Introduction to Machine Learning and Deep Learning

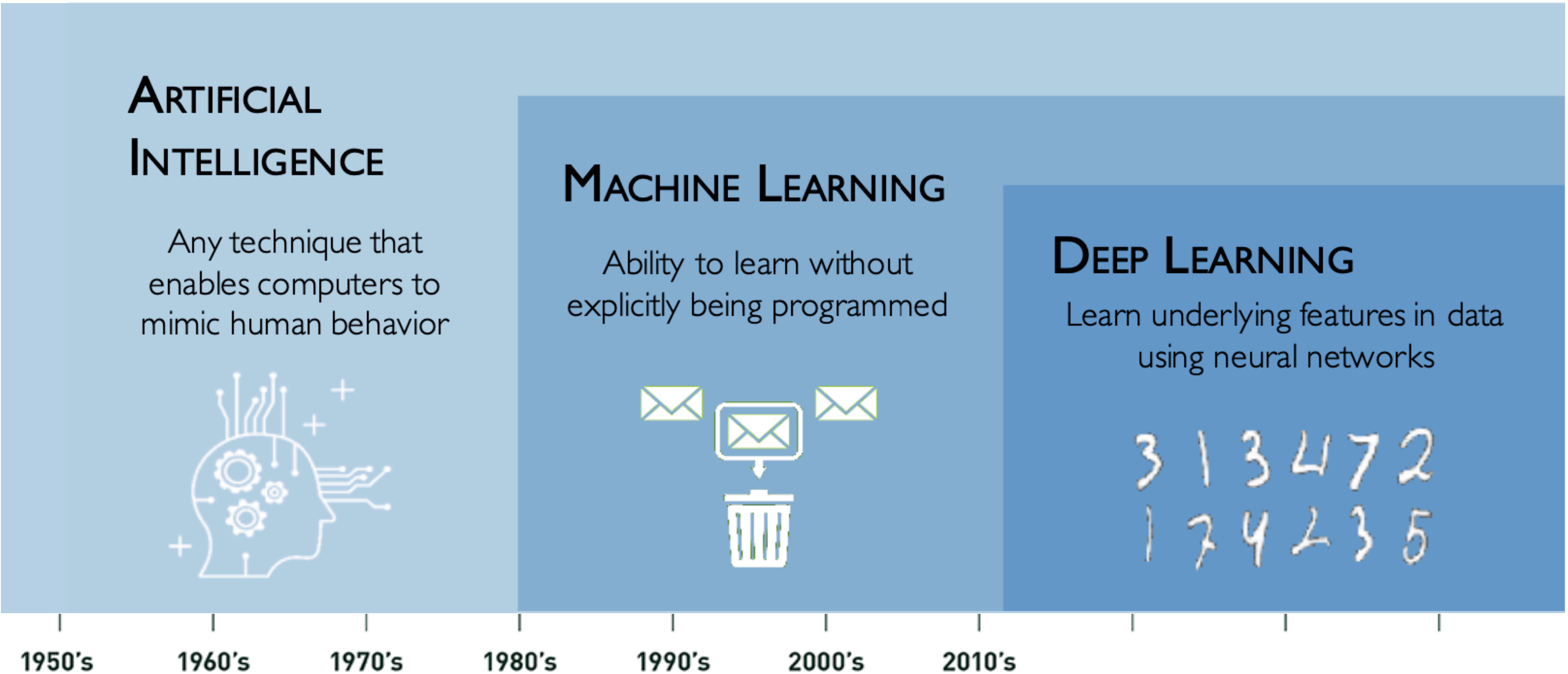
Itthi Chatnuntawech
Nanoinformatics and Artificial Intelligence (NAI)
National Nanotechnology Center (NANOTEC)
National Science and Technology Development Agency (NSTDA)
September 5, 2022

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator



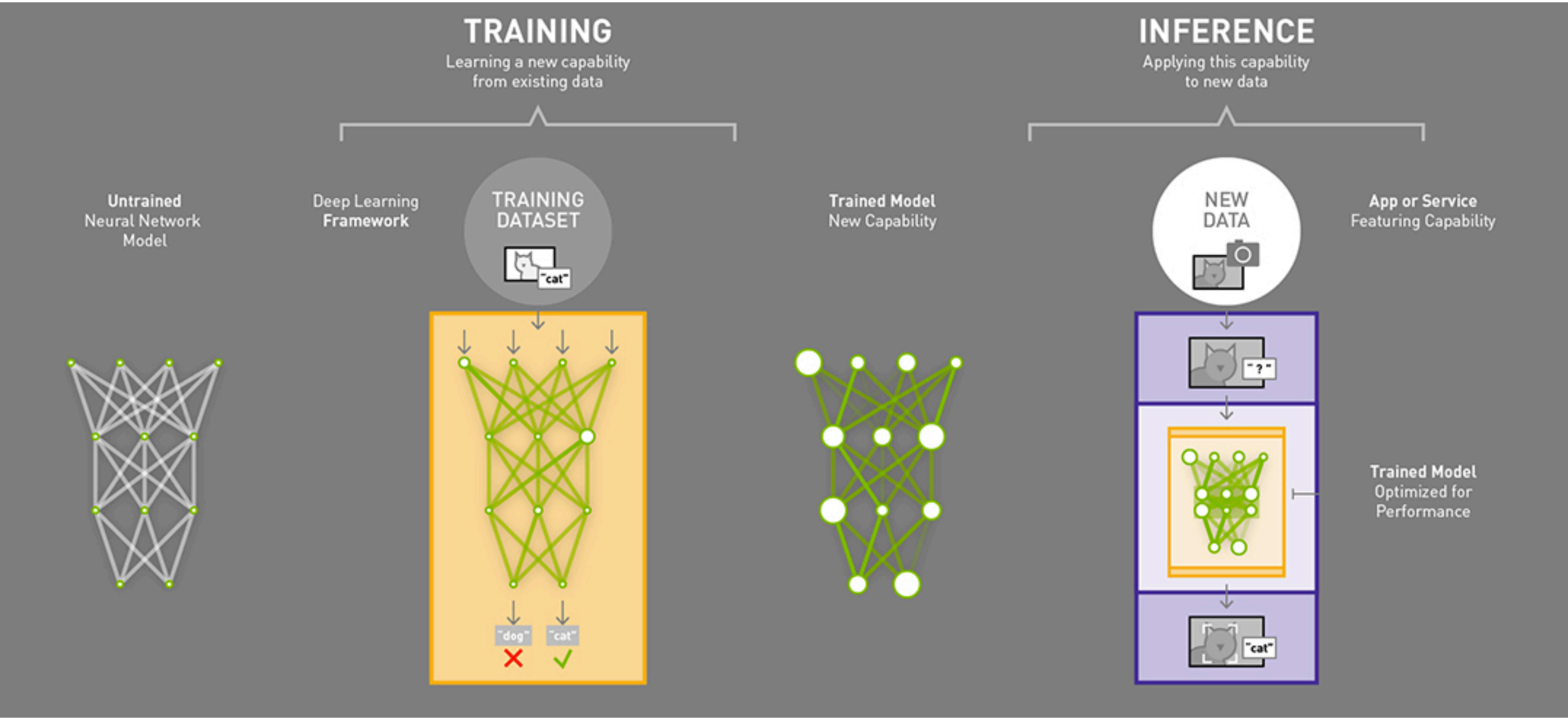
→ Fasting blood sugar test → x mg/dL

normal if $x < 100$

prediabetes if $100 \leq x \leq 125$

diabetes if $125 < x$

Overview: Machine Learning



Create a model

Train the model with training data

Test the trained model with test data

Source: NVIDIA

keywords: feature extraction,
classification task, classifier

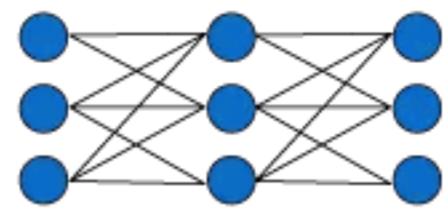
Example: Watermelon or Orange?

Traditional Machine Learning

Input image

Feature extraction

random forest, SVM,
kNN, XGBoost, LGBM



Classification

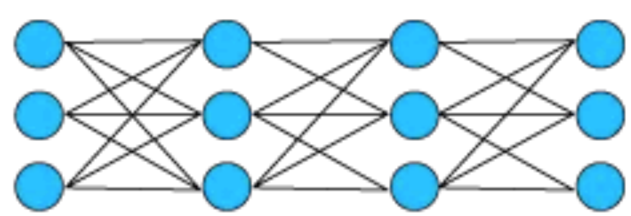
Output

watermelon

Deep Learning

Input image

deep neural network



Feature extraction + Classification

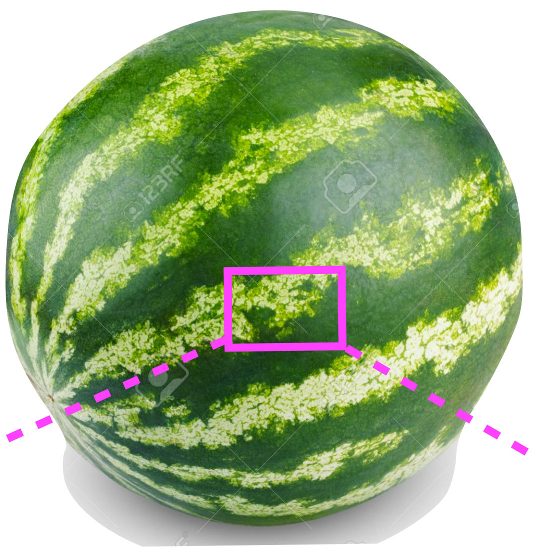
Output

watermelon

Some parts taken from: Azua Tech

keywords: vector, matrix, tensor, pixel

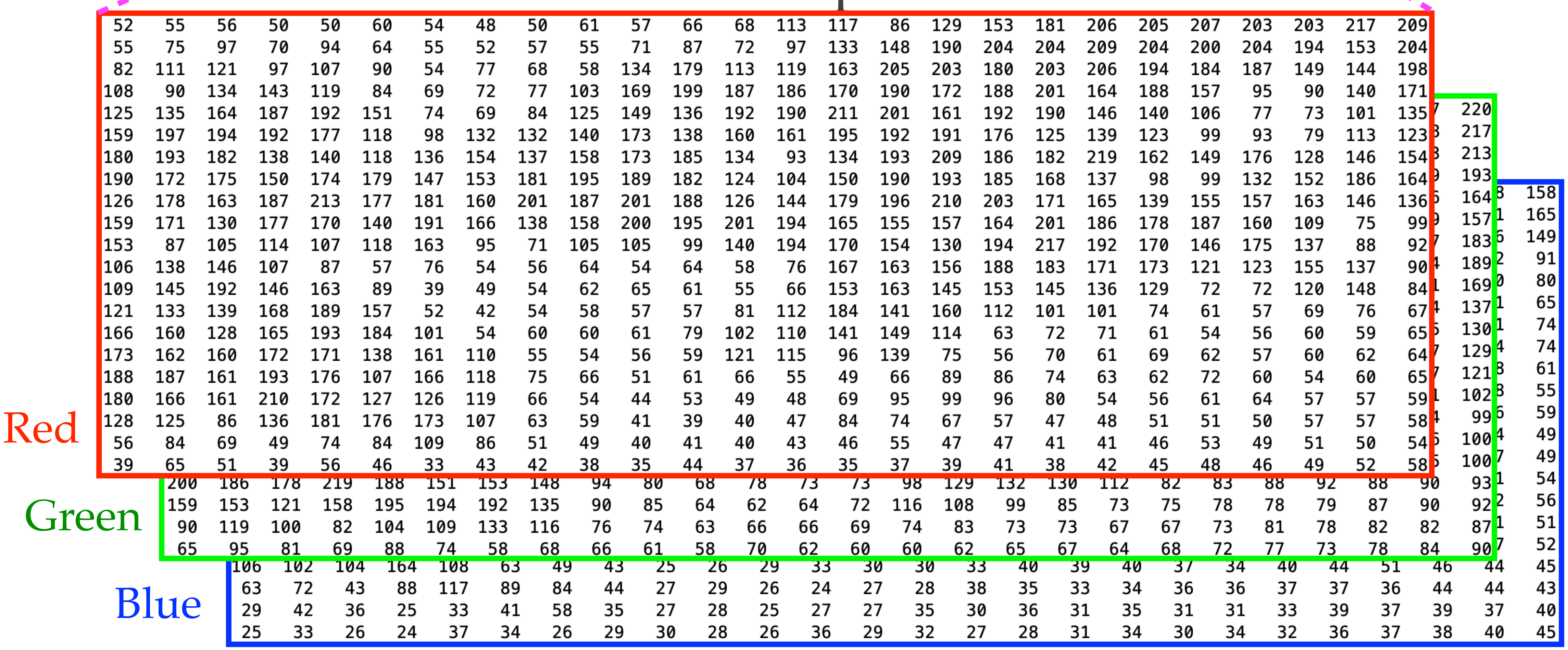
What we see



orange: 0

watermelon: 1

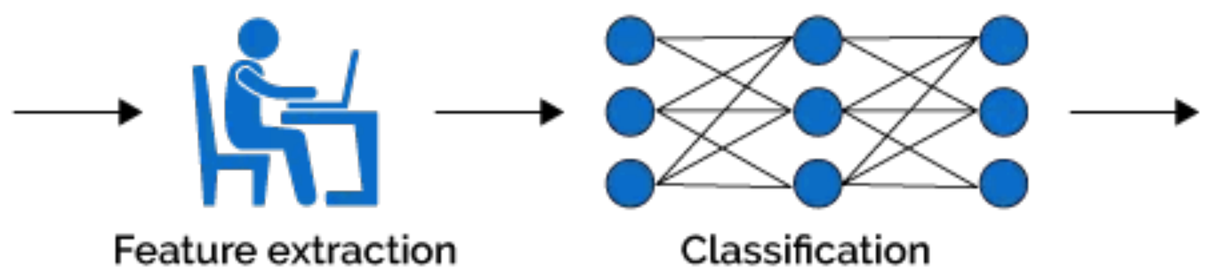
What computers see



Watermelon or Orange?

Traditional Machine Learning

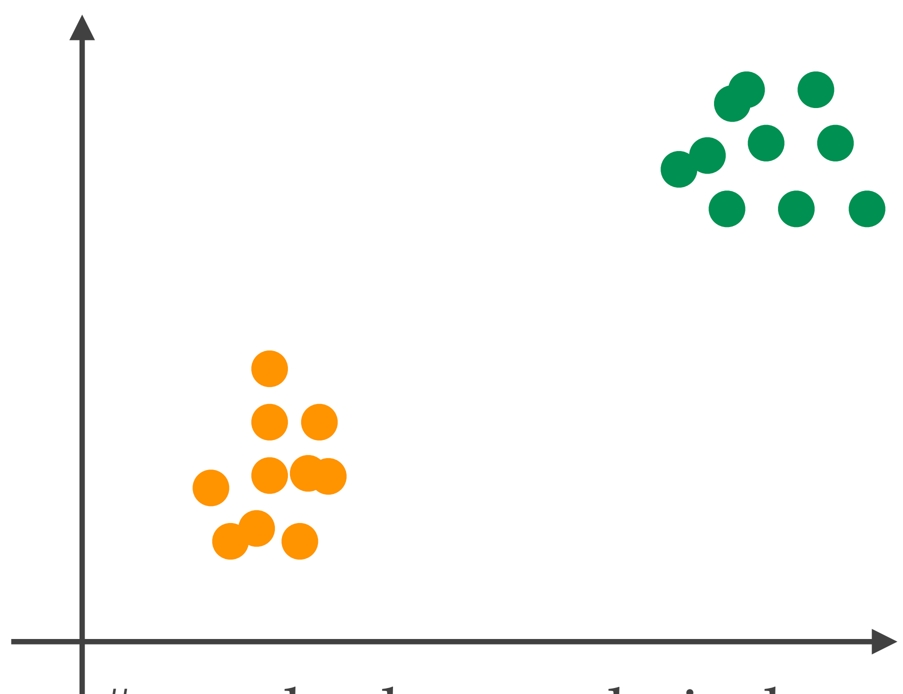
Input image



Output

watermelon

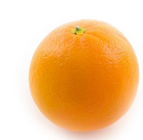
Phase: Training



“radius”
feature 2

non-background pixels
feature 1

current sample

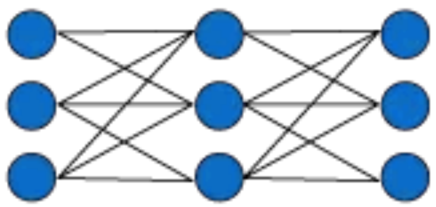


keywords: feature extraction, feature engineering, handcrafted/hand-designed features, feature space, descriptors

Watermelon or Orange?

Traditional Machine Learning

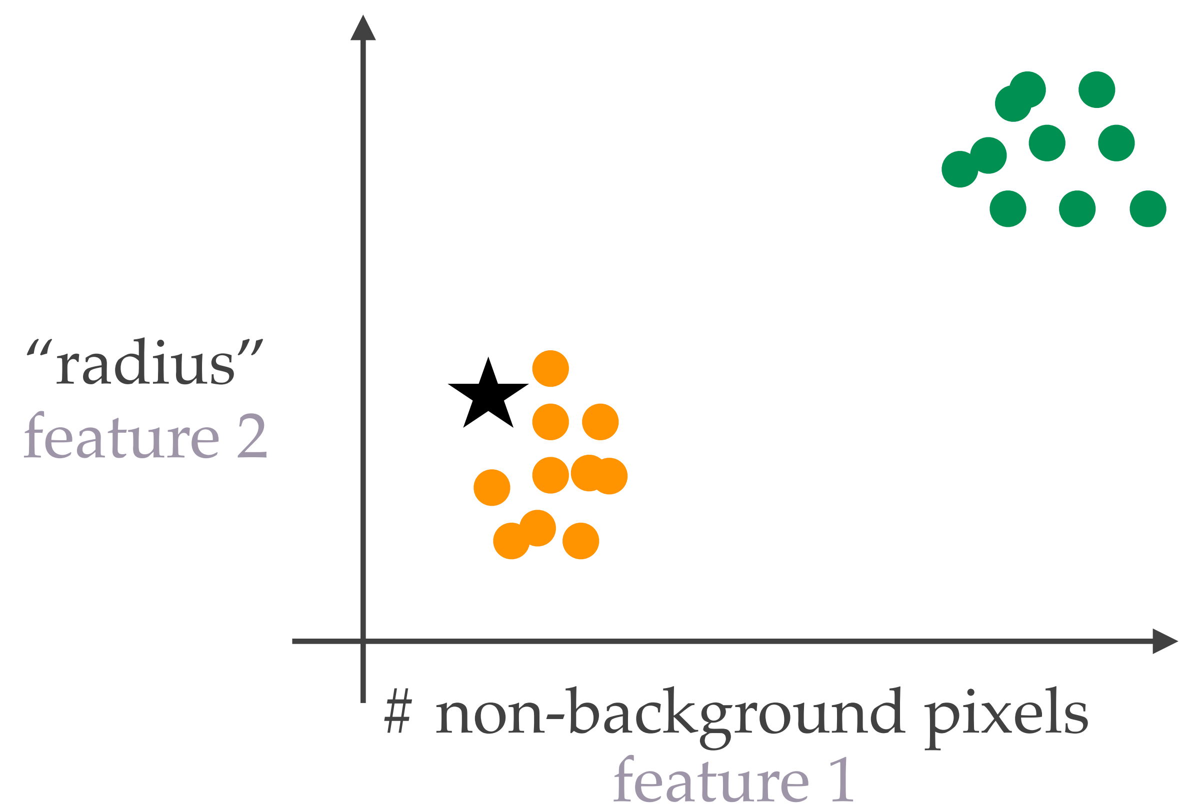
Input image



Output

watermelon

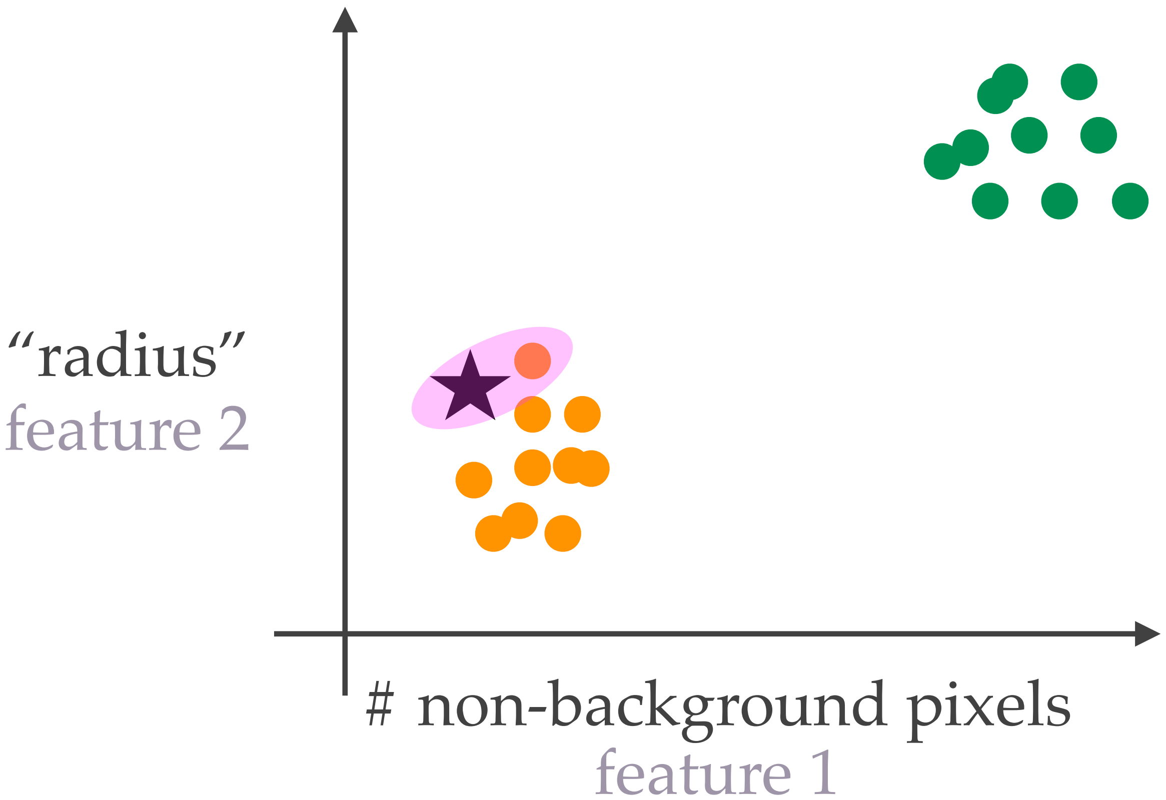
Phase: Testing



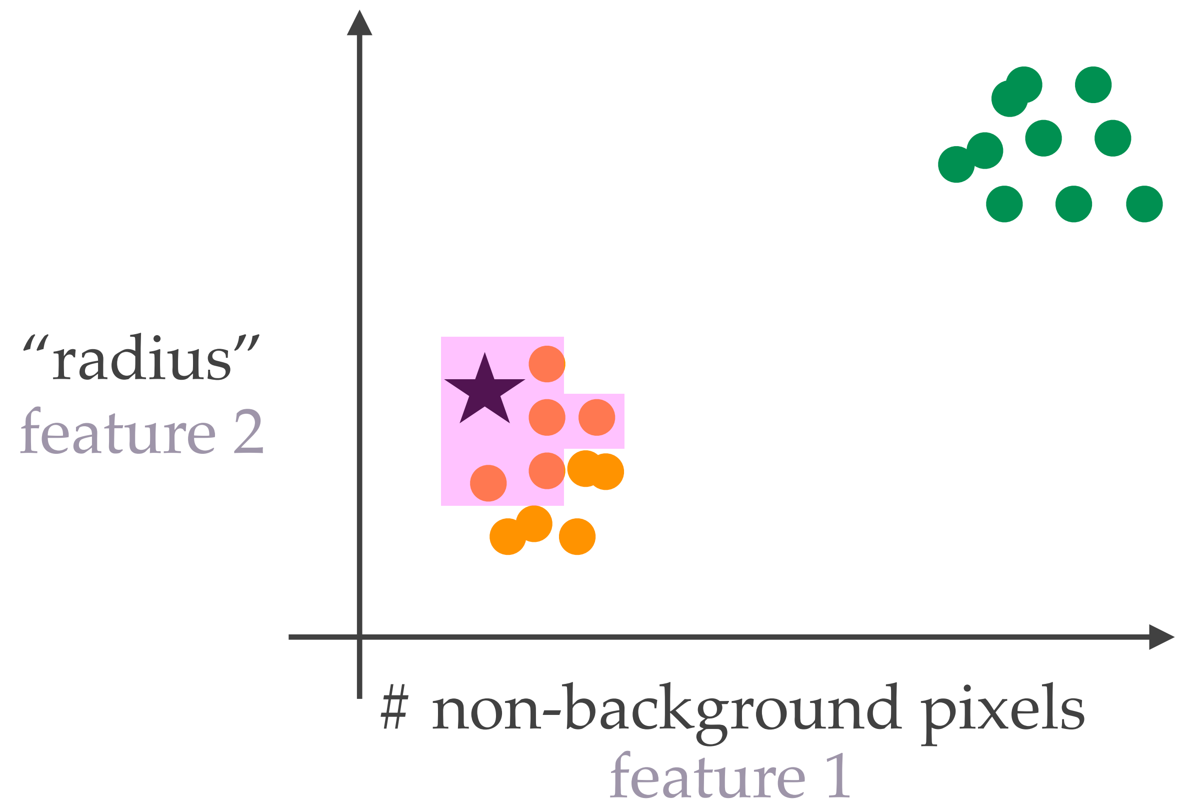
current sample

Orange!

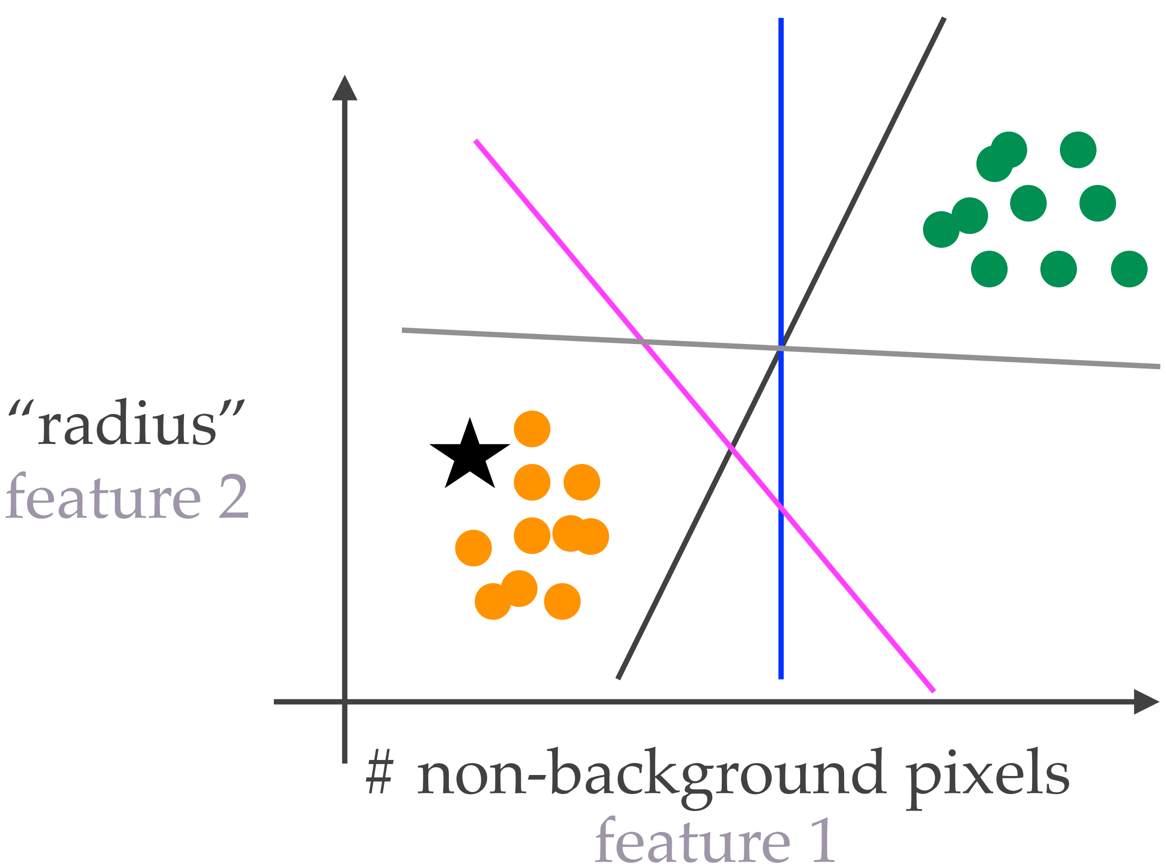
Nearest Neighbor



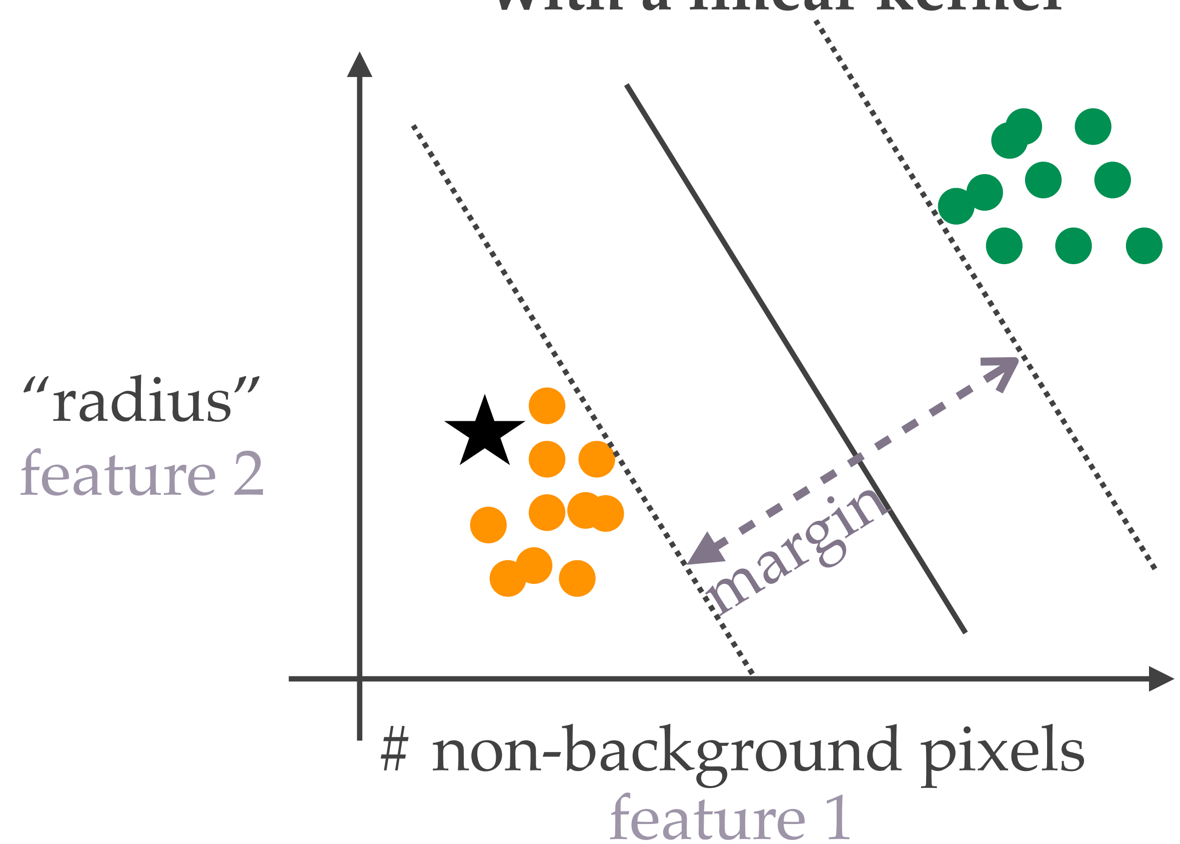
k-Nearest Neighbors (kNN)



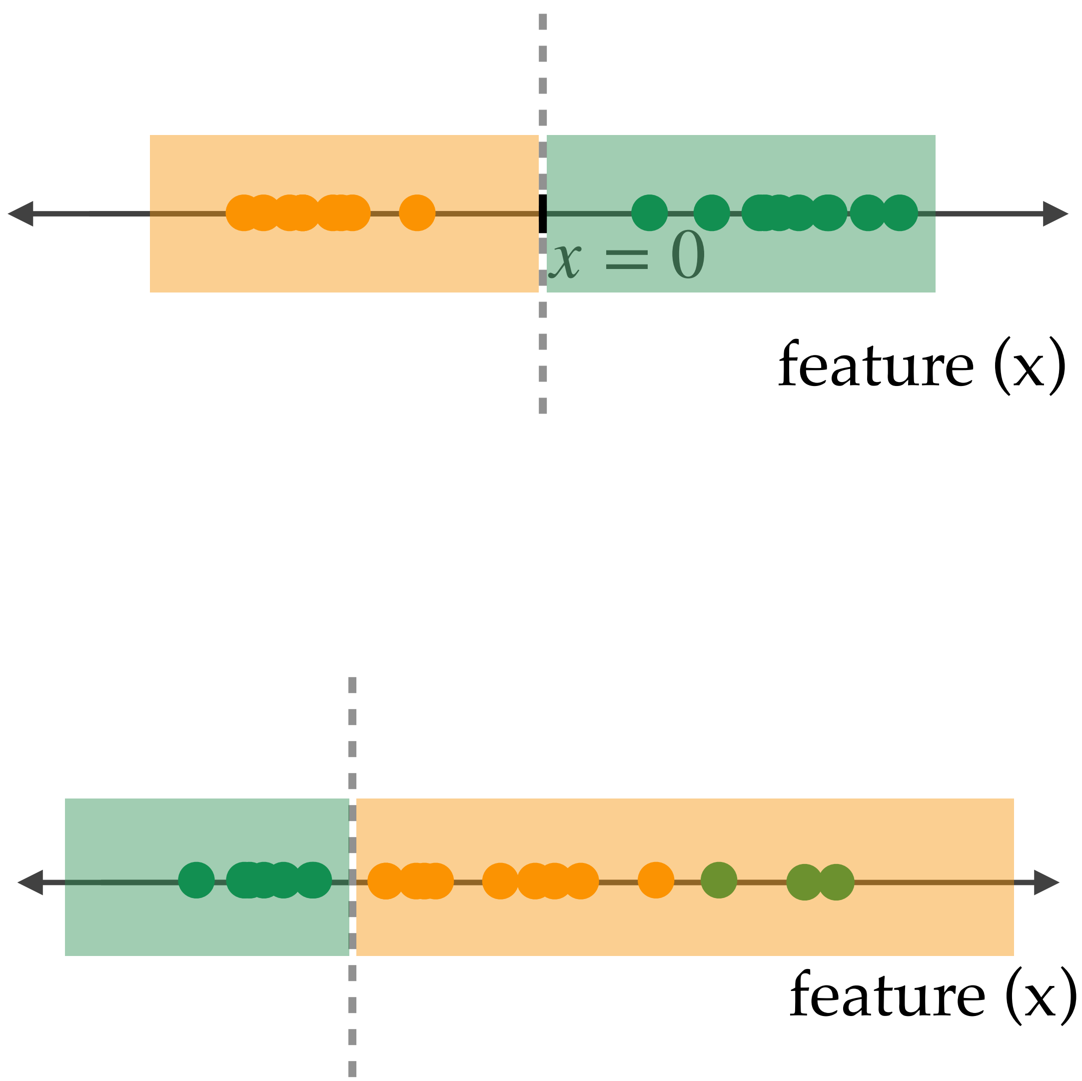
Linear Classifier



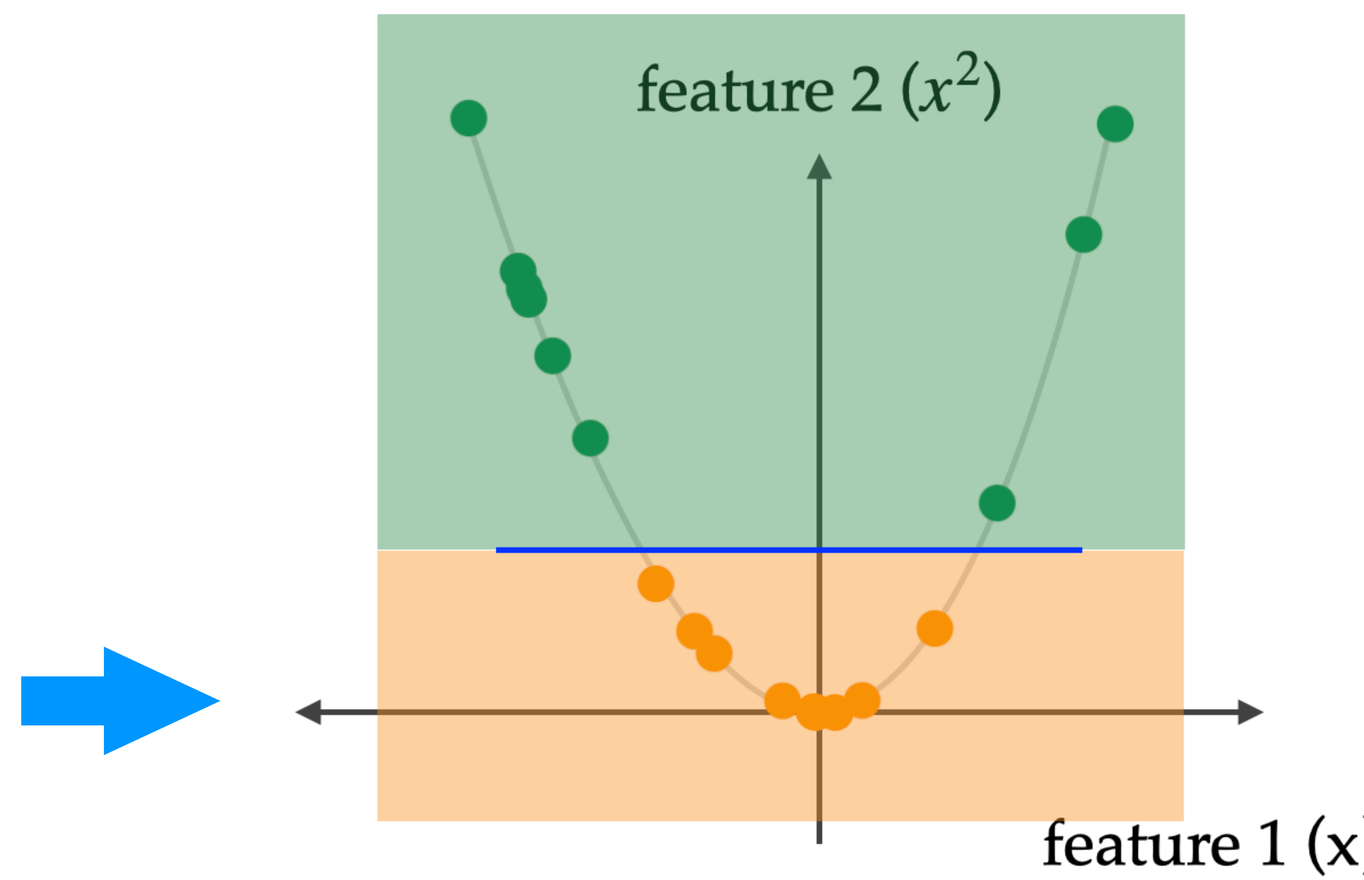
Support Vector Machine (SVM) with a linear kernel



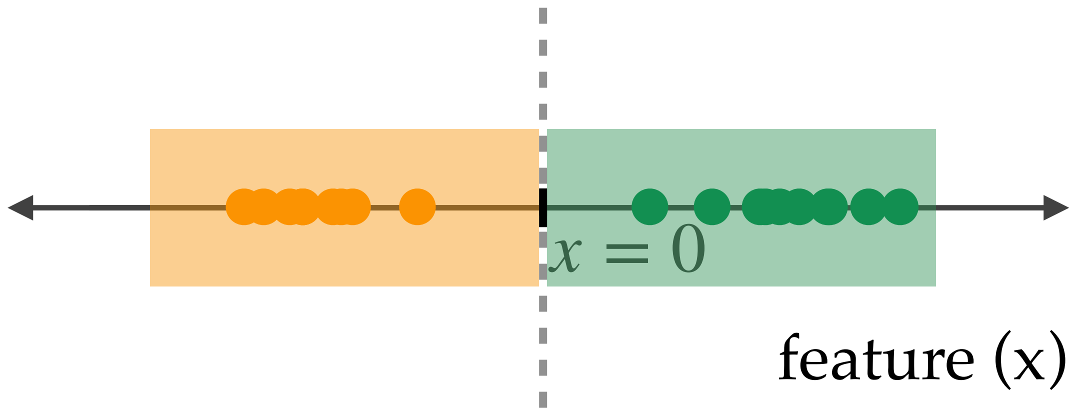
Linear Classification?

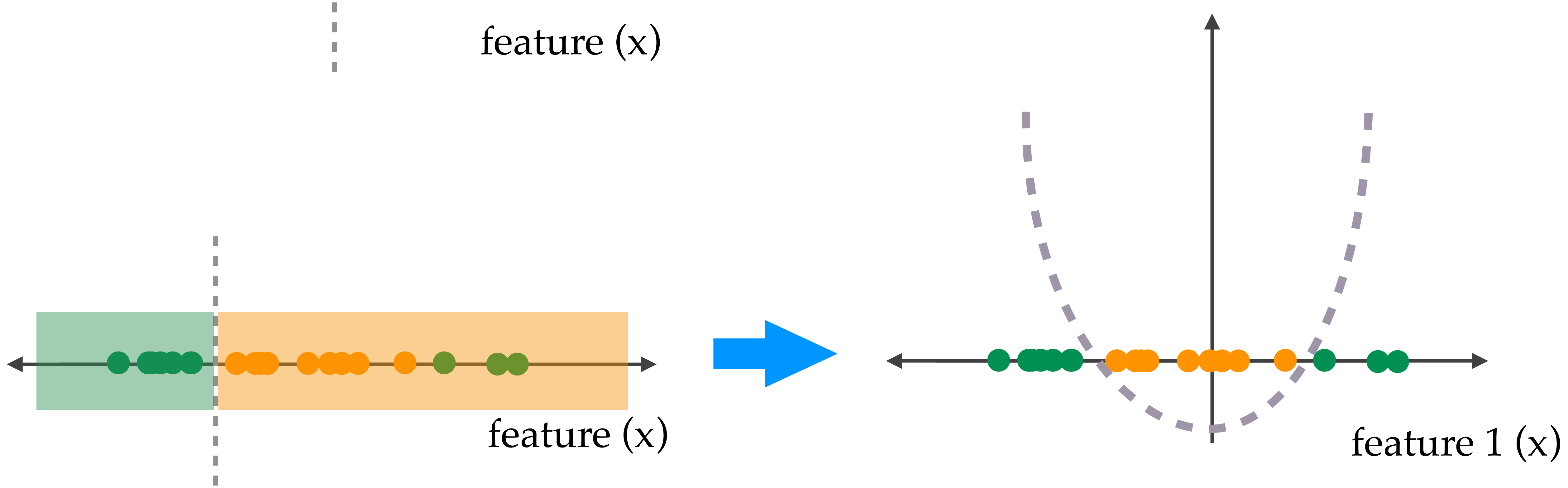


Good feature engineering helps.
A linear classifier still works!



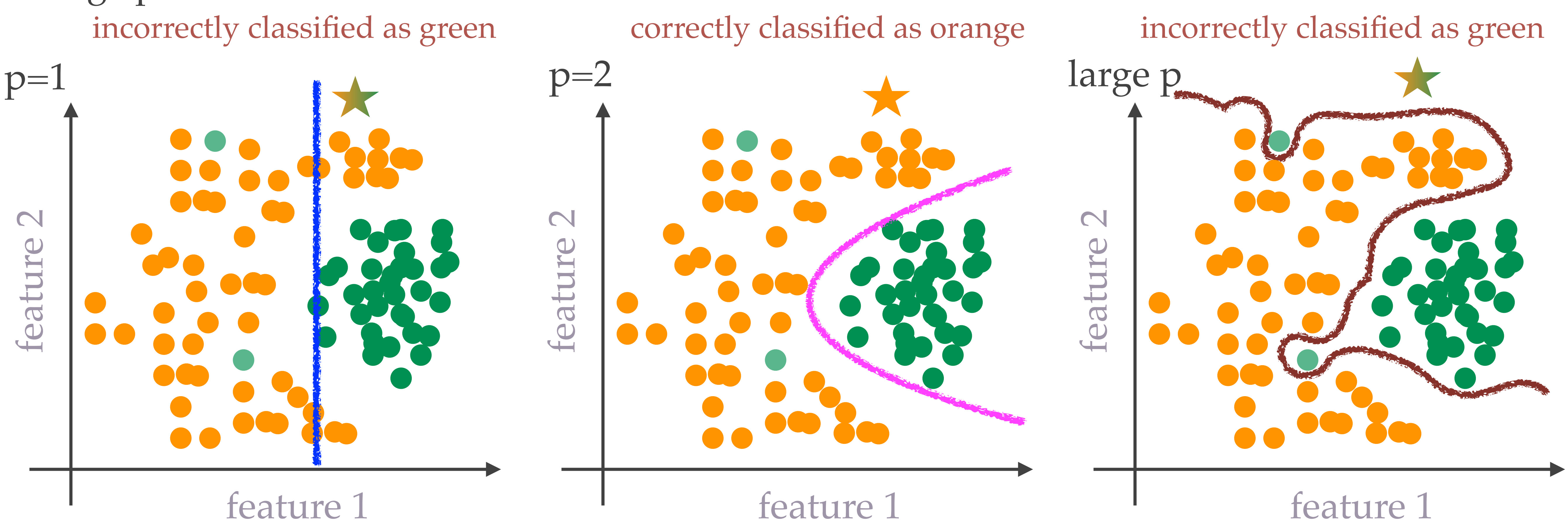
Linear Classification?





What should we pick as our model?

If we use a p^{th} order polynomial to separate the classes, what happens when p=1? p=2? large p?



Underfit

Doesn't perform well for both seen (training) and unseen (test) data

- High training error
- High test error

Overfit

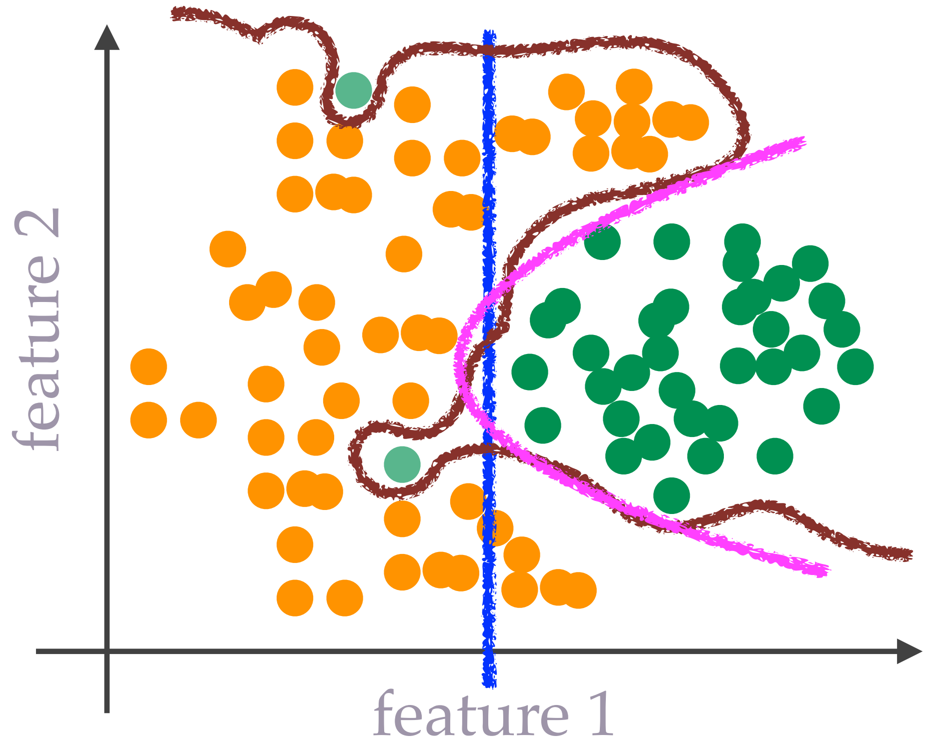
Perform well for training data, but not test data

- Low training error
- High test error

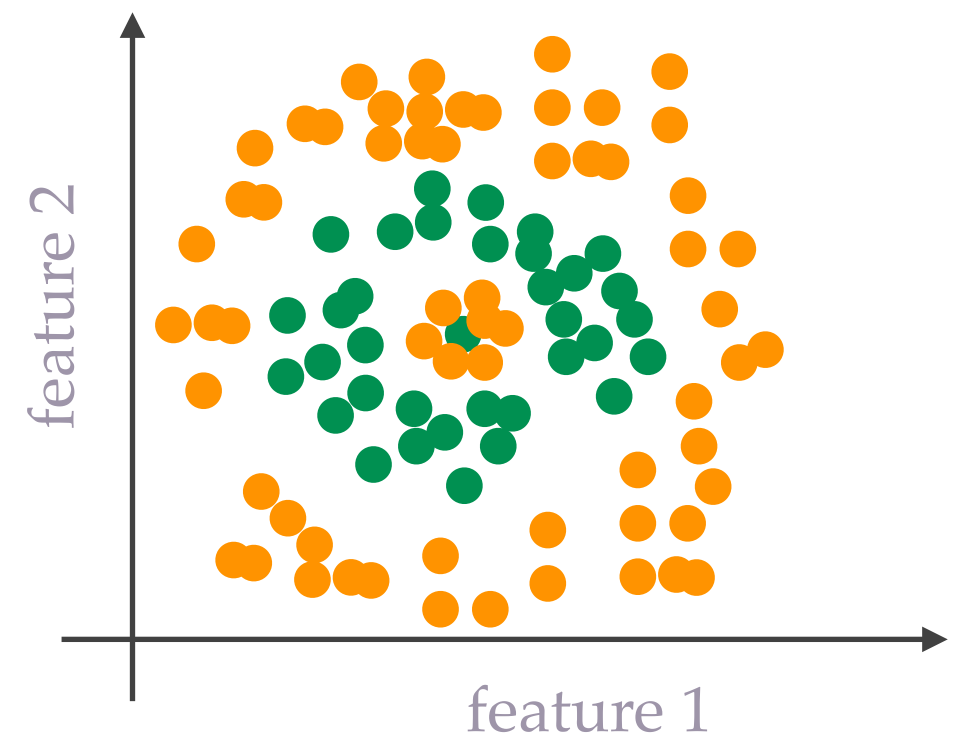
keywords: model selection,
hyper-parameter tuning,
training/validation/test data

What should we pick as our model?

Polynomial order: p = ?



k-Nearest Neighbors: k = ?



If we use a p^{th} order polynomial to separate the classes, what happens when p=1? p=2? p=1000?

What happens when k is large (e.g., k=1000)?

What happens when k=1?

How do we pick a model (e.g., simple linear classifier, p^{th} order polynomial, k-nearest neighbors, logistic regressor)? How do we pick p and k?

Validation data (in addition to training and test data, which we have seen earlier)!

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

Simplified Pipeline & Training, Validation, and Test Data

1. Given a task, pick a model   linear vs nonlinear?   Polynomial of order p = 3 vs p = 21?

2. Split the data into three groups: training, validation, and test

Training	Validation	Test

3. Optimize the selected model using the training data

4. Evaluate the trained model using the validation data

Ex. Try a linear model, 8 kNNs with different k, and 20 polynomials with p = 2, 3, …, 21

5. Pick more models, optimize the models, and evaluate the trained models on the validation data

Ex. Pick the model with the smallest validation error (out of the 30 models)

6. Use the model with the lowest validation error as your final model

7. Evaluate the final model on the test data

Example: Training, Validation, and Test Data

Scenario: A time-limited, two-choice materials science exam will be held next month.
We have come up with four strategies (i.e., 4 models) for acing the upcoming exam

- AI #1: Always pick choice 1 ☐ choice 1 ☐ choice 2
- AI #2: Pick choice 1 for the first half of the exam and learn what to answer for the rest
- AI #3: Pick choice 2 for the first half of the exam and learn what to answer for the rest
- AI #4: Learn what to answer for all questions

Training	Validation	Test
Practice problems and previous exams from the last decade (except 2018-2021)	2018-2021 exams	The upcoming exam (unseen)

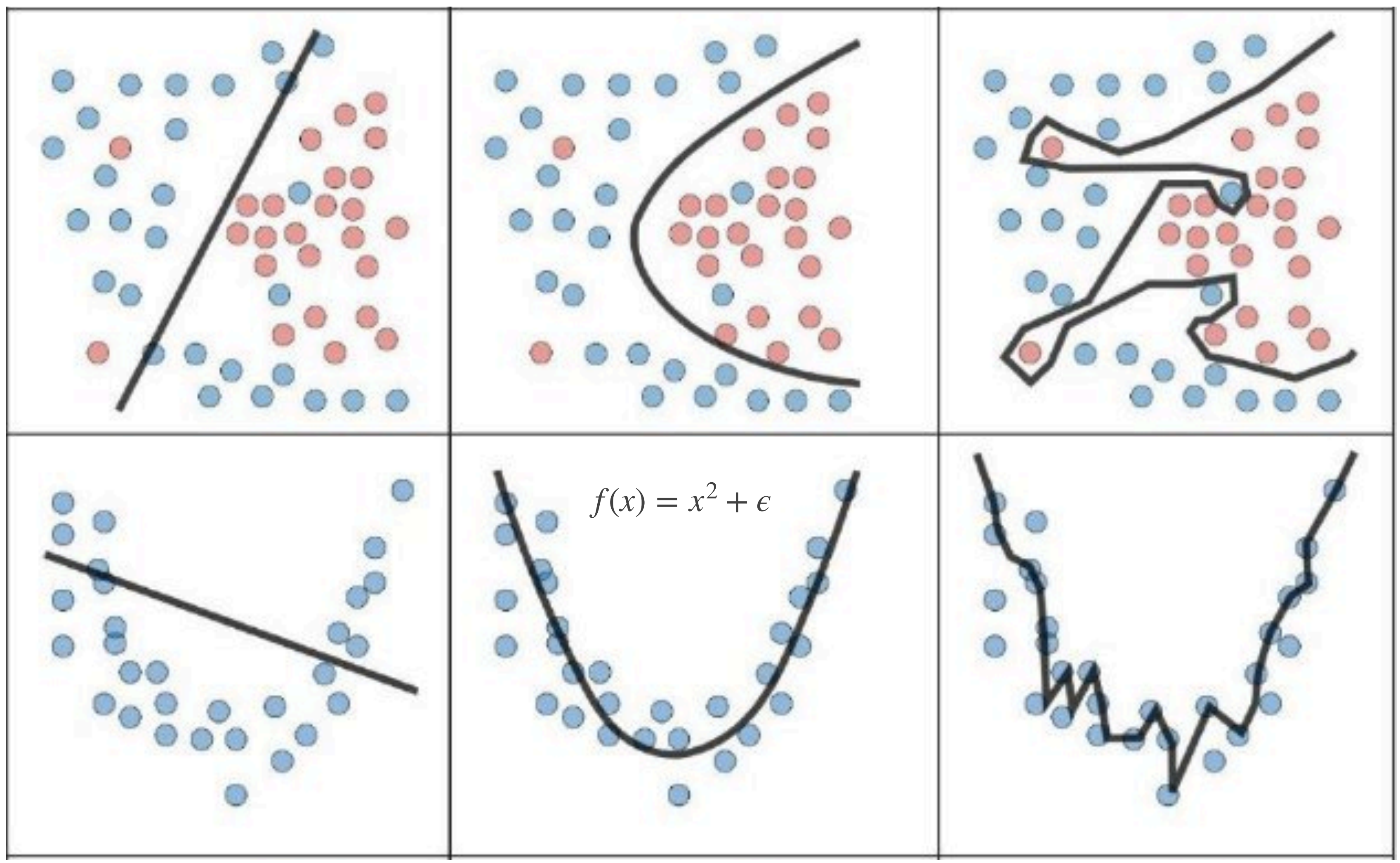
Steps
1. Use the training problems and exams to teach all the AI models.
2. Use the validation exams to see how the trained models perform.
3. Select the AI model that achieves the highest score on the validation exams to take the upcoming exam (test exam) for you

Overfitting and Underfitting

Underfitting | Overfitting

Classification

Regression



Likely symptoms

Underfitting:
- High training error
- Training error close to test error

Middle:
- Training error slightly lower than test error

Overfitting:
- Low training error
- Test error much higher than training error

https://i.pinimg.com/originals/72/e2/22/72e222c1542539754df1d914cb671bd7.png

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

Watermelon or Orange?

Traditional Machine Learning

Input image

hand-engineered / hand-crafted features

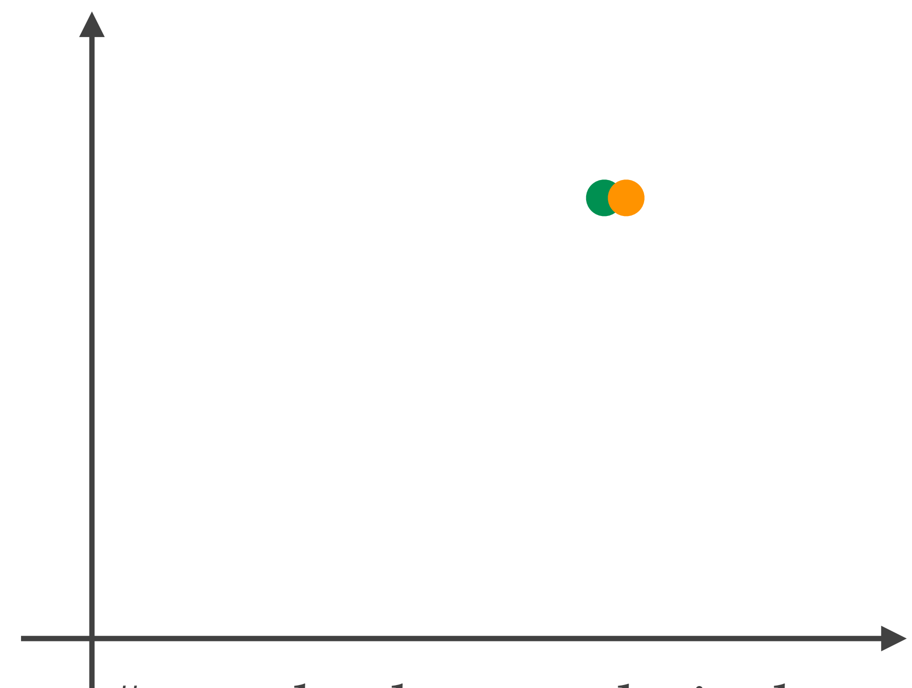
random forest, SVM, kNN, XGBoost, LGBM

Output

watermelon

Problem with feature engineering?

"radius"
feature 2



non-background pixels
feature 1

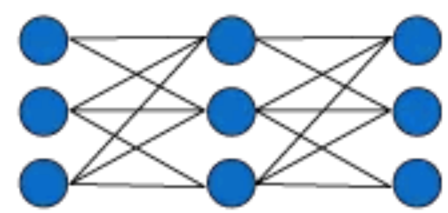
Some parts taken from: Azua Tech

Watermelon or Orange?

Traditional Machine Learning

Input image

random forest, SVM, kNN, XGBoost, LGBM



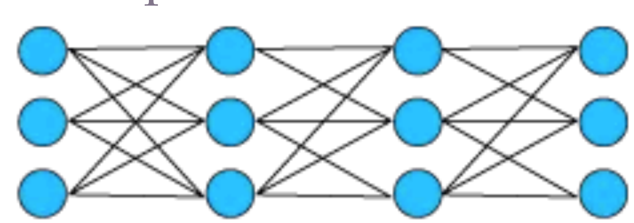
Output

watermelon

Deep Learning

Input image

deep neural network



Output

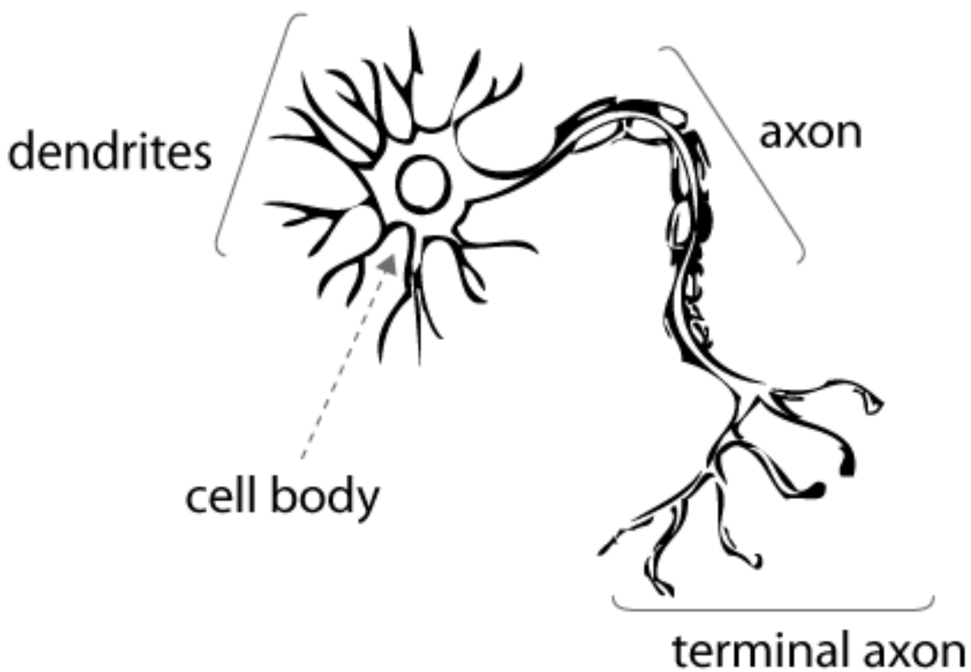
watermelon

Some parts taken from: Azua Tech

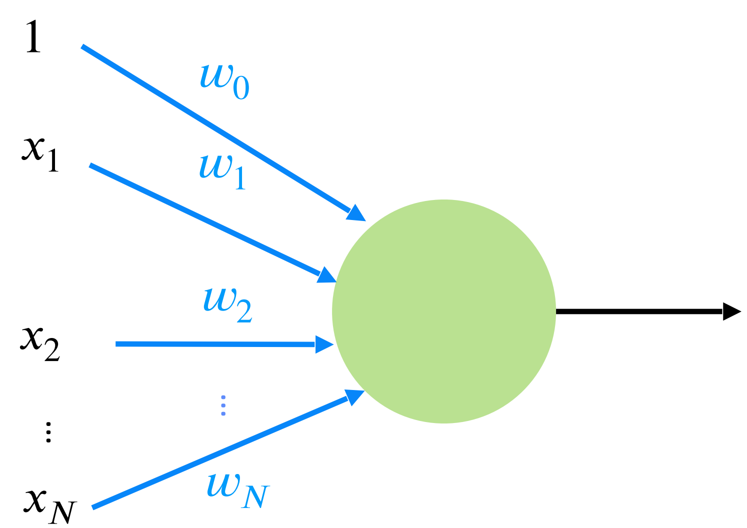
Deep Learning

- Deep learning is a subfield of machine learning that stems from artificial neural networks (ANN)

Single neuron

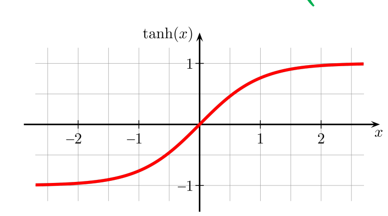


Artificial neuron



$$y = a\left(\sum_{n=1}^{N} x_n w_n + w_0\right)$$

non-linear function (activation)



If we set all the weights to 0, then $y = 0$.
If we set $w_1 = 1$ and the rest to 0, then $y = a(x_1)$.
If we set $w_0 = 0$ and the rest to 1, then $y = a(x_1 + x_2 + \ldots + x_N)$.

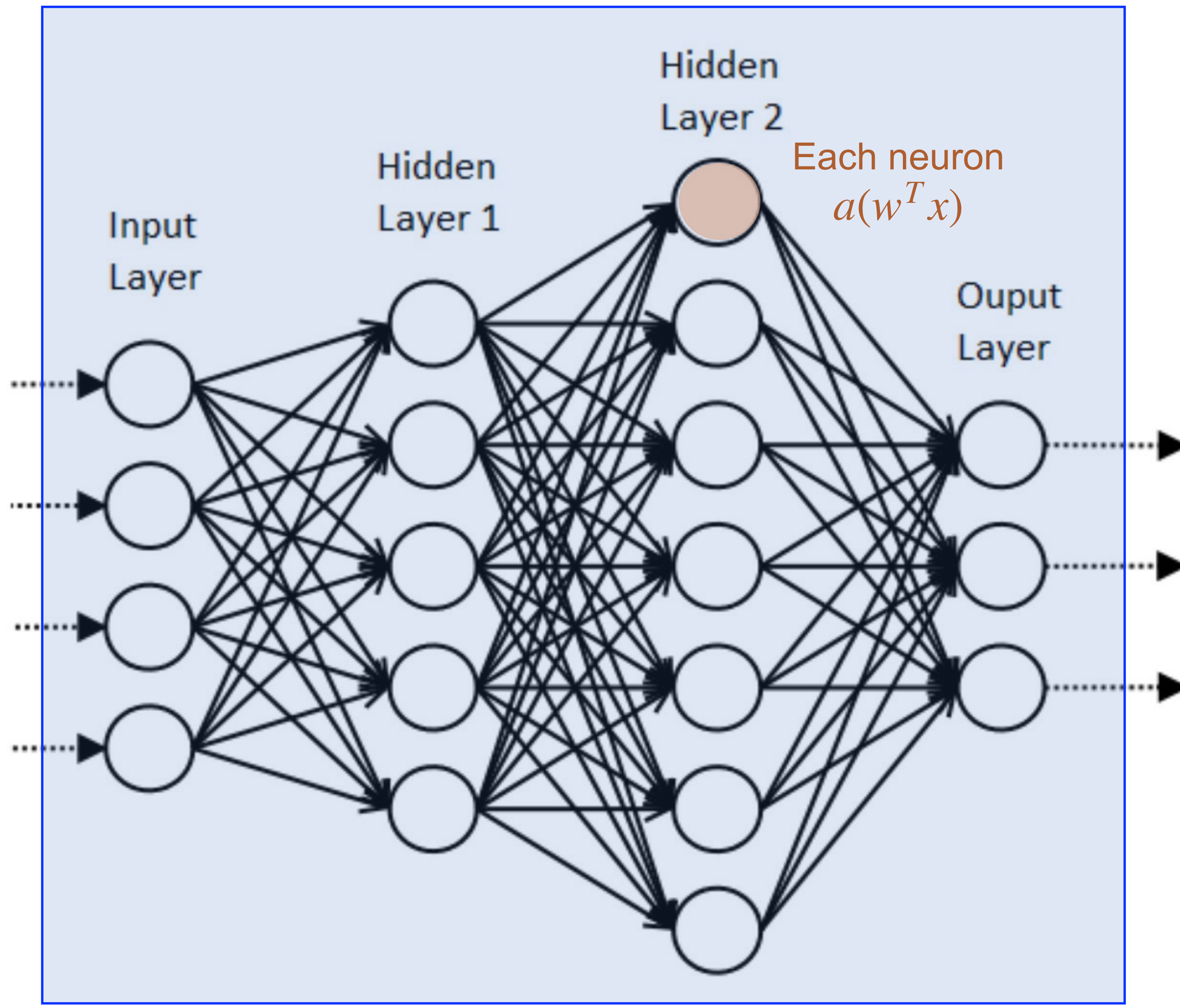
Different weights give rise to different behavior

Source: The Essense of Artificial Neural Networks by Ivan Liljeqvist (Medium)

Deep Learning

- An artificial neural network with many hidden layers is called a deep artificial neural network (ANN)

Input
x



Output
y

Weights
W

$$y = f_W(x)$$

A function that can change behavior depending on the current value of W

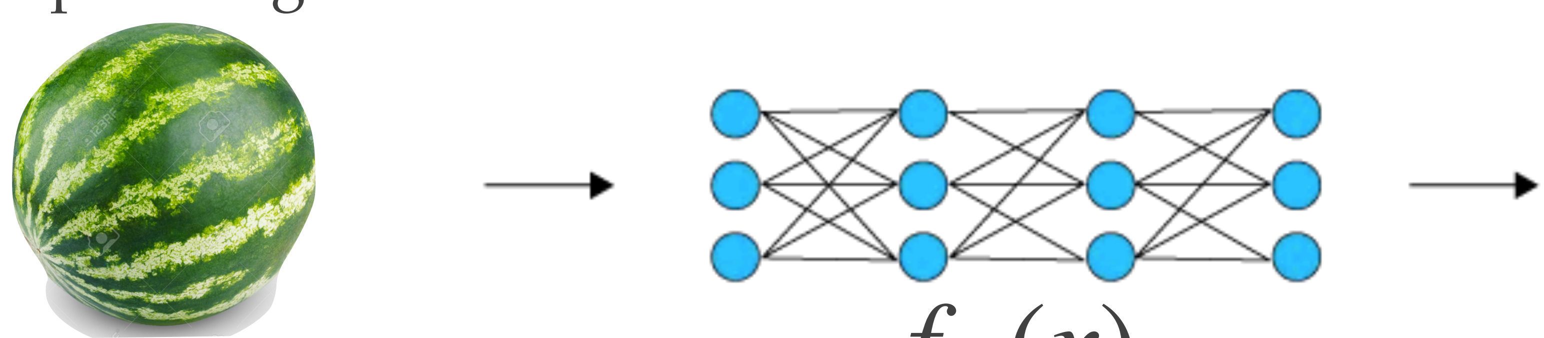
Deep Learning

- Think of a deep NN as a universal function approximator

Image classification

Input image

Output: Label



1 (watermelon)

$f_W(x)$

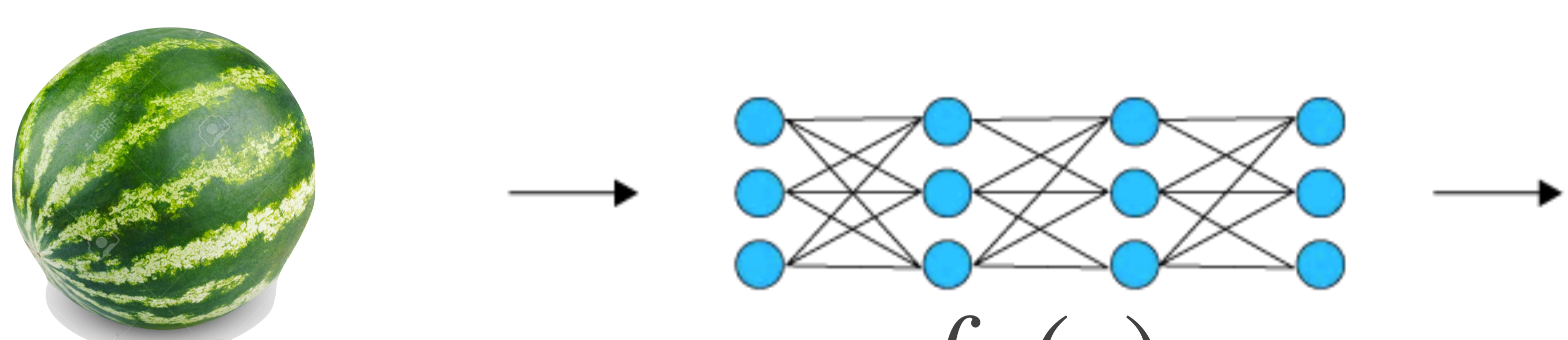
Adjustable weight

Input

Object detection

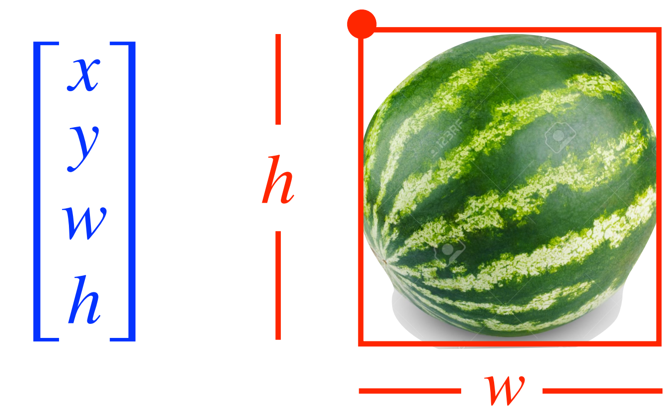
Input image

Output: rectangular box



$f_W(x)$

(x, y)



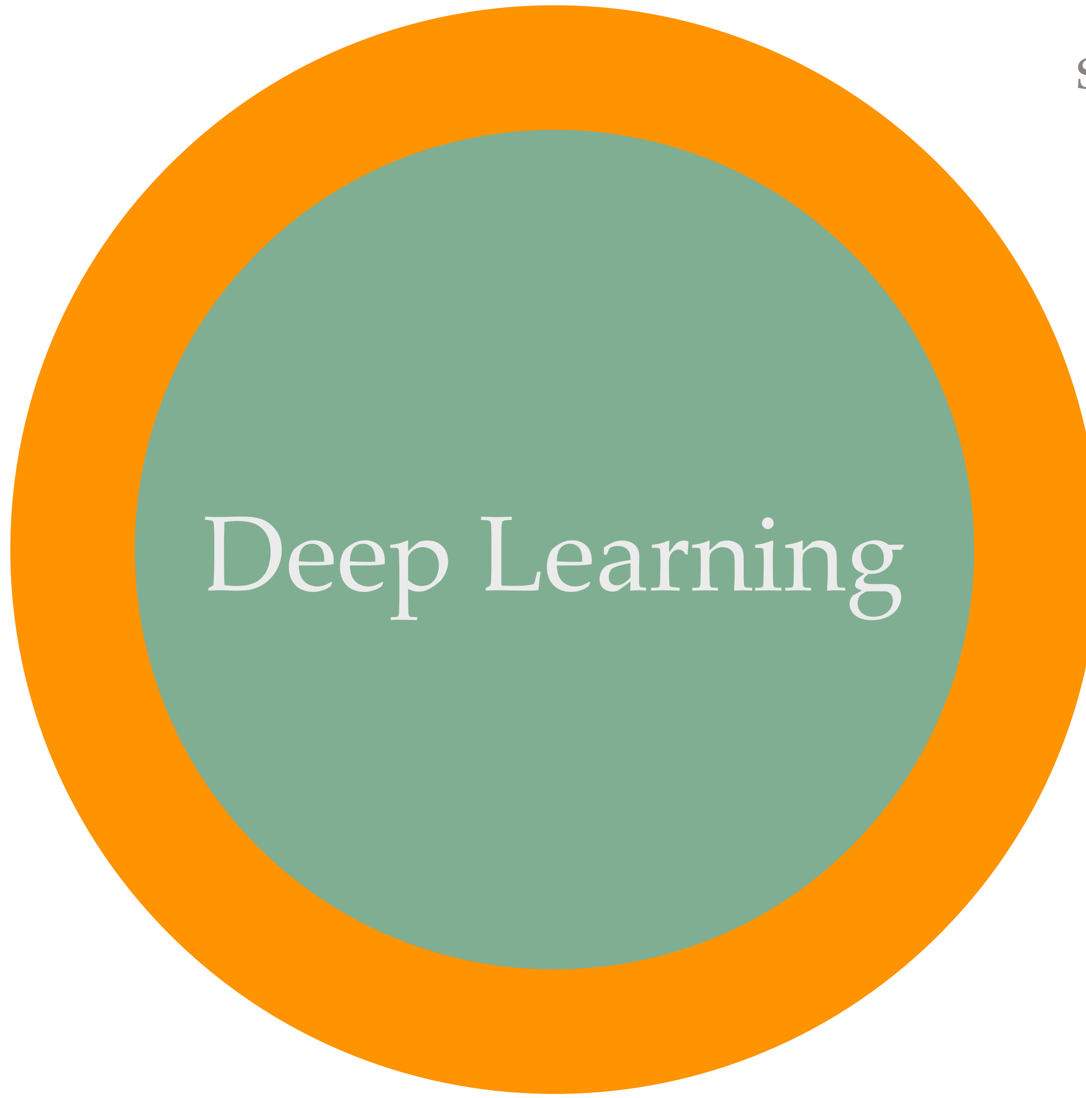
Fully connected layer
(Dense)

Convolutional layer

Conv1D, 2D, 3D, …
separable Conv

Optimizer

SGD
Adam
RMSprop



Pooling layer

max-pooling
average-pooling

Evaluation metric

accuracy
F1-score
AUC
confusion matrix

Activation function

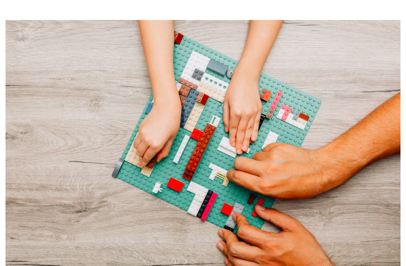
sigmoid
softmax
ESP (swish)
ReLU

Loss function

categorical crossentropy
binary crossentropy
mean squared error
mean absolute error

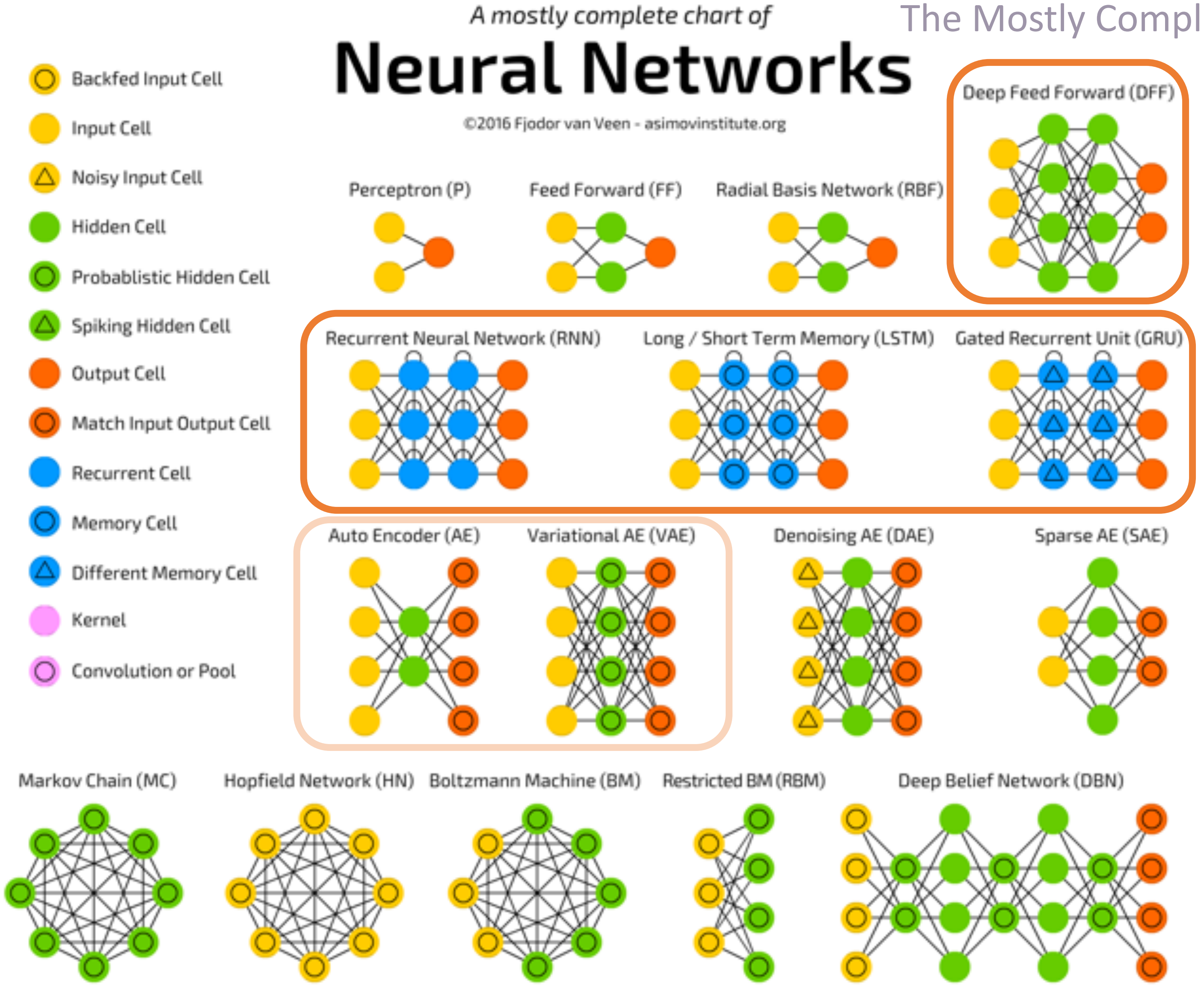
Regularization

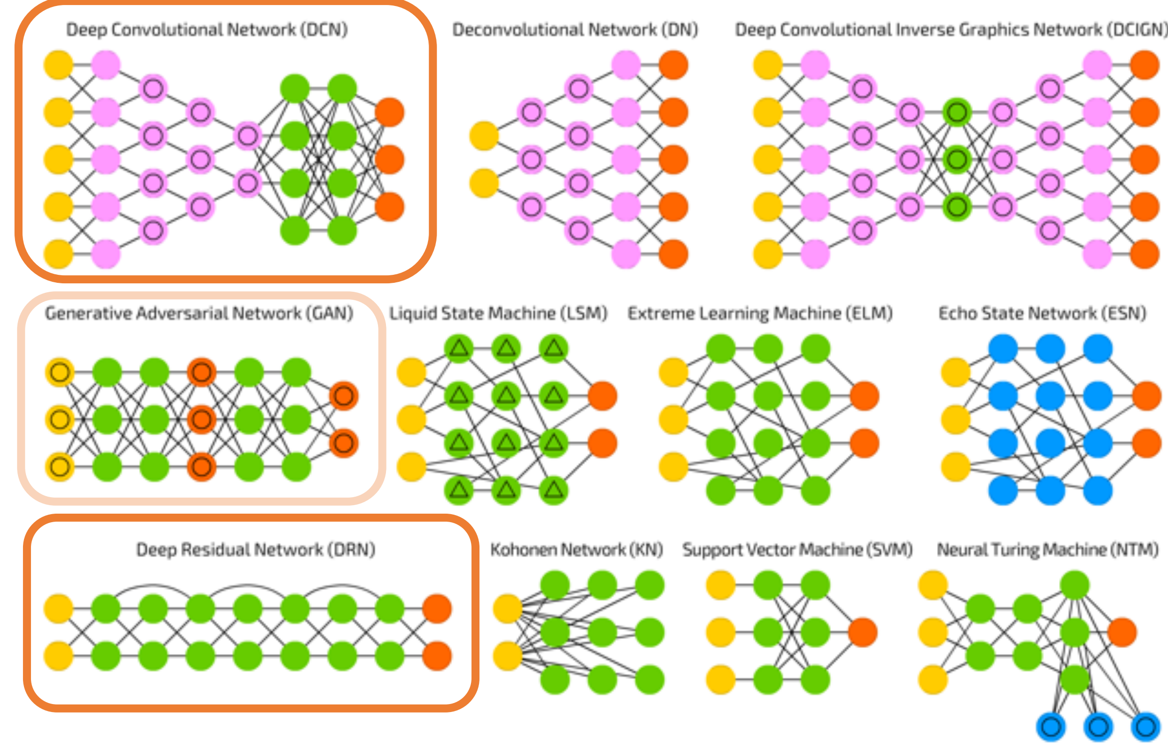
Dropout
Data augmentation
l_1, l_2 regularizations

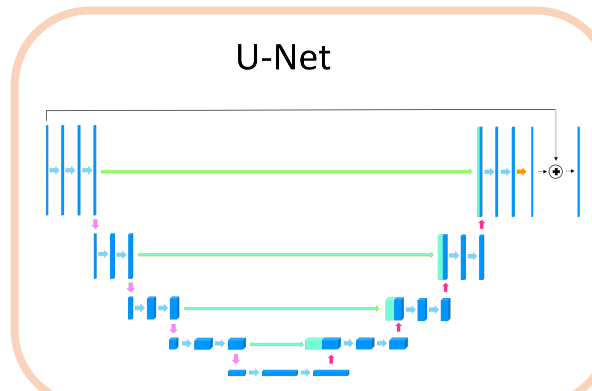


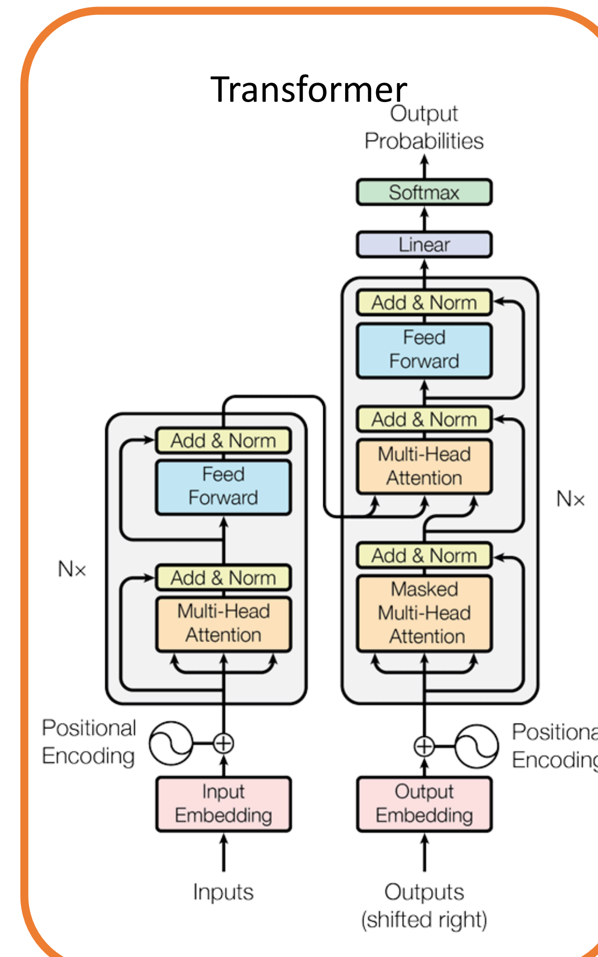
❖ Combine basic components to build a neural network
- More components → “More” representative power

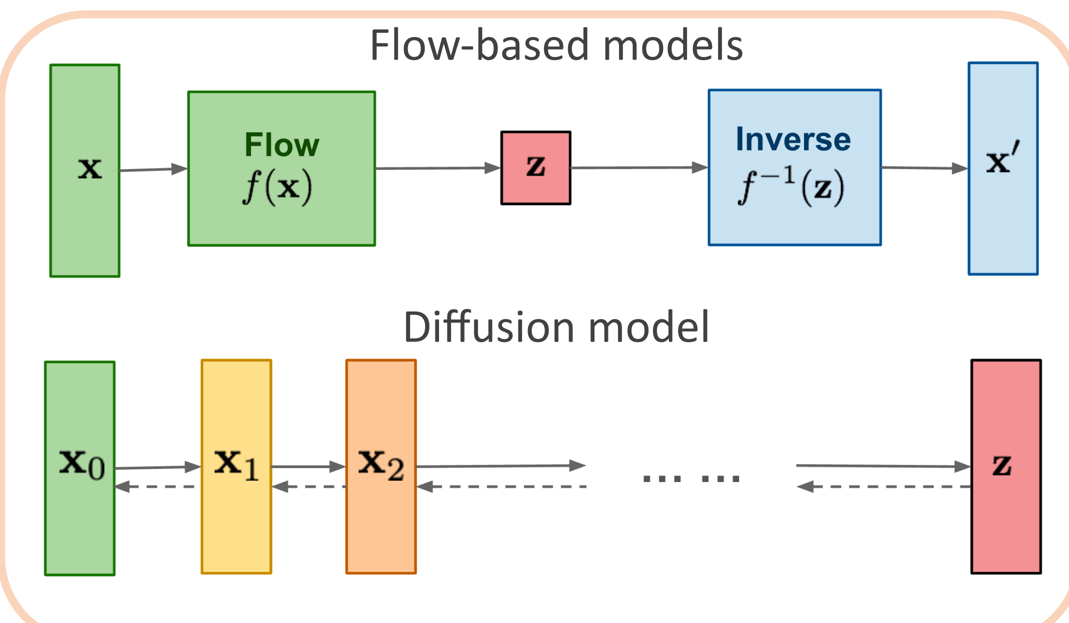
The Mostly Complete Chart of Neural Networks by the team at the Asimov Institute

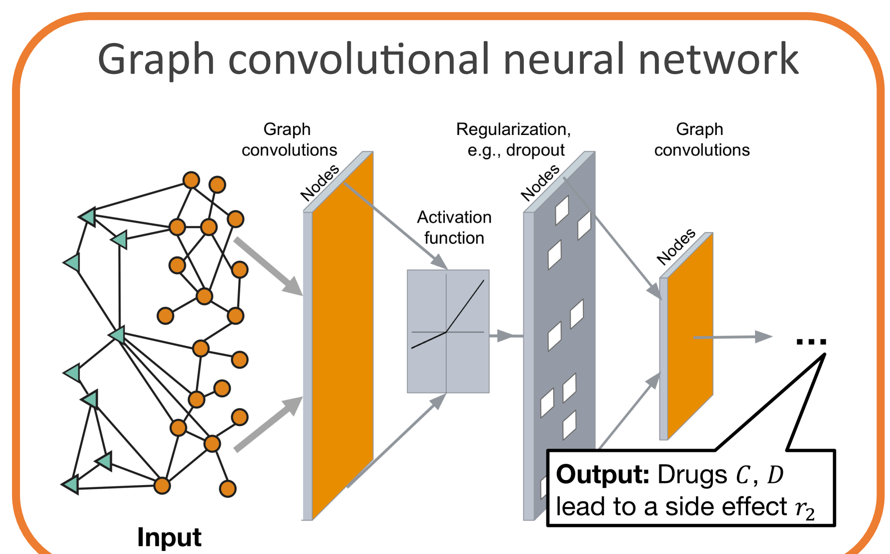












What are Diffusion Models?

Graph Neural Networks for Multirelational Link Prediction

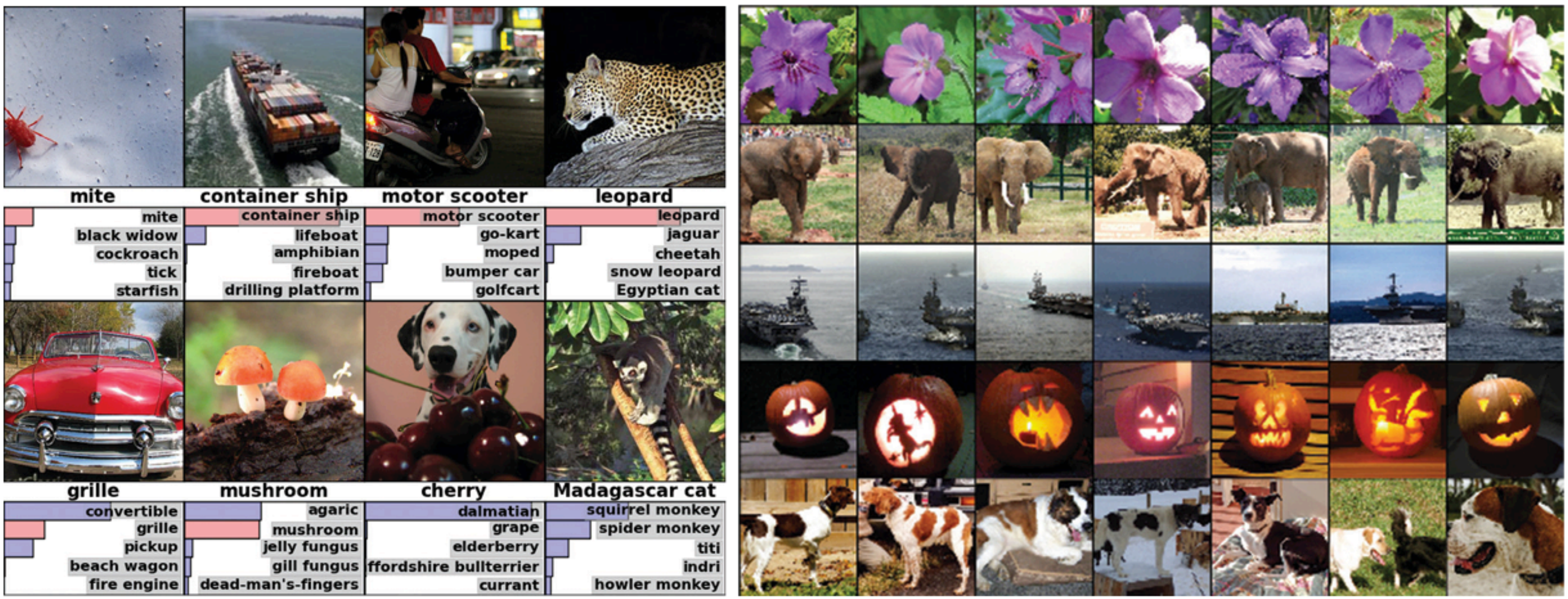
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

ImageNet

IMAGENET

1000 classes, 14,197,122 images

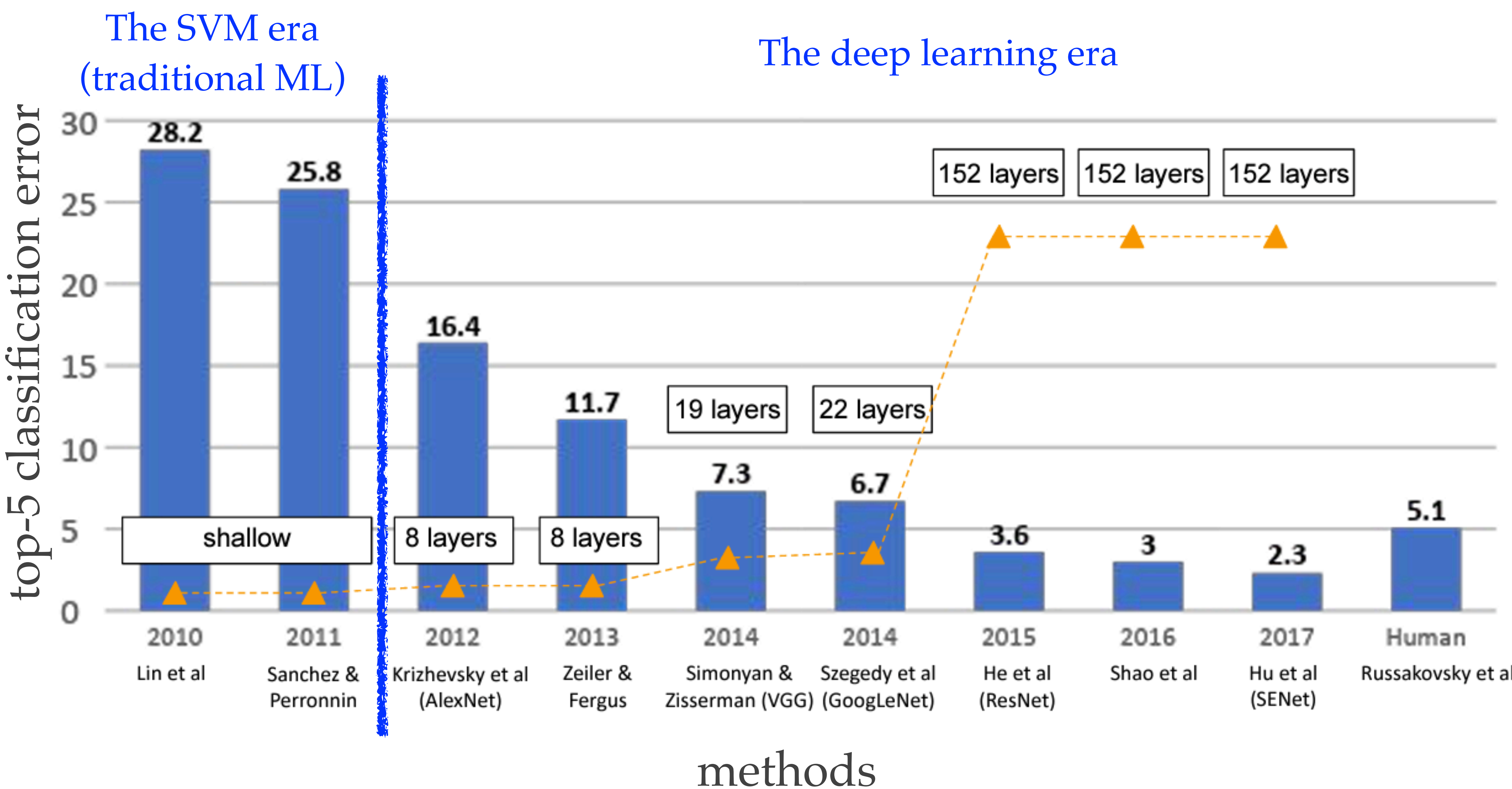
Figure 4. (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

ImageNet

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Stanford CS231n (Spring 2022): Lecture 6.

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

Supervised Model Training

Step 1: Create a neural network with some initial weight f_{W_0}

Neural network

Training iteration 0
f_{W_0}

keywords: dataset preparation, target, labels, ground truth, true, input-output pairs

Supervised Model Training

Step 2: Prepare a dataset which is a collection of input-output pairs

Prepared inputs
x

Neural network

Training iteration 0
f_{W_0}

Prepared outputs
(true labels)
y

1

0

0

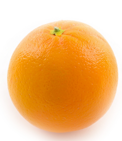
1

orange: 0
watermelon: 1

Supervised Model Training

Step 3: Pass the prepared inputs to the network

Prepared inputs
x



Neural network

Training iteration 0
f_{W_0}

Estimated outputs $\hat{y} = f_{W_0}(x)$	Prepared outputs (true labels) y
0.6	1
0.6	0
0.5	0
0.5	1

orange: 0
watermelon: 1

Supervised Model Training

Step 4: Compare $\hat{y}$ to y and modify the weights of the neural network to make $\hat{y}$ approach y using the backpropagation algorithm

A simple loss function: $L(y, \hat{y}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

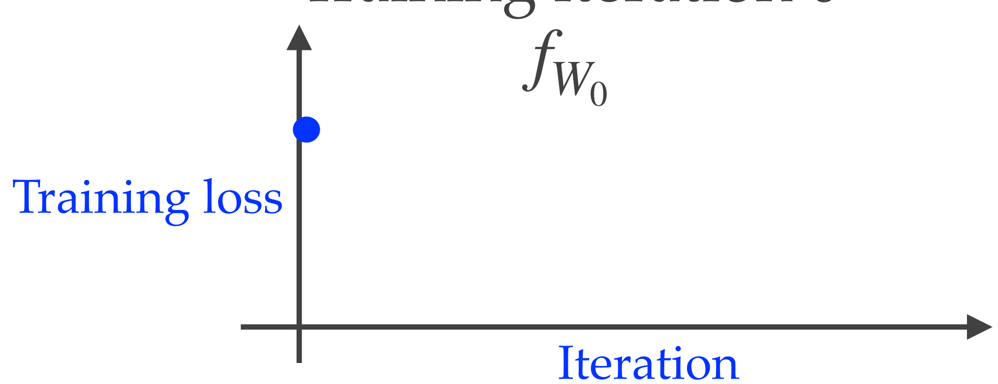
$$L(y, \hat{y}) = (0.6-1)^2 + (0.6-0)^2 + (0.5-0)^2 + (0.5-1)^2 = 1.02$$

Prepared inputs x	Estimated outputs $\hat{y} = f_{W_0}(x)$	Prepared outputs (true labels) y
	0.6	1
	0.6	0
	0.5	0
	0.5	1

Neural network

Training iteration 0

f_{W_0}



orange: 0

watermelon: 1

Supervised Model Training

Repeat steps 3 and 4 to continuously improve the weights of the neural network

Prepared inputs
x

A simple loss function: $L(y, \hat{y}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

$L(y, \hat{y}) = (0.6 - 1)^2 + (0.4 - 0)^2 + (0.4 - 0)^2 + (0.6 - 1)^2 = 0.64$

Estimated outputs
$\hat{y} = f_{W_1}(x)$

Prepared outputs
(true labels)
y

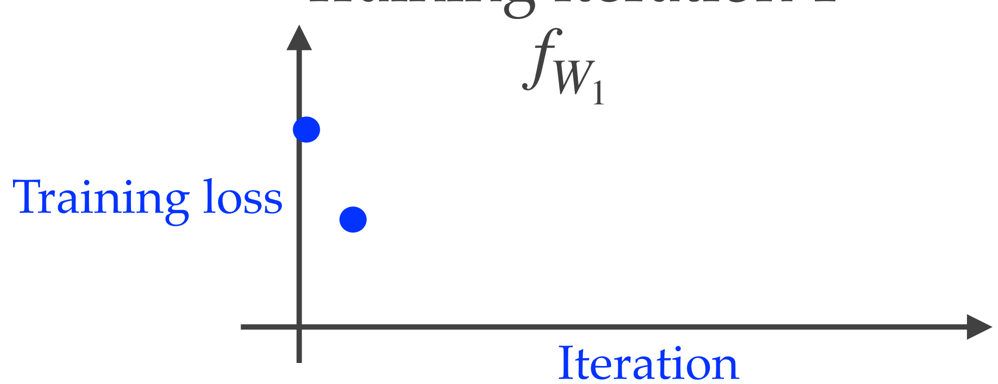
0.6 | 1

Neural network

0.4 | 0

0.4 | 0

Training iteration 1
f_{W_1}



0.6 | 1

orange: 0
watermelon: 1

Supervised Model Training

Repeat steps 3 and 4 to continuously improve the weights of the neural network

Prepared inputs
x

A simple loss function: $L(y, \hat{y}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

$L(y, \hat{y}) = (0.7 - 1)^2 + (0.35 - 0)^2 + (0.4 - 0)^2 + (0.6 - 1)^2 = 0.5325$

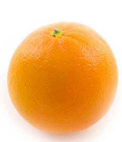
Estimated outputs
$\hat{y} = f_{W_2}(x)$

Prepared outputs
(true labels)
y

0.7 1

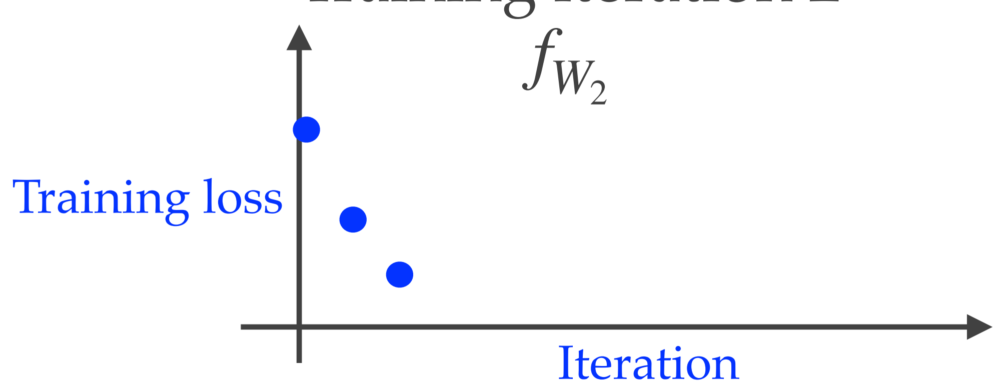
Neural network

0.35 0



0.4 0

Training iteration 2
f_{W_2}



0.6 1

orange: 0
watermelon: 1

Supervised Model Training

Repeat steps 3 and 4 to continuously improve the weights of the neural network

A simple loss function: $L(y, \hat{y}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

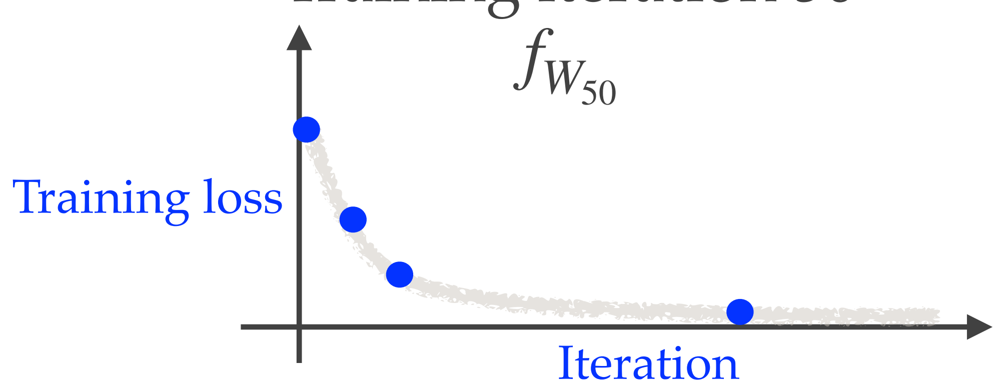
$L(y, \hat{y}) = (0.8 - 1)^2 + (0.15 - 0)^2 + (0.1 - 0)^2 + (0.9 - 1)^2 = 0.0825$

Prepared inputs x	Estimated outputs $\hat{y} = f_{W_{50}}(x)$	Prepared outputs (true labels) y
	0.8	1
	0.15	0
	0.1	0
	0.9	1

Neural network

Training iteration 50

$f_{W_{50}}$



orange: 0
watermelon: 1

Supervised Model Training

Repeat steps 3 and 4 to continuously improve the weights of the neural network

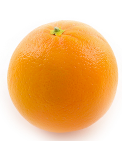
Prepared inputs
x

A simple loss function: $L(y, \hat{y}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

$L(y, \hat{y}) = (0.99 - 1)^2 + (0.05 - 0)^2 + (0.01 - 0)^2 + (0.97 - 1)^2 = 0.0036$

Estimated outputs
$\hat{y} = f_{W_{1000}}(x)$

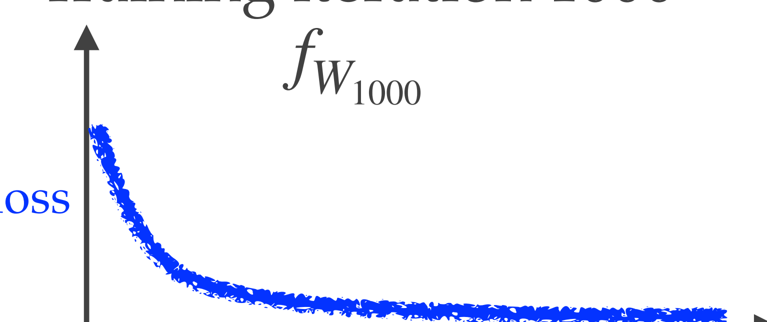
Prepared outputs
(true labels)
y



Neural network

Training iteration 1000

$f_{W_{1000}}$



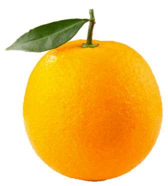
Estimated outputs	Prepared outputs
0.99	1
0.05	0
0.01	0
0.97	1

orange: 0
watermelon: 1

keywords: overfitting, underfitting, regularization, early stopping, data augmentation, training/validation/test data

Test the Trained Model

Test image (unseen)



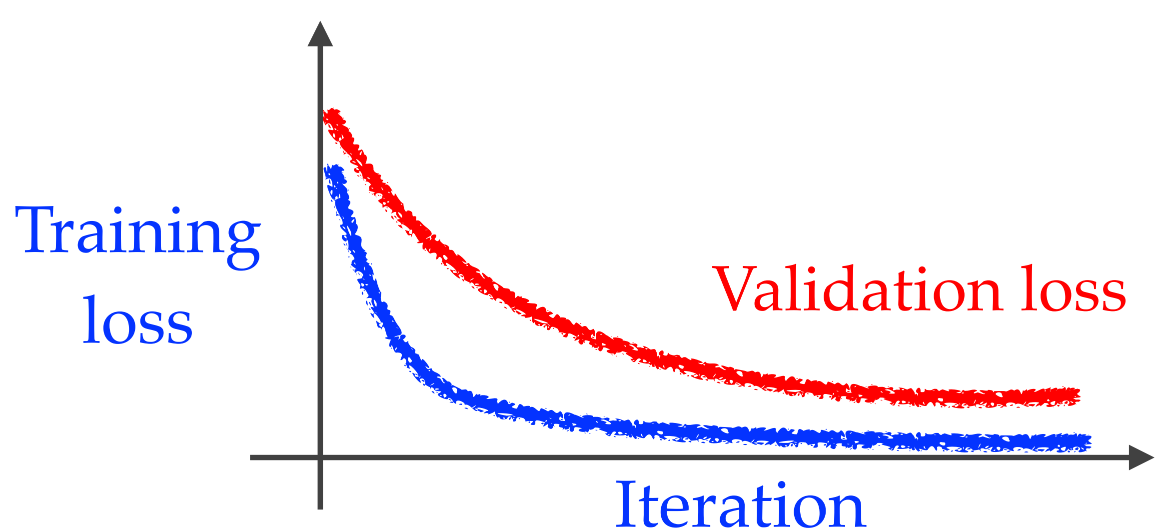
Trained neural network

Predicted Label

0.01

less than 0.5, so it is likely orange

orange: 0
watermelon: 1

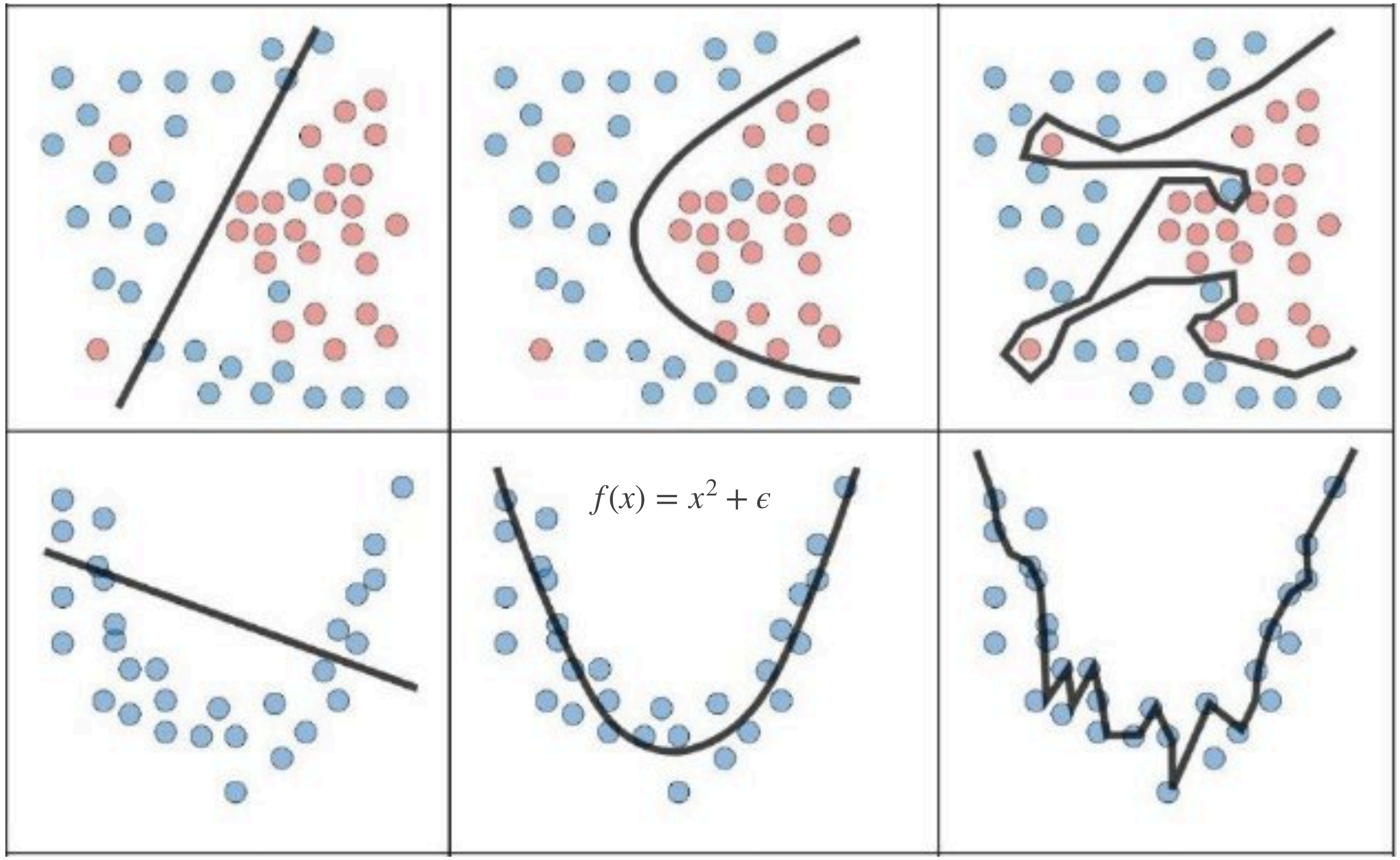


https://i.pinimg.com/originals/72/e2/22/72e222c1542539754df1d914cb671bd7.png

	Underfitting		Overfitting

Classification

Regression



Likely symptoms

- Underfitting:
  - High training error
  - Training error close to test error
- Middle:
  - Training error slightly lower than test error
- Overfitting:
  - Low training error
  - Test error much higher than training error

A way to check for the deep-learning-based methods (both classification and regression)

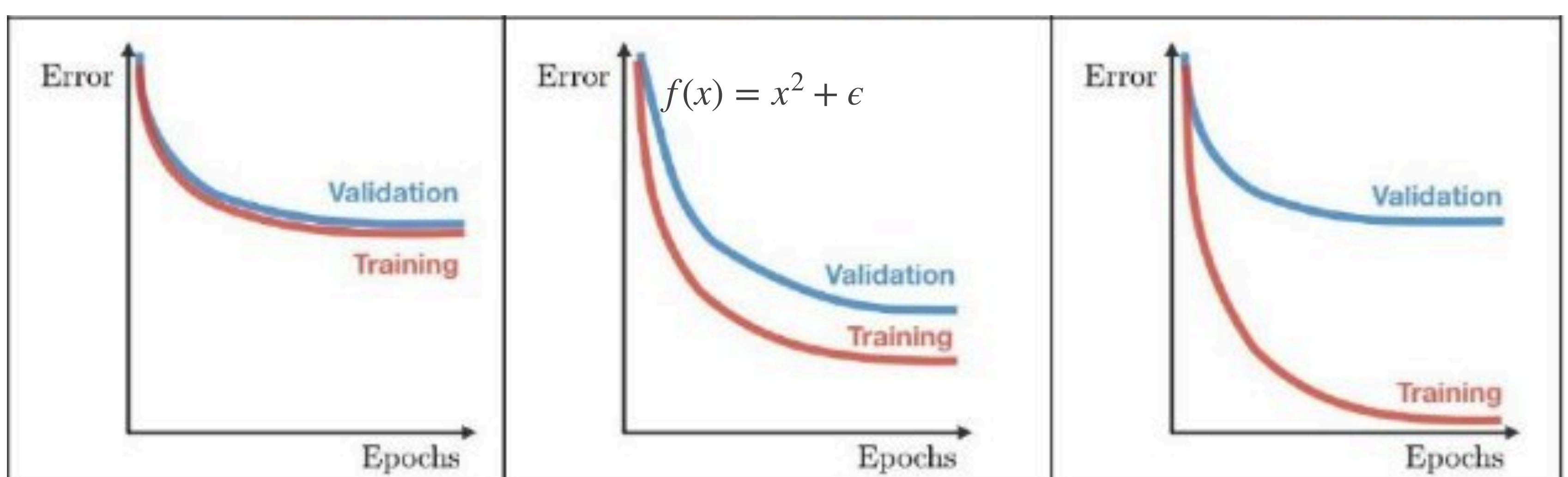
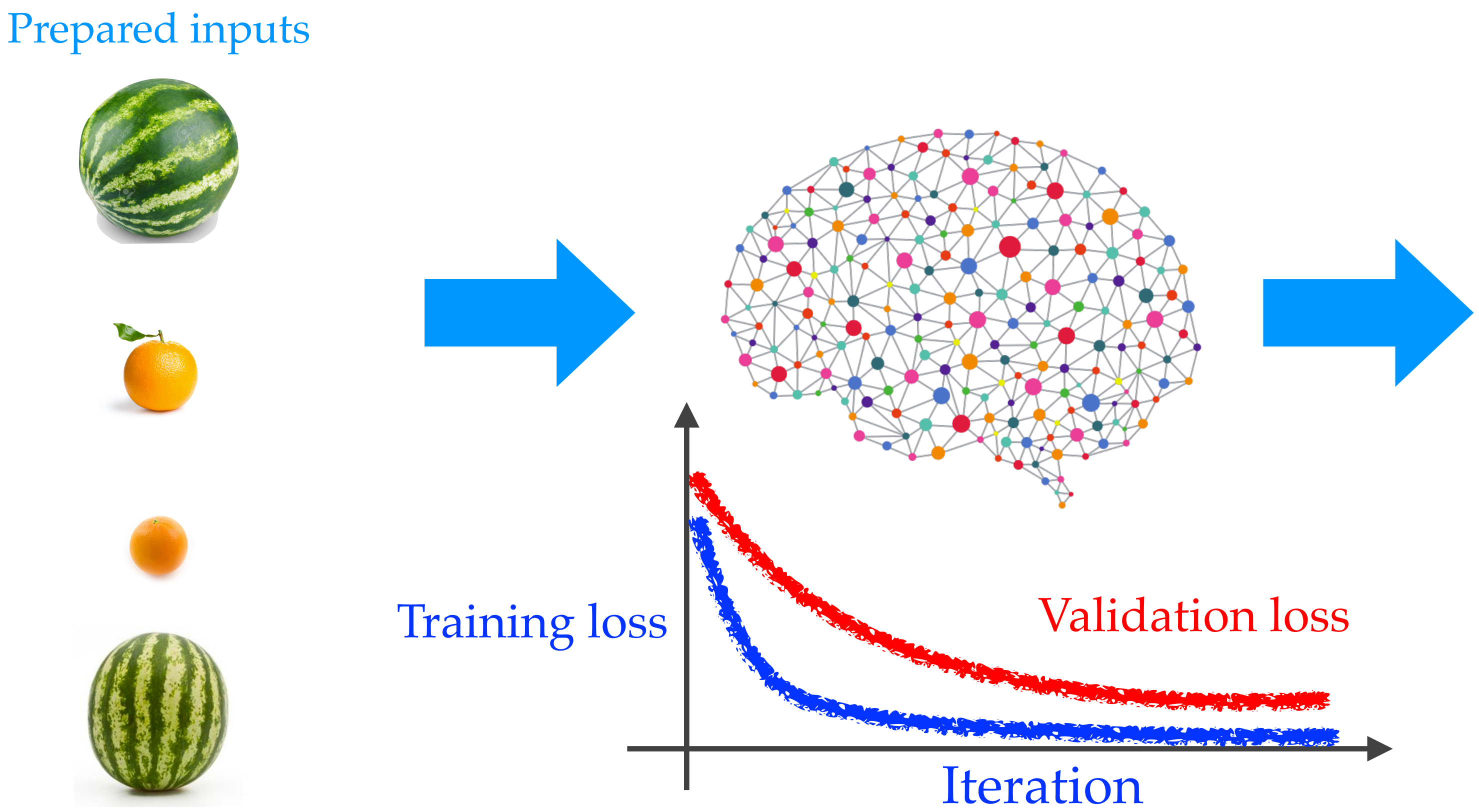


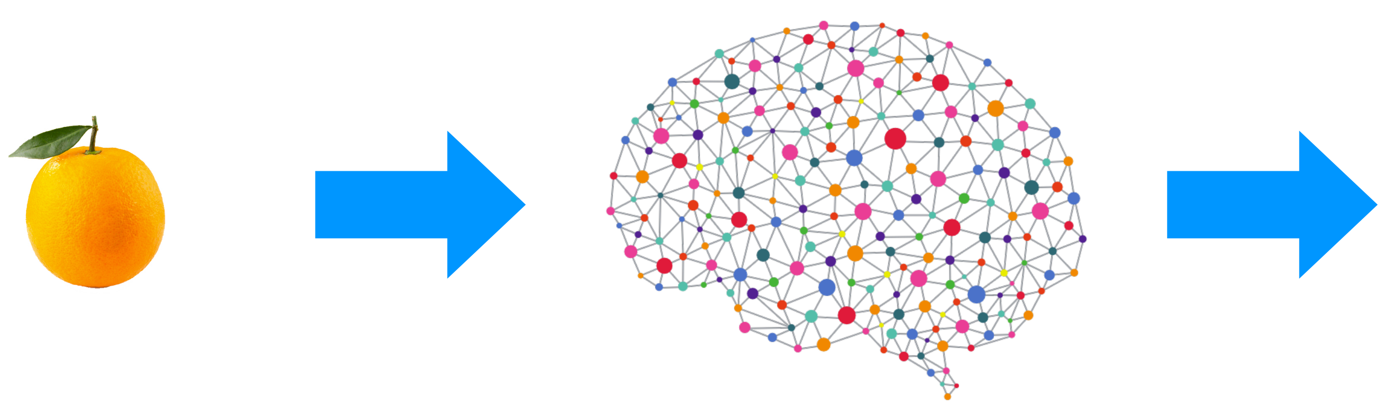
Image Classification Using Deep Learning

- Training phase: Optimize the weights of a deep neural network



Estimated outputs	Prepared outputs
0.7	1
0.2	0
0.1	0
0.9	1

- Test phase: Perform prediction using the trained neural network



0.01

orange!

Outline

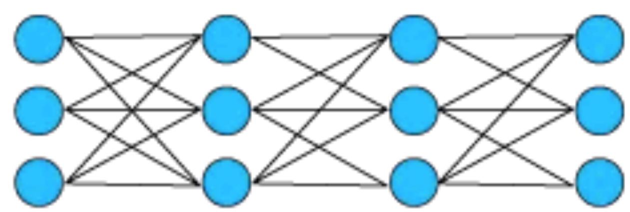
- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

Deep Learning

- Train a deep neural network (NN) to perform some task from data
- Think of a deep NN as a universal function approximator

Image classification

Input: image



$f_W(x)$

Output: Label

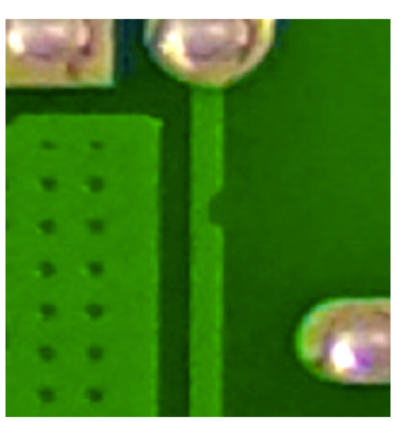
1: good condition

Adjustable weight

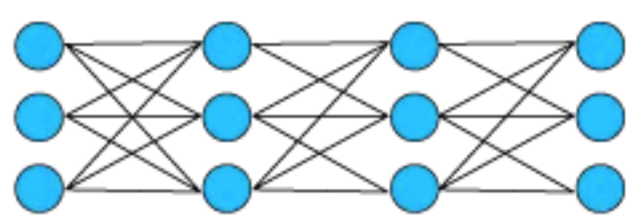
Input

Defect detection

Input: image



Huang, Weibo, and Peng Wei. "A PCB dataset for defects detection and classification." arXiv preprint arXiv:1901.08204 (2019).

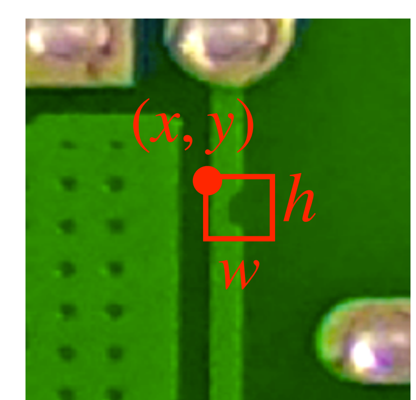


$f_W(x)$

Output: rectangular box(es) and the corresponding class(es)

$$\begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix}$$

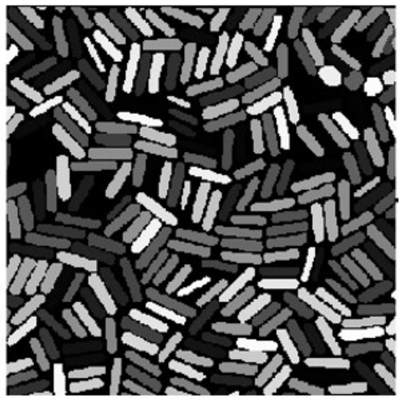
corresponding class



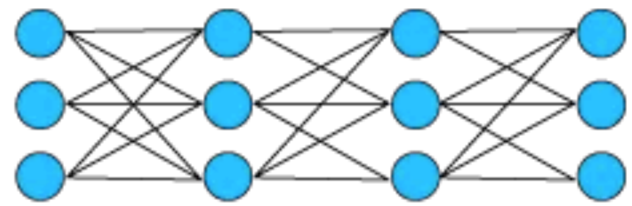
Electron Microscopy Particle Segmentation

Input

EM image



Yildirim, Batuhan, and Jacqueline M. Cole. "Bayesian particle instance segmentation for electron microscopy image quantification." Journal of Chemical Information and Modeling 61.3 (2021): 1136-1149.



$f_W(x)$

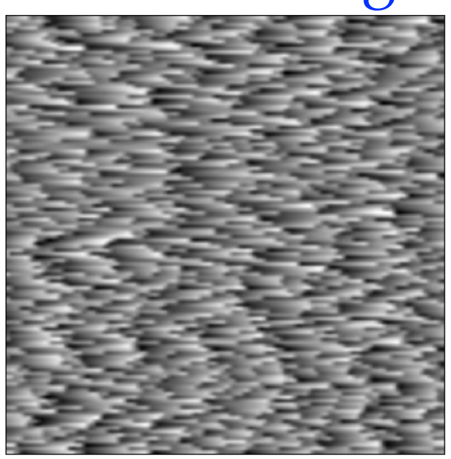
Output

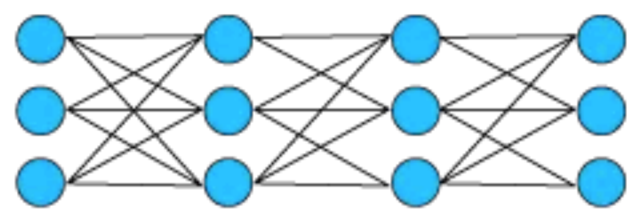
segmentation map

Super-resolution

Input

low-res image

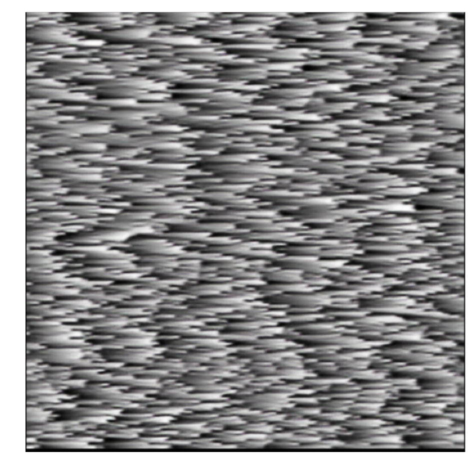




$f_W(x)$

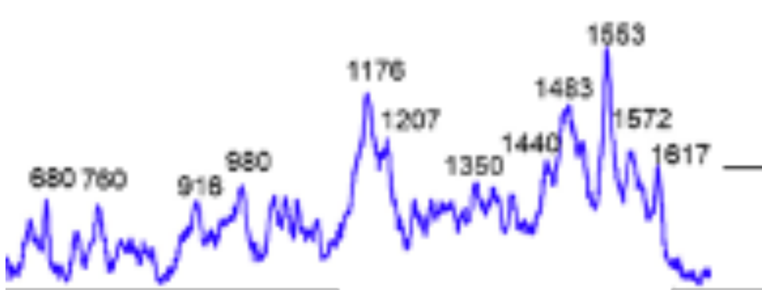
Output

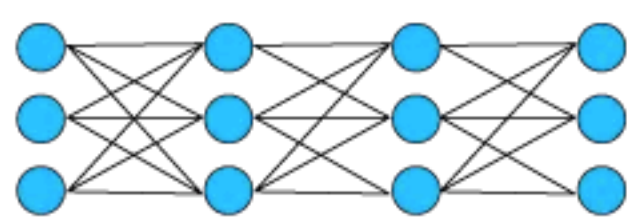
high-res image



Spectrum classification

Input





$f_W(x)$

Output

Category

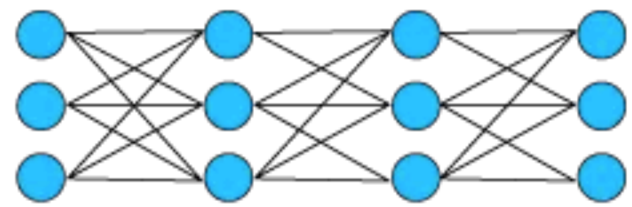
Synthesis parameter optimization

Min, Kyoungmin, et al. "Machine learning assisted optimization of electrochemical properties for Ni-rich cathode materials." Scientific reports 8.1 (2018): 1-7.

Input

Experimental parameters

- Composition
- Temperature
- Dopant
- Washing
- Coating
- ICP
- XRD



$$f_W(x)$$

Output

Electrochemical properties

- The 1st discharge capacity
- Capacity retention rate
- Amount of residual Li after initial synthesis

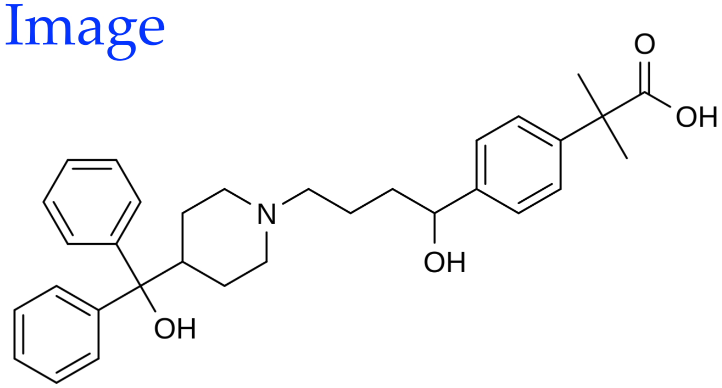
Molecule classification

Input

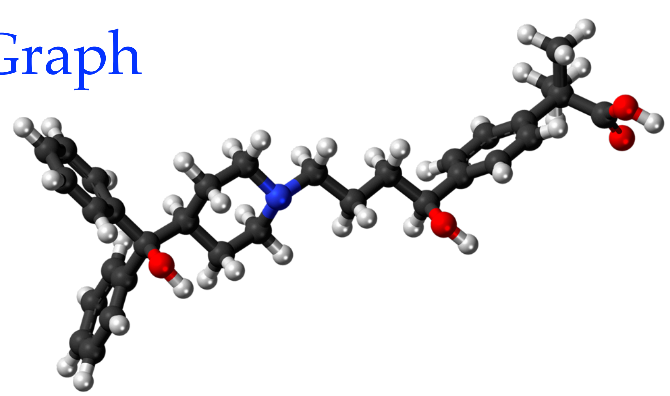
String of characters

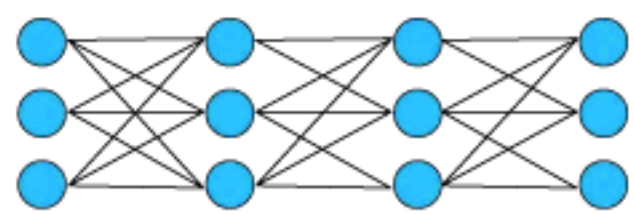
$C_{32}H_{39}NO_4$

Image



Graph





$$f_W(x)$$

Output

2nd-generation antihistamine drug

Outline

- Introduction to Supervised Machine Learning
  - AI vs ML vs DL
  - Traditional Machine Learning
  - Training, Validation, and Test Data
  - Overfitting and Underfitting
- Introduction to Supervised Deep Learning
  - Traditional ML vs Deep Learning
  - Artificial Neuron and Neural Network
  - Supervised Learning
- Deep Learning as a Function Approximator

What's Next for Deep Learning?

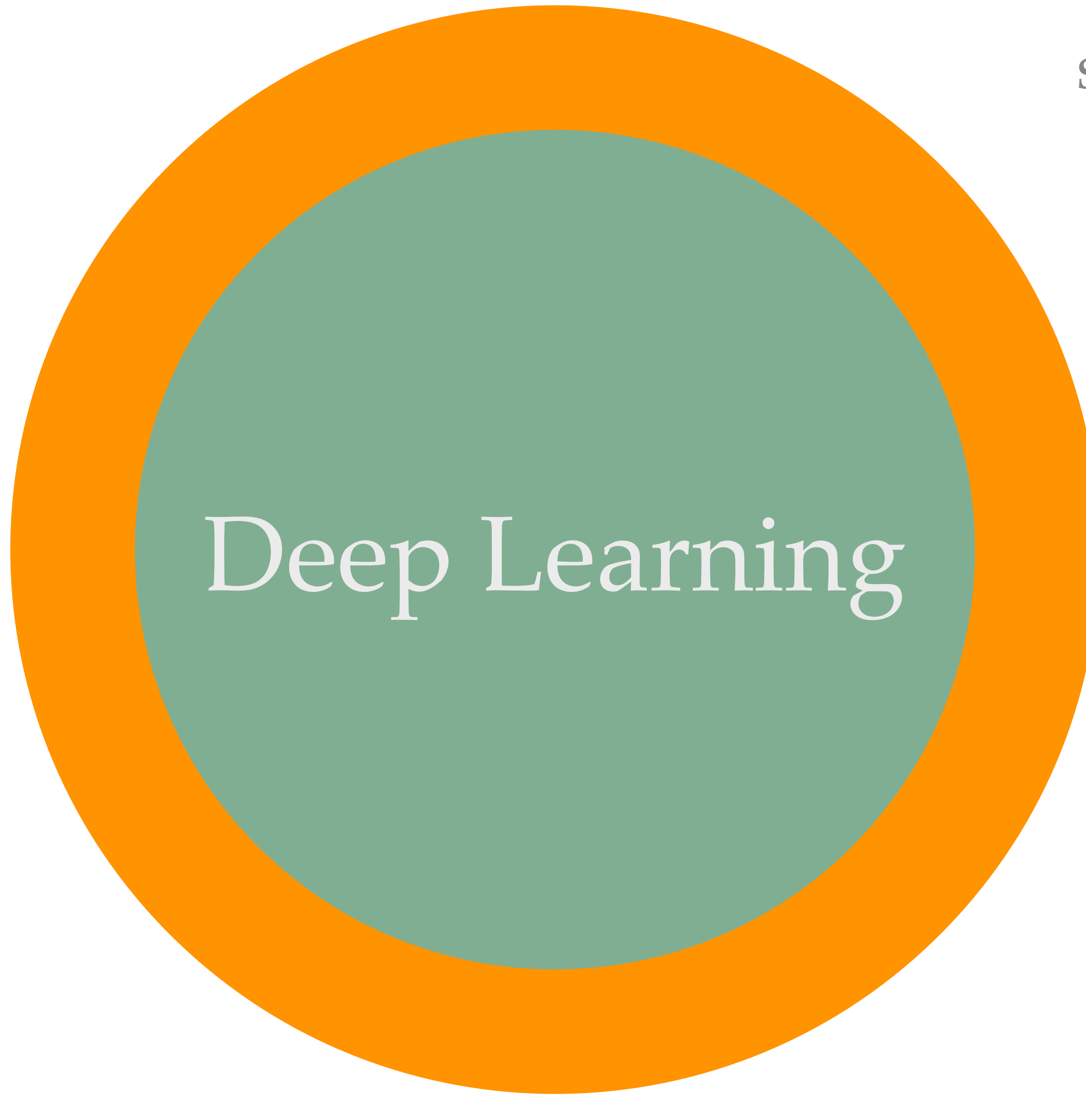
Fully connected layer (Dense)

Convolutional layer

Conv1D, 2D, 3D, …
separable Conv

Optimizer

SGD
Adam
RMSprop



Pooling layer

max-pooling
average-pooling

Evaluation metric

accuracy
F1-score
AUC
confusion matrix

Activation function

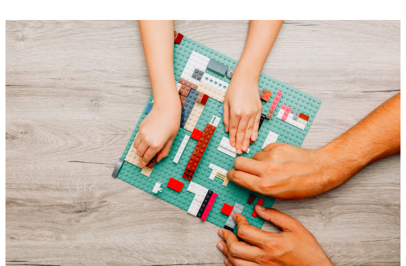
sigmoid
softmax
ESP (swish)
ReLU

Loss function

categorical crossentropy
binary crossentropy
mean squared error
mean absolute error

Regularization

Dropout
Data augmentation
l_1, l_2 regularizations



- Combine basic components to build a neural network
  - More components → "More" representative power