Introduction to DL

DB/AI Bootcamp

2018, Summer

WD Chen, TY Lin @ DataLab, CS, NTHU

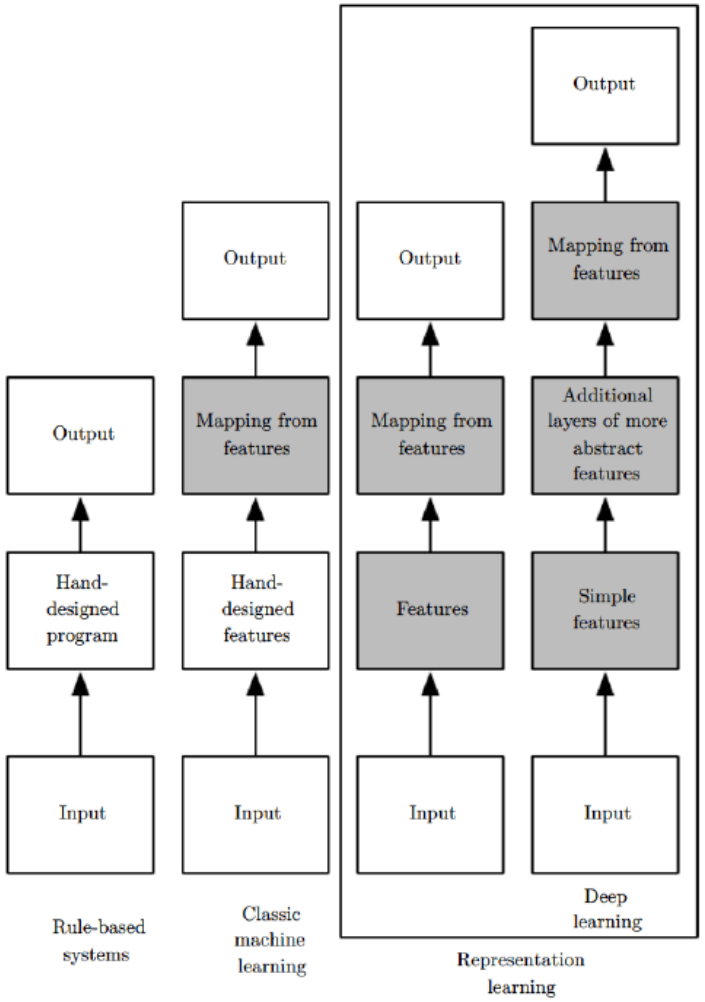
Outline

- The Rise of Deep Learning
  - Representation Learning
  - Transfer Learning
  - Curse of Dimensionality
- Basics of Neural Networks
  - Perceptron
  - Optimization - Gradient Descent
  - Multi-layer Perceptron
  - Optimization - Backpropagation
- Main Stream Models
  - Fully Connected Network (FC)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Generative Adversarial Network (GAN)

Outline

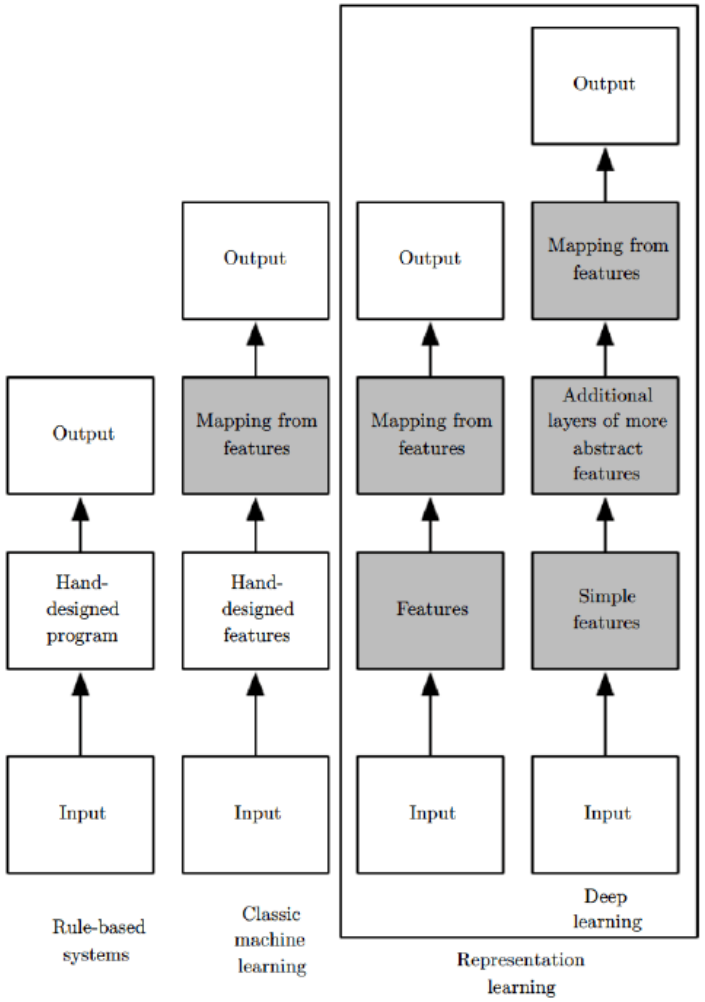
- The Rise of Deep Learning
  - Representation Learning
  - Transfer Learning
  - Curse of Dimensionality
- Basics of Neural Networks
  - Perceptron
  - Optimization - Gradient Descent
  - Multi-layer Perceptron
  - Optimization - Backpropagation
- Main Stream Models
  - Fully Connected Network (FC)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Generative Adversarial Network (GAN)

Representation Learning



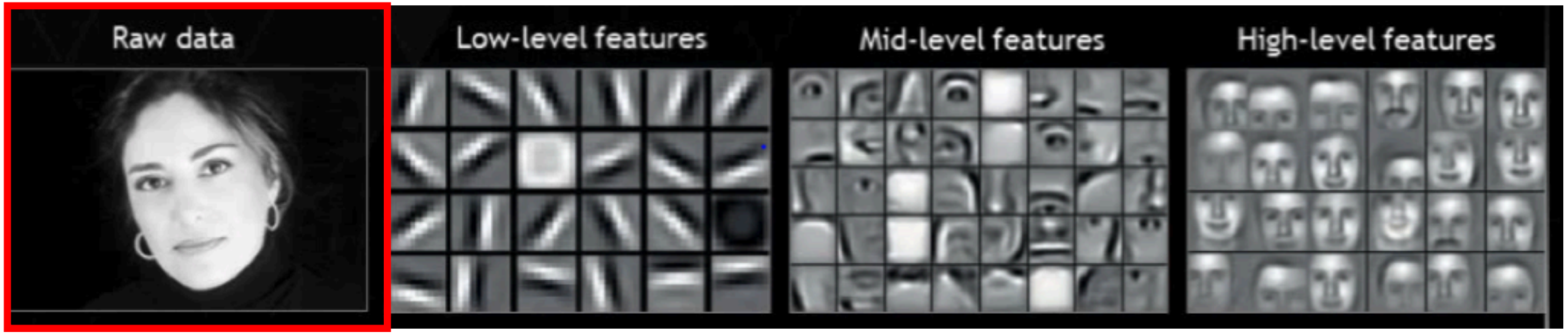
- For traditional machine learning, we need to do some feature engineering to get the feature
- Gray boxes are learned automatically.

Representation Learning



- For traditional machine learning, we need to do some feature engineering to get the feature
- Gray boxes are learned automatically.
- DL learns high-level features by **reusing** the low-level ones, and maps the most abstract (deepest) features to the output.

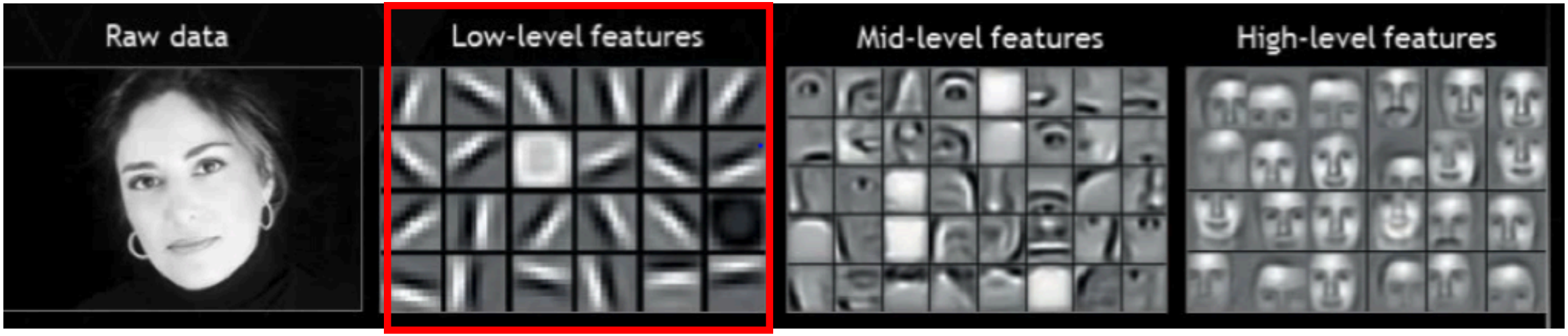
Representation Learning



Input feature is pixel

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, ICML 2009

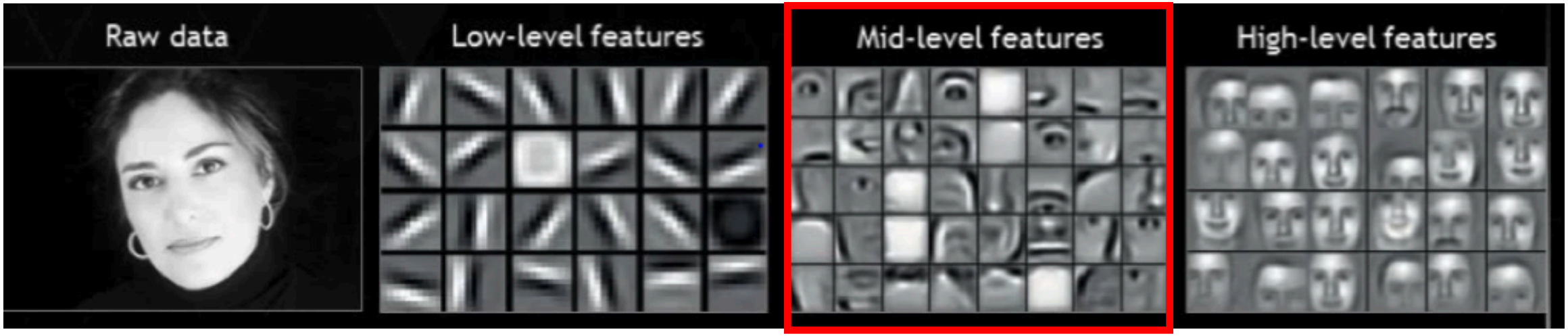
Representation Learning



Different edges

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, ICML 2009

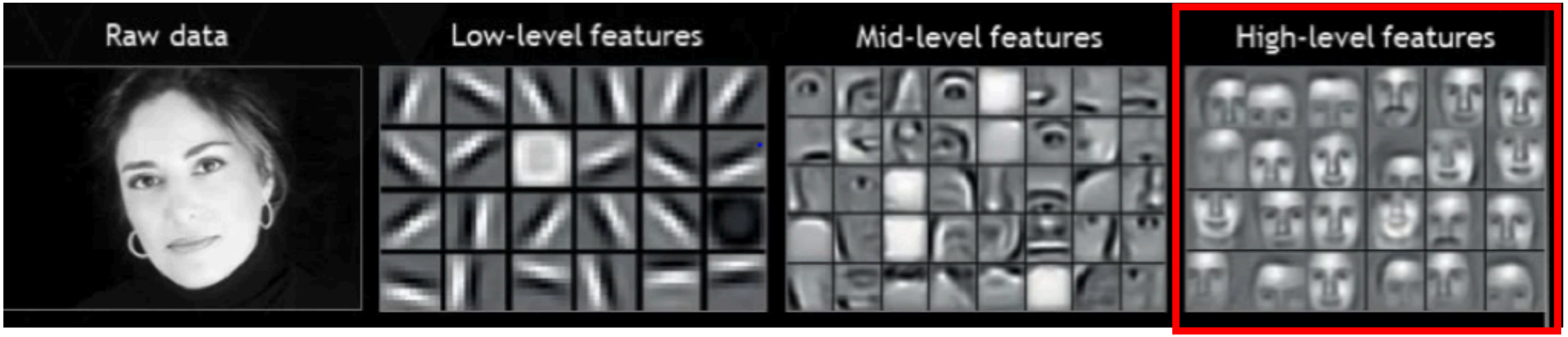
Representation Learning



Ear, nose, eyes

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, ICML 2009

Representation Learning

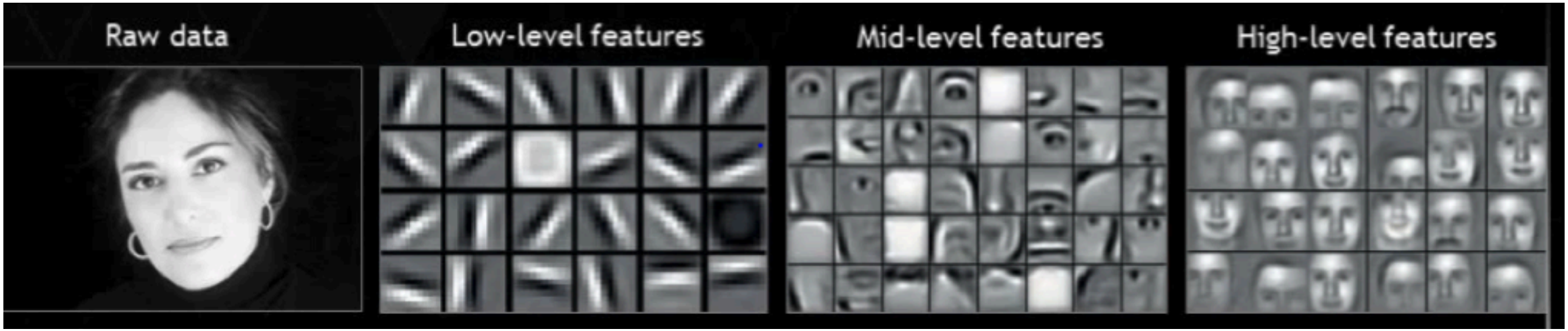


Different faces

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, ICML 2009

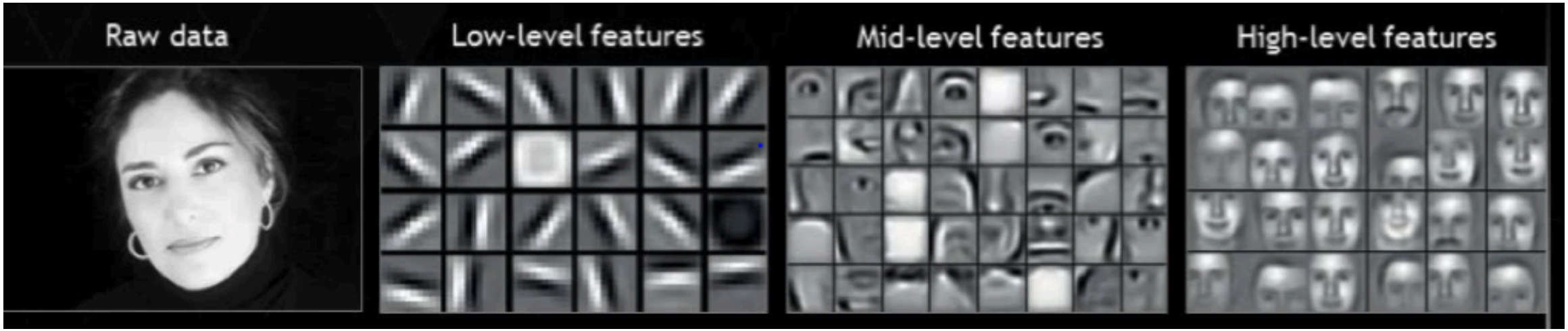
Transfer Learning

- Transfer learning: to reuse the knowledge learned from a task to help another task



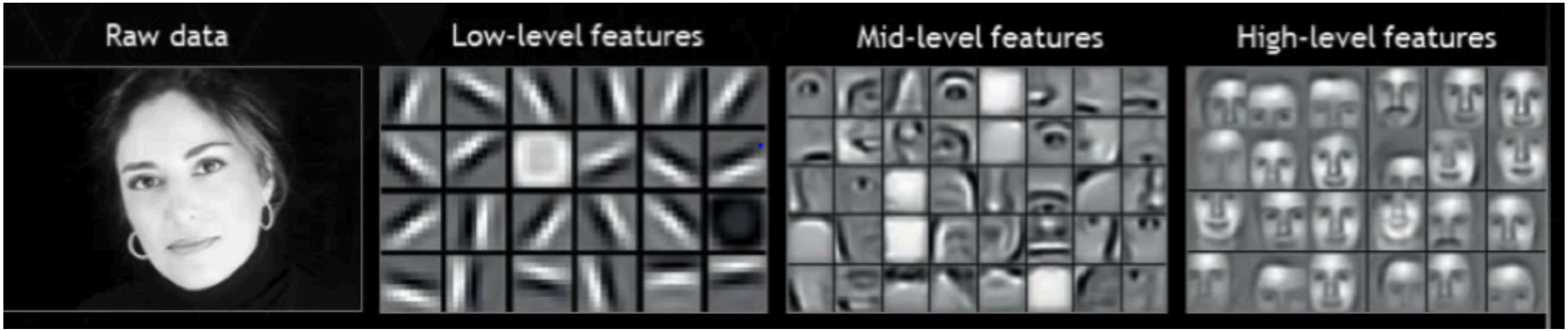
Transfer Learning

- Transfer learning: to reuse the knowledge learned from a task to help another task
- In deep learning, it's common to reuse the feature extraction network from one task in another.



Transfer Learning

- Transfer learning: to reuse the knowledge learned from a task to help another task
- In deep learning, it's common to reuse the feature extraction network from one task in another.
  - Weights may be further updated when training model in a new task.



Curse of Dimensionality

- Cause by the volume of the space increasing exponentially when the data dimension grows high, the learning task will suffer from the curse of dimensionality.

Curse of Dimensionality

- Cause by the volume of the space increasing exponentially when the data dimension grows high, the learning task will suffer from the curse of dimensionality.
- Through representation learning, DL can form the **high-level features** which has **lower dimension**
  - Overcome the curse of dimensionality

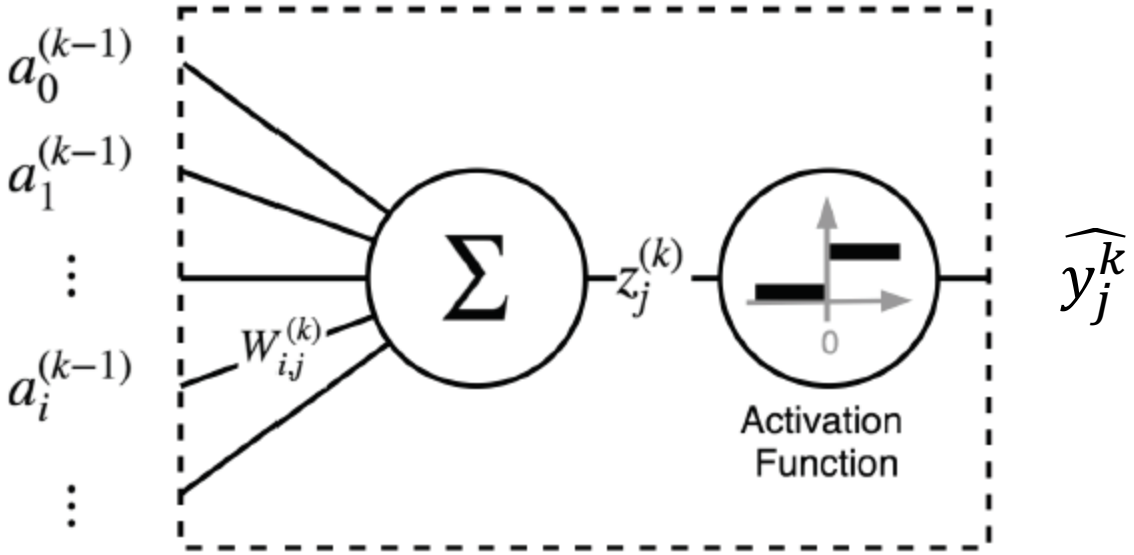
Outline

- The Rise of Deep Learning
  - Representation Learning
  - Transfer Learning
  - Curse of Dimensionality
- Basics of Neural Networks
  - Perceptron
  - Optimization - Gradient Descent
  - Multi-layer Perceptron
  - Optimization - Backpropagation
- Main Stream Models
  - Fully Connected Network (FC)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Generative Adversarial Network (GAN)

Perceptron

- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1

$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

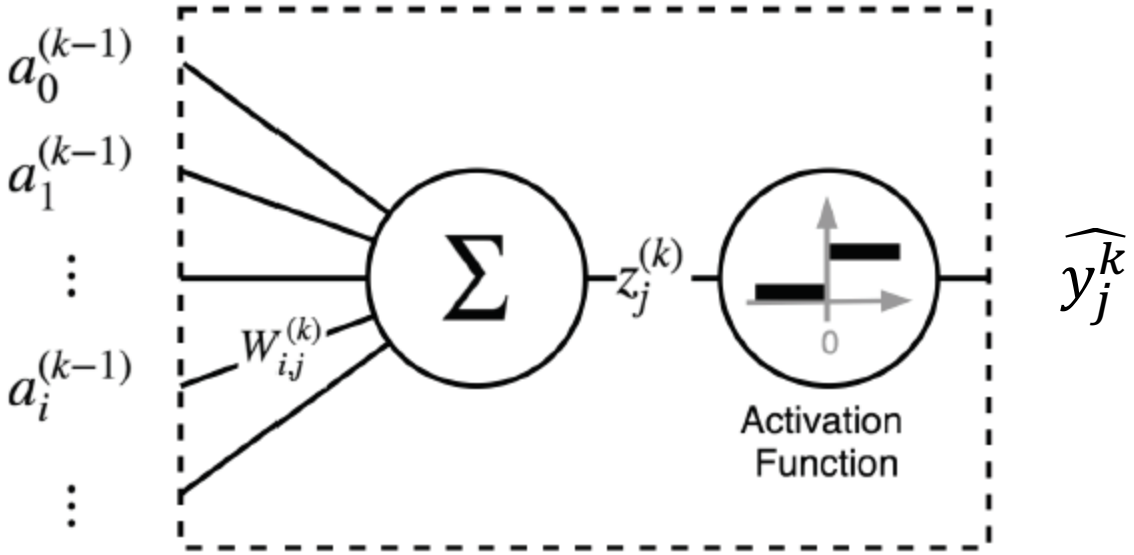


Perceptron

- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1

- Calculates the linear combination of these inputs
  - $z = \Sigma_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$

$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

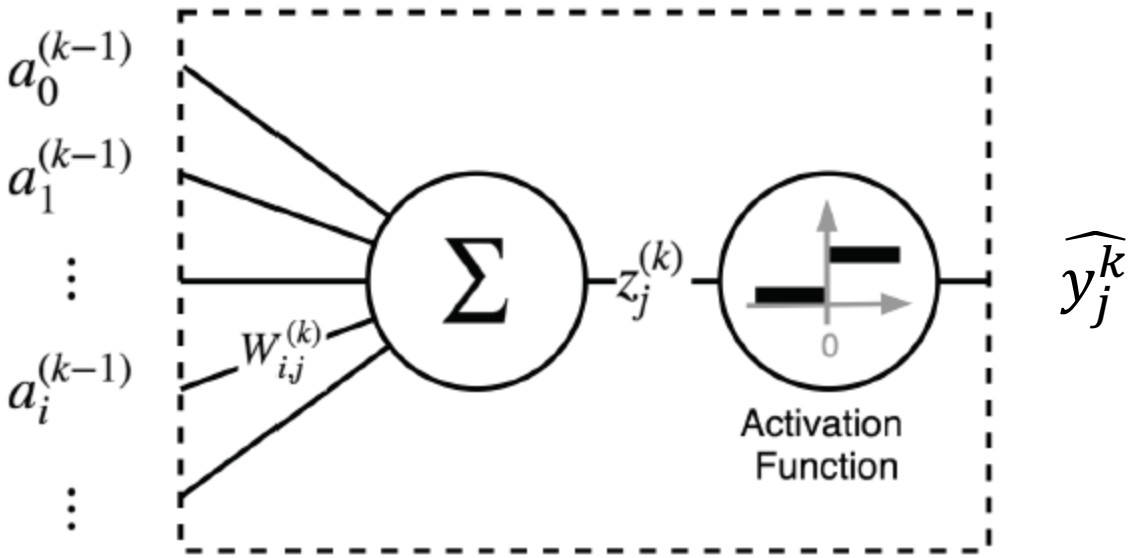


Perceptron

- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1

- Calculates the linear combination of these inputs
  - $z = \Sigma_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$
  - $\mathrm{y} = act(z)$

$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

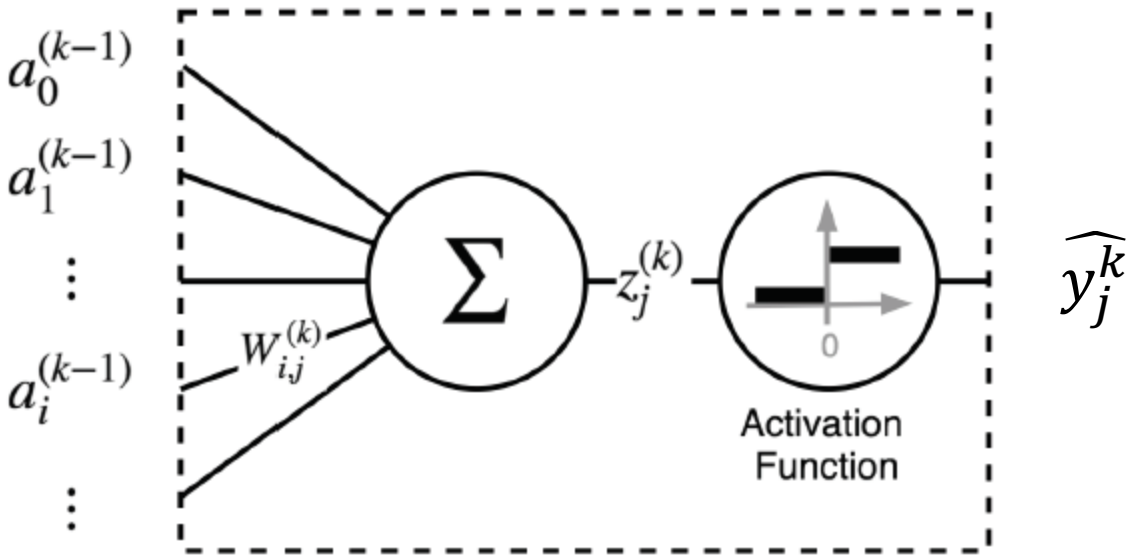


Perceptron

- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1

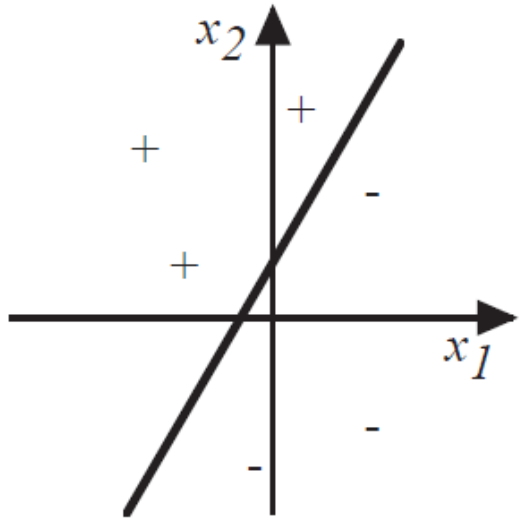
- Calculates the linear combination of these inputs
  - $z = \Sigma_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$
  - $\mathrm{y} = act(z)$
  - $act(.)$: activation function
    - E.g. sigmoid function

$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

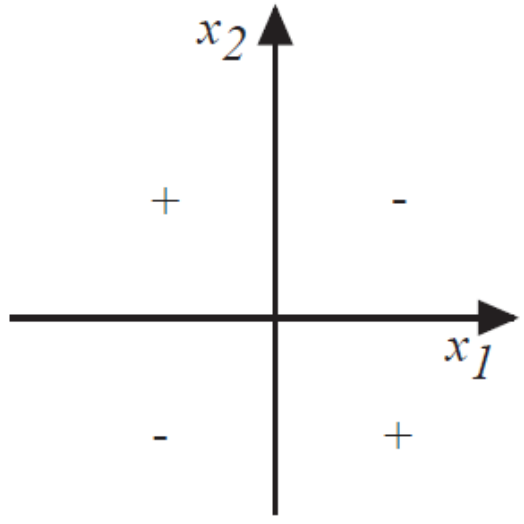


Representation Power

- A perceptron represents a **hyperplane decision surface** in the **n**-dimensional space of instances.



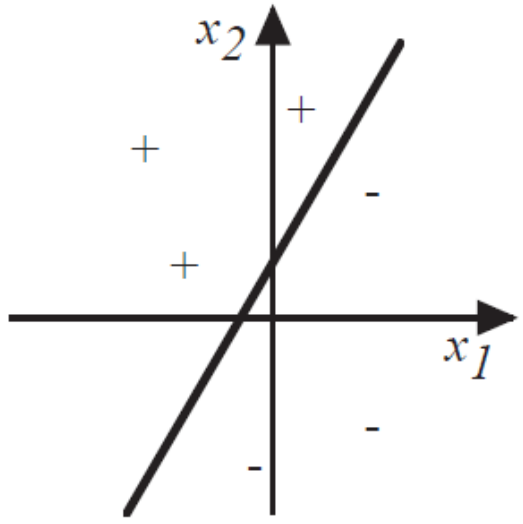
(a)



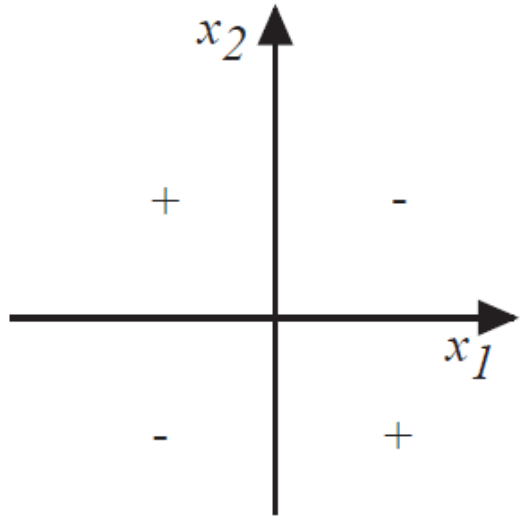
(b)

Representation Power

- A perceptron represents a **hyperplane decision surface** in the **n**-dimensional space of instances.
- Perceptron is a **linear model** which is the **basic** component of neural networks.



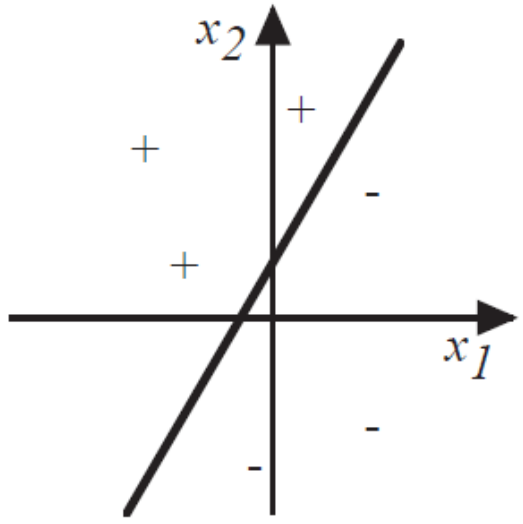
(a)



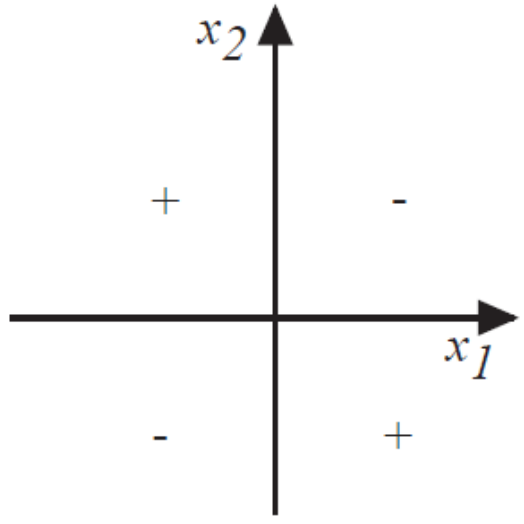
(b)

Representation Power

- A perceptron represents a **hyperplane decision surface** in the **n**-dimensional space of instances.
- Perceptron is a **linear model** which is the **basic** component of neural networks.
- Those that can be separated are called **linearly separable**.



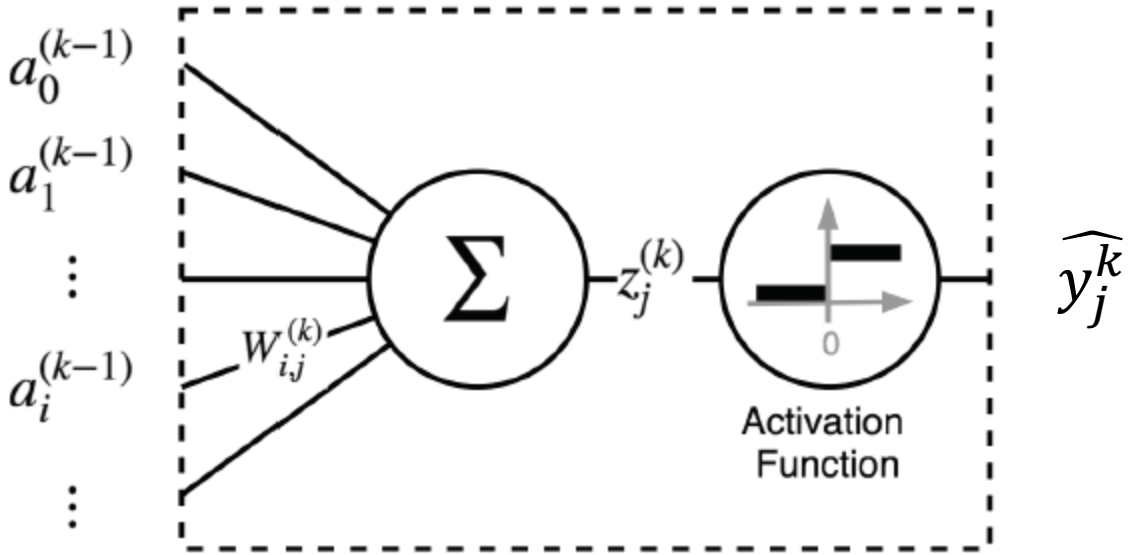
(a)



(b)

Perceptron

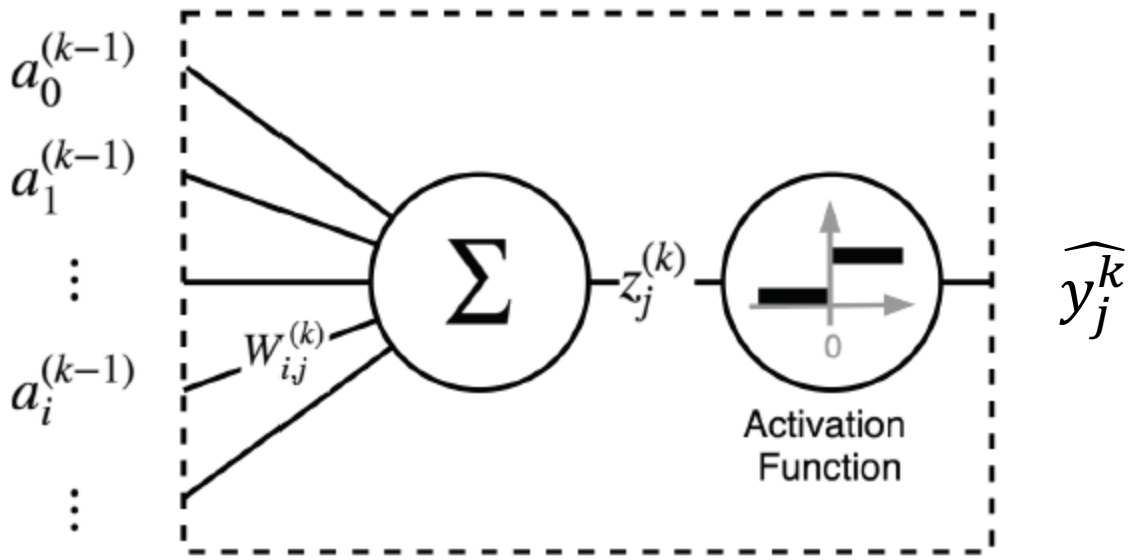
- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1



$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

Perceptron

- Input: a vector of real-valued inputs $(x_1, x_2, \ldots, x_n)$
- Output: 1 if the result is greater than 1, otherwise -1



$$\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$$

- But how do we get the weights such that the output is what we want?

Gradient Descent

- A popular optimization technique for neural networks.
- Update the weights of perceptron.

Gradient Descent

- A popular optimization technique for neural networks.
- Update the weights of perceptron.
- Measure the goodness:
  - $L(w) = loss(\hat{y}, y)$

Gradient Descent

- A popular optimization technique for neural networks.
- Update the weights of perceptron.
- Measure the goodness:
  - $L(w) = loss(\hat{y}, y)$
- Goal:
  - Find the best w such that the loss is minimum.
  - $w^* = argmin_w L(w)$

Gradient Descent

- A popular optimization technique for neural networks.
- Update the weights of perceptron.
- Measure the goodness:
  - $L(w) = loss(\hat{y}, y)$
- Goal:
  - Find the best w such that the loss is minimum.
  - $w^* = argmin_w L(w)$

Input: $\boldsymbol{w^{(0)}} \in \mathbb{R}^d$ an initial guess, a small $\eta > 0$

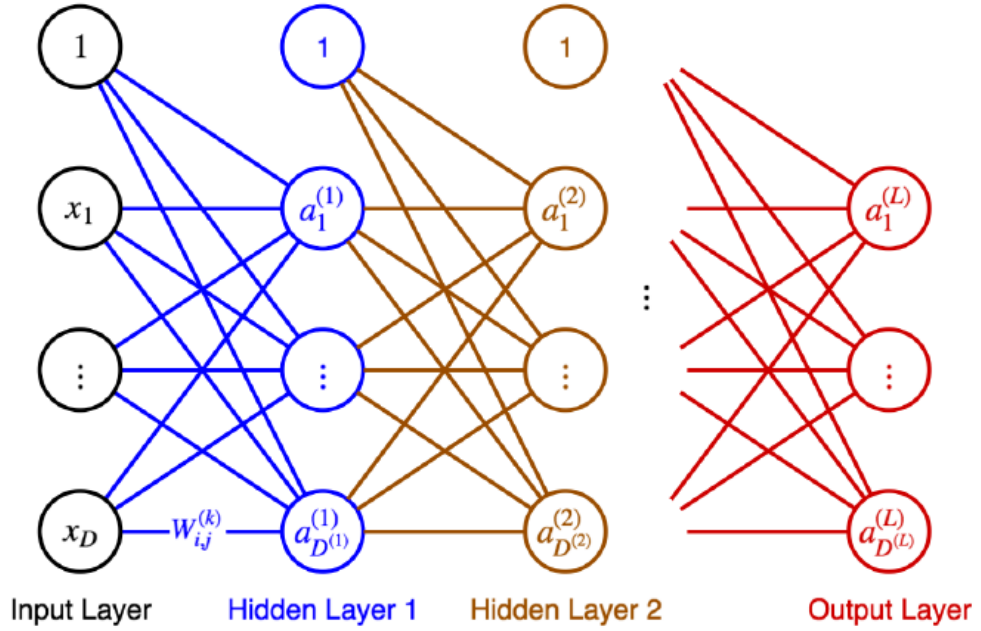
repeat

  $\boldsymbol{w^{(t+1)}} \leftarrow \boldsymbol{w^{(t)}} - \eta \nabla L(\boldsymbol{w^{(t)}})$;

until *convergence criterion is satisfied*;

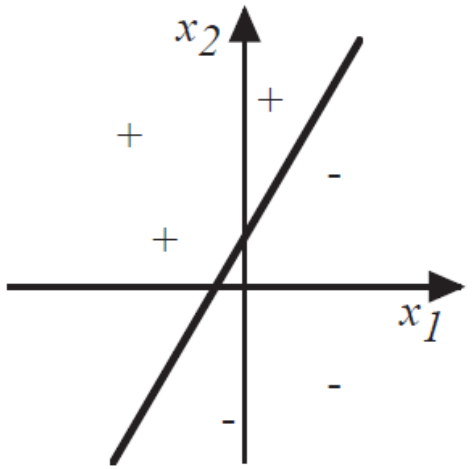
Multilayer Perceptron

- Stack of perceptron forms multi-layer perceptron.
- Capable of learning **nonlinear decision surfaces**.

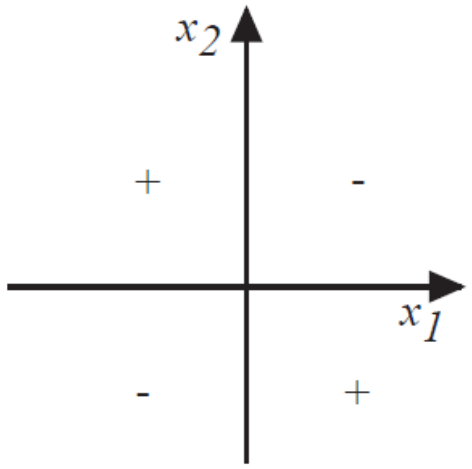


Representation Power

- Nonlinear function
- More powerful than single perceptron.
- Hidden layers can extract feature representation and increase/reduce dimension.



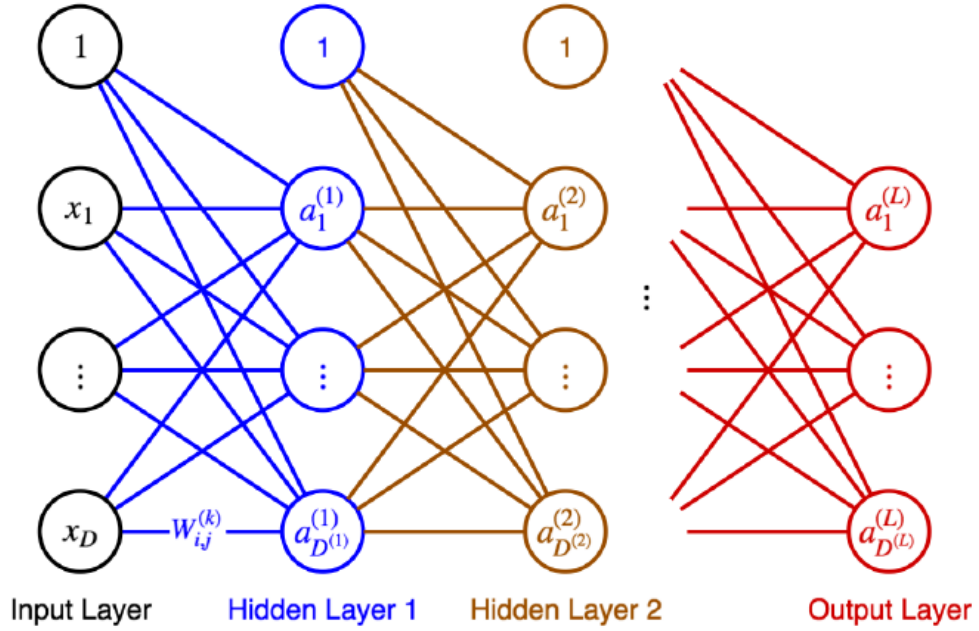
(a)



(b)

Multilayer Perceptron

- Now, we have multiple perceptron in the model.
- How do we update the weights ($W_{i,j}^{(k)}$)?

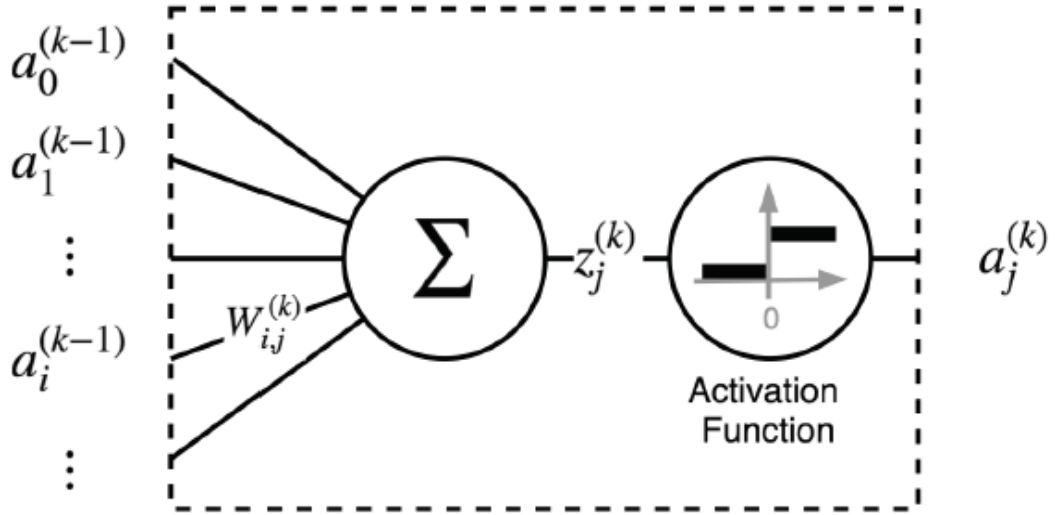


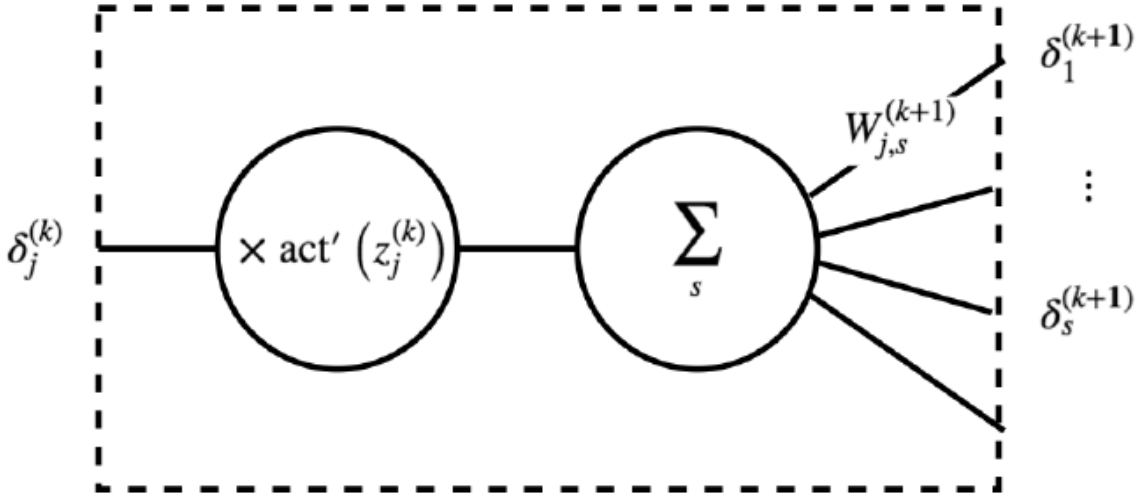
Backpropagation

- Efficient way to evaluate multiple partial derivatives at once.
- Employ gradient descent to minimize error.

Backpropagation

- Efficient way to evaluate multiple partial derivatives at once.
- Employ gradient descent to minimize error.
- Forward pass + backward pass.



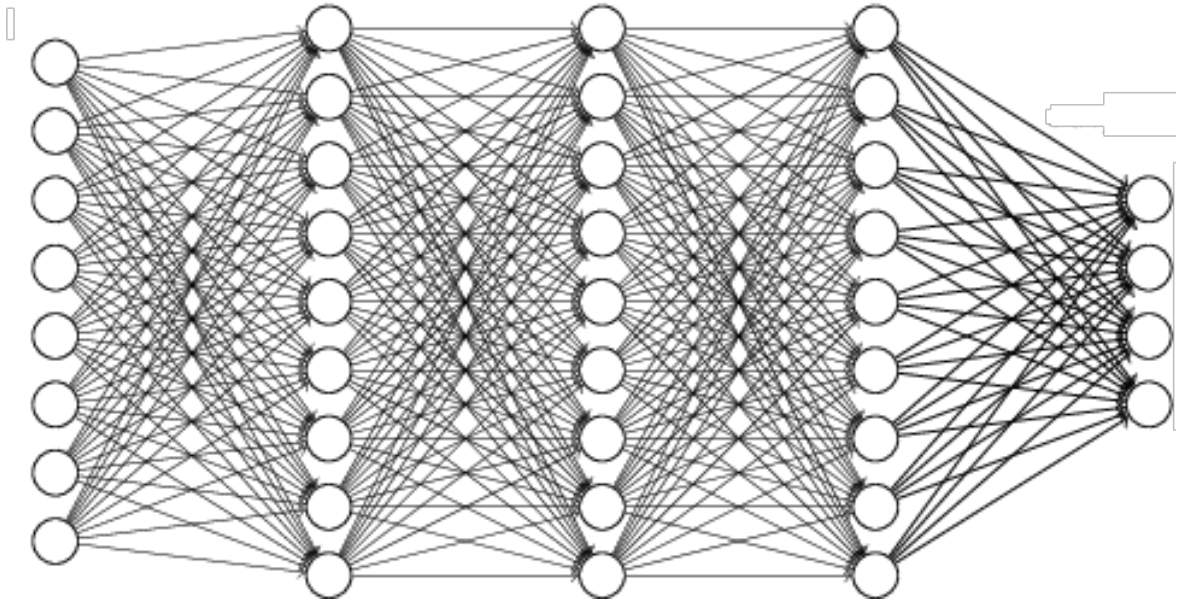


Outline

- The Rise of Deep Learning
  - Representation Learning
  - Transfer Learning
  - Curse of Dimensionality
- Basics of Neural Networks
  - Perceptron
  - Optimization - Gradient Descent
  - Multi-layer Perceptron
  - Optimization - Backpropagation
- Main Stream Models
  - Fully Connected Network (FC)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Generative Adversarial Network (GAN)

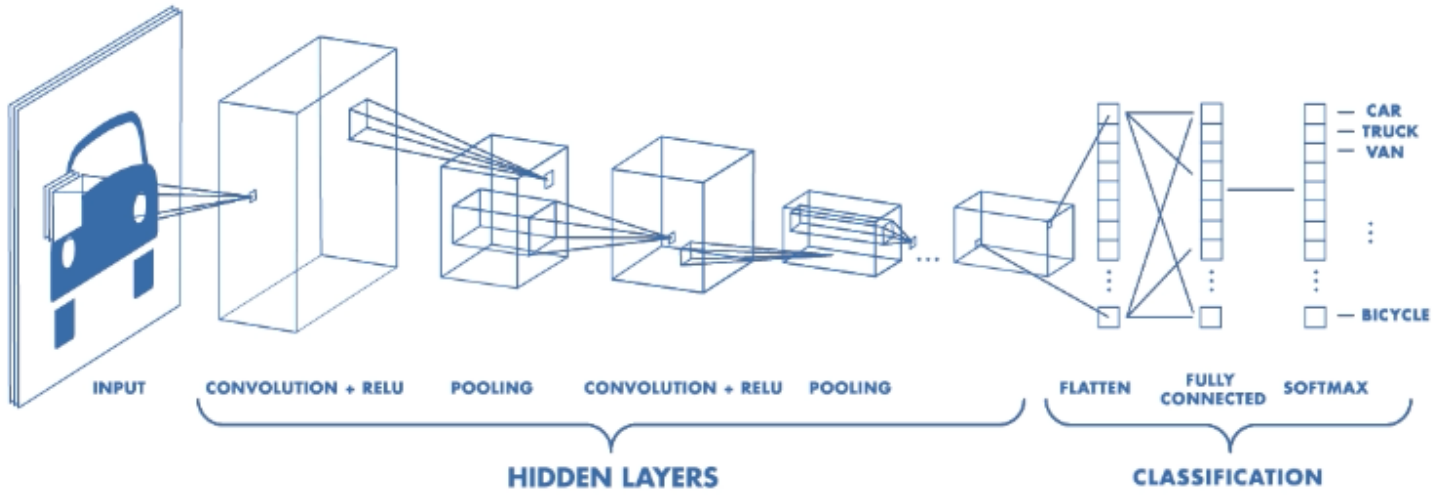
Fully Connected Network

- Define a family of functions that are parameterized by the weights of the network.
- Application:
  - Classify categories
  - Compute regression value.



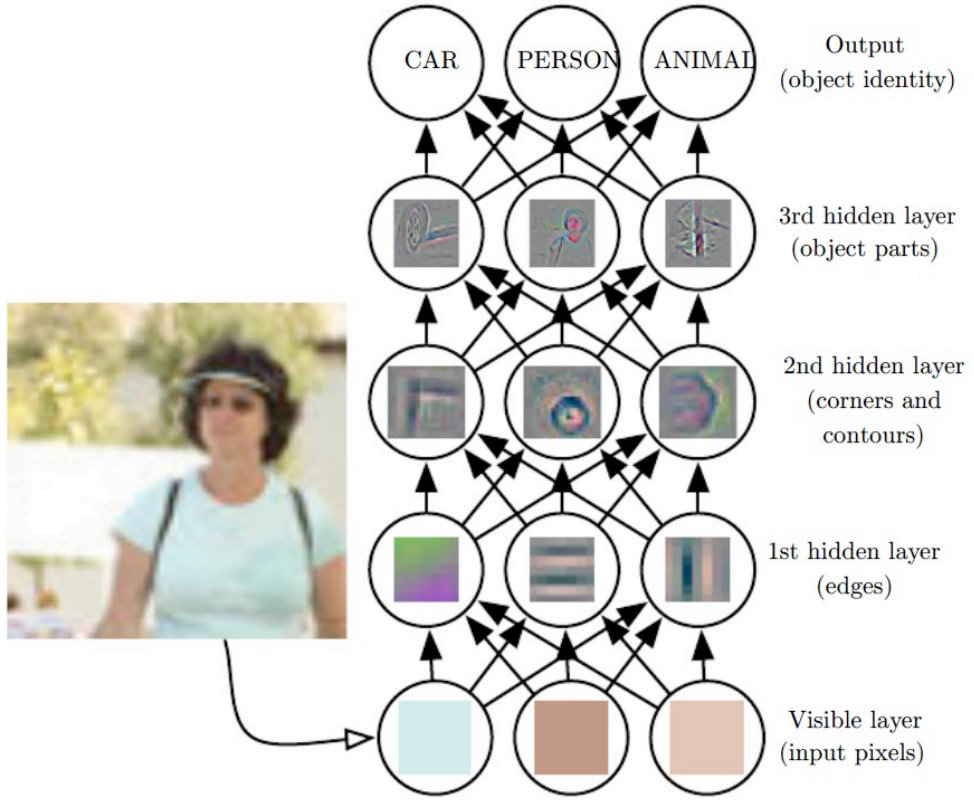
Convolutional Neural Networks (CNN)

- Use filters to extract local and spatial features.



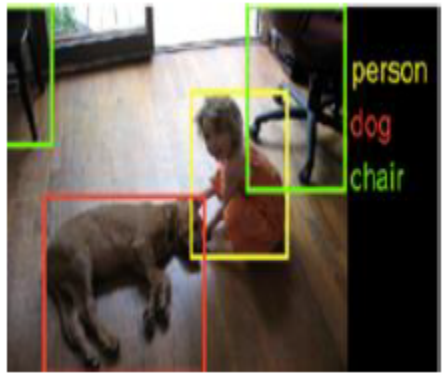
Convolutional Neural Networks (CNN)

- Use filters to extract local and spatial features.
- Abstract features in hierarchical levels.



Convolutional Neural Networks (CNN)

- Use filters to extract local and spatial features.
- Abstract features in hierarchical levels.
- CNN is popular for image recognition, image segmentation, object detection, etc.



Object detection

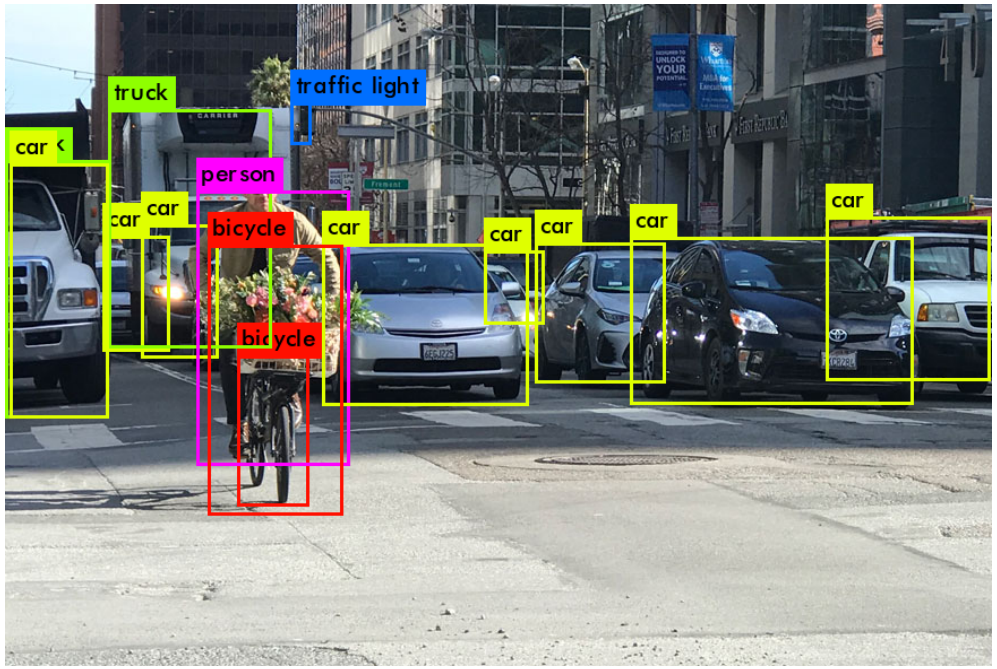

Object localization


Image or video Segmentation

YOLO

"You Only Look Once: Unified, Real-Time Object Detection", Redmon et al.

- Real time object detection



(https://pjreddie.com/darknet/yolo/)

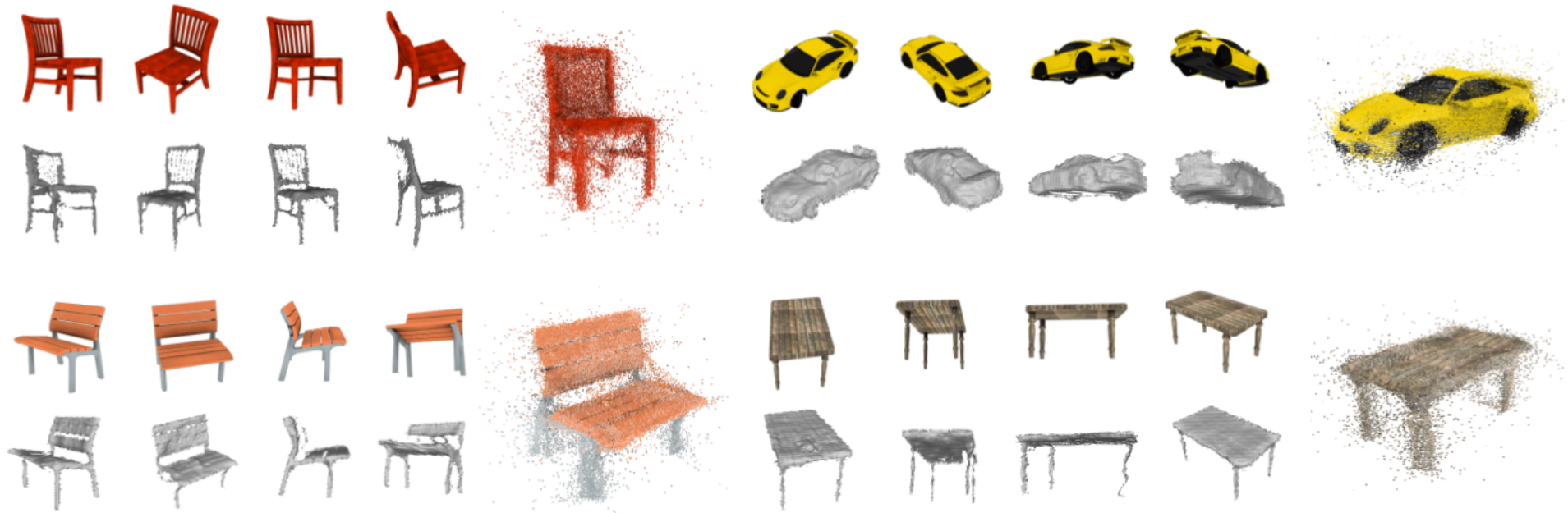
Style Transfer "A Neural Algorithm of Artistic Style", Gatys et al.

- Wonder woman in "Rain Princess" way.



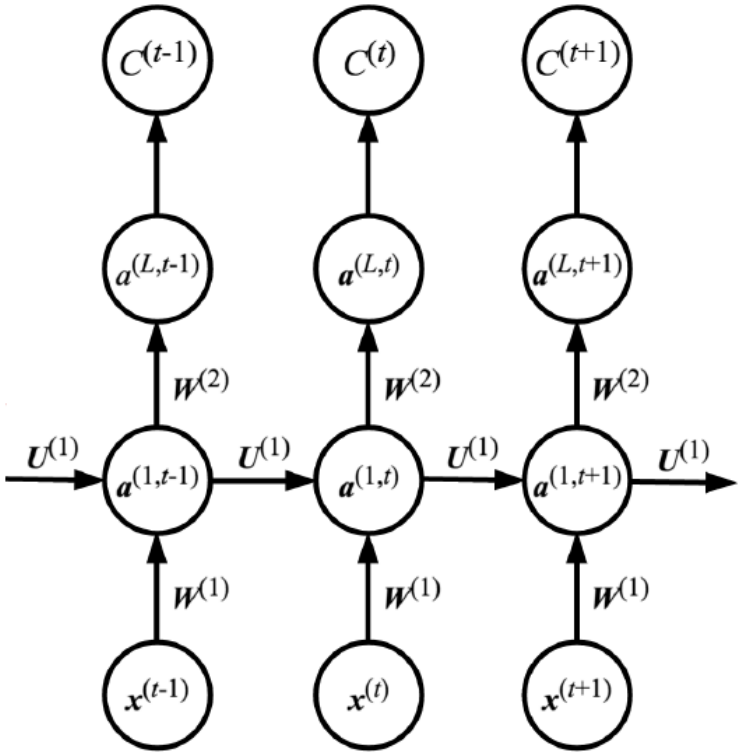
3D Reconstruction "Learning a Multi-View Stereo Machine", Kar.

- Reconstruct the 3D model



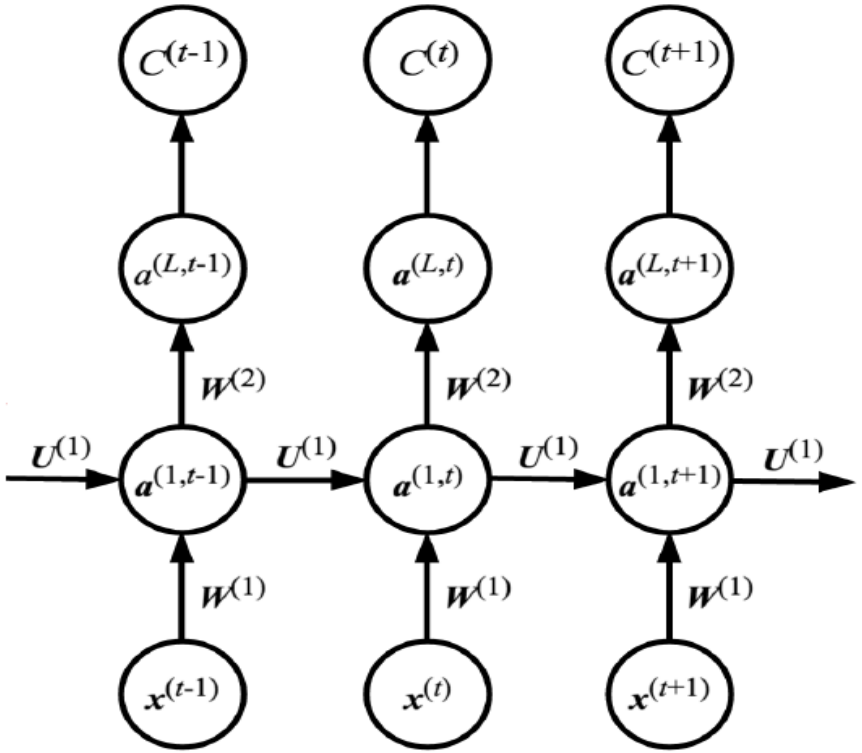
Recurrent Neural Networks (RNN)

- Sequential data: data points come in order and successive points may be dependent, e.g.,
  - Letters in a word
  - Words in a sentence/document
  - Phonemes in a spoken word utterance
  - Page clicks in a Web session
  - Frames in a video, etc.



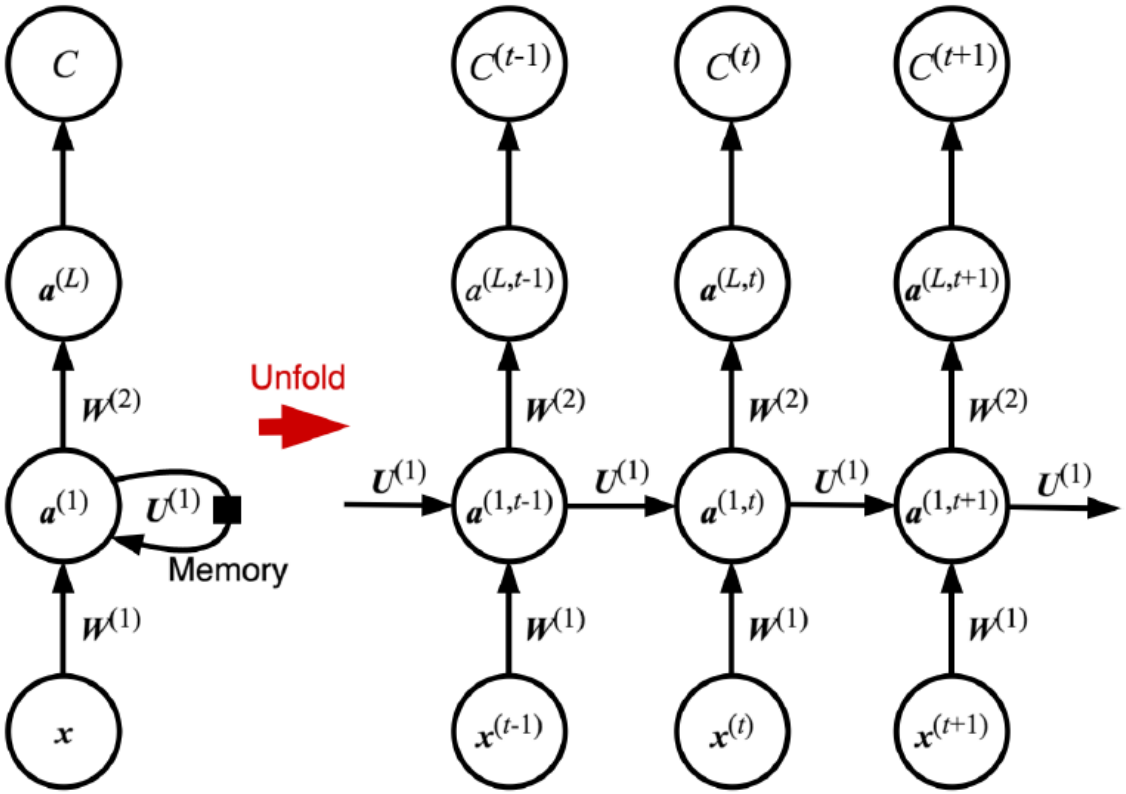
Recurrent Neural Networks (RNN)

- Weights are shared across time instances.
- Assumes that the "transition functions" are time invariant.
- Our goal is to learn $\boldsymbol{U}^k$ and $\boldsymbol{W}^k$ for k = 1, …, L



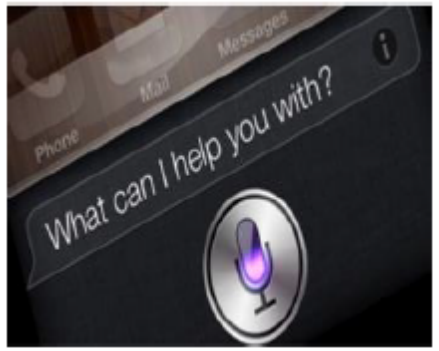
Recurrent Neural Networks (RNN)

- The computational graph of an RNN can be folded in time



Recurrent Neural Networks (RNN)

- RNN is popular for sequential data.
- Application: speech recognition, machine translation, chat bot, etc.



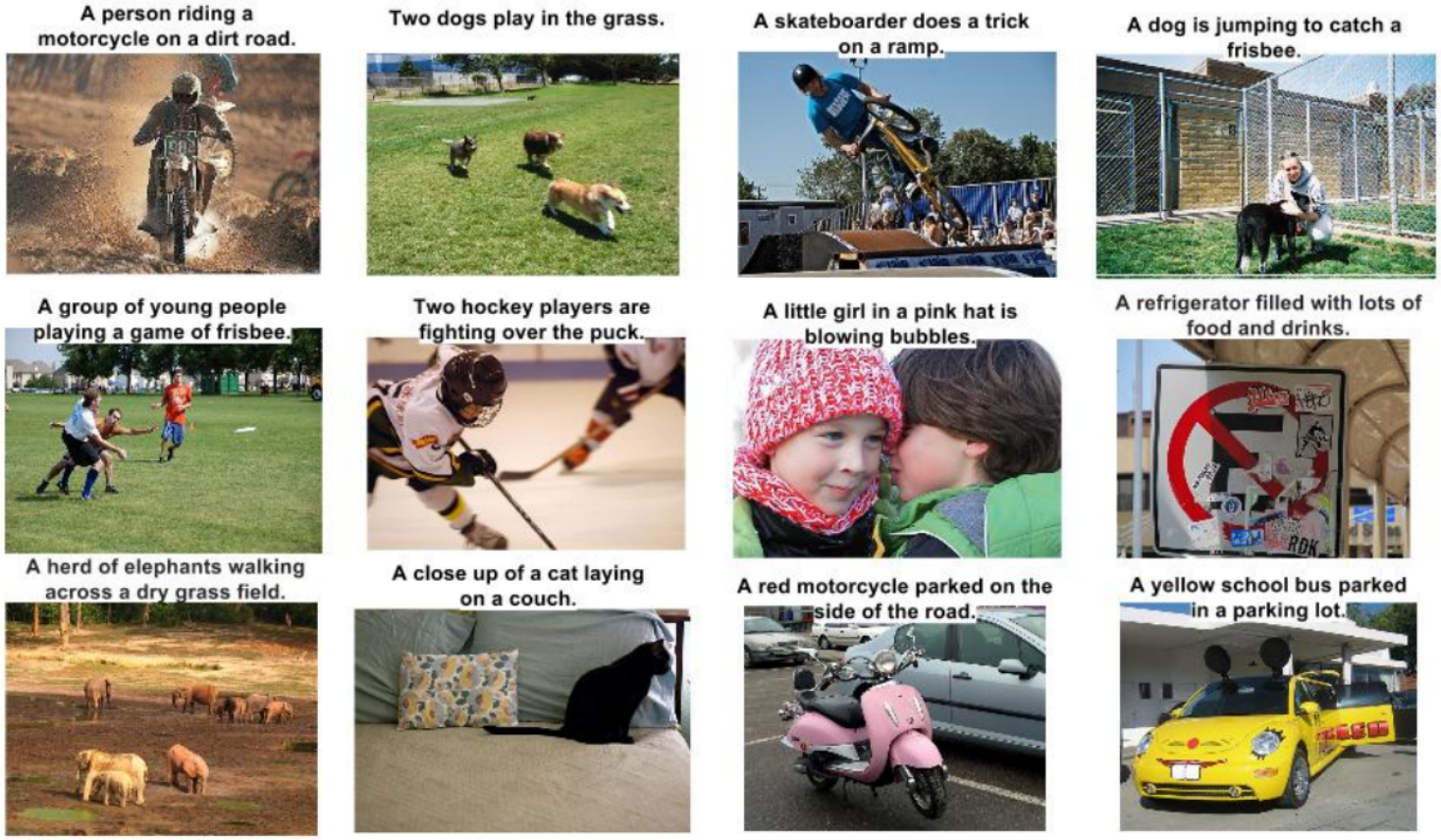
Speech recognition

Machine translation


Chat bot

Image Caption "Show and Tell: A Neural Image Caption Generator", Viny

- Given an image, Generate a sentence.



Generate Poetry

"Chinese Poetry Generation with Planning based Neural Network", Wang.

- Generate poems from titles of modern concepts

啤酒	冰心
Beer	Xin Bing
今宵啤酒两三缸，	一片冰心向月明，
I drink glasses of beer tonight,	I open up my pure heart to the moon,
杯底香醇琥珀光。	千山春水共潮生。
With the bottom of the glass full of aroma and amber light.	With the spring river flowing past mountains.
清爽金风凉透骨，	繁星闪烁天涯路，
Feeling cold as the autumn wind blows,	Although my future is illuminated by stars,
醉看明月挂西窗。	往事萦怀梦里行。
I get drunk and enjoy the moon in sight by the west window.	The past still lingers in my dream.

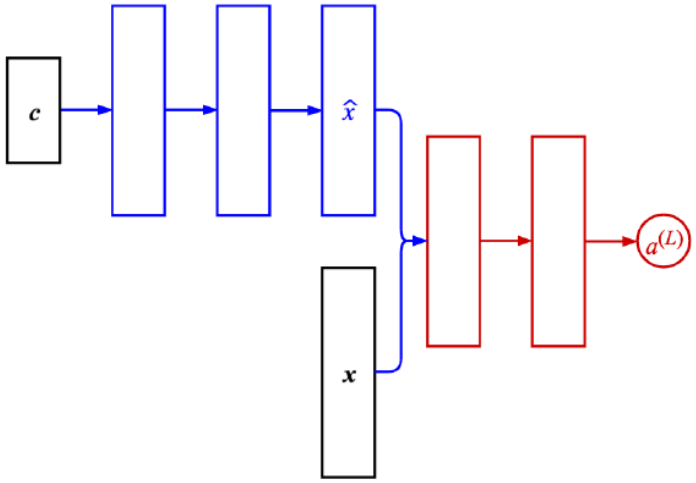
Sketch Drawings

"A Neural Representation of Sketch Drawings", Ha et al.

- Let machine draw something.

Generative Adversarial Networks (GANs)

- Generator
  - To generate data points from random codes.
- Discriminator
  - To separate generated points from real ones.



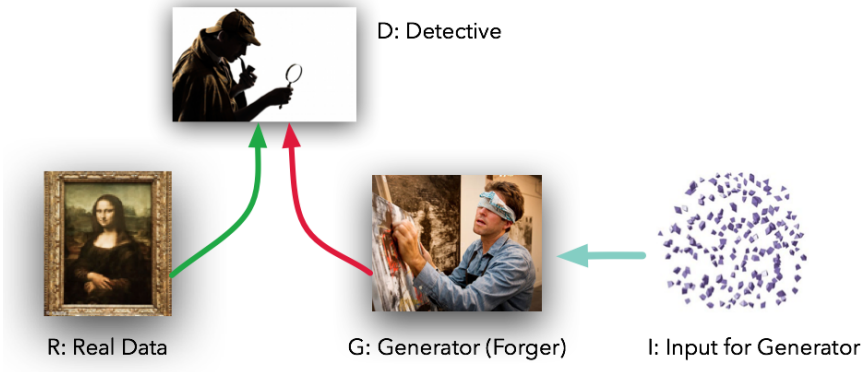
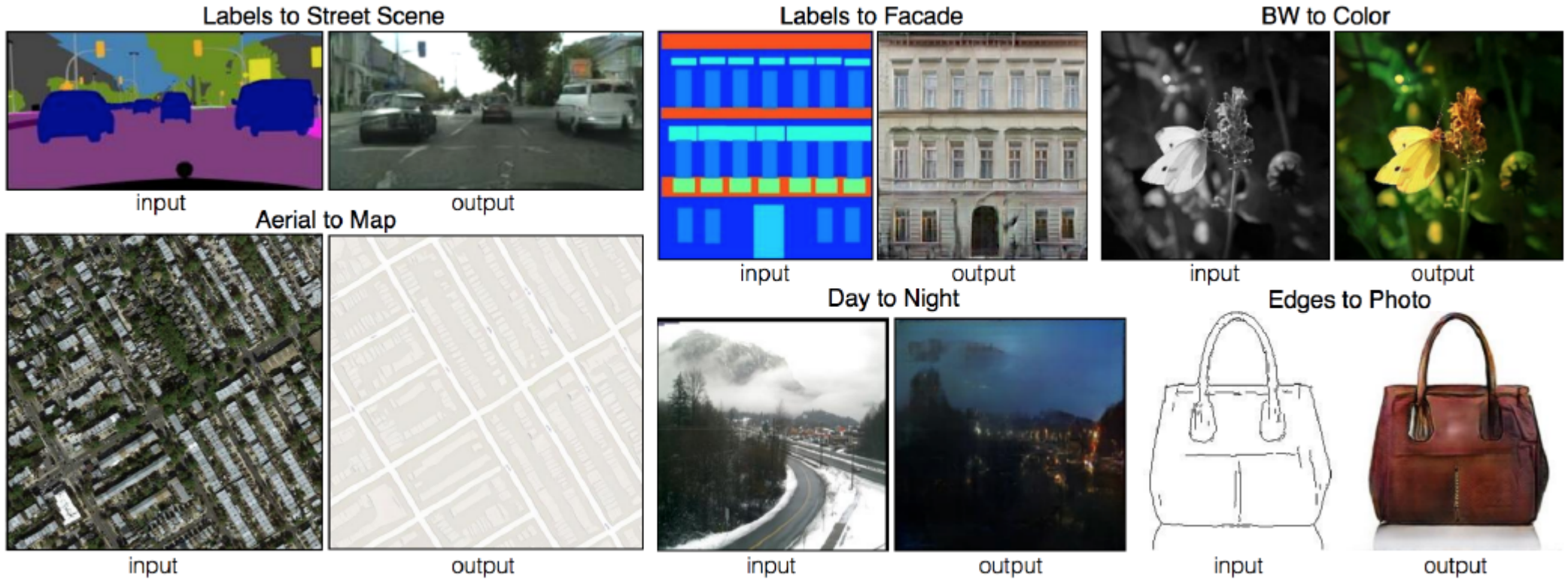


Image-to-Image Translation

"Image-to-Image Translation with Conditional Adversarial Networks", Isola et al.

- Given an image in source domain, generate images in target domain.



(https://affinelayer.com/pixsrv/)

Reverse Image Caption

“Generative Adversarial Text to Image Synthesis”, Reed e

- Given a sentence, Generate an image.

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

Generate Human Face

"Progressive Growing of GANs for Improved Quality, Stability, and Variation"
Karras et al.

- Generate very high quality human face.

(https://www.youtube.com/watch?time_continue=1&v=XOxxPcy5Gr4&t=40s)

Thanks!

- If you have any question or want to learn more please refer to Prof. Shan-Hung Wu's teaching website
  - https://nthu-datalab.github.io/ml/index.html