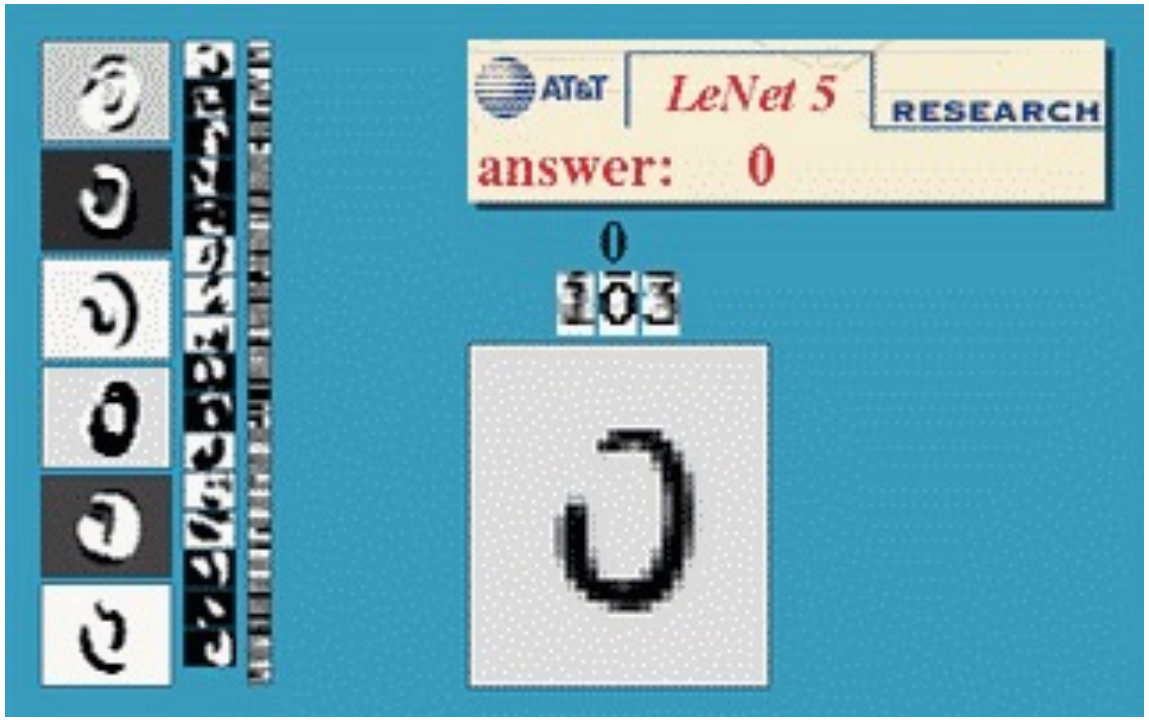
- Input size dependent
- Resembles an MLP
- Usually found at the end of a CNN



LeNet in action



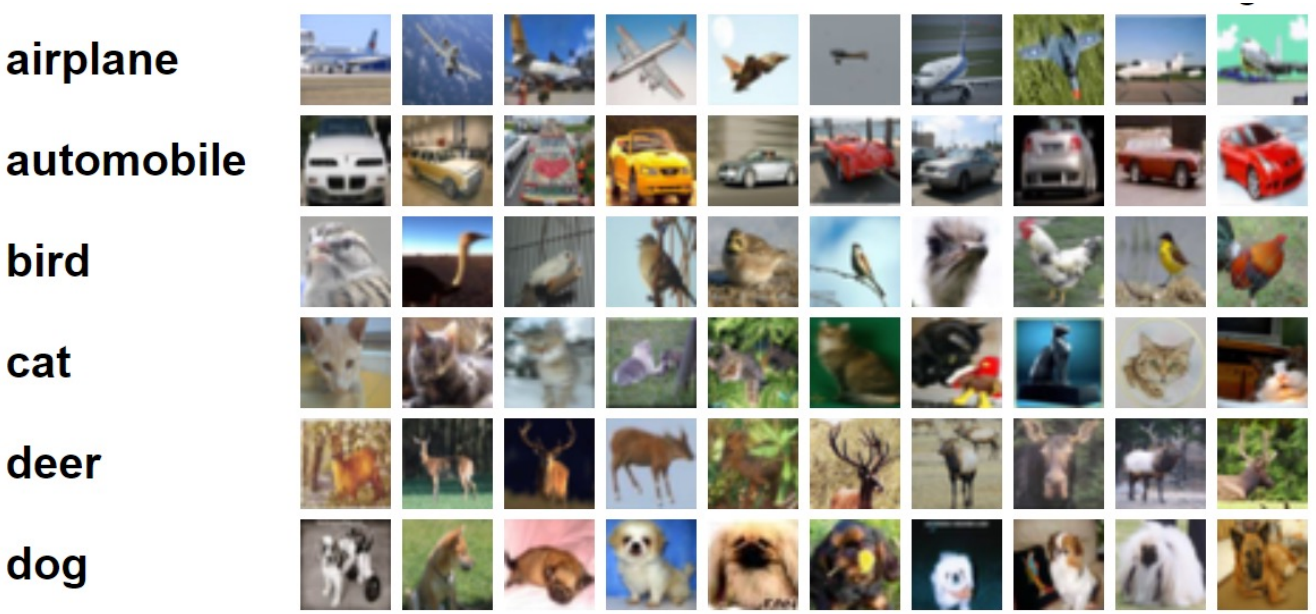
- 5 layers CNN
- One of the most popular
- Reaches 99% accuracy with MNIST dataset



http://yann.lecun.com/exdb/lenet/

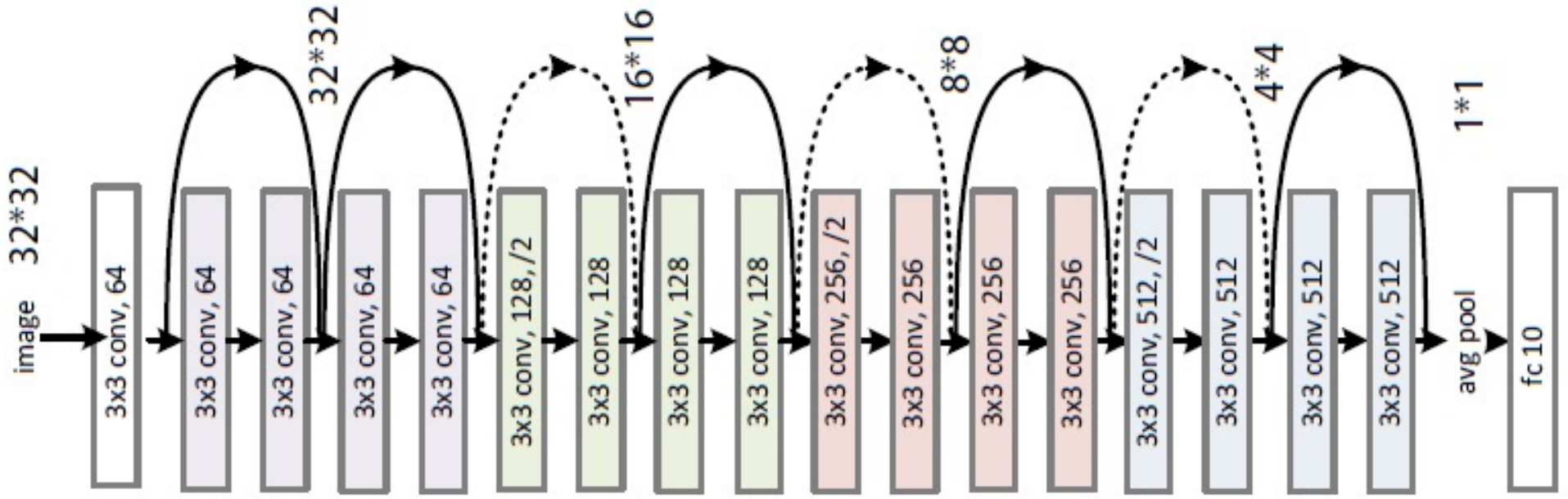
CIFAR-10/100 dataset

- 32x32 sized color images
  - 10 classes: Numbers 0-9
  - 50000 images for training
  - 10000 images for testing
    - *Unseen data*
- Widely used in research



https://www.cs.toronto.edu/~kriz/cifar.html

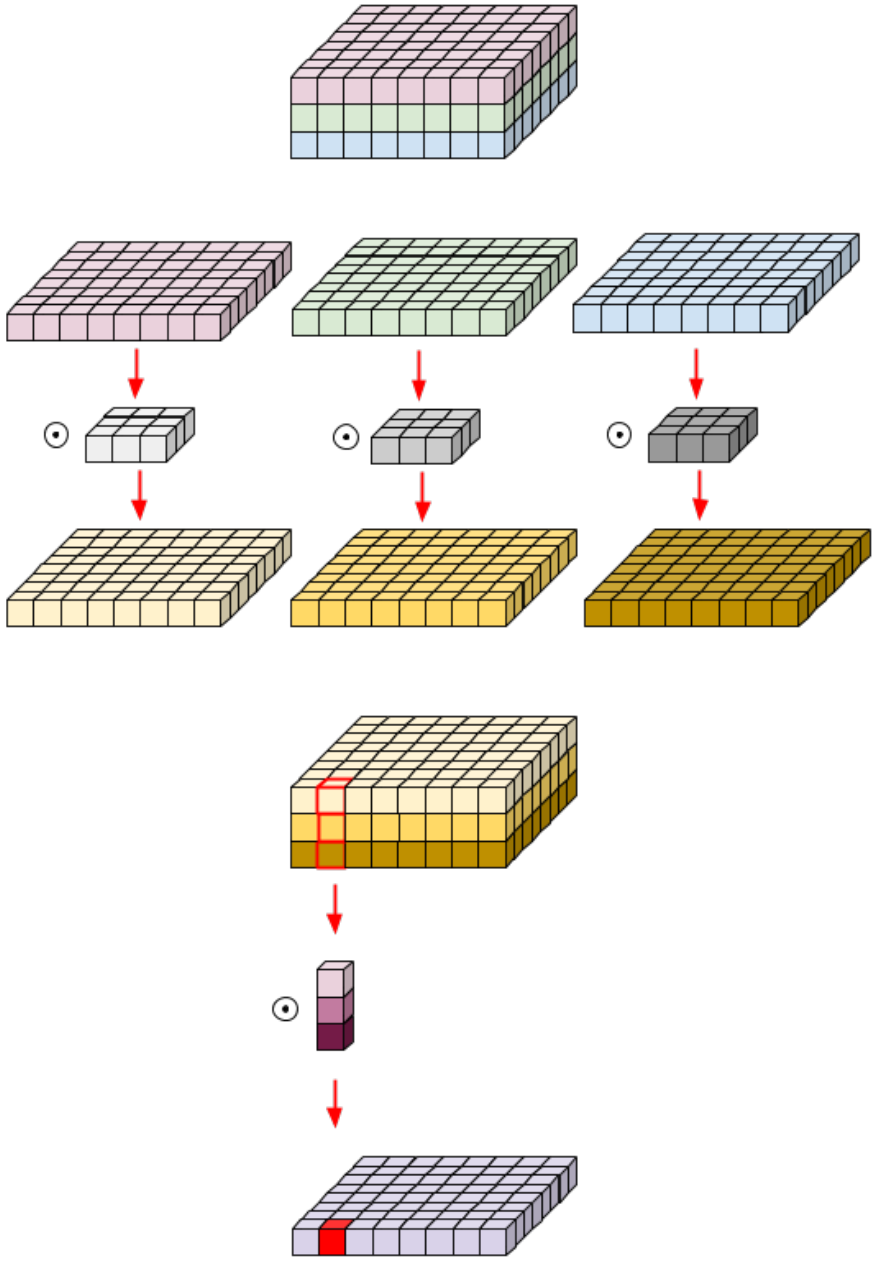
ResNet

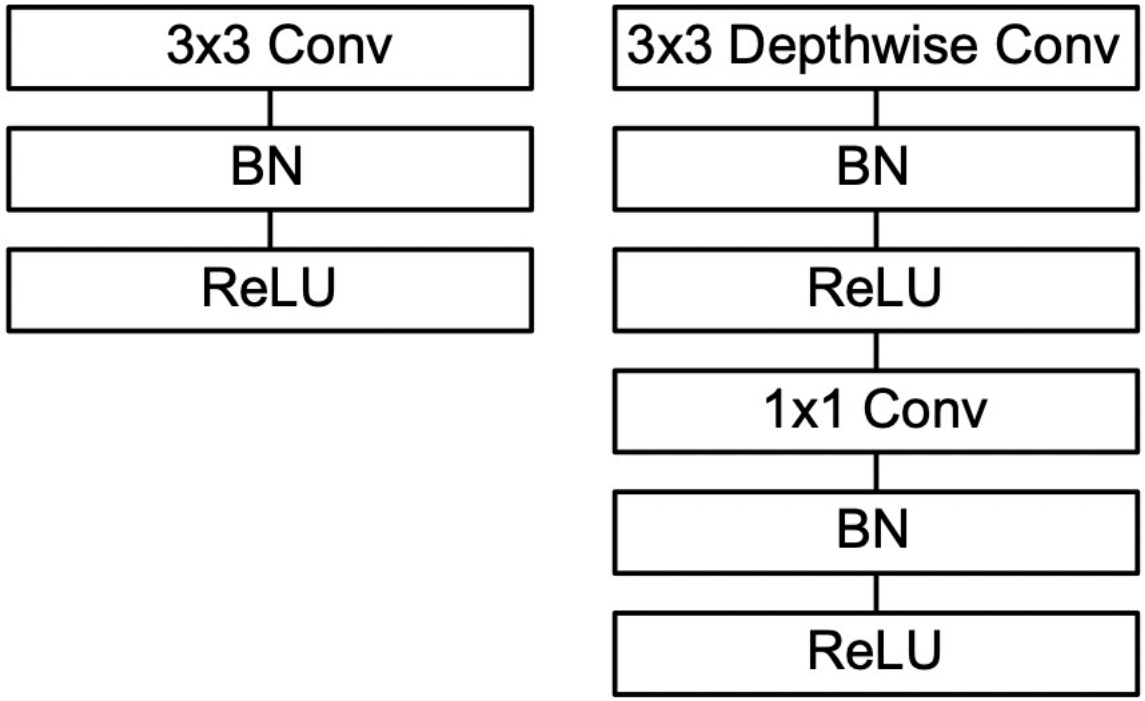


- **Includes skip connections (residual connections)**
- **Less weights but more deep**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

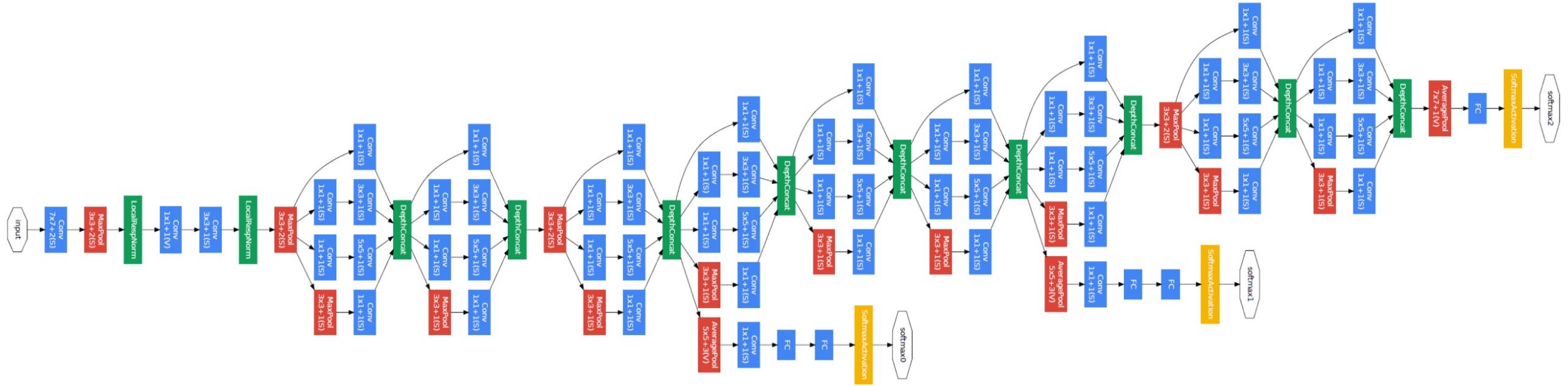
MobileNet





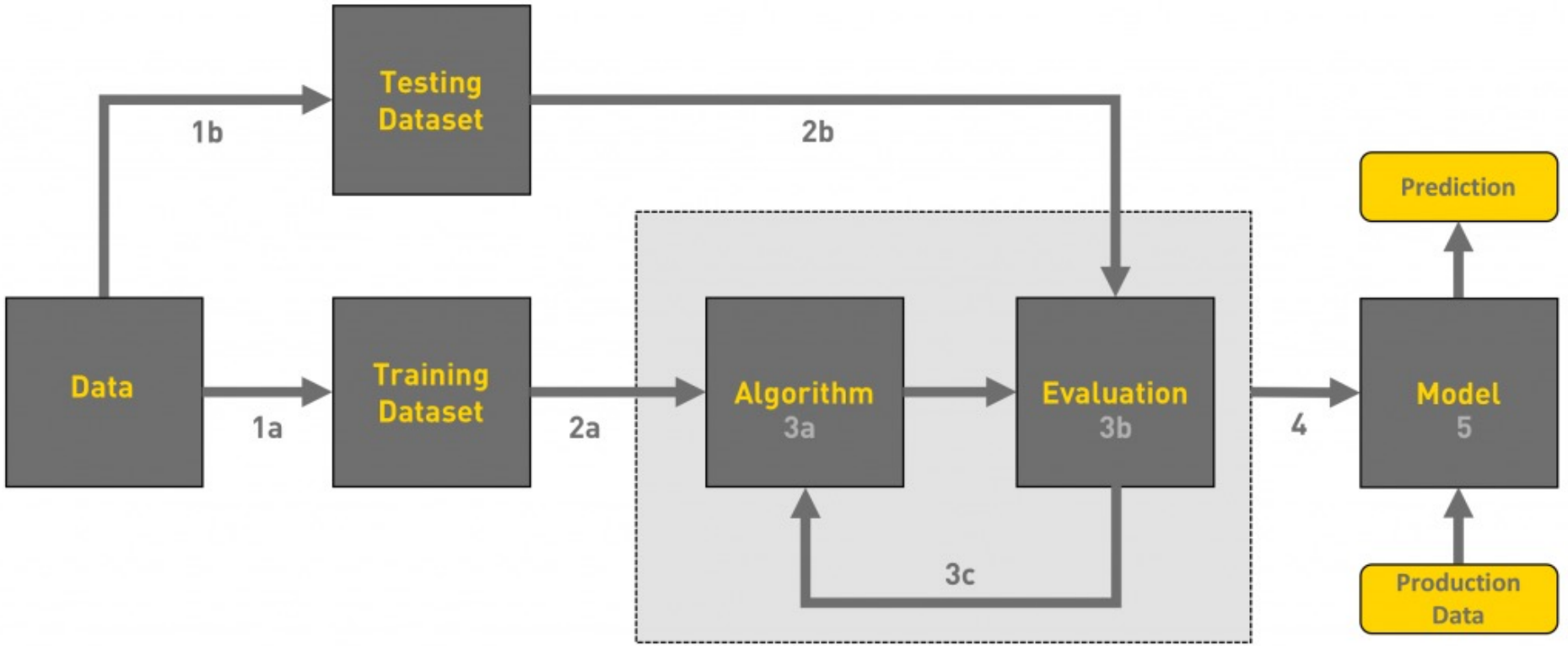
- Different convolution layer
  - Depthwise convolution
- Fewer weights
- Good for mobile devices

GoogleNet



- **Uses different size filters at each convolution layer**
- **Multiple branches**

Training ML model



CNNs using Pytorch

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

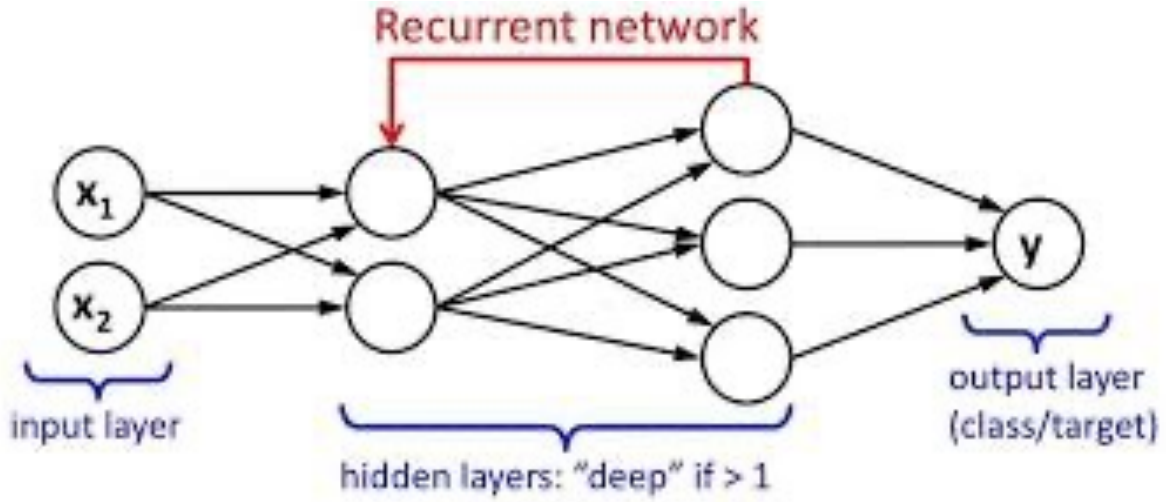
```python
for epoch in range(2):  # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

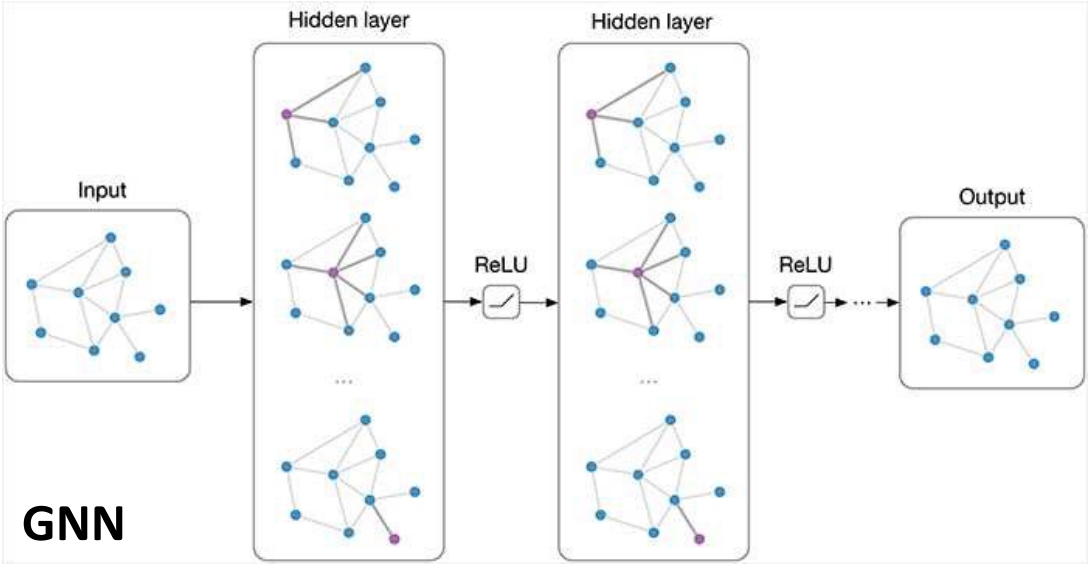
Other Deep Learning algorithms



RNN

- RNN can work with unequal size data
- Popular in NLP
- Sample RNN inputs
  - I ate an apple
  - Yesterday, I ate an apple
  - I like apple

- GNNs work with graph data
- Node order invariant representation
- Sample GNN inputs
  - Facebook userbase (person = node, friend=edge)
  - Chemical molecules



GNN

Some other ML variants

- **ML can be used for classification, regression, clustering, etc.**
- **Many ML algorithms exist**
  - **Underlying math is similar**
- ***We will use some of these algorithms for this class***