

Deep Learning in Finance

Reinforcement Learning in Finance

1111AIFQA09

MBA, IM, NTPU (M6132) (Fall 2022)

Tue 2, 3, 4 (9:10-12:00) (B8F40)



<https://meet.google.com/paj-zhji-mya>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week	Date	Subject/Topics
1	2022/09/13	Introduction to Artificial Intelligence in Finance and Quantitative Analysis
2	2022/09/20	AI in FinTech: Metaverse, Web3, DeFi, NFT, Financial Services Innovation and Applications
3	2022/09/27	Investing Psychology and Behavioral Finance
4	2022/10/04	Event Studies in Finance
5	2022/10/11	Case Study on AI in Finance and Quantitative Analysis I
6	2022/10/18	Finance Theory

Syllabus

Week	Date	Subject/Topics
7	2022/10/25	Data-Driven Finance
8	2022/11/01	Midterm Project Report
9	2022/11/08	Financial Econometrics and Machine Learning
10	2022/11/15	AI-First Finance
11	2022/11/22	Deep Learning in Finance; Reinforcement Learning in Finance
12	2022/11/29	Case Study on AI in Finance and Quantitative Analysis II

Syllabus

Week	Date	Subject/Topics
13	2022/12/06	Industry Practices of AI in Finance and Quantitative Analysis
14	2022/12/13	Algorithmic Trading; Risk Management; Trading Bot and Event-Based Backtesting
15	2022/12/20	Final Project Report I
16	2022/12/27	Final Project Report II
17	2023/01/03	Self-learning
18	2023/01/10	Self-learning

Deep Learning in Finance

Reinforcement Learning in Finance

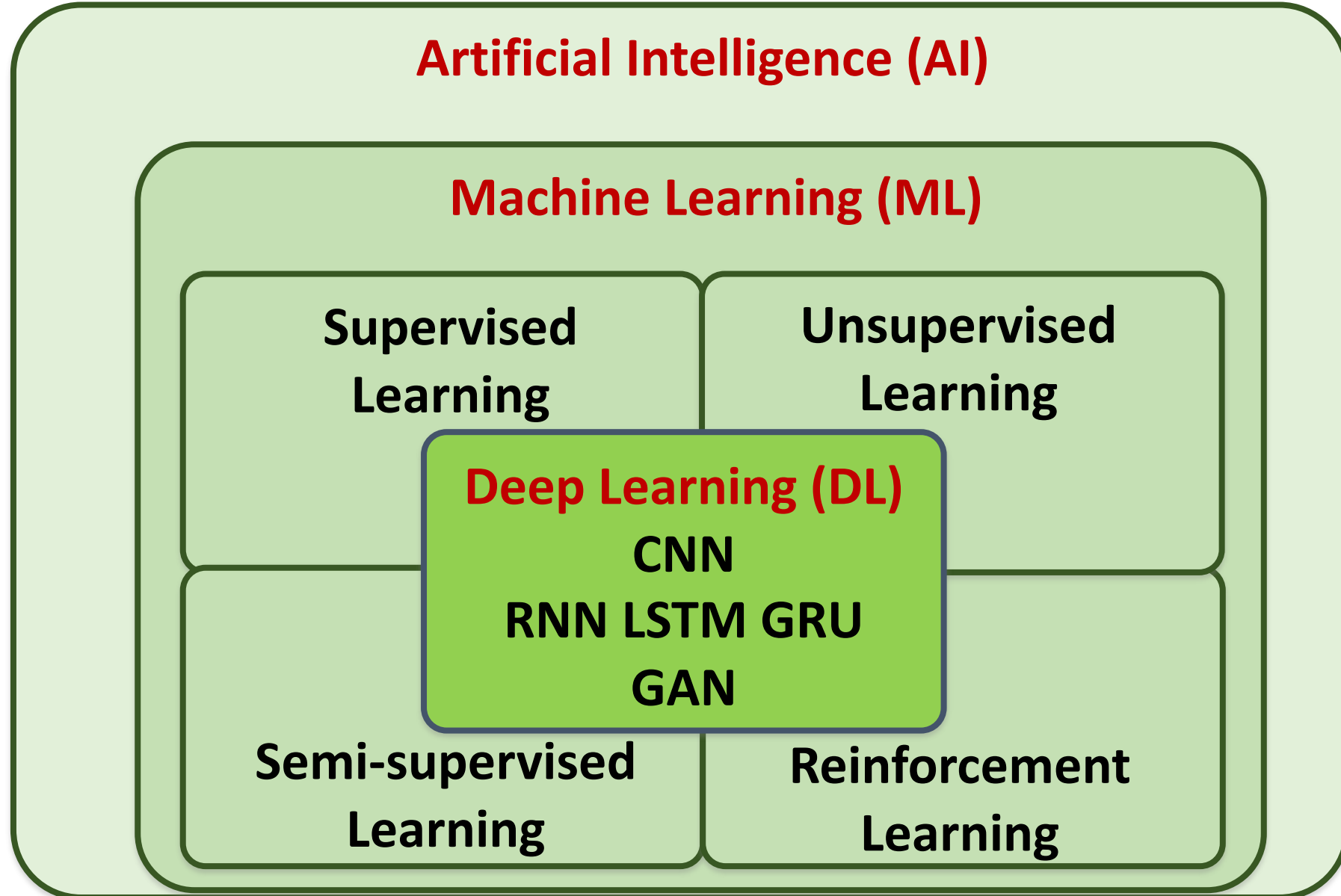
Outline

- **Deep Learning (DL) in Finance**
 - **Dense Neural Networks (DNN)**
 - **Recurrent Neural Networks (RNN)**
 - **Convolutional Neural Networks (CNN)**
- **Reinforcement Learning (RL) in Finance**
 - **Q Learning (QL)**
 - **Improved Finance Environment**
 - **Improved Financial QL Agent**

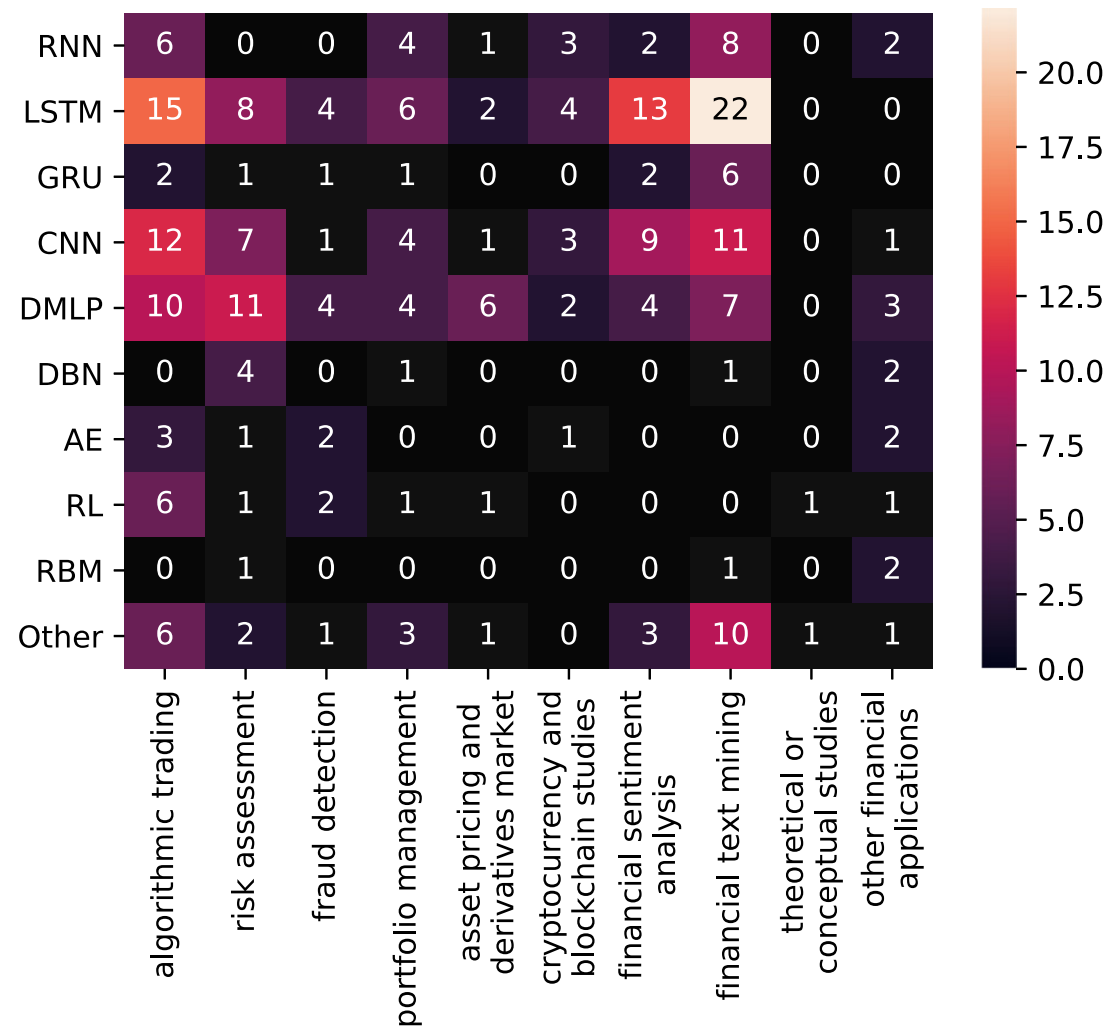
Deep Learning in Finance

- **Dense Neural Networks (DNN)**
- **Recurrent Neural Networks (RNN)**
- **Convolutional Neural Networks (CNN)**

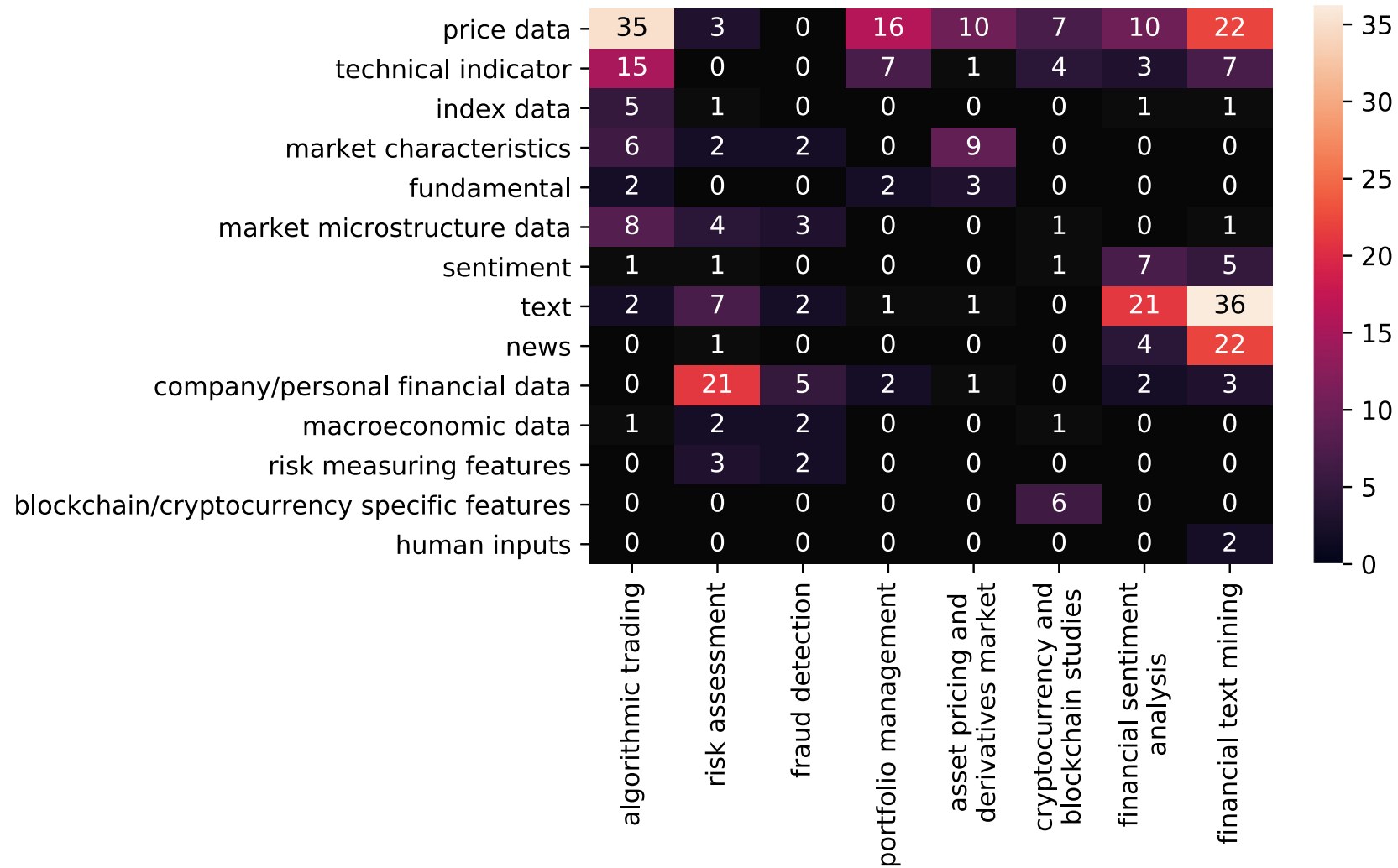
AI, ML, DL



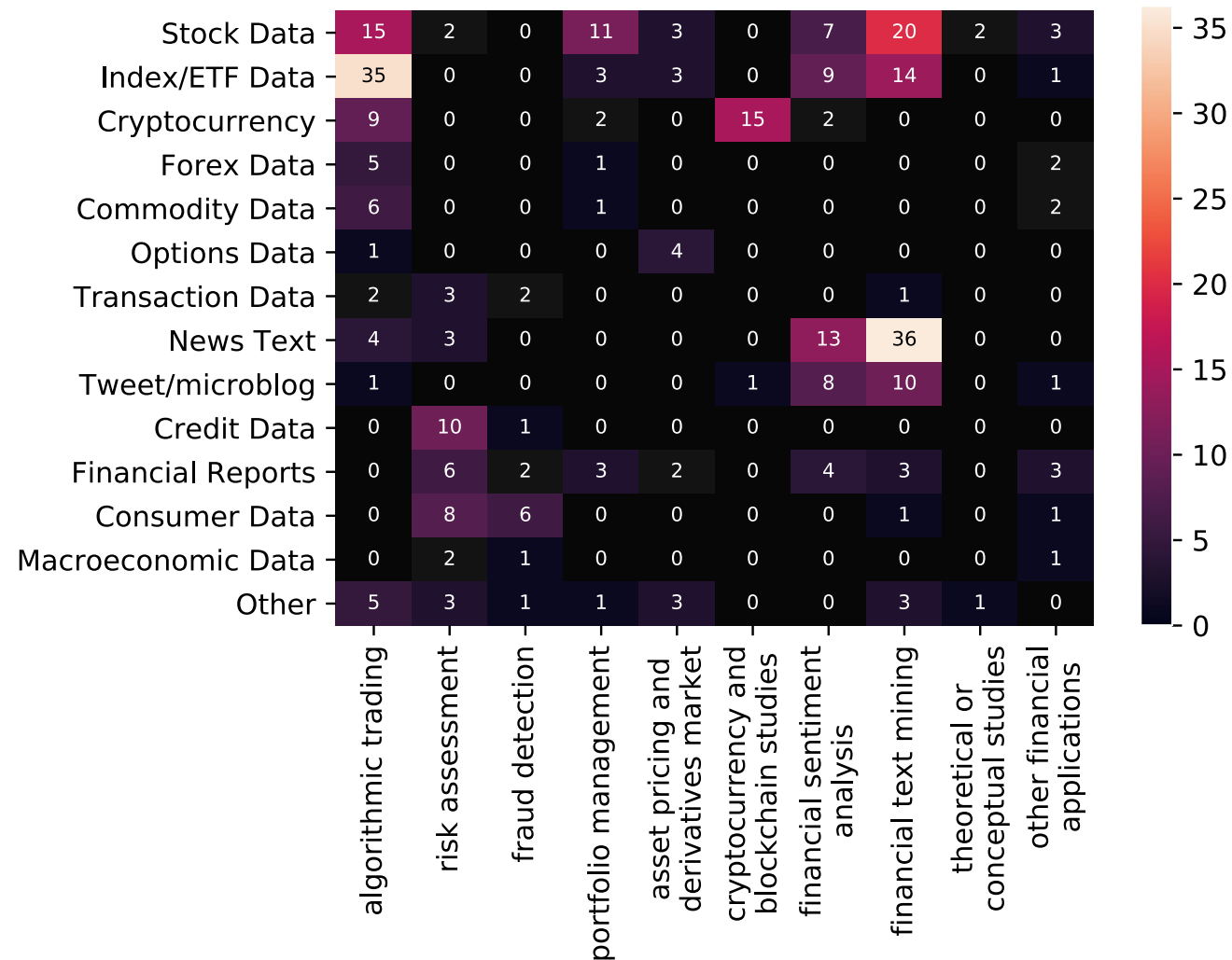
Deep learning for financial applications: Topic-Model Heatmap



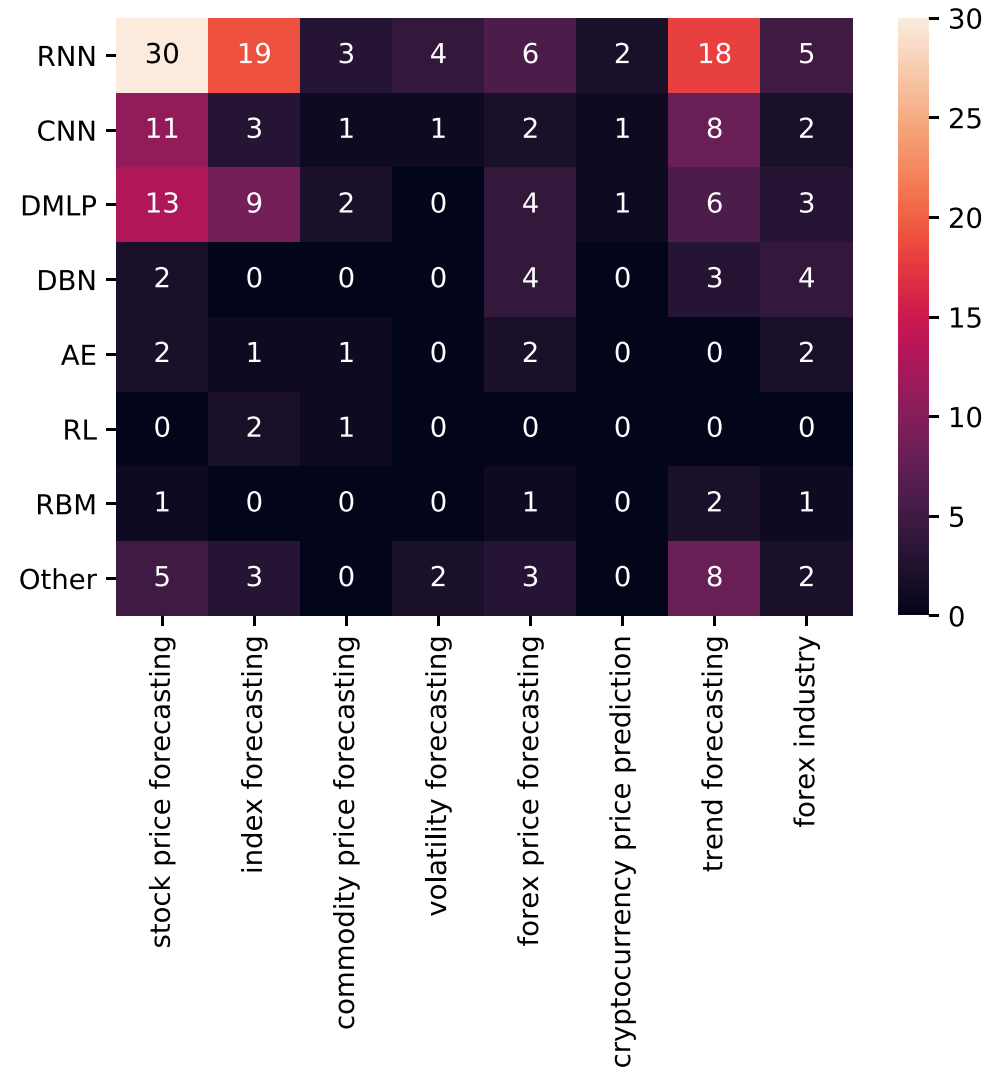
Deep learning for financial applications: Topic-Feature Heatmap



Deep learning for financial applications: Topic-Dataset Heatmap

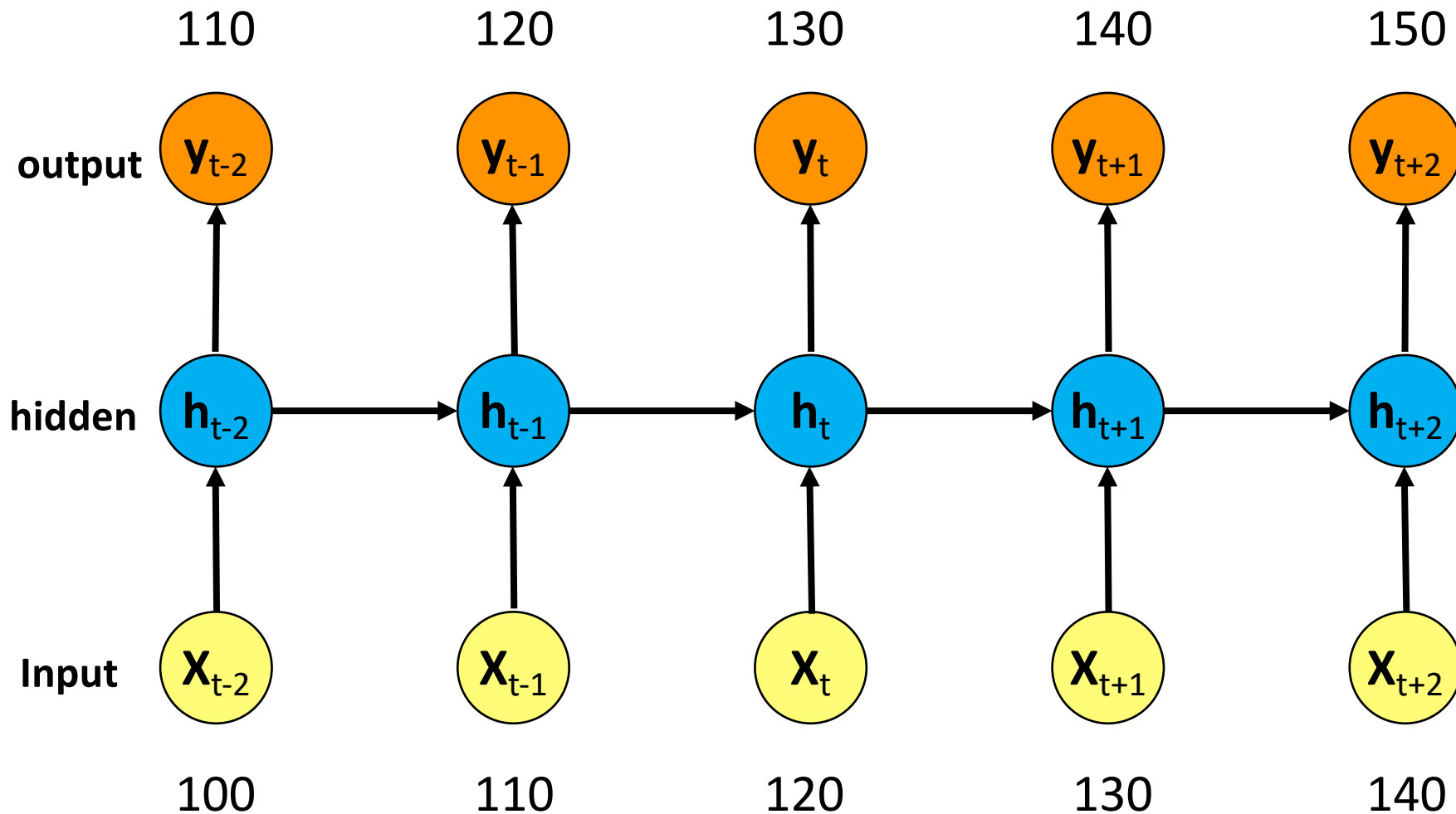


Financial time series forecasting with deep learning: Topic-model heatmap



Recurrent Neural Networks (RNN)

Time Series Forecasting



Deep Learning

Deep Learning and Neural Networks



TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Iterations
000,582

Learning rate
0.03

Activation
Tanh

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

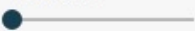
Which dataset do you want to use?



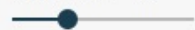
Ratio of training to test data: 50%



Noise: 0



Batch size: 10



INPUT

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

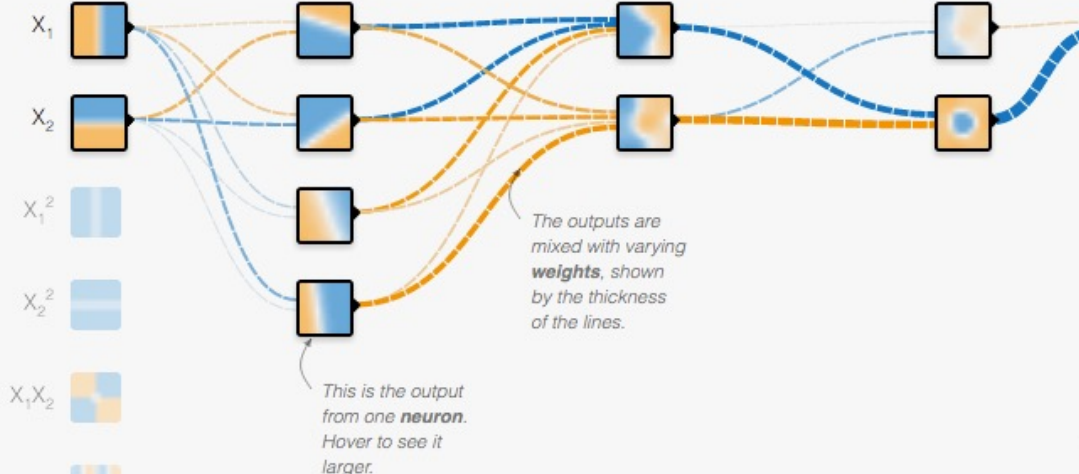
$X_1 X_2$

3 HIDDEN LAYERS

4 neurons

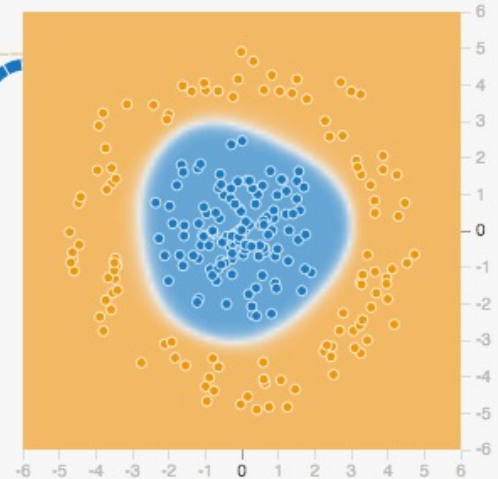
2 neurons

2 neurons



OUTPUT

Test loss 0.000
Training loss 0.000



<http://playground.tensorflow.org/>

Tensor

- 3
 - # a rank 0 tensor; this is a **scalar** with shape []
- [1., 2., 3.]
 - # a rank 1 tensor; this is a **vector** with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
 - # a rank 2 tensor; a **matrix** with shape [2, 3]
- [[[1., 2., 3.], [7., 8., 9.]]]
 - # a rank 3 **tensor** with shape [2, 1, 3]

Scalar

80

Vector

[50 60 70]

Matrix

$$\begin{bmatrix} 50 & 60 & 70 \\ 55 & 65 & 75 \end{bmatrix}$$

Tensor

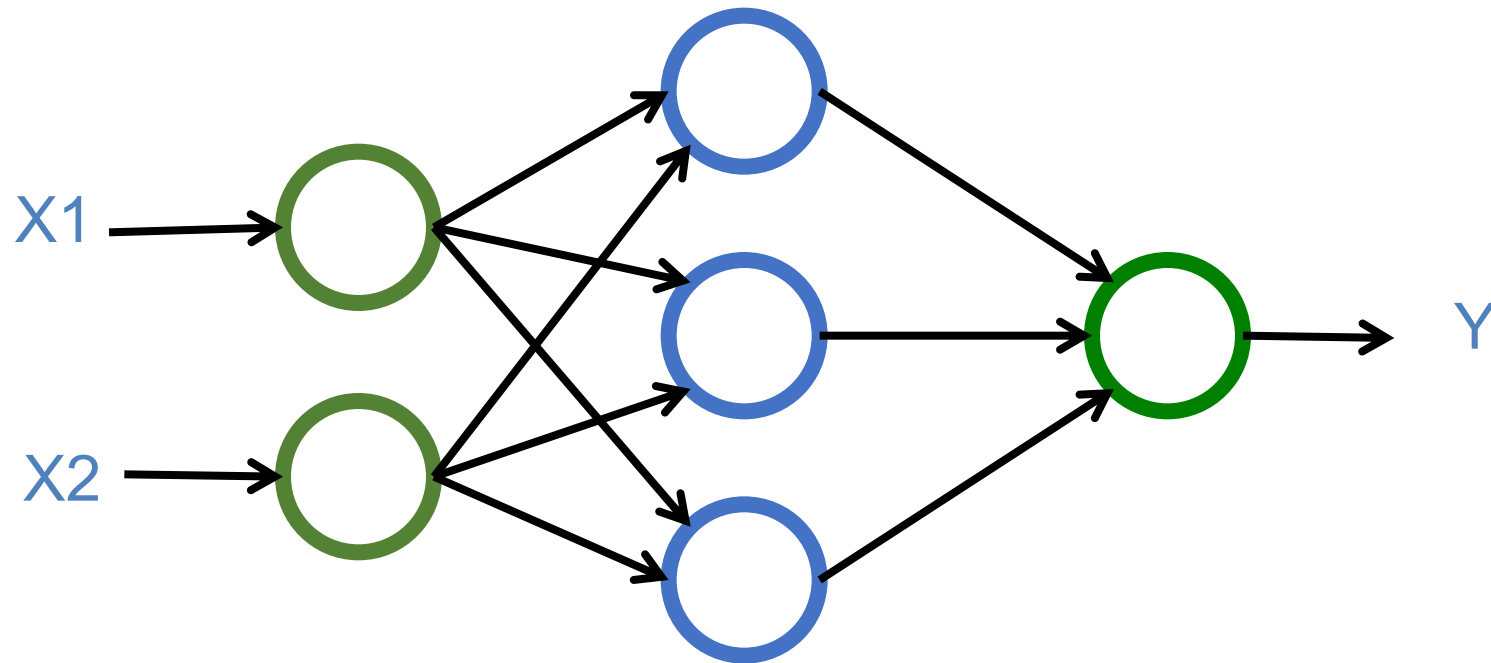
$$\begin{bmatrix} [50 & 60 & 70] & [70 & 80 & 90] \\ [55 & 65 & 75] & [75 & 85 & 95] \end{bmatrix}$$

Deep Learning and Neural Networks

Deep Learning Foundations: Neural Networks

Deep Learning and Neural Networks

Input Layer (X) Hidden Layer (H) Output Layer (Y)

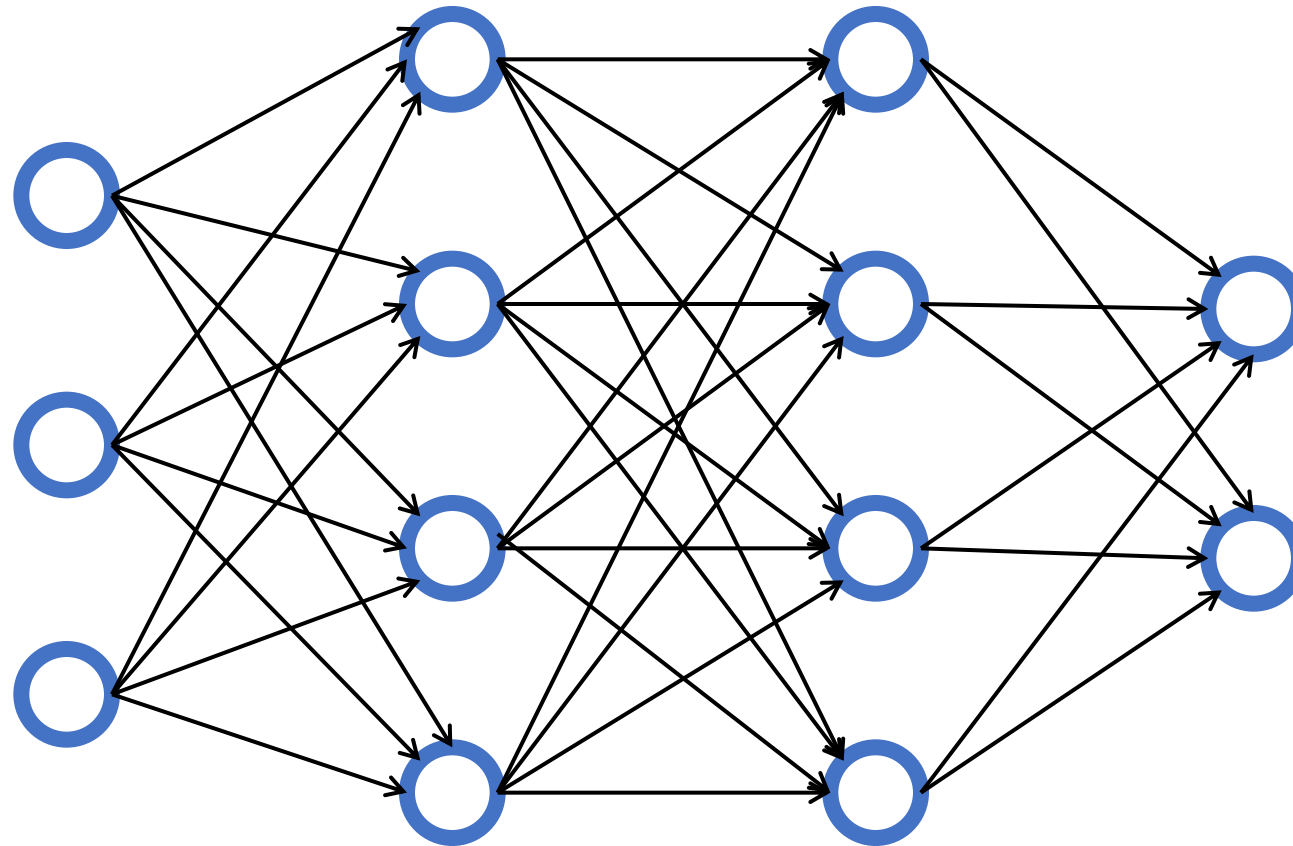


Deep Learning and Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



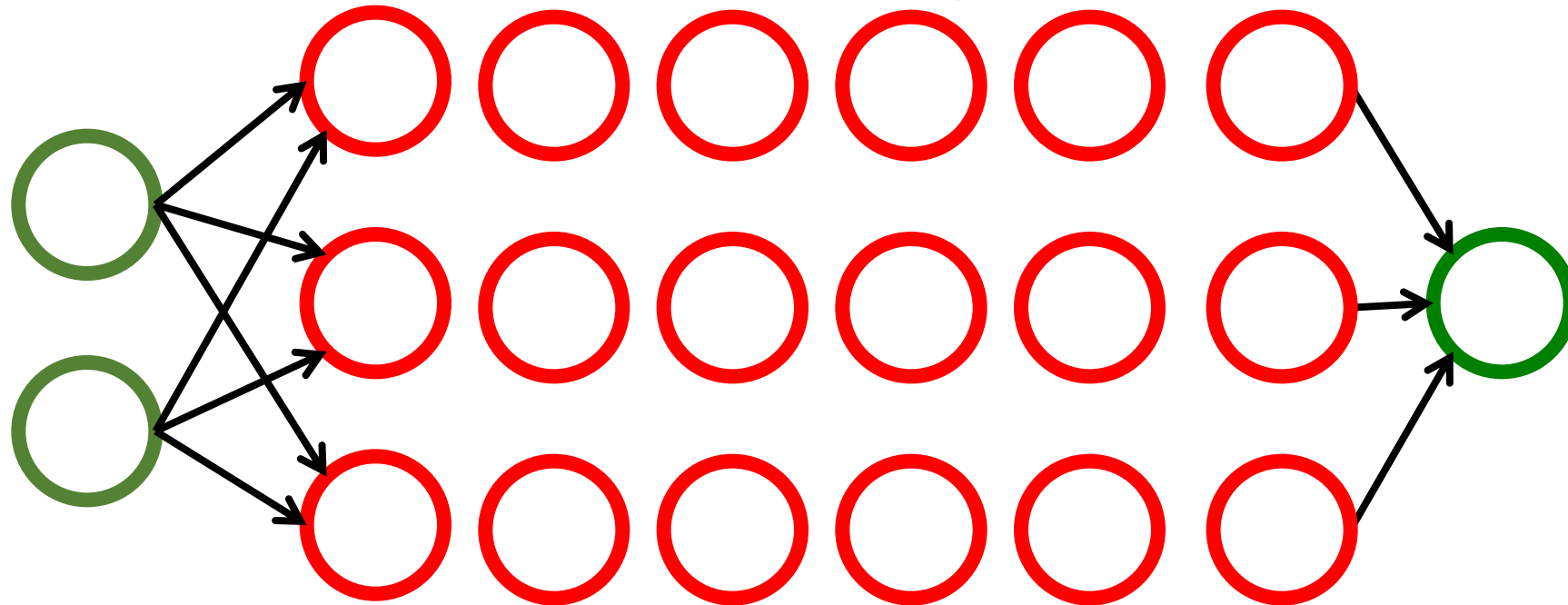
Deep Learning and Neural Networks

Input Layer
(X)

Hidden Layers
(H)

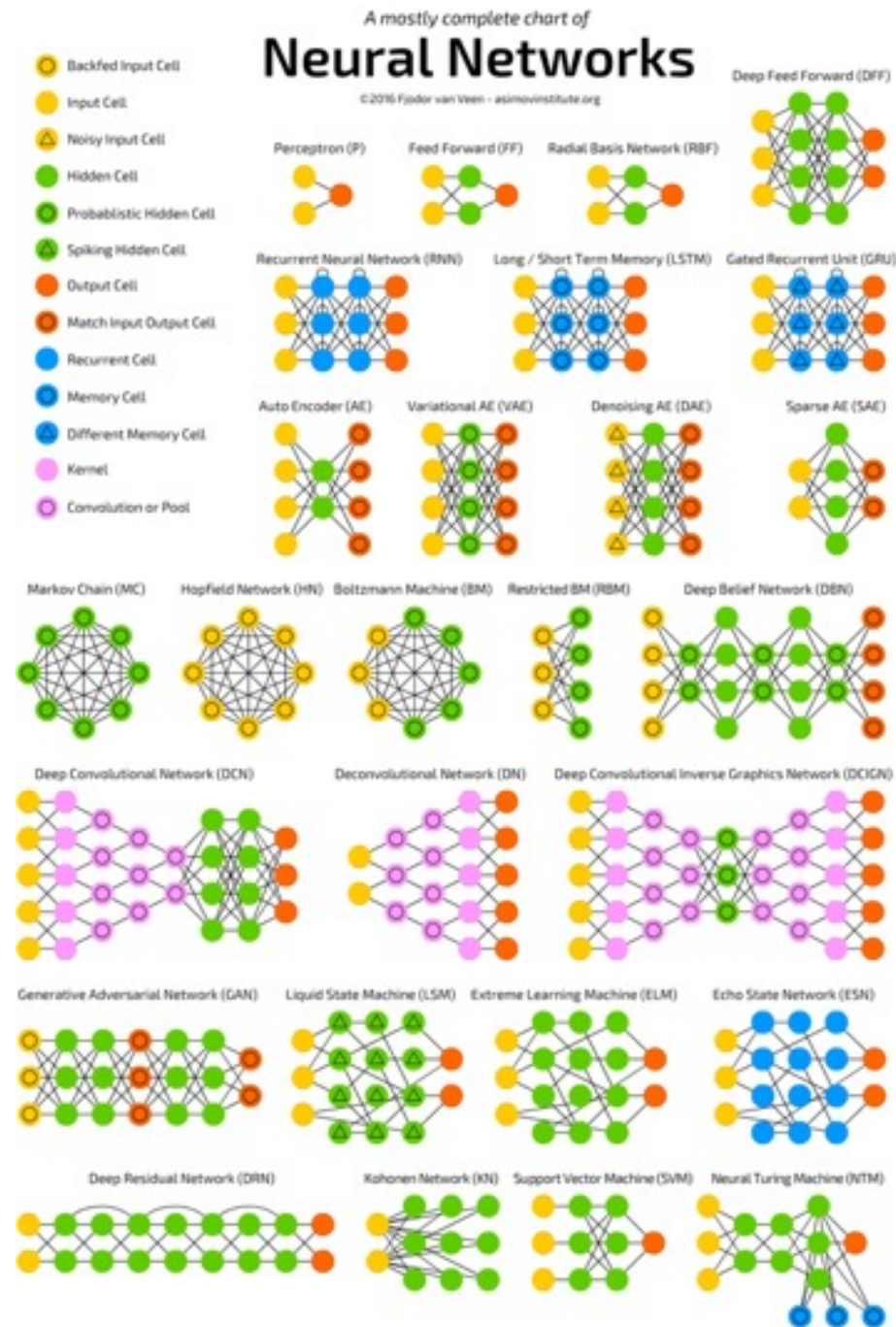
Output Layer
(Y)

Deep Neural Networks
Deep Learning



Deep Learning and Deep Neural Networks

Neural Networks (NN)



Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probablistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

Perceptron (P)



Feed Forward (FF)



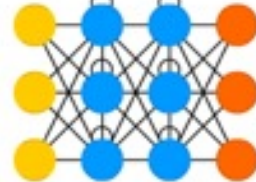
Radial Basis Network (RBF)



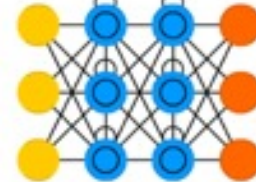
Deep Feed Forward (DFF)



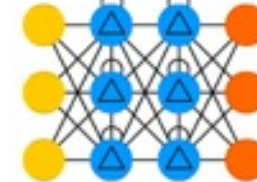
Recurrent Neural Network (RNN)



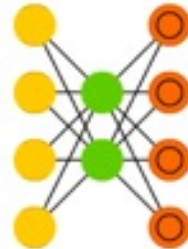
Long / Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



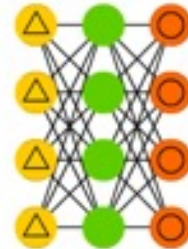
Auto Encoder (AE)



Variational AE (VAE)



Denosing AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



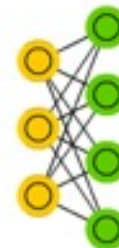
Hopfield Network (HN)



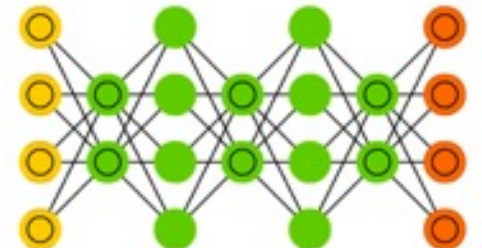
Boltzmann Machine (BM)



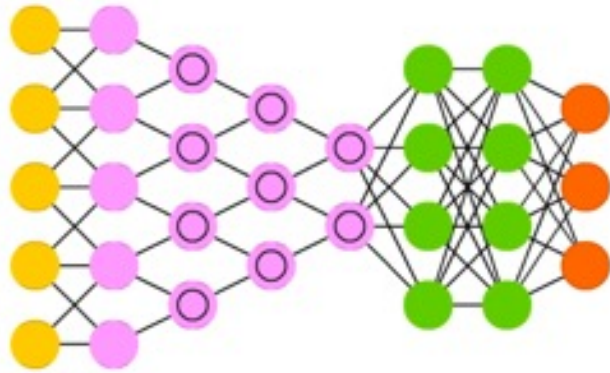
Restricted BM (RBM)



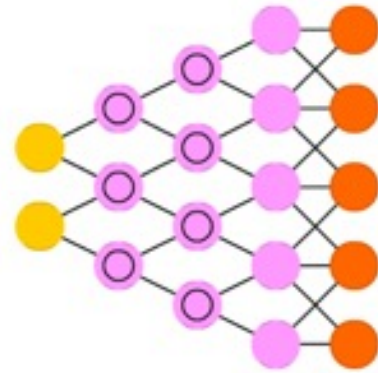
Deep Belief Network (DBN)



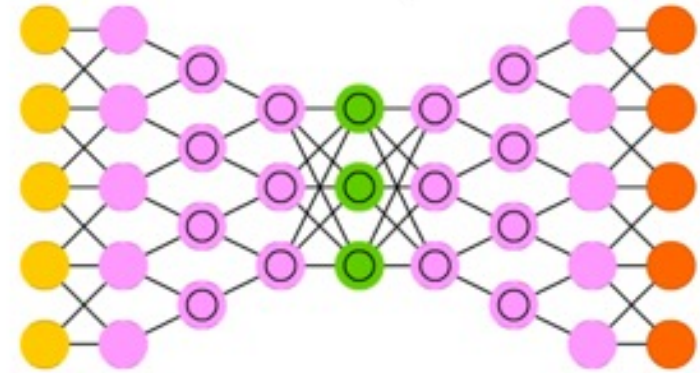
Deep Convolutional Network (DCN)



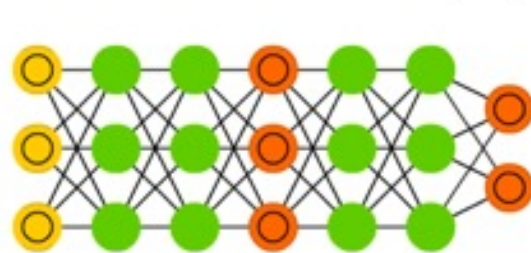
Deconvolutional Network (DN)



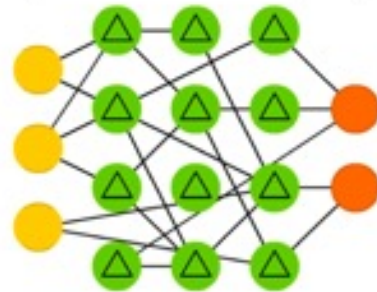
Deep Convolutional Inverse Graphics Network (DCIGN)



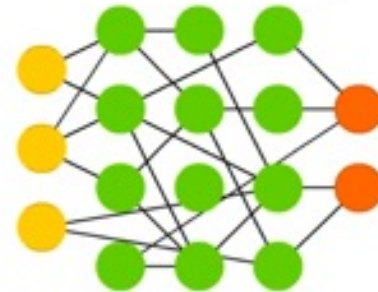
Generative Adversarial Network (GAN)



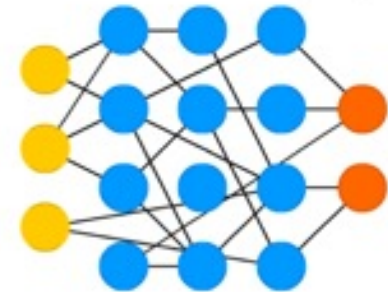
Liquid State Machine (LSM)



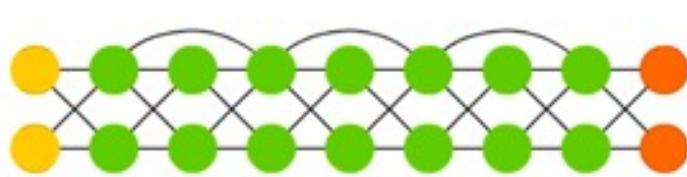
Extreme Learning Machine (ELM)



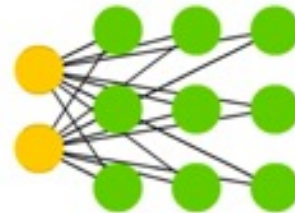
Echo State Network (ESN)



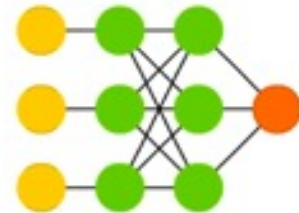
Deep Residual Network (DRN)



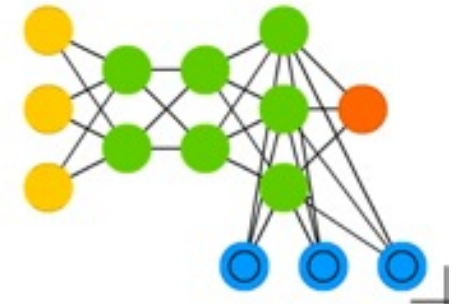
Kohonen Network (KN)



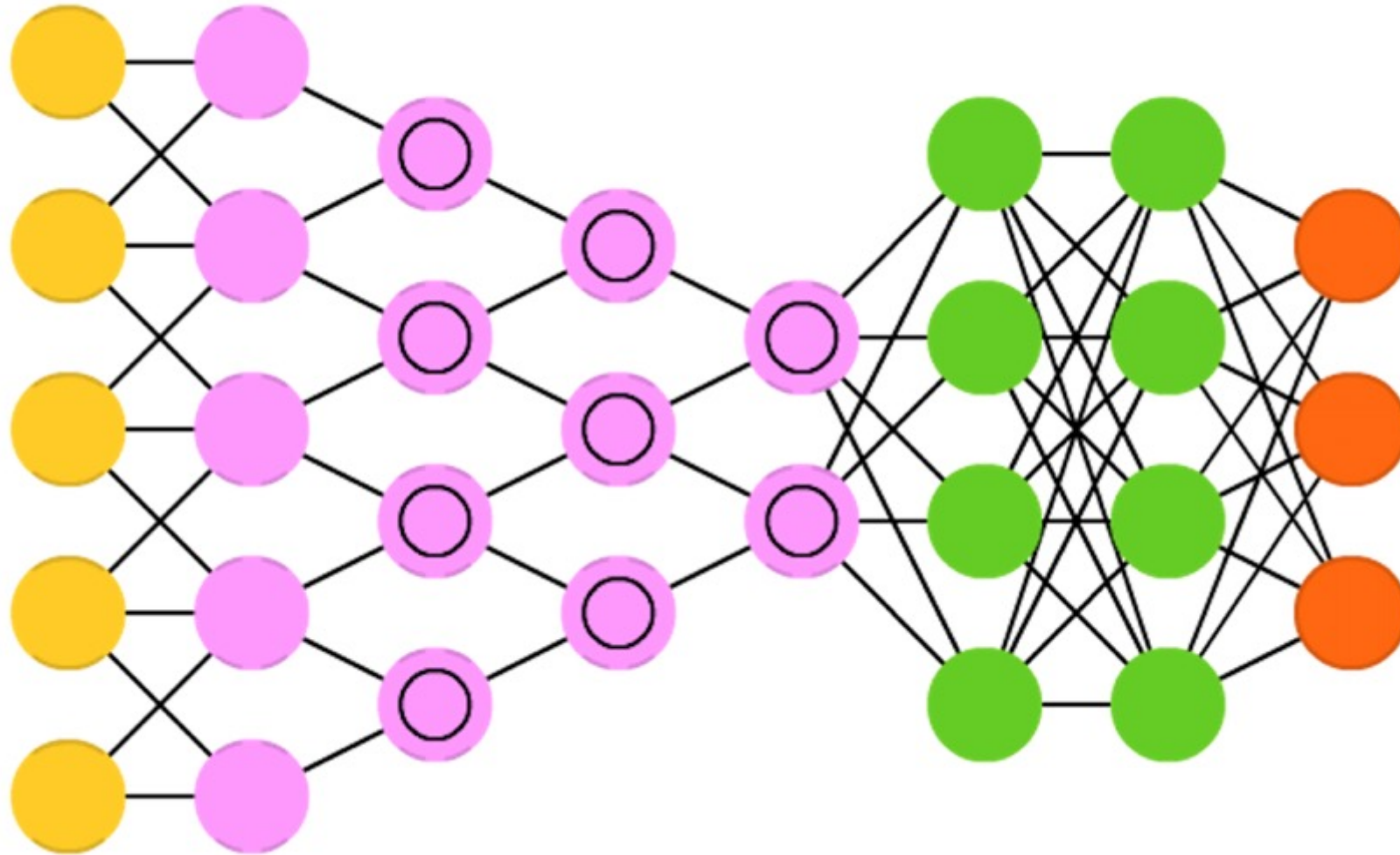
Support Vector Machine (SVM)



Neural Turing Machine (NTM)



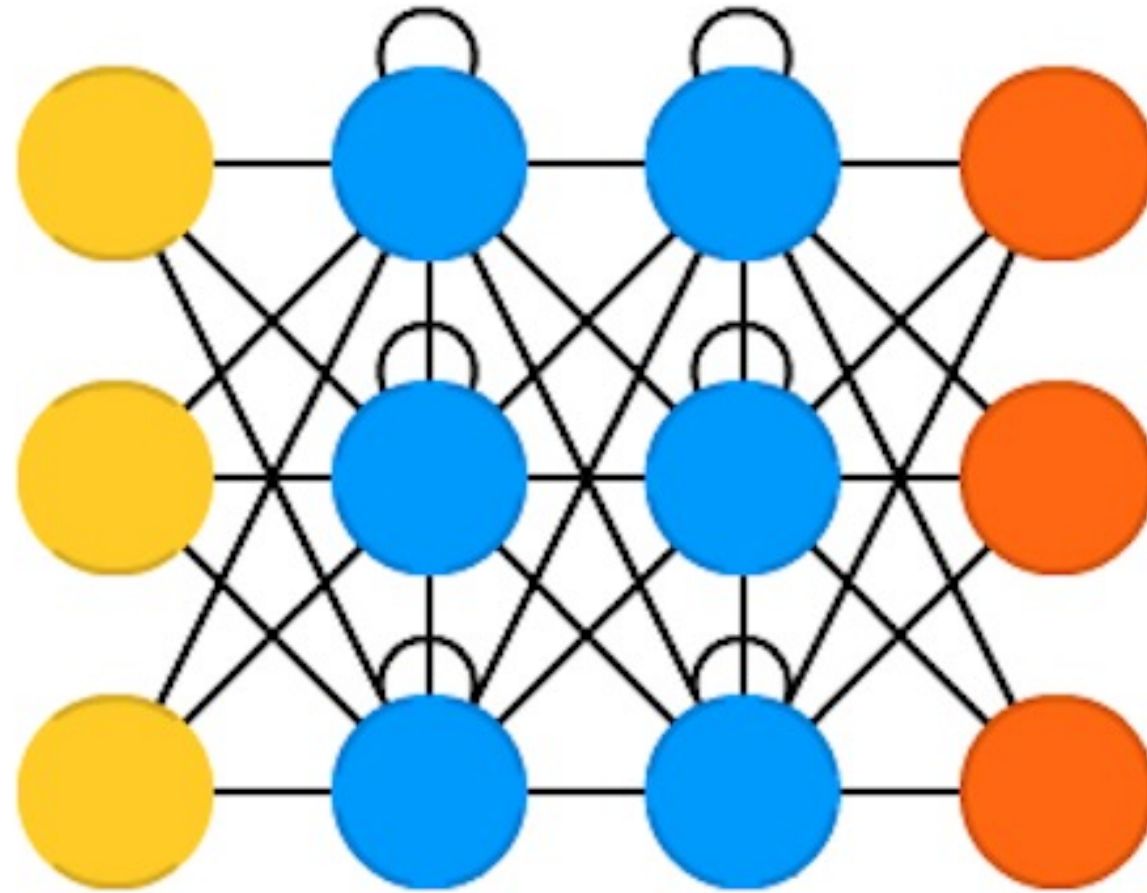
Convolutional Neural Networks (CNN or Deep Convolutional Neural Networks, DCNN)



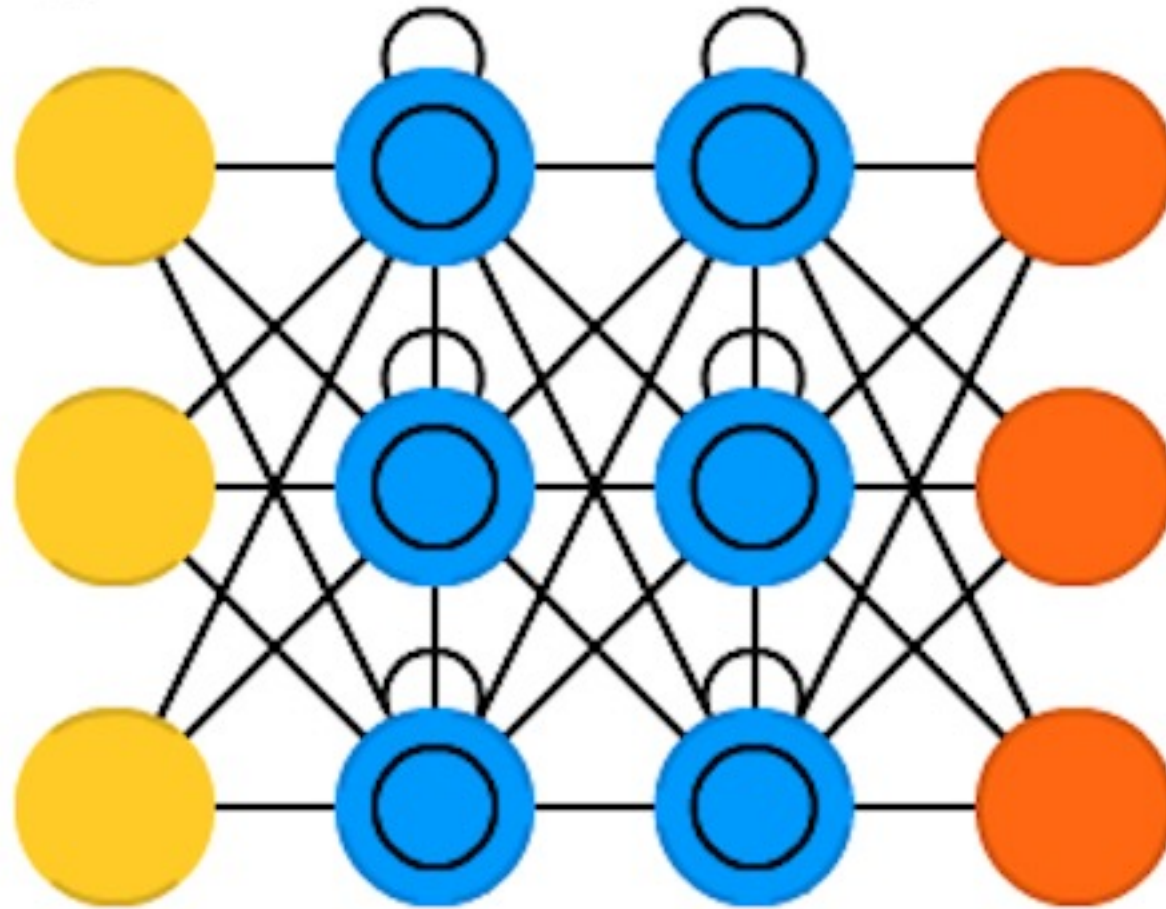
LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

Source: <http://www.asimovinstitute.org/neural-network-zoo/>

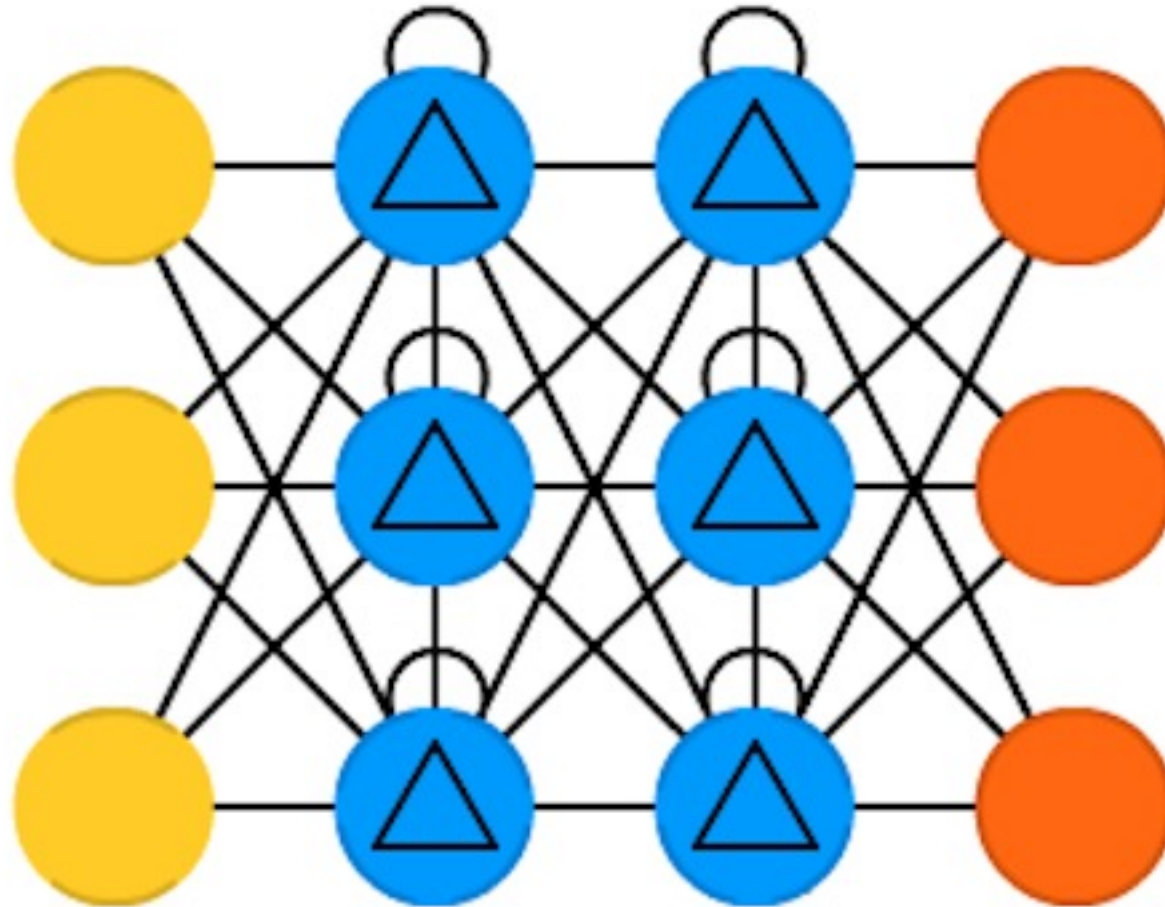
Recurrent Neural Networks (RNN)



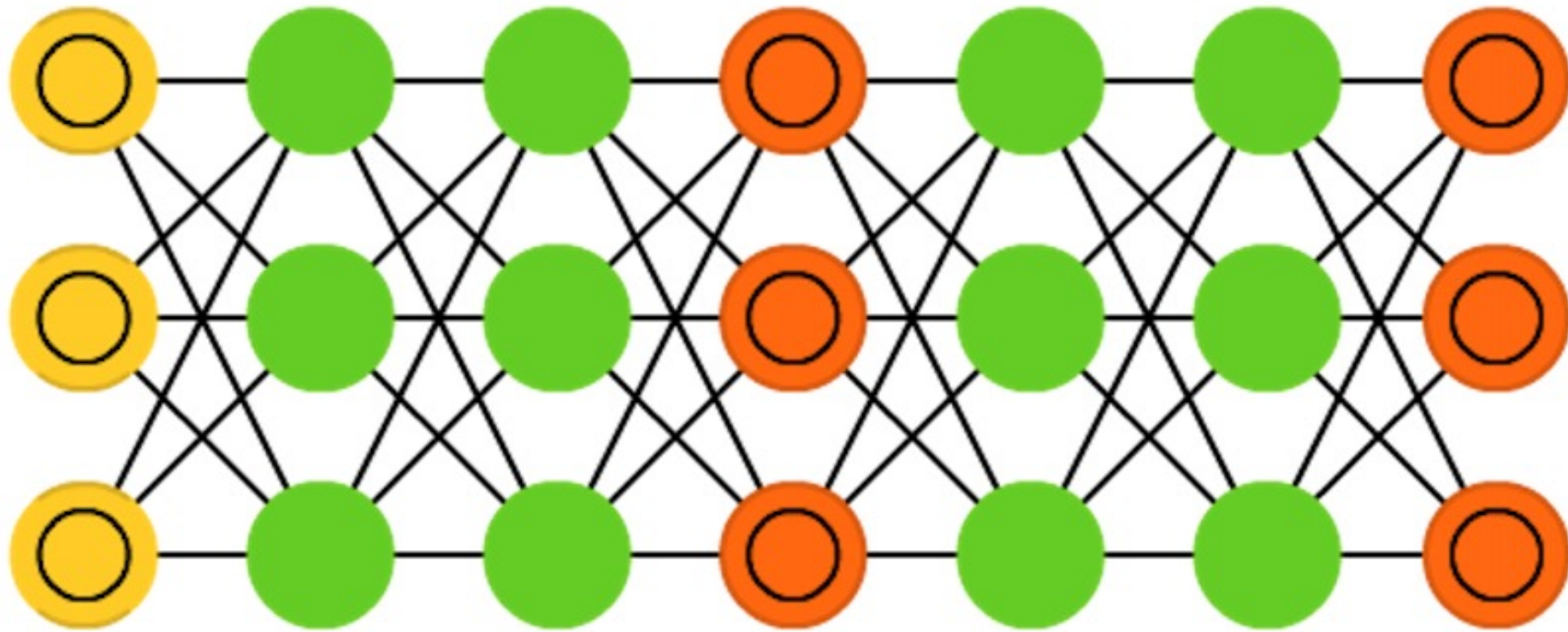
Long / Short Term Memory (LSTM)



Gated Recurrent Units (GRU)



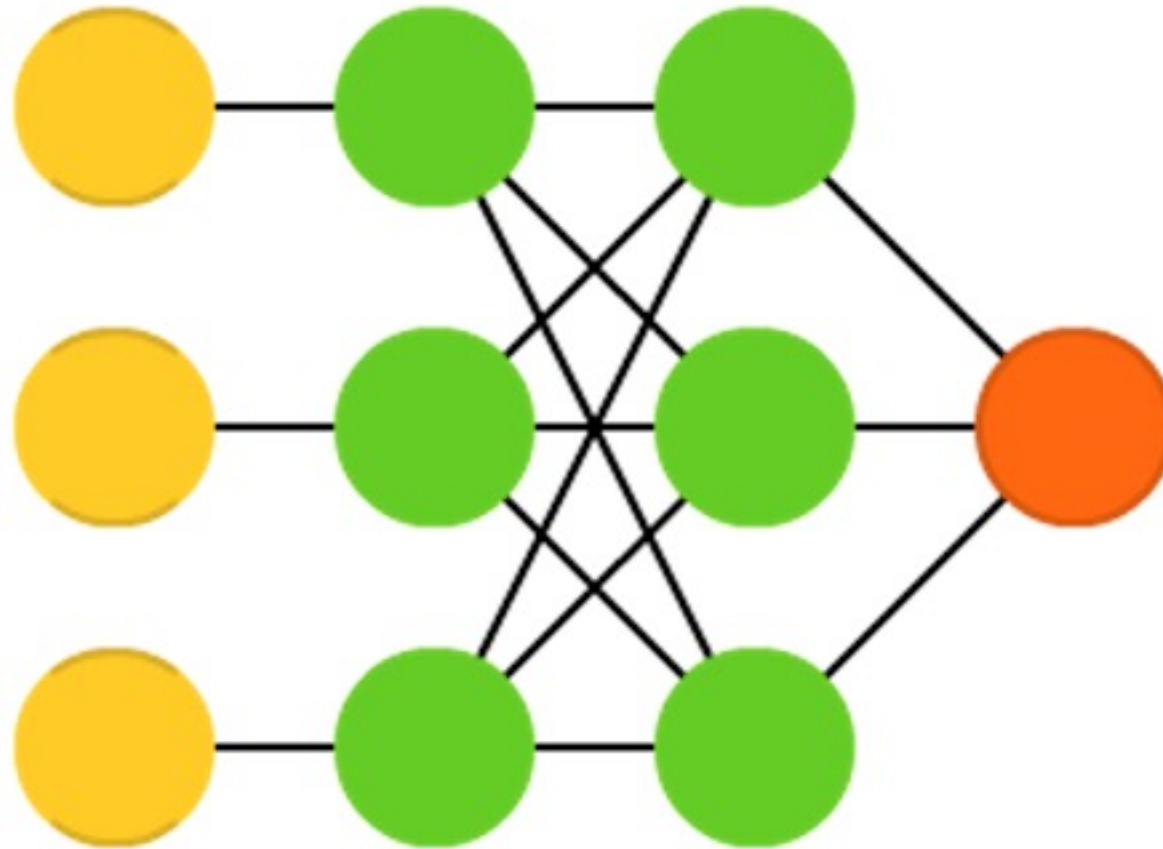
Generative Adversarial Networks (GAN)



Goodfellow, Ian, et al. "Generative adversarial nets." Advances in Neural Information Processing Systems. 2014.

Source: <http://www.asimovinstitute.org/neural-network-zoo/>

Support Vector Machines (SVM)



Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

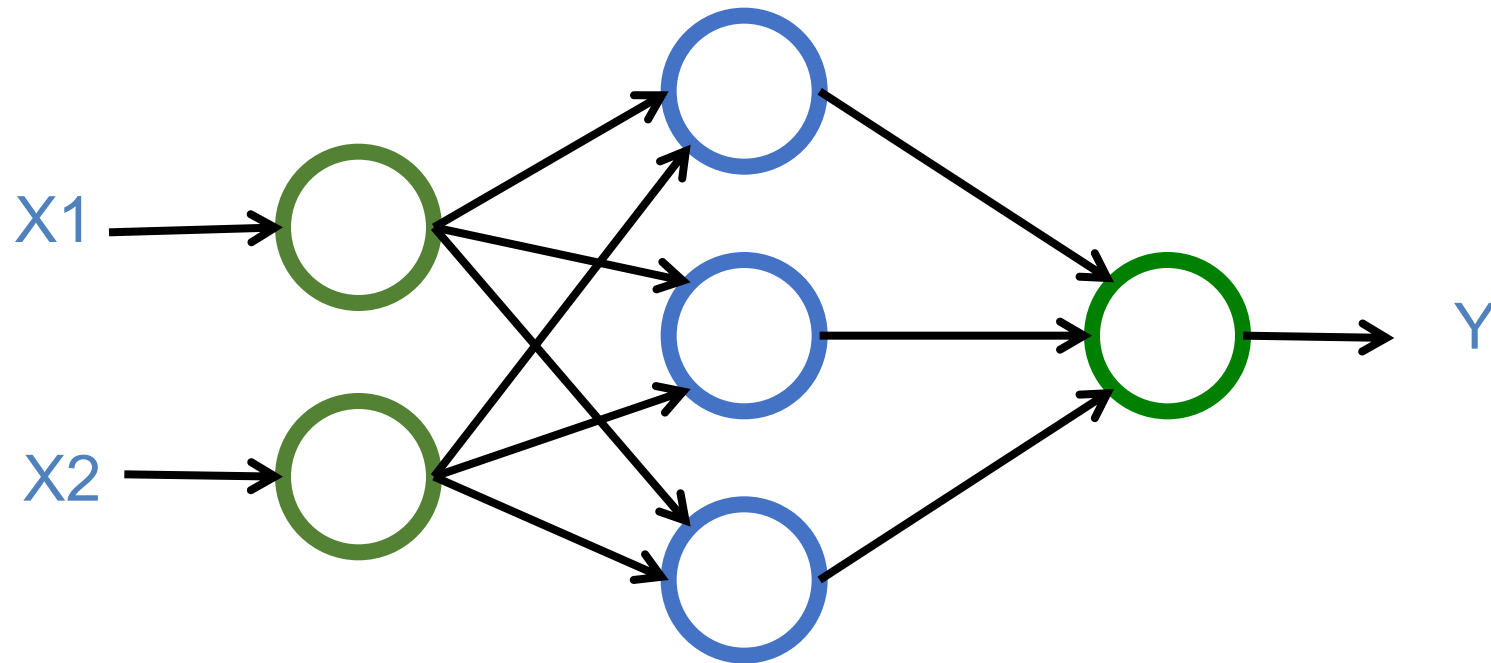
Source: <http://www.asimovinstitute.org/neural-network-zoo/>

Neural Networks

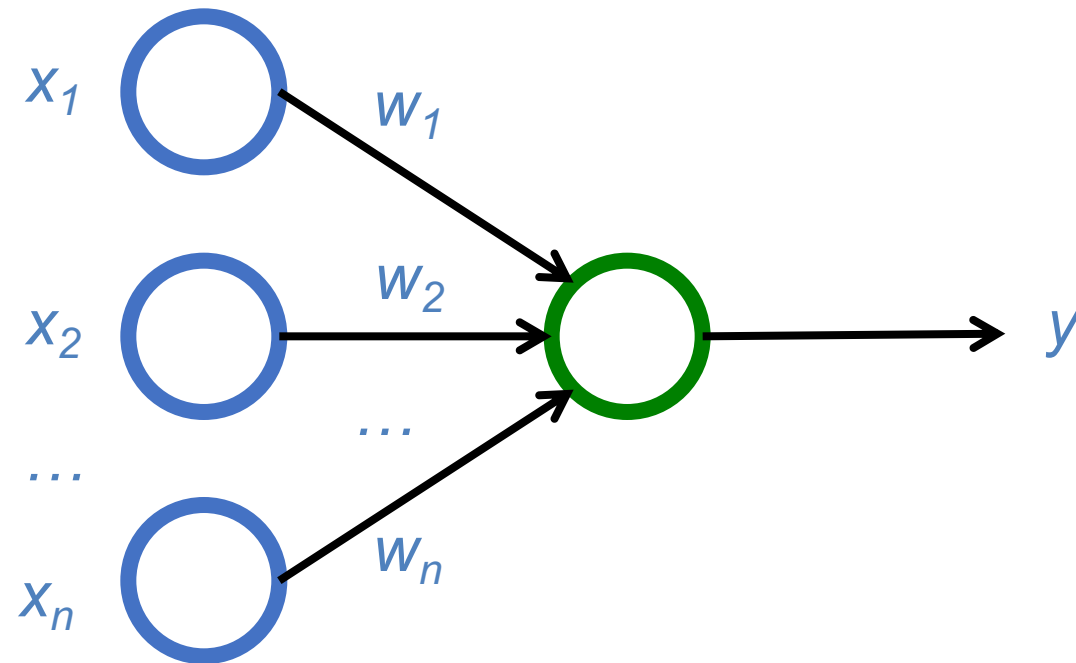
Input Layer
(X)

Hidden Layer
(H)

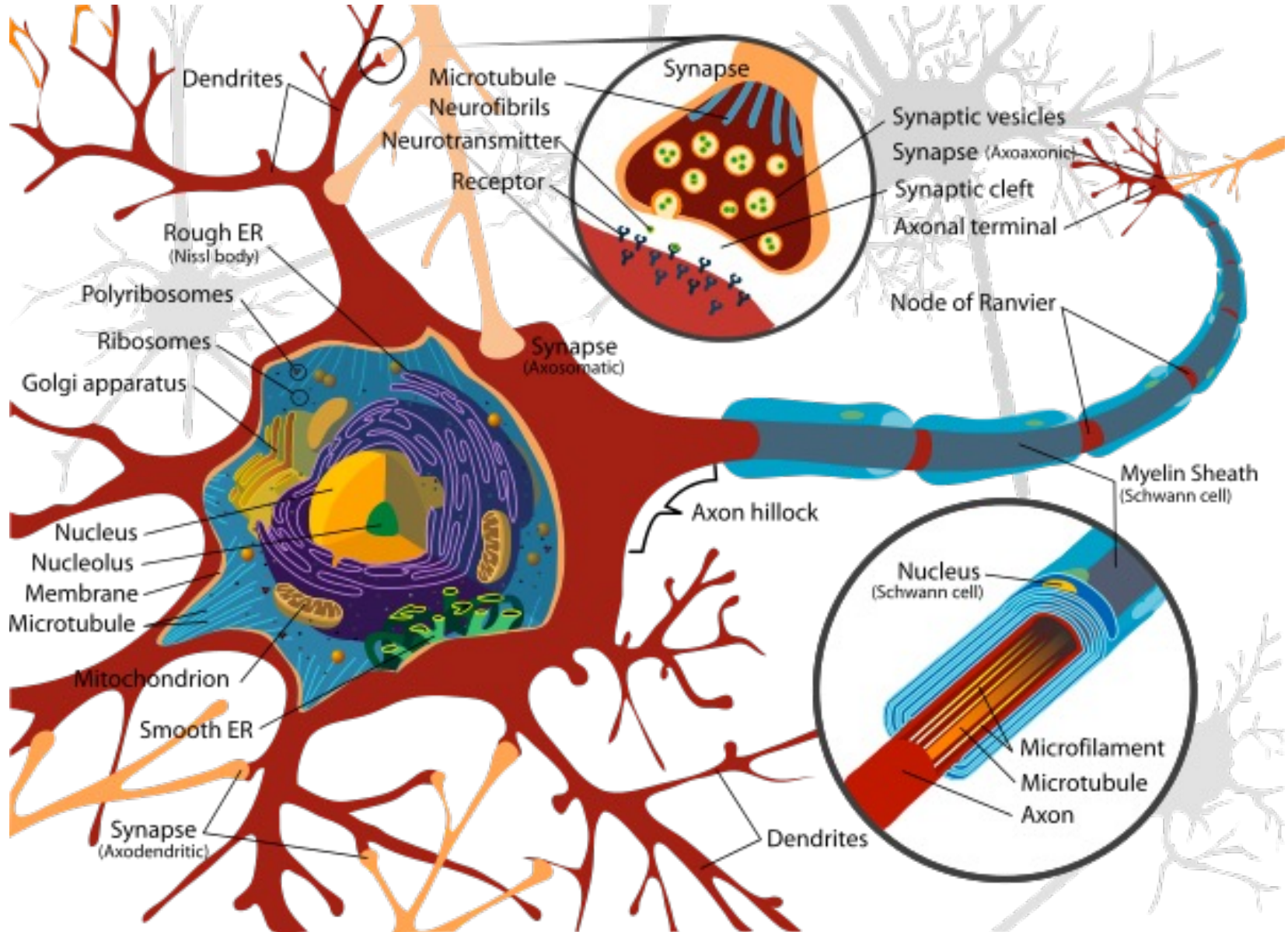
Output Layer
(Y)



The Neuron

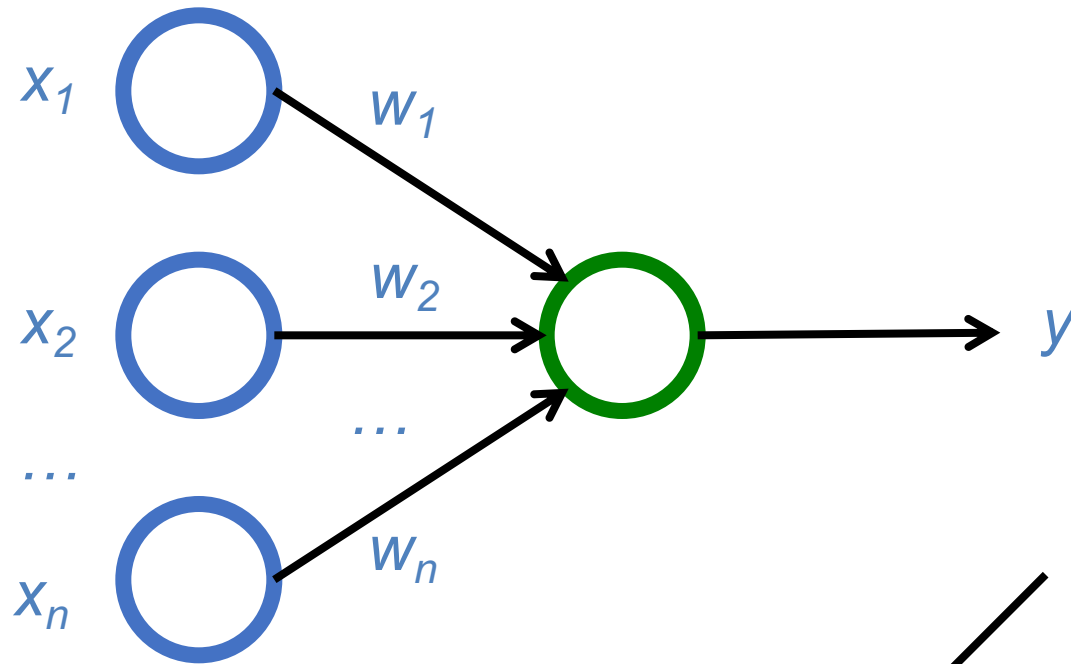


Neuron and Synapse



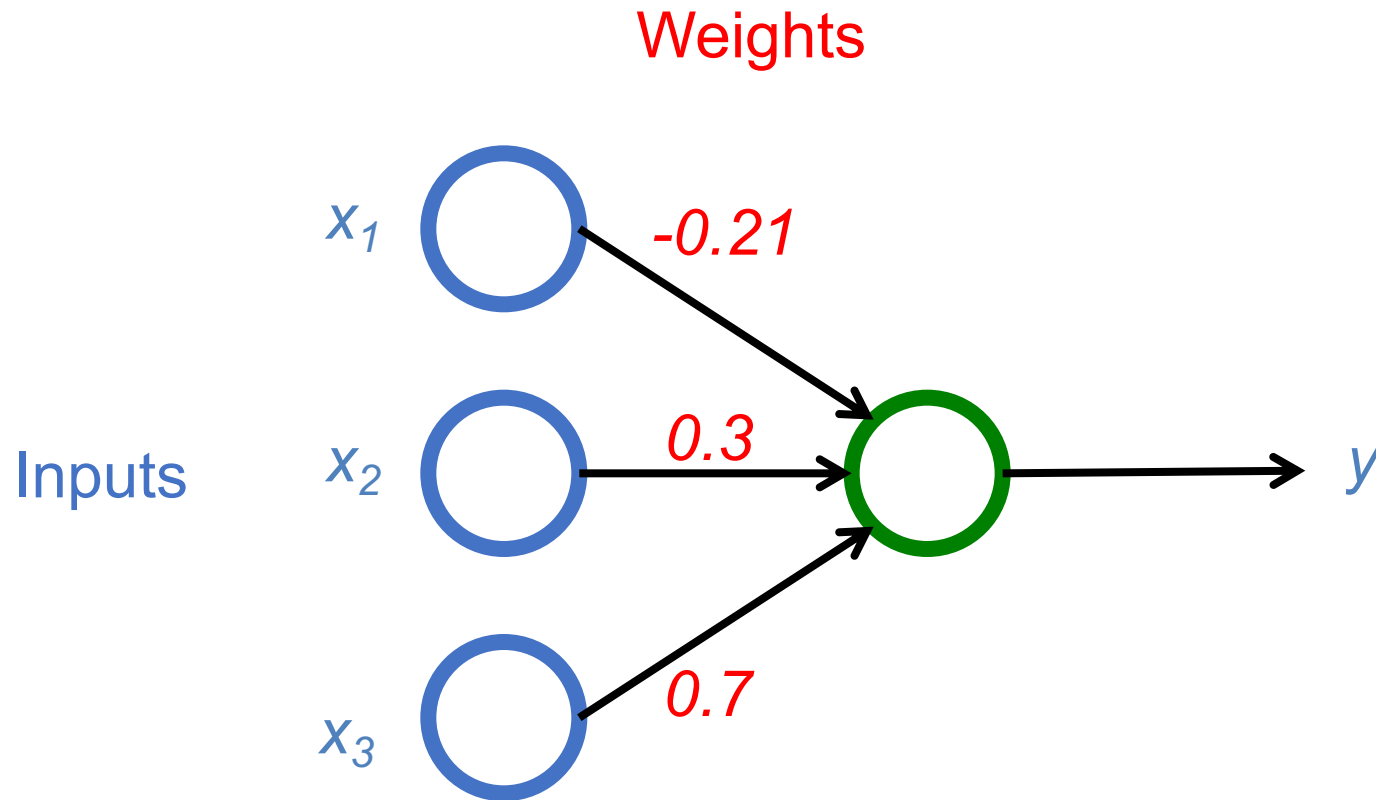
The Neuron

$$y = F\left(\sum_i w_i x_i\right)$$

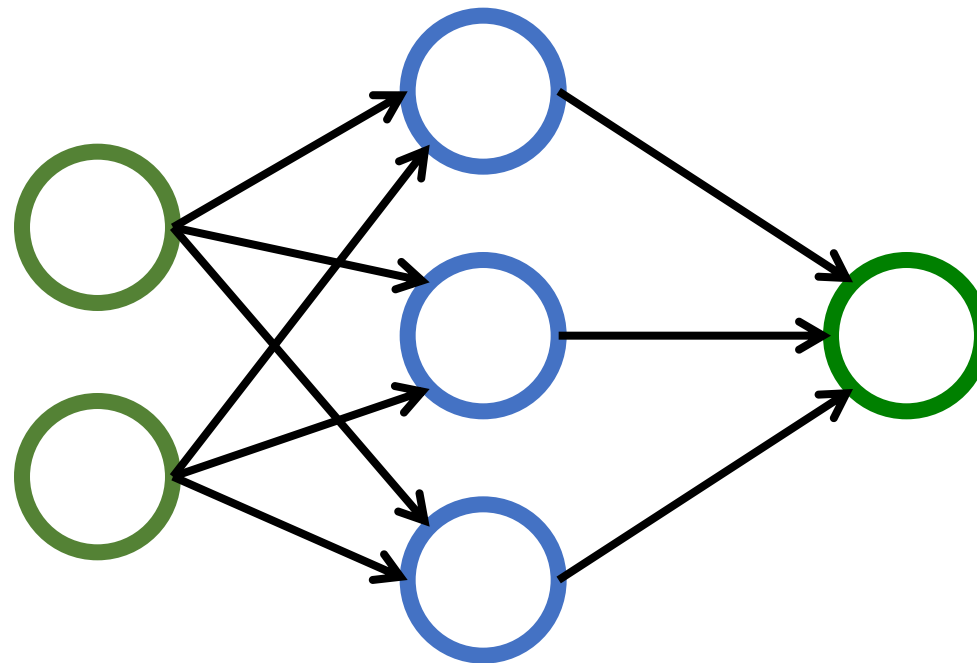


$$F(x) = \max(0, x)$$

$$y = \max (0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$



Neural Networks

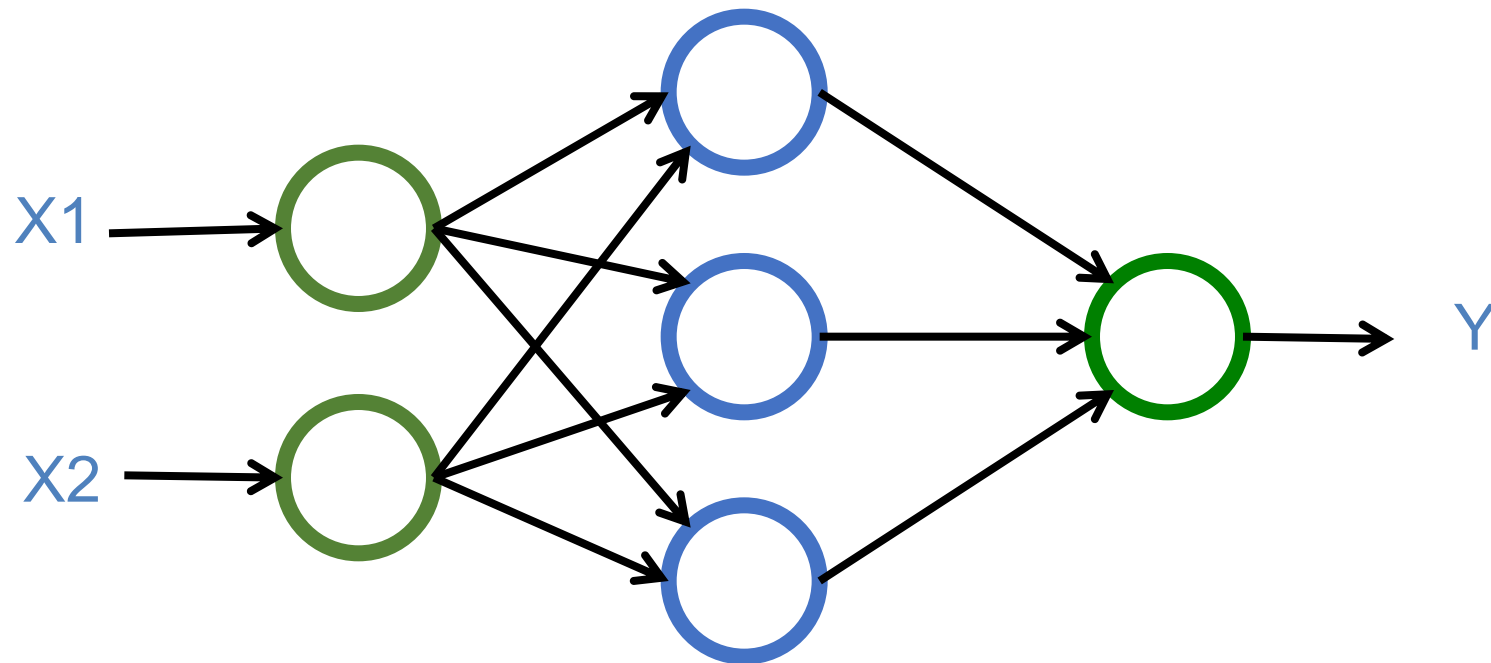


Neural Networks

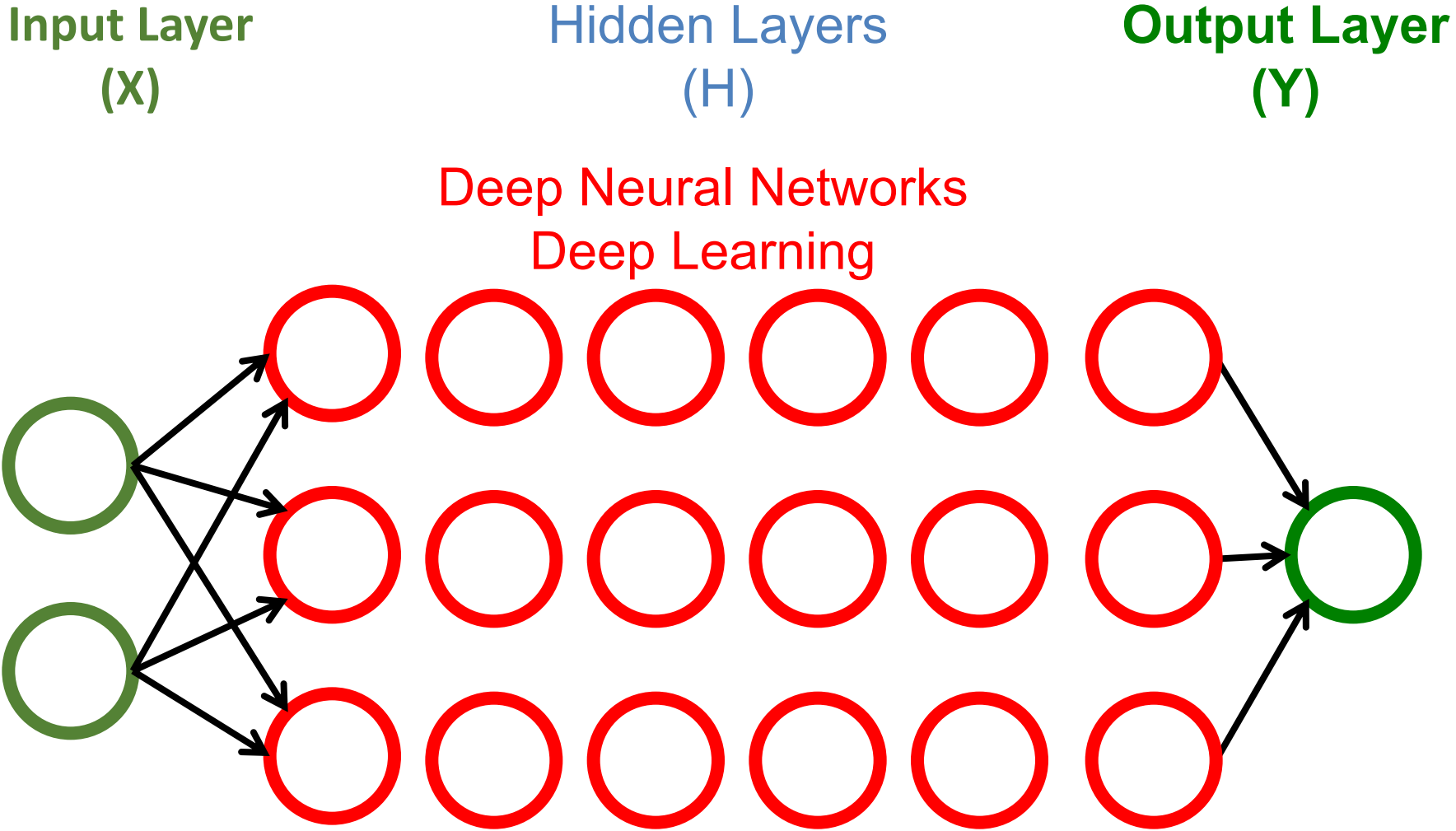
Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



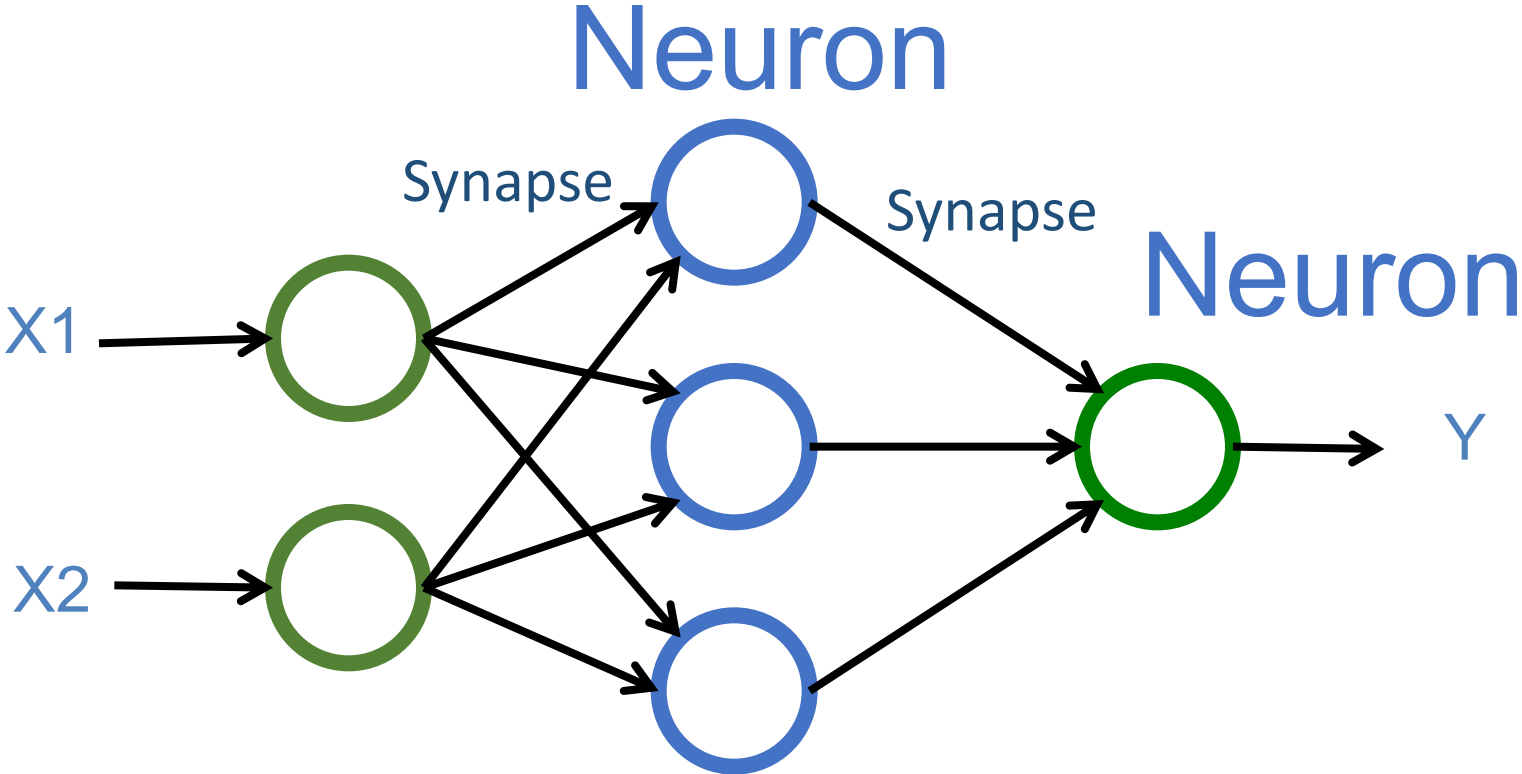
Neural Networks



Source: <https://www.youtube.com/watch?v=bxe2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZR1PoU>

Neural Networks

Input Layer (X) Hidden Layer (H) Output Layer (Y)

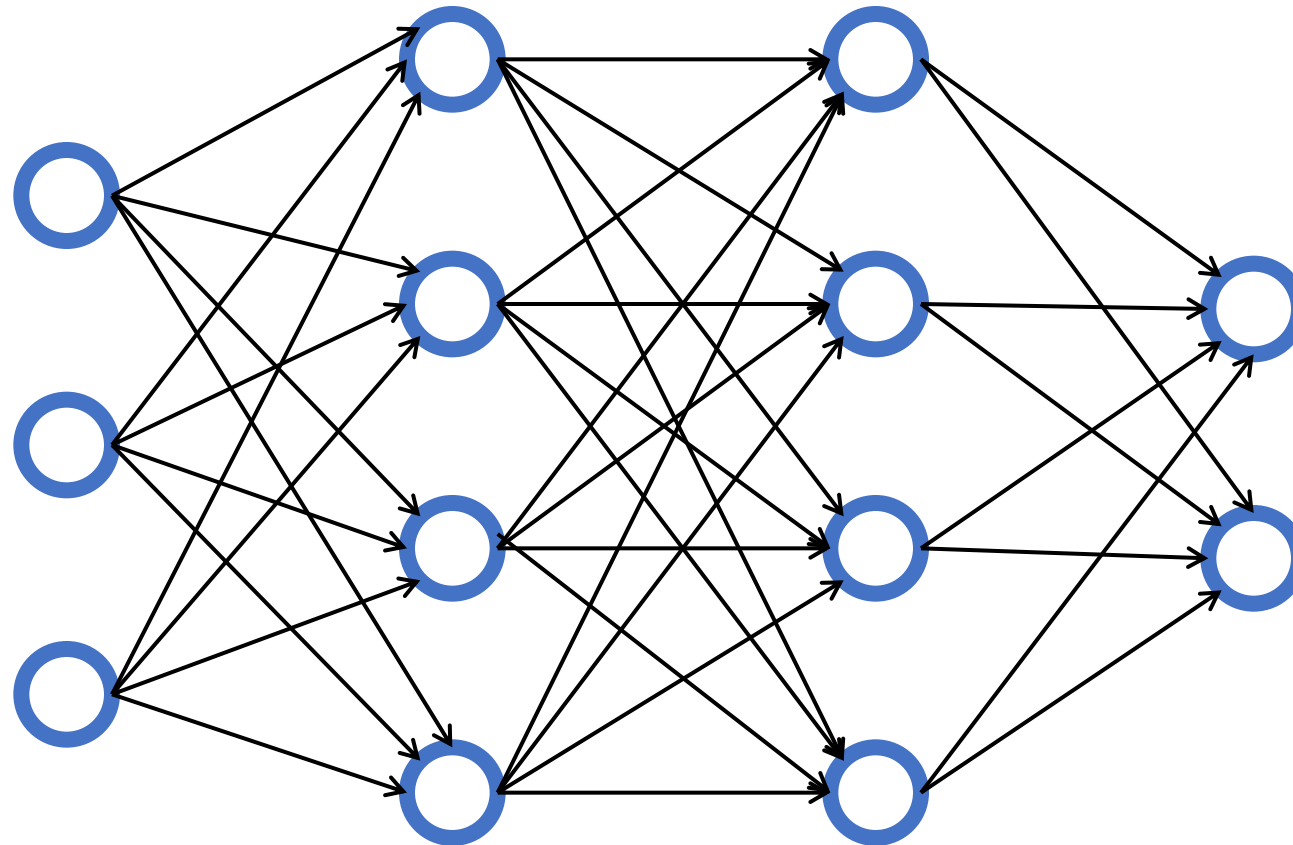


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

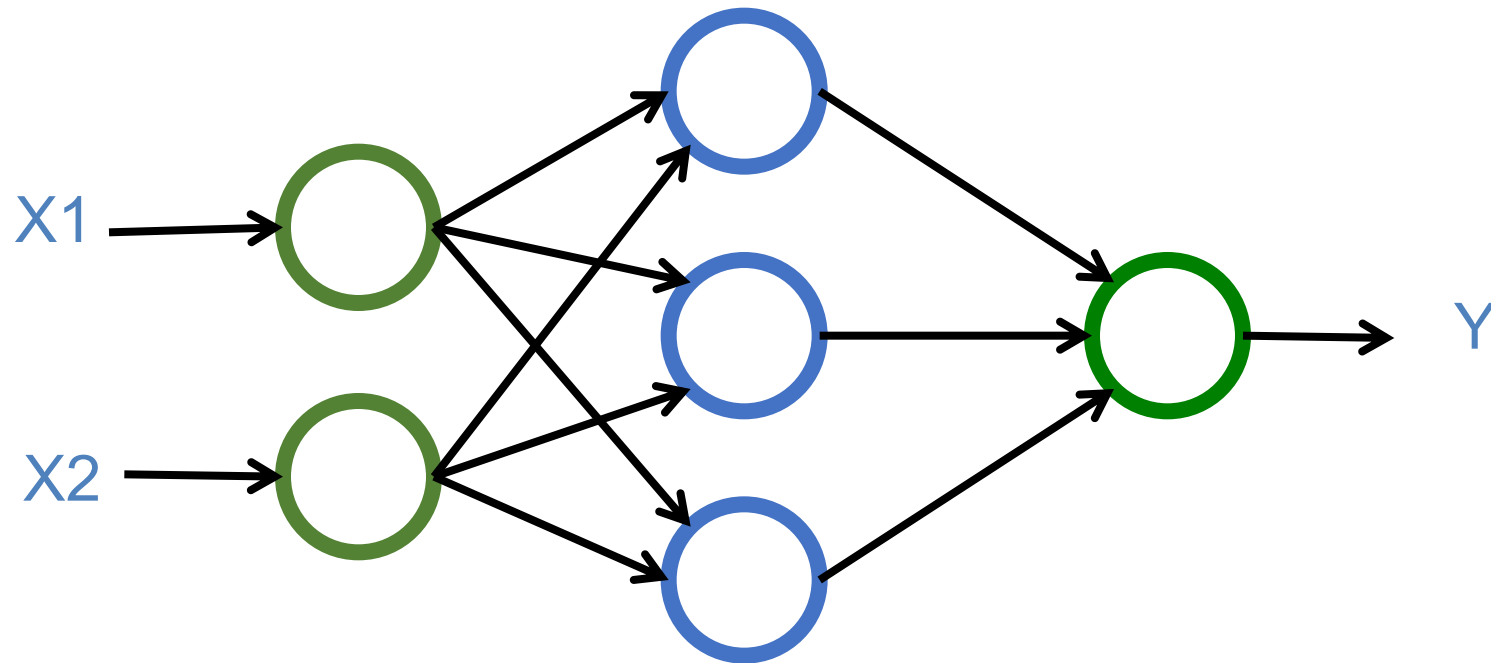


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



Linear function

$$y = f(x)$$

$$y = w_1 x + w_0$$

$$h_w(x) = w_1 x + w_0$$

$$Y = W X + b$$

Output

input

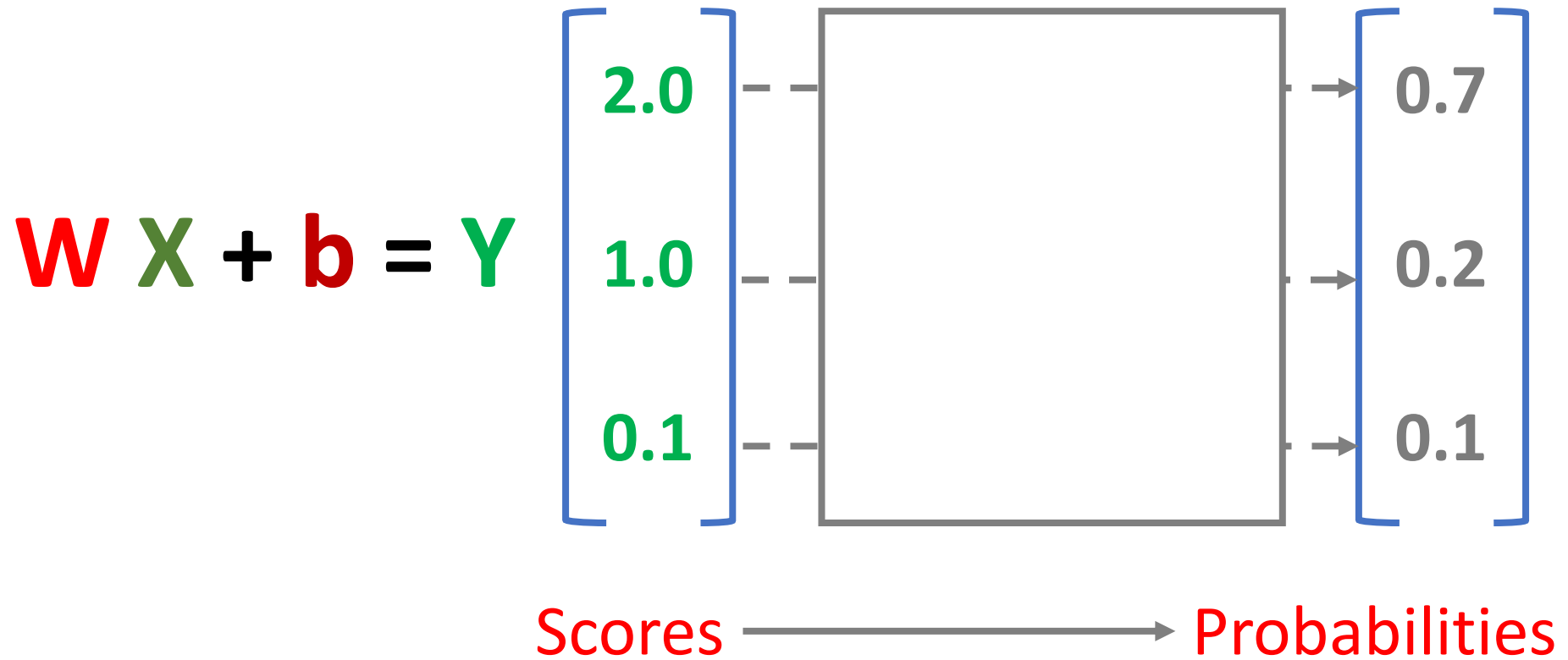
$$Y = W X + b$$

Weights

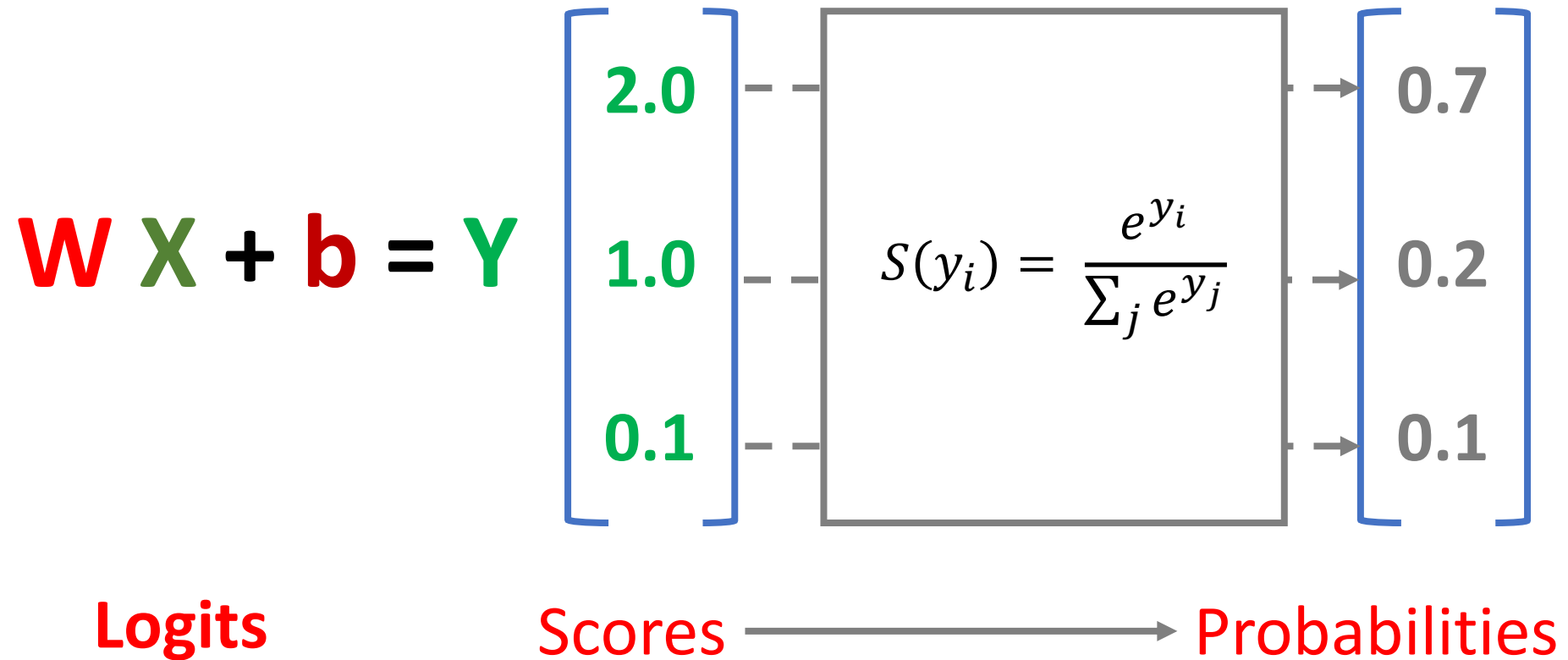
bias

Trained

The diagram illustrates the linear equation $Y = WX + b$. The variable Y is labeled as 'Output' with a downward arrow. The variable X is labeled as 'input' with a downward arrow. The variable W is labeled as 'Weights' with an upward arrow, and the variable b is labeled as 'bias' with an upward arrow. A red arrow labeled 'Trained' points to both the 'Weights' and 'bias' labels, indicating that these parameters are learned from data.



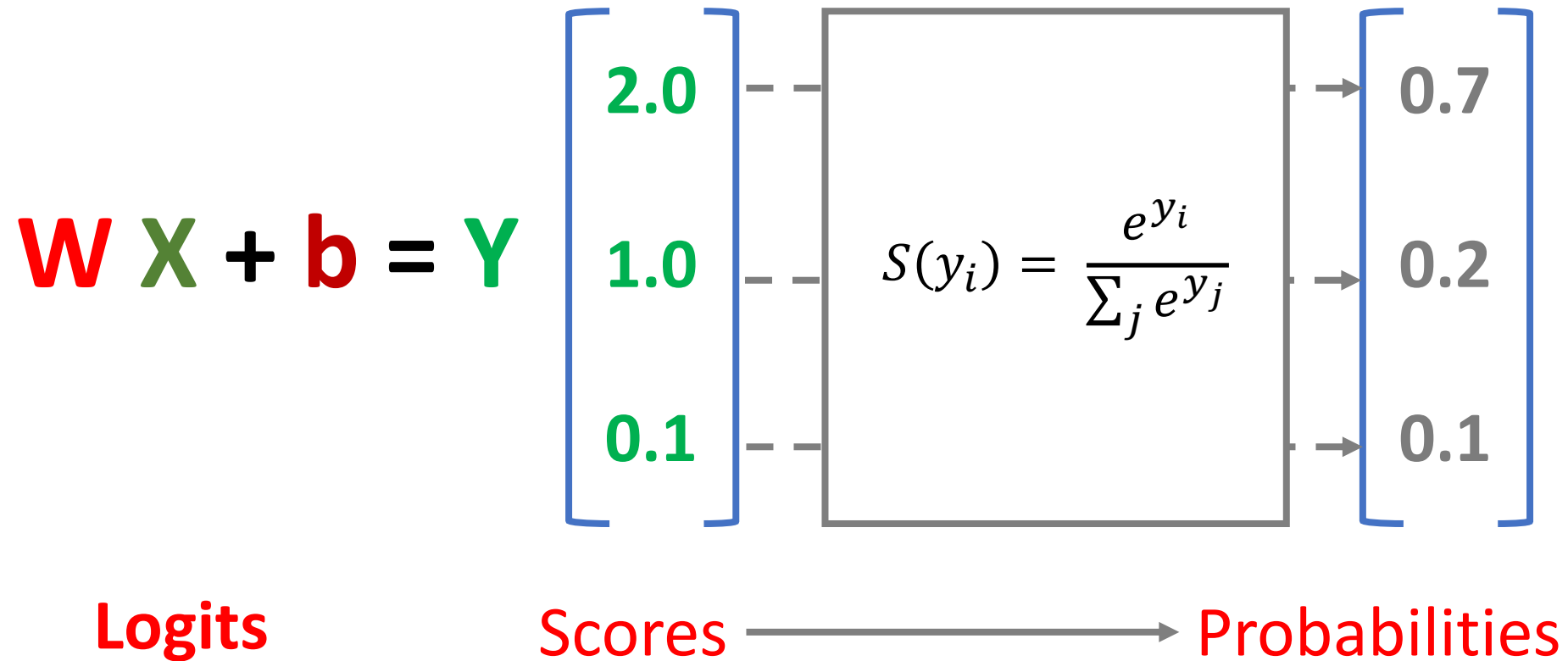
SoftMAX



$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1$$



Training a Network
=
Minimize the Cost Function

Training a Network

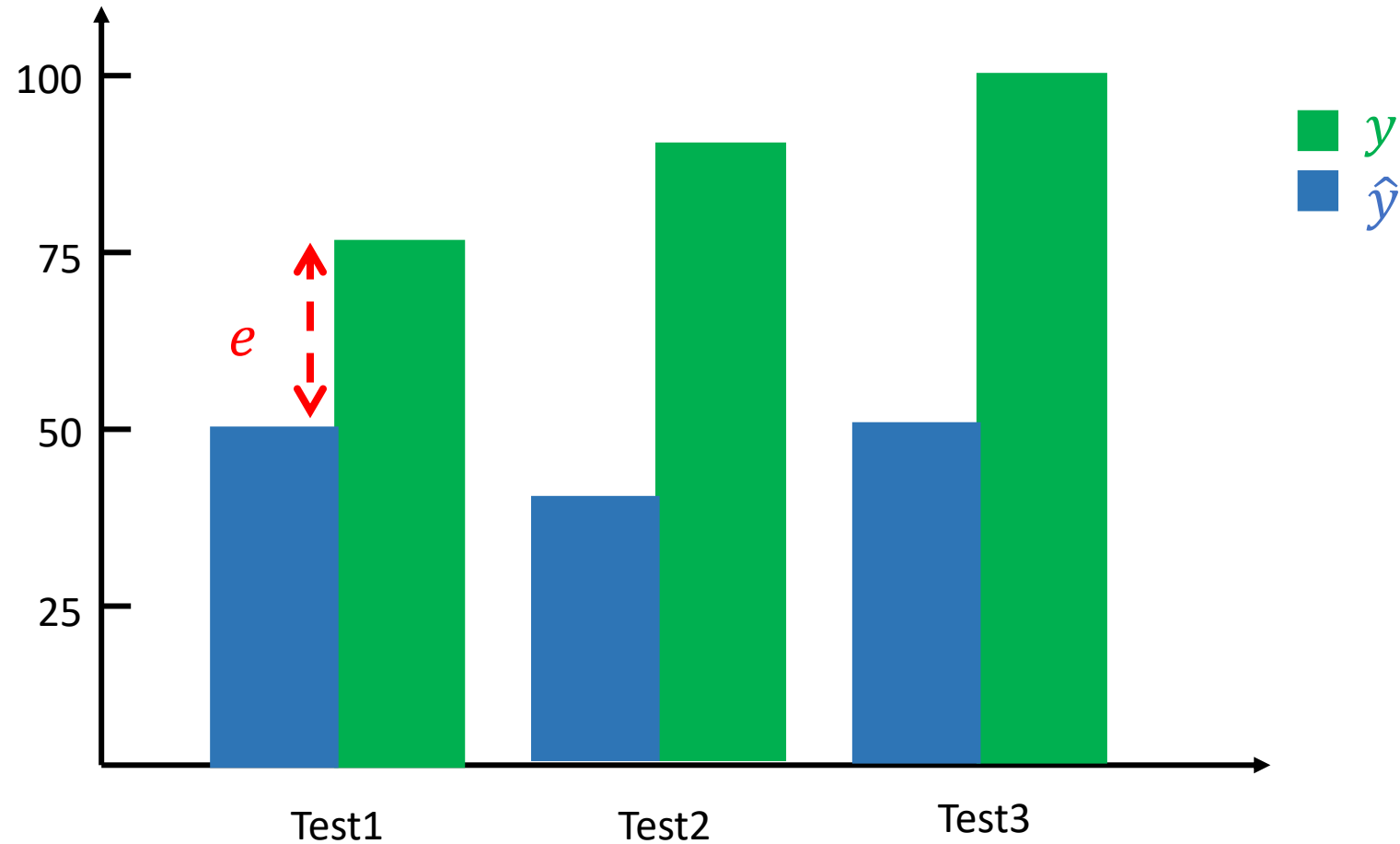
=

Minimize the **Cost** Function

Minimize the **Loss** Function

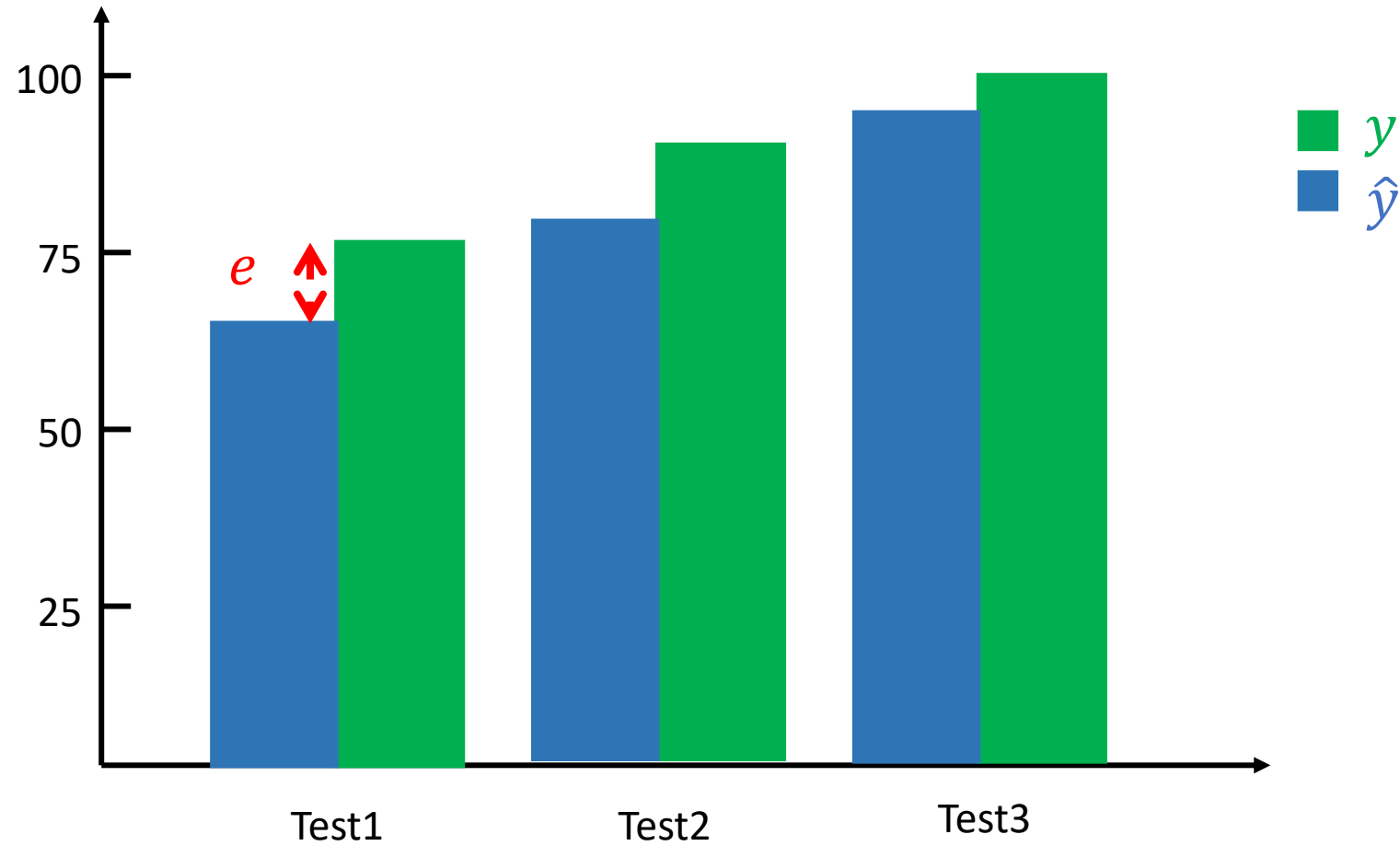
Error = Predict Y - Actual Y

Error : Cost : Loss



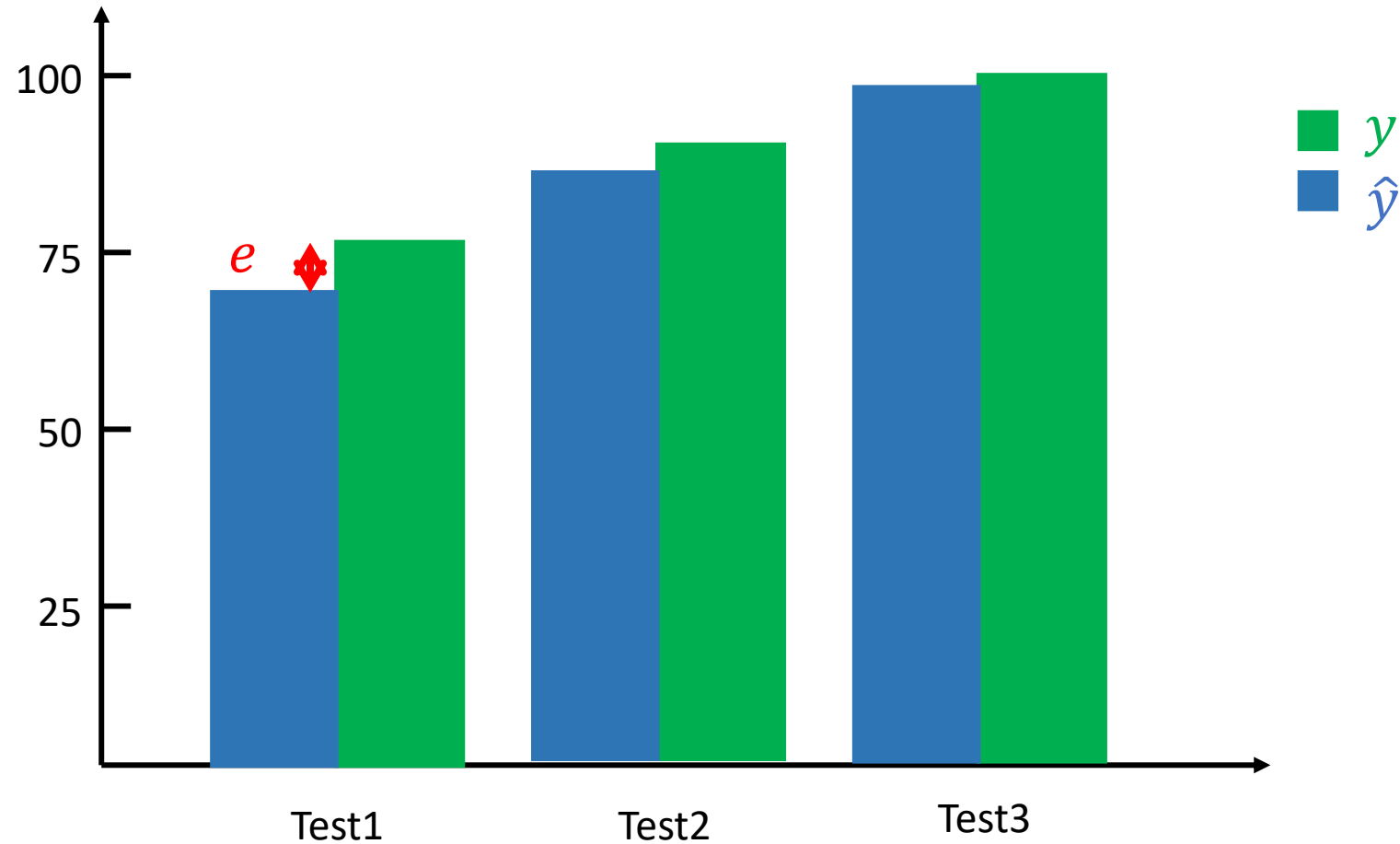
Error = Predict Y - Actual Y

Error : Cost : Loss



Error = Predict Y - Actual Y

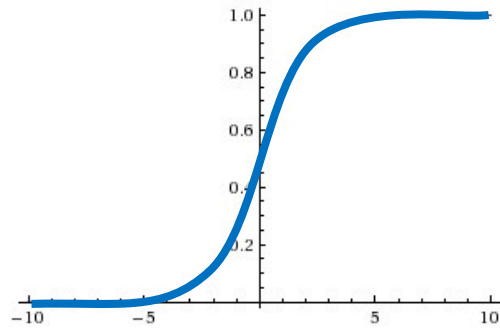
Error : Cost : Loss



Activation Functions

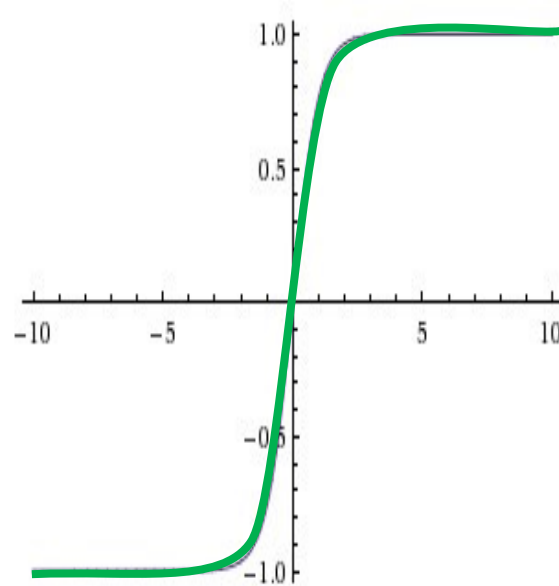
Activation Functions

Sigmoid



[0, 1]

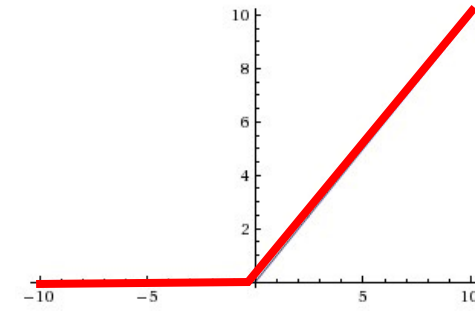
TanH



[-1, 1]

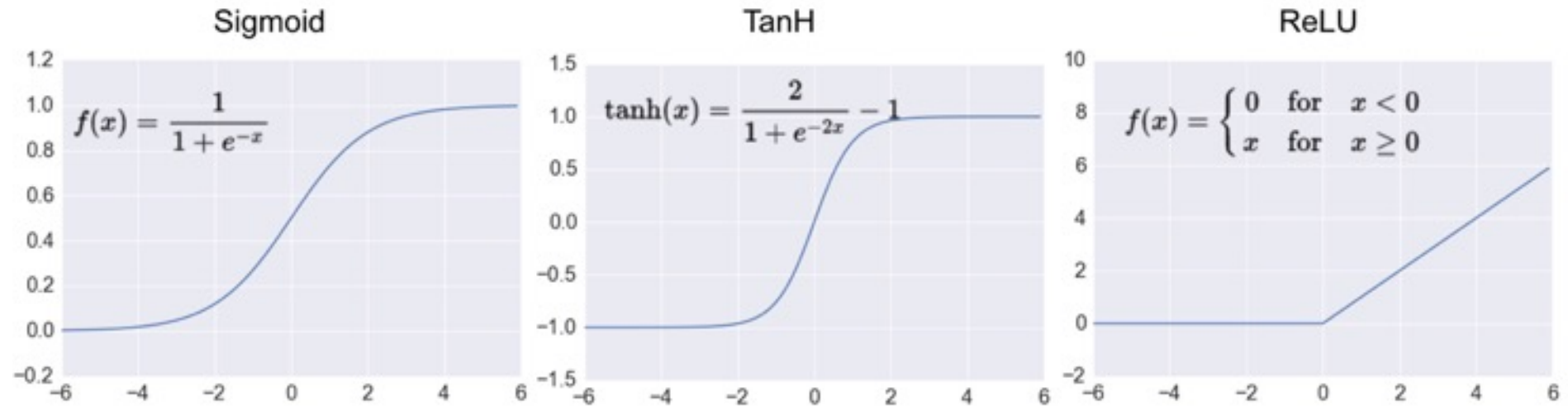
ReLU

(Rectified Linear Unit)



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

Activation Functions



Loss Function

Binary Classification: 2 Class

**Activation Function:
Sigmoid**

**Loss Function:
Binary Cross-Entropy**

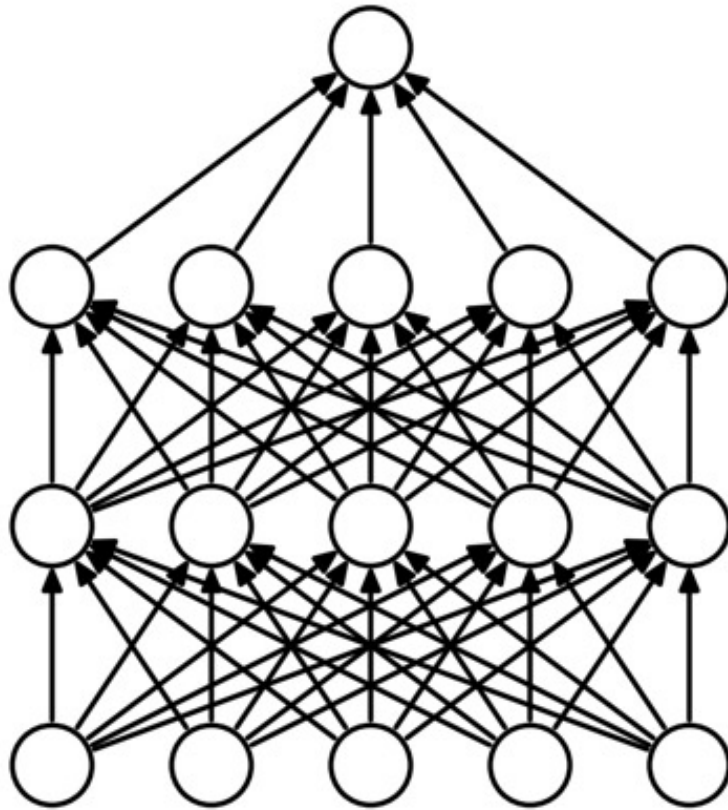
Multiple Classification: 10 Class

**Activation Function:
SoftMAX**

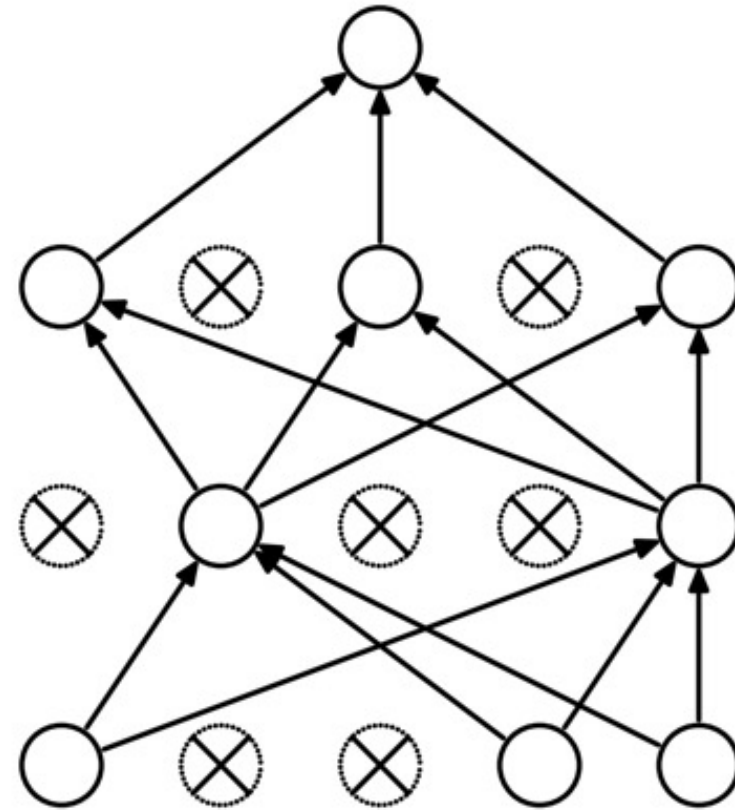
**Loss Function:
Categorical Cross-Entropy**

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net



(b) After applying dropout.

Source: Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.

"Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15, no. 1 (2014): 1929-1958.

Learning Algorithm

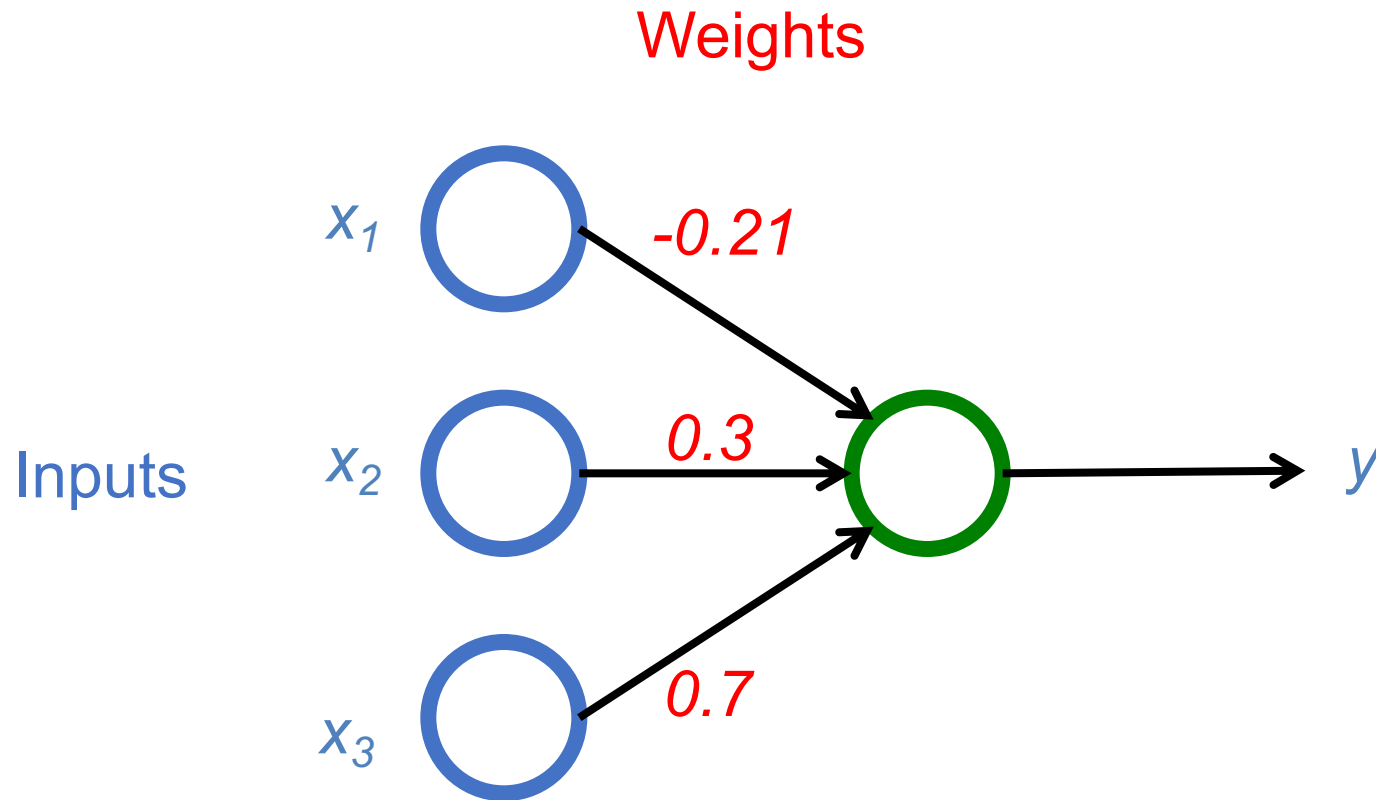
While not done:

Pick a random training example “(input, label)”

Run neural network on “input”

Adjust weights on edges to make output closer to “label”

$$y = \max (0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

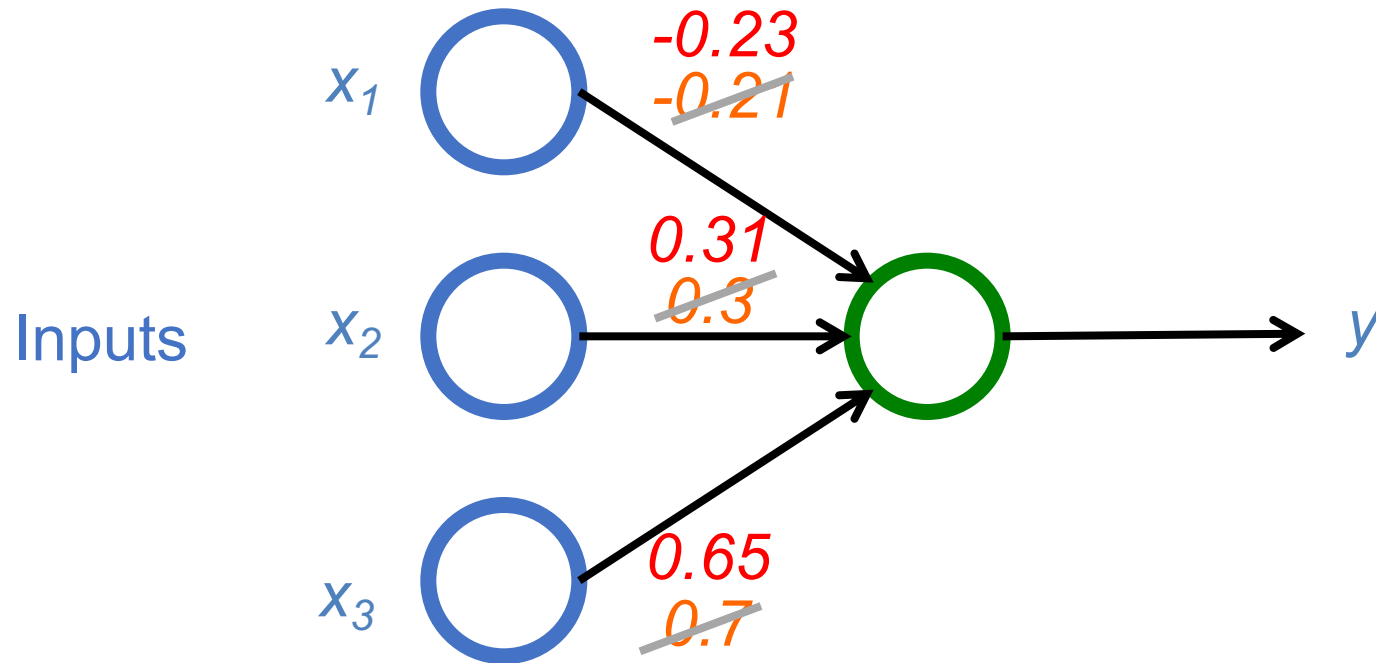


Next time:

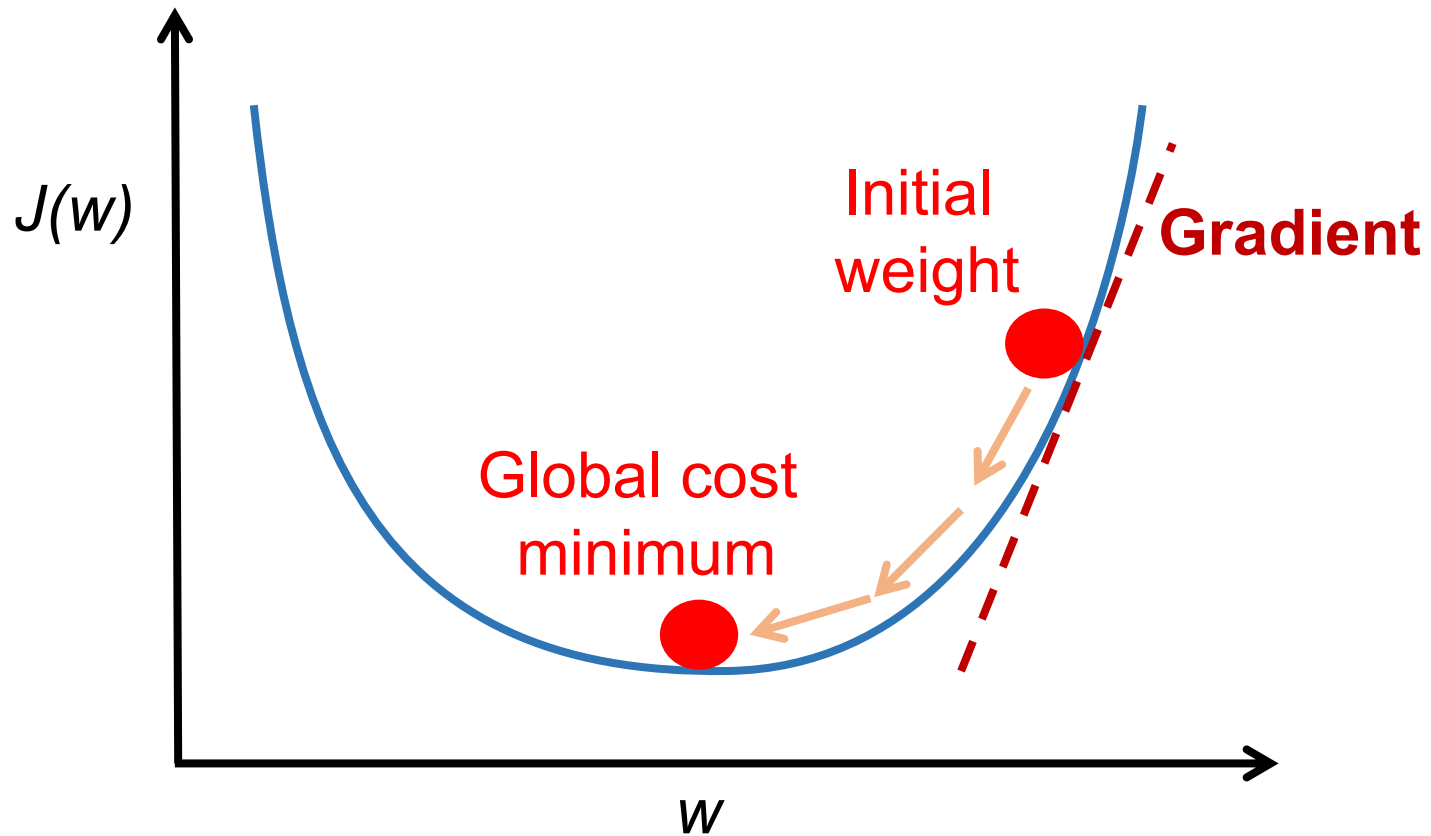
$$y = \max(0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3)$$

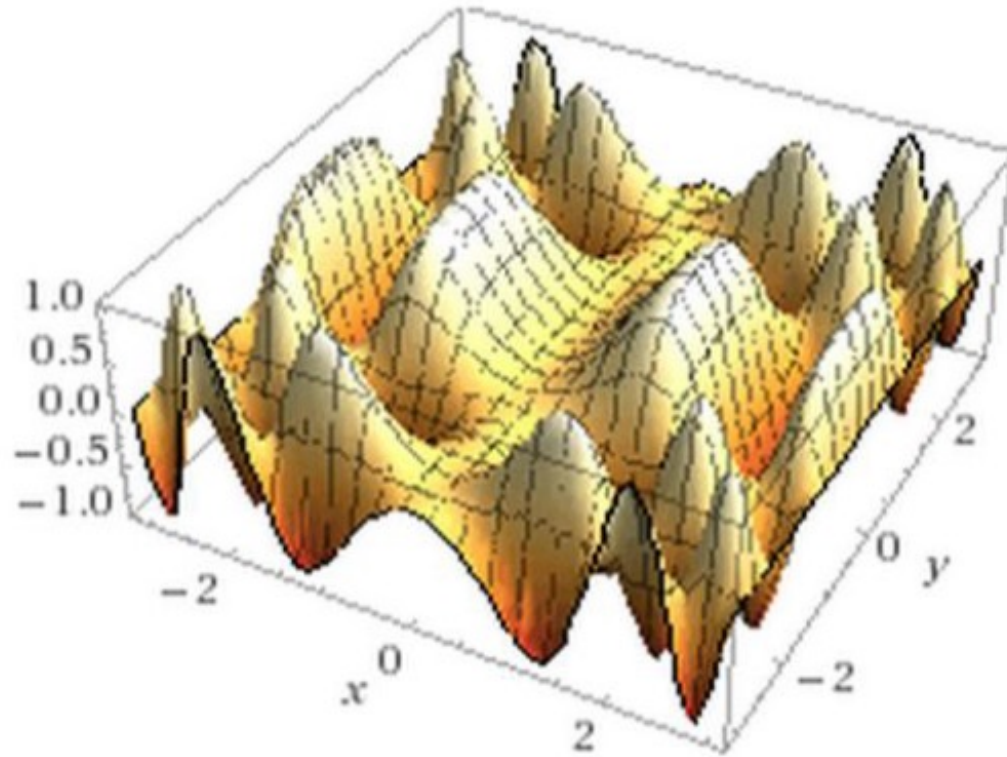
~~$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$~~

Weights



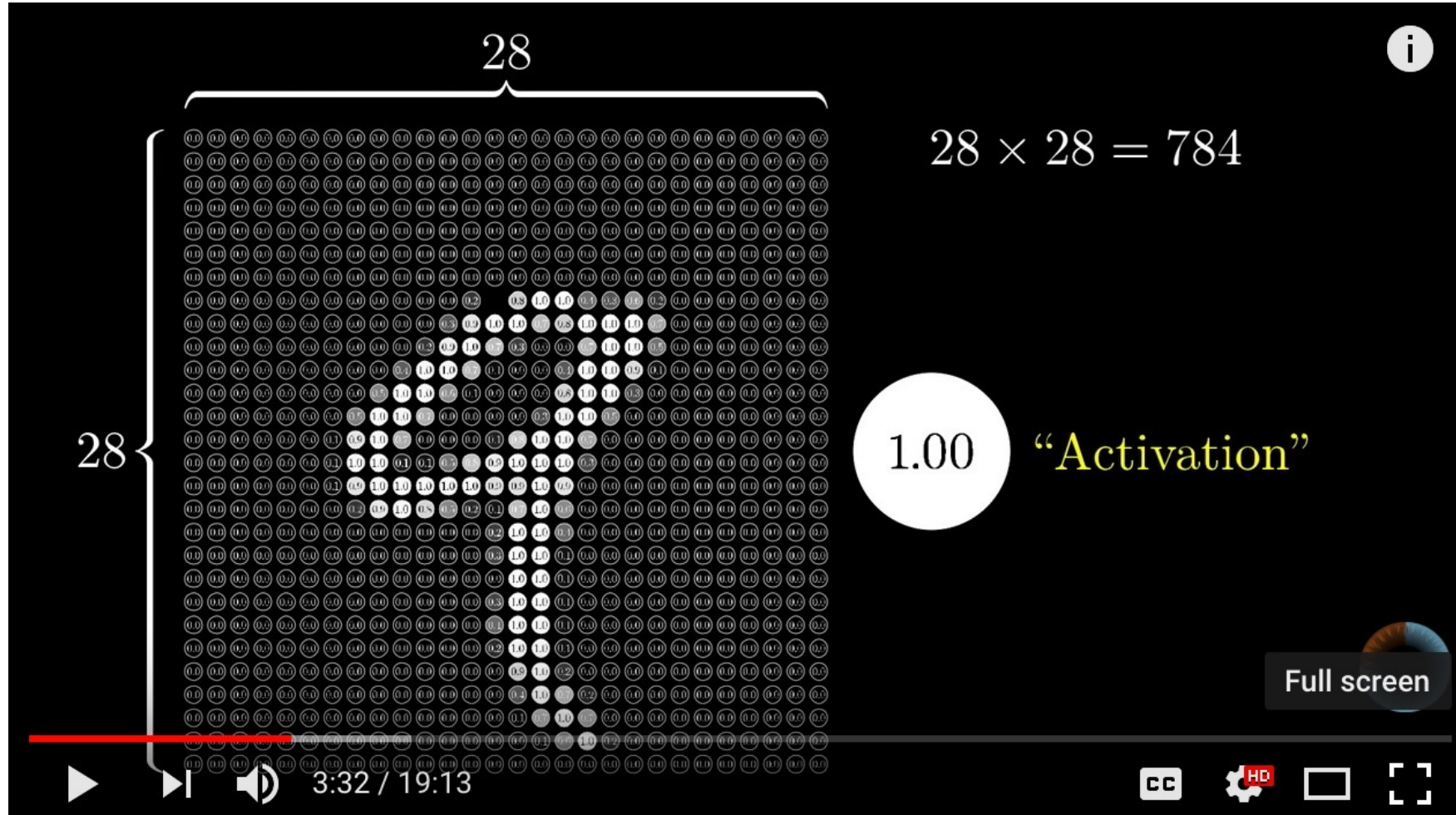
Optimizer: Stochastic Gradient Descent (SGD)





This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

Neural Network and Deep Learning




Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning,

<https://www.youtube.com/watch?v=aircAruvnKk>

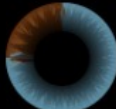
Gradient Descent

how neural networks learn

Average cost of all training data...

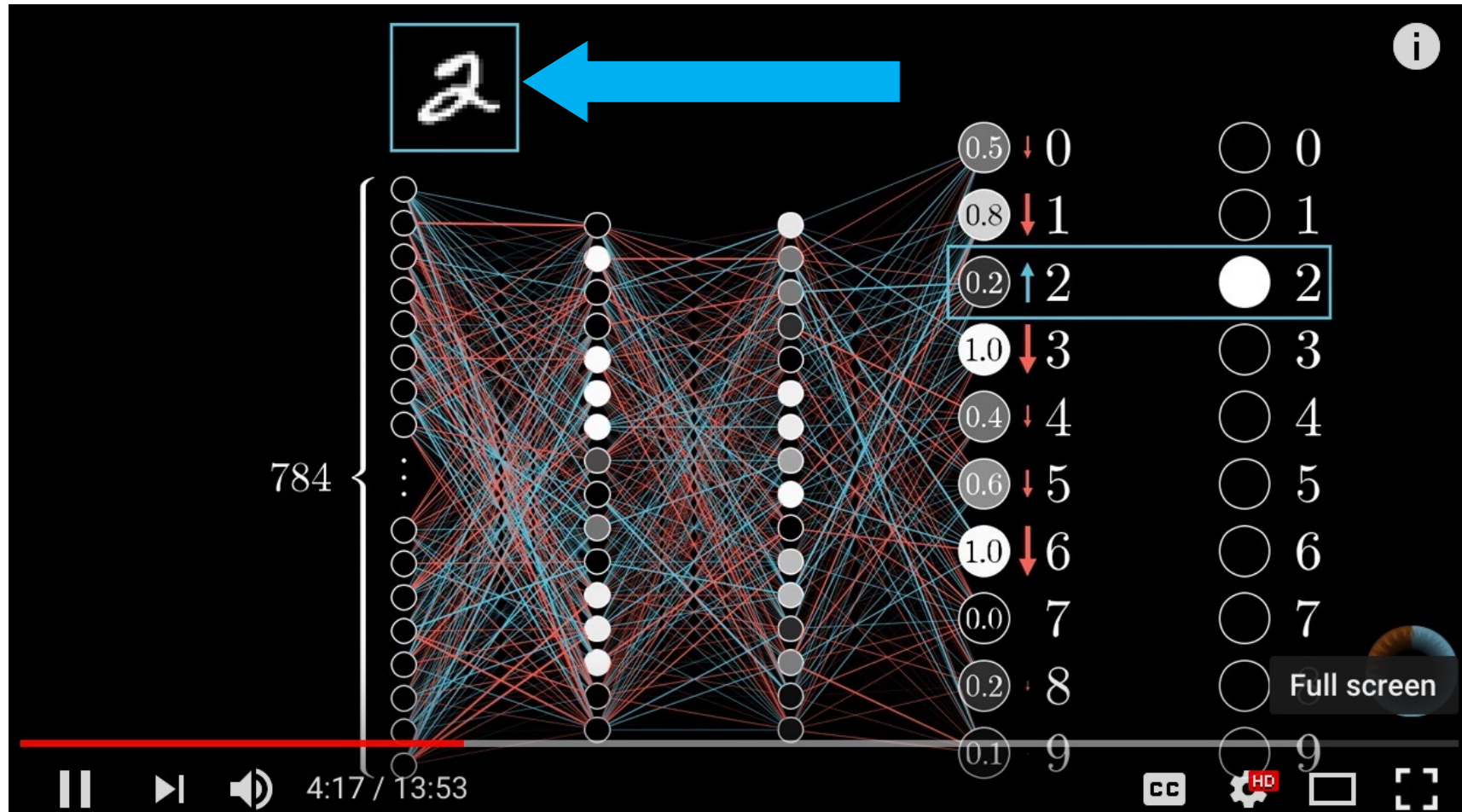
Cost of 

What's the "cost" of this difference?

Utter trash 

$(0.18 - 0.00)^2 +$	<input type="radio"/>	0
$(0.29 - 0.00)^2 +$	<input type="radio"/>	1
$(0.58 - 0.00)^2 +$	<input type="radio"/>	2
$(0.77 - 0.00)^2 +$	<input type="radio"/>	3
$(0.20 - 0.00)^2 +$	<input type="radio"/>	4
$(0.36 - 0.00)^2 +$	<input type="radio"/>	5
$(0.93 - 0.00)^2 +$	<input type="radio"/>	6
$(1.00 - 0.00)^2 +$	<input type="radio"/>	7
$(0.95 - 1.00)^2 +$	<input checked="" type="radio"/>	8
$(0.35 - 0.00)^2$	<input type="radio"/>	9

Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=llg3gGewQ5U>

Learning Algorithm

While not done:

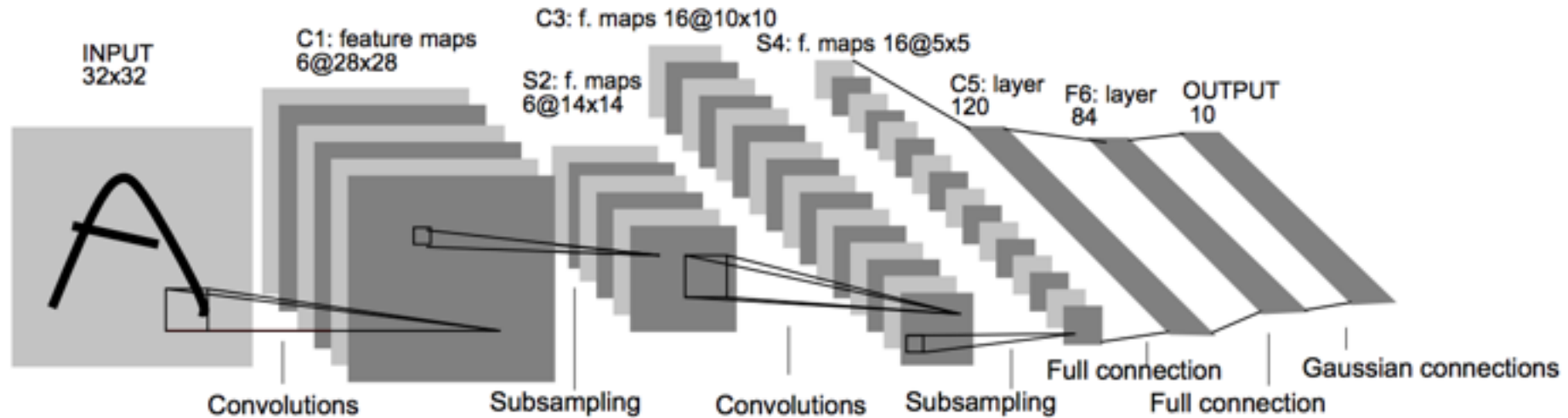
Pick a random training example “(input, label)”

Run neural network on “input”

Adjust weights on edges to make output closer to “label”

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN)



Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)

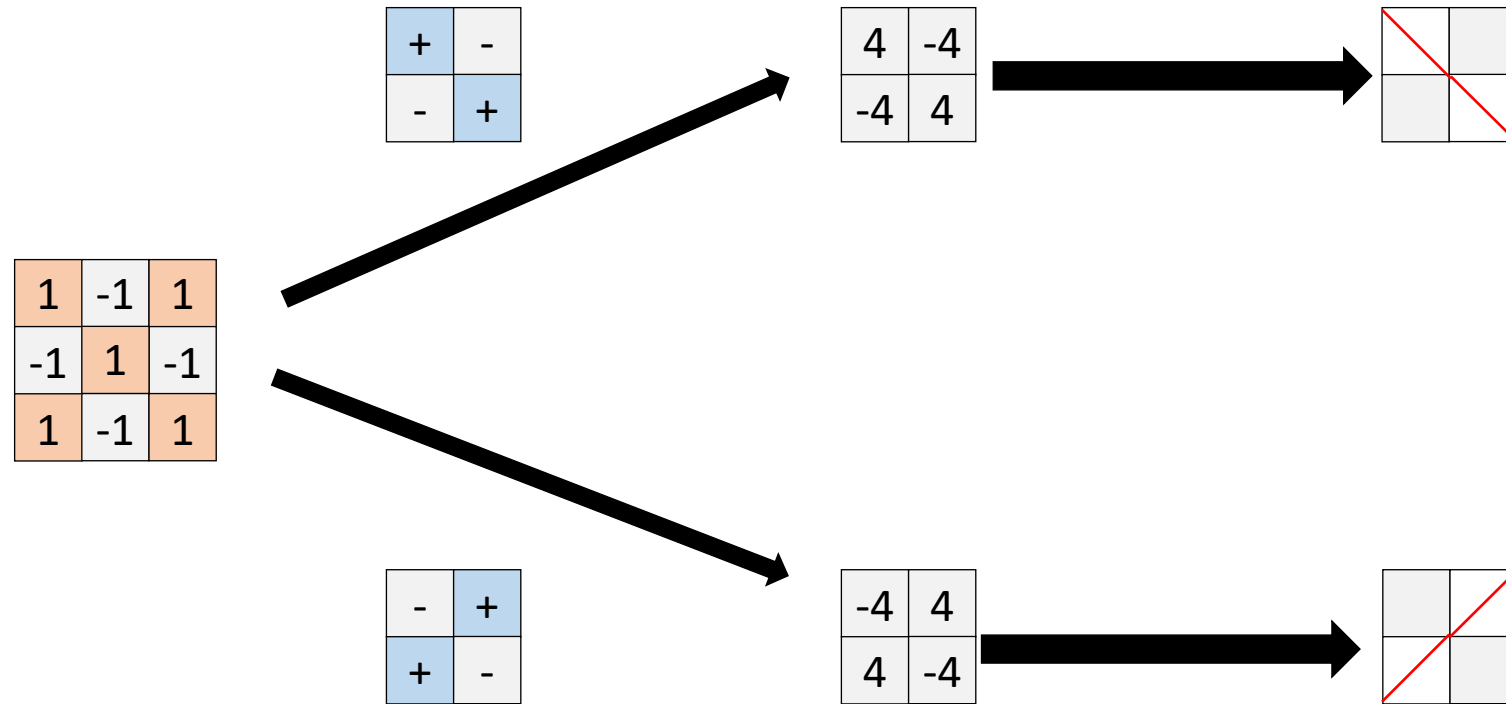
Source: <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

Source: LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
"Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

Convolutional Neural Networks (CNN)

- **Convolution**
- **Pooling**
- **Fully Connection (FC) (Flattening)**

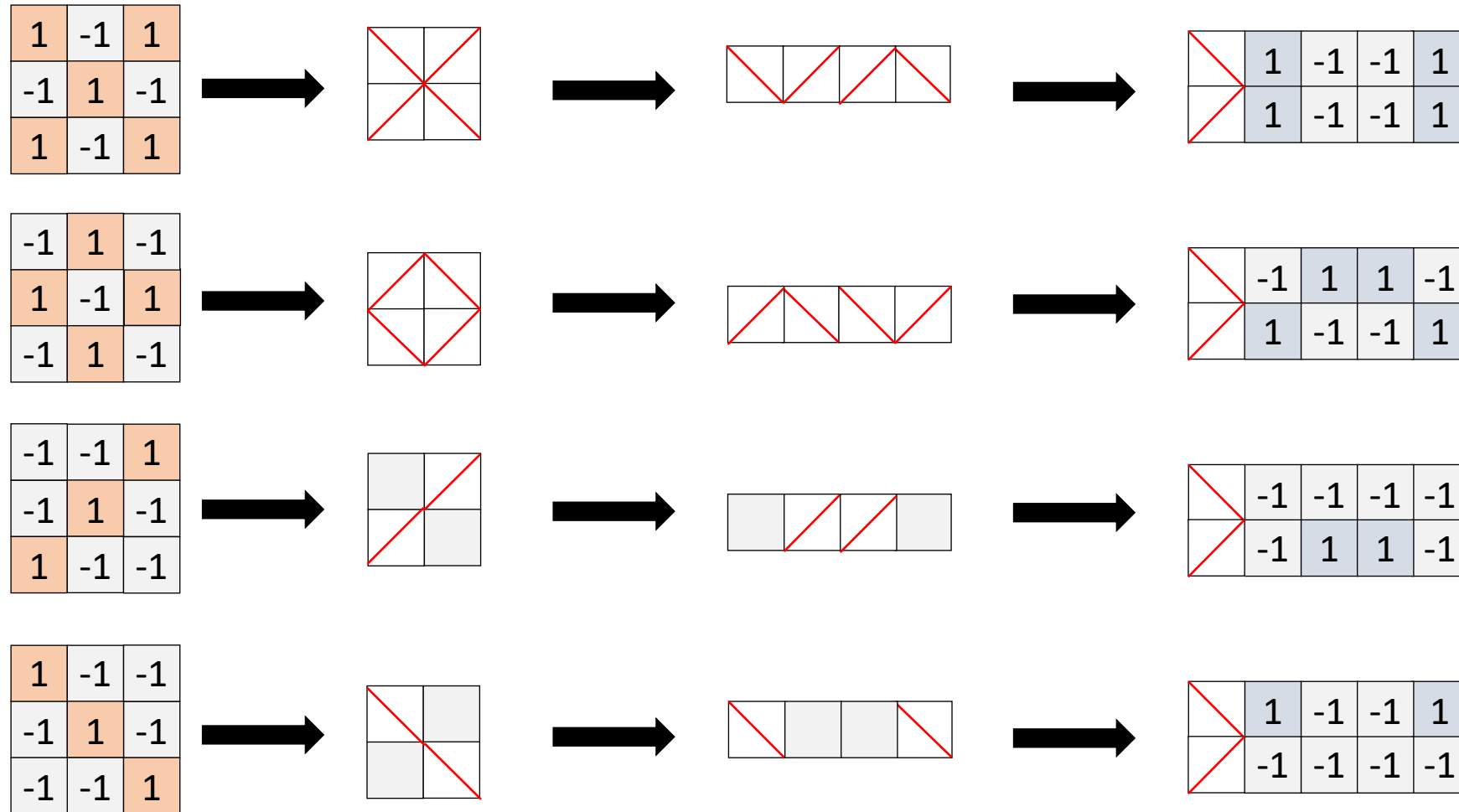
A friendly introduction to Convolutional Neural Networks and Image Recognition



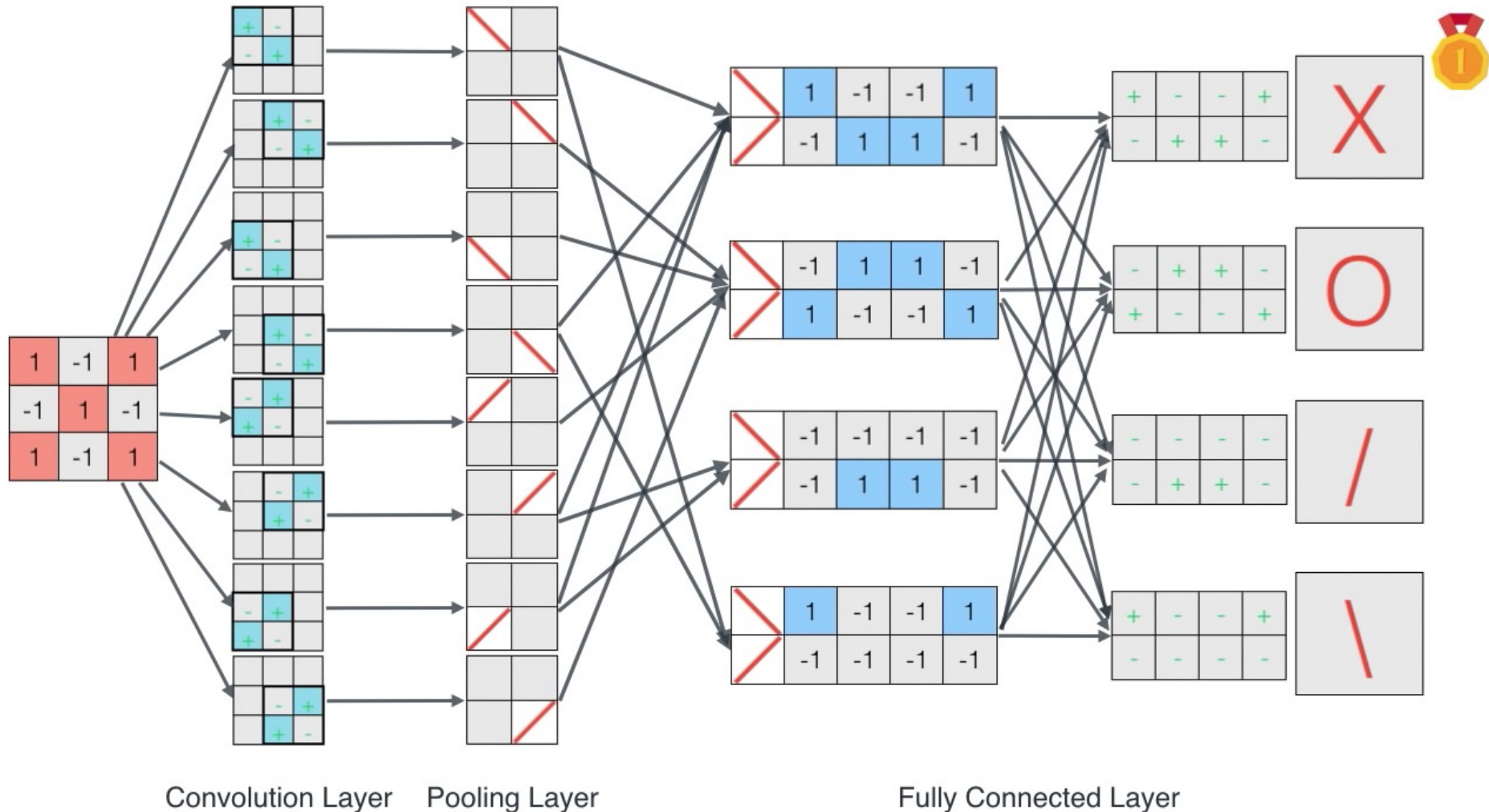
Convolution Layer

Pooling Layer

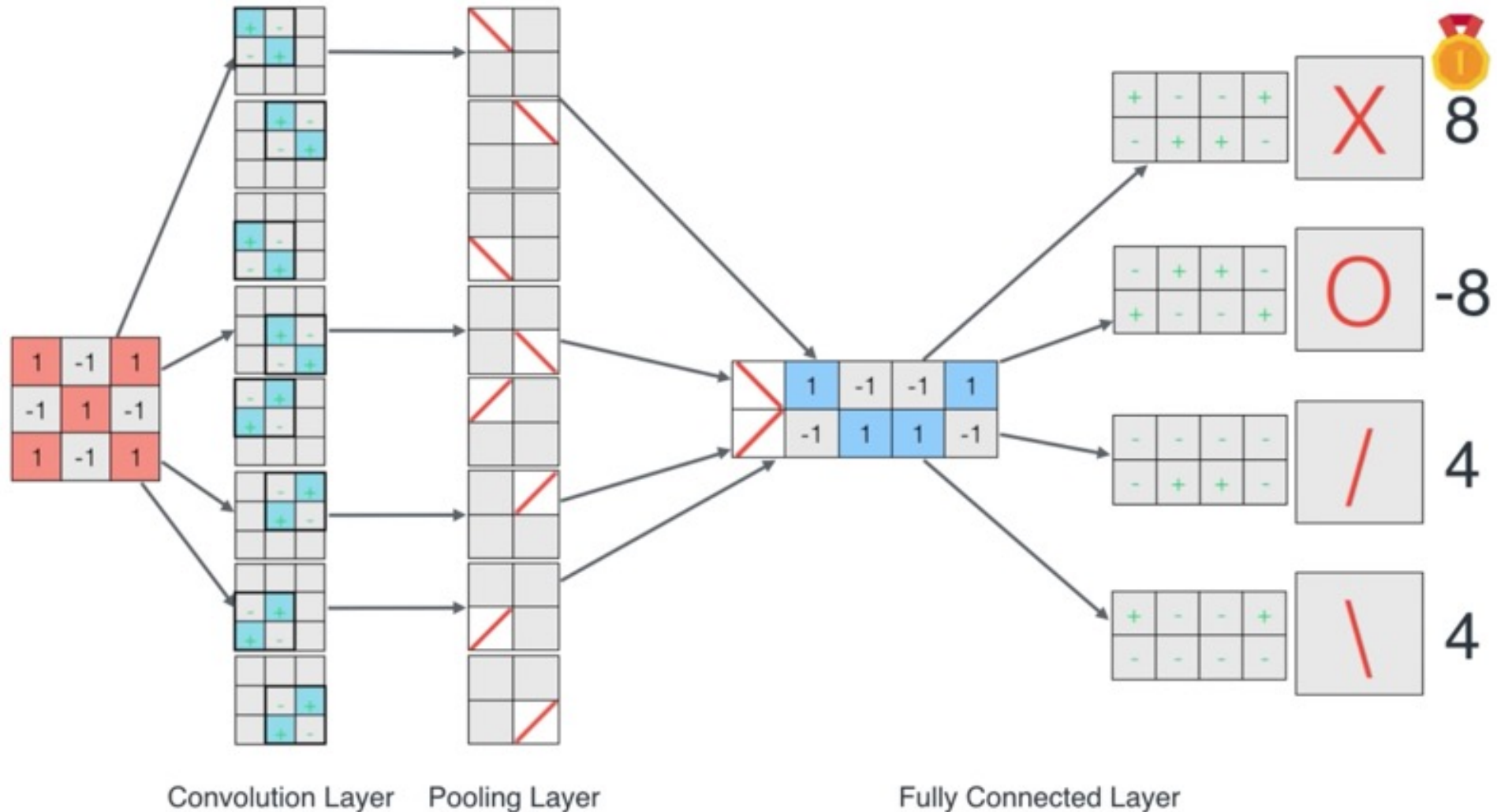
A friendly introduction to Convolutional Neural Networks and Image Recognition



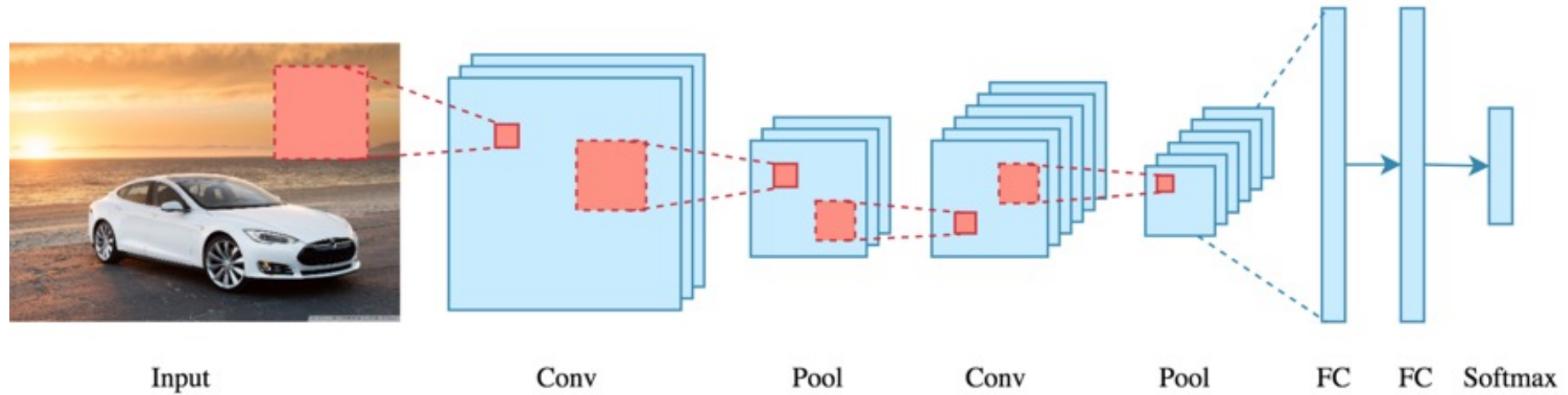
A friendly introduction to Convolutional Neural Networks and Image Recognition



A friendly introduction to Convolutional Neural Networks and Image Recognition



CNN Architecture



CNN Convolution Layer

Convolution is a mathematical operation to merge two sets of information

3x3 convolution

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

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

CNN Convolution Layer

Input x Filter --> Feature Map

receptive field: 3x3

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4		

Feature Map

CNN Convolution Layer

Input x Filter --> Feature Map

receptive field: 3x3

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4	3	

Feature Map

CNN Convolution Layer

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

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

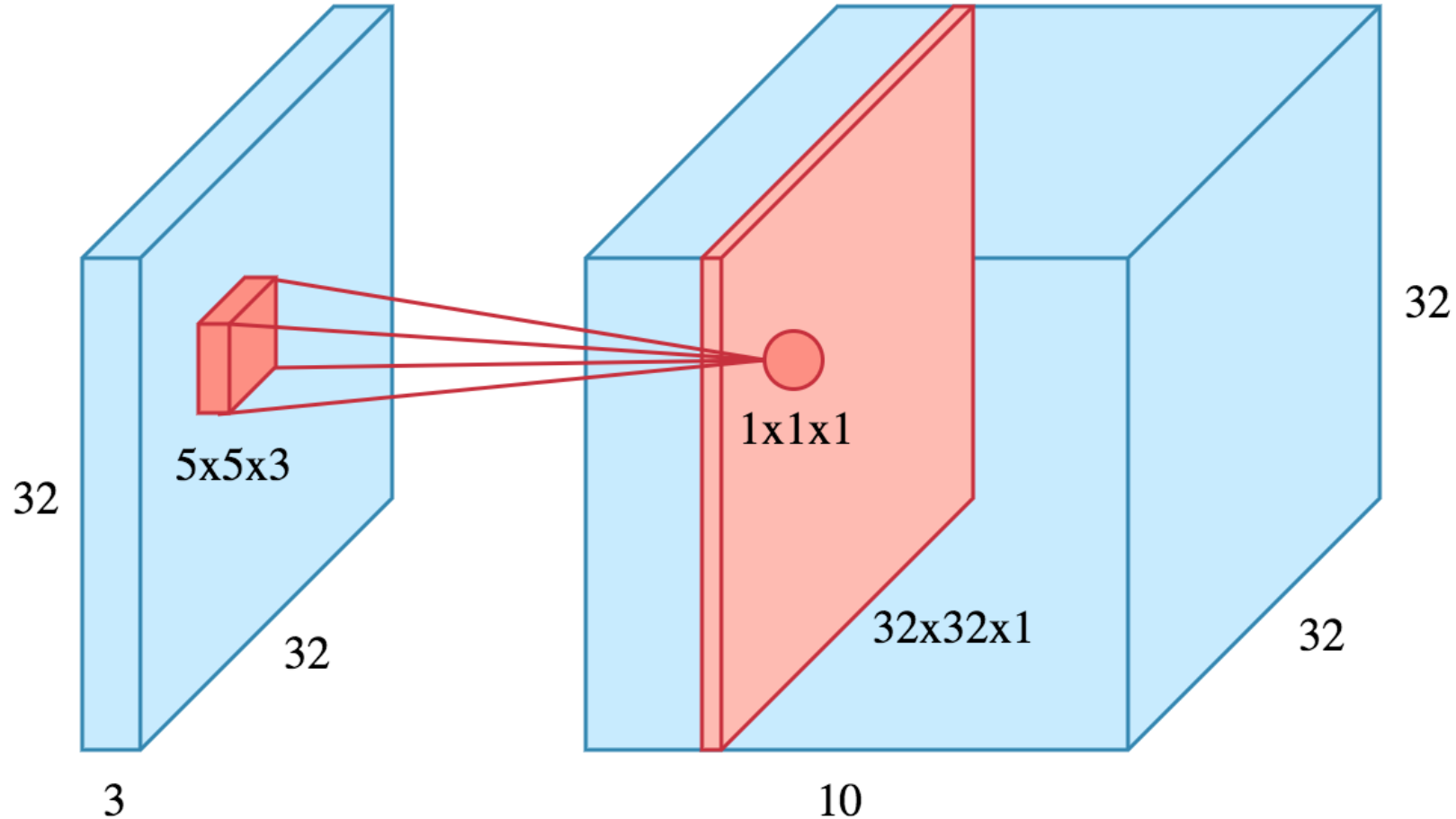
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Example convolution operation shown in 2D using a 3x3 filter

CNN Convolution Layer

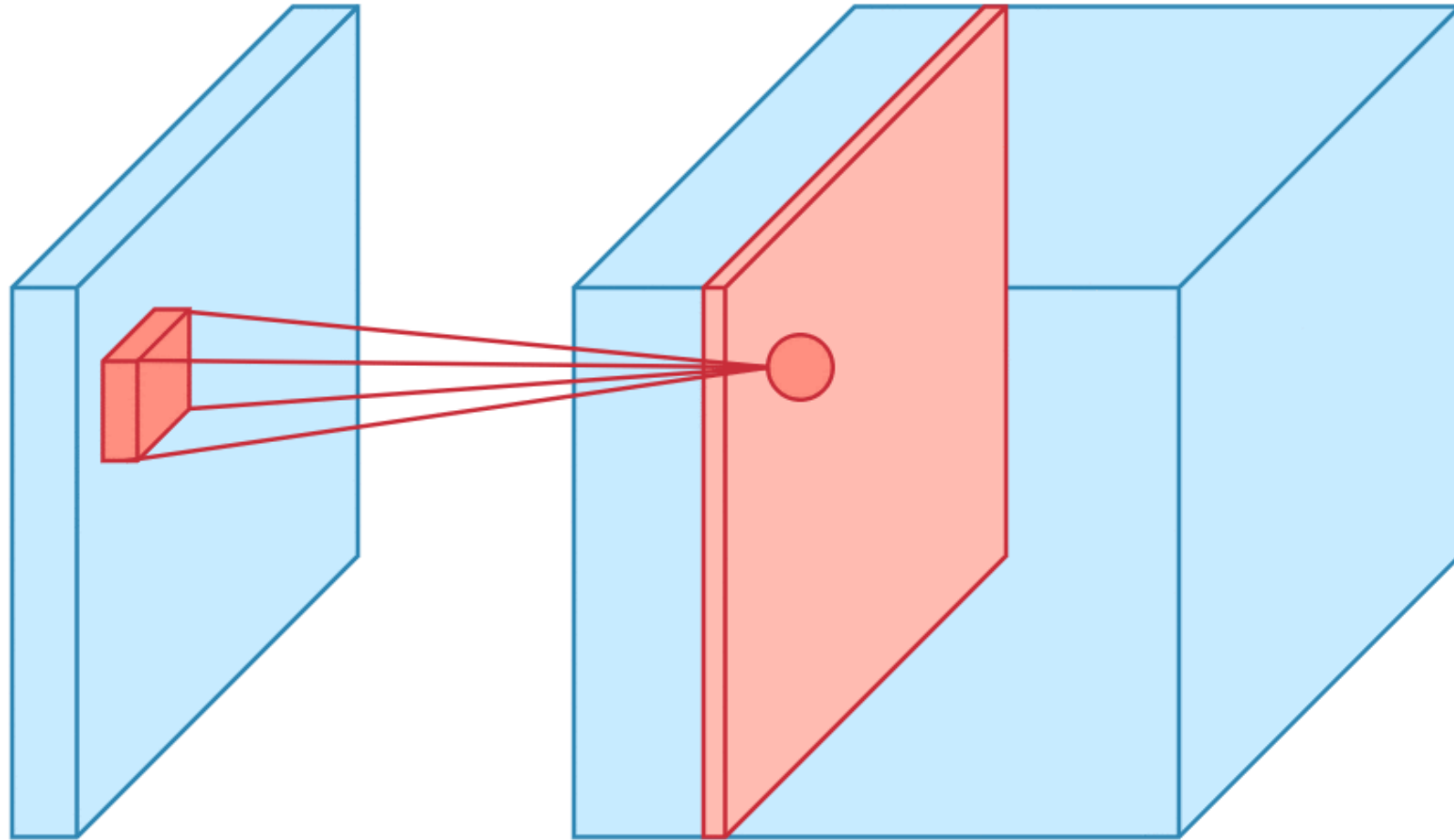
10 different filters 10 feature maps of size 32x32x1



final output of the convolution layer:
a volume of size 32x32x10

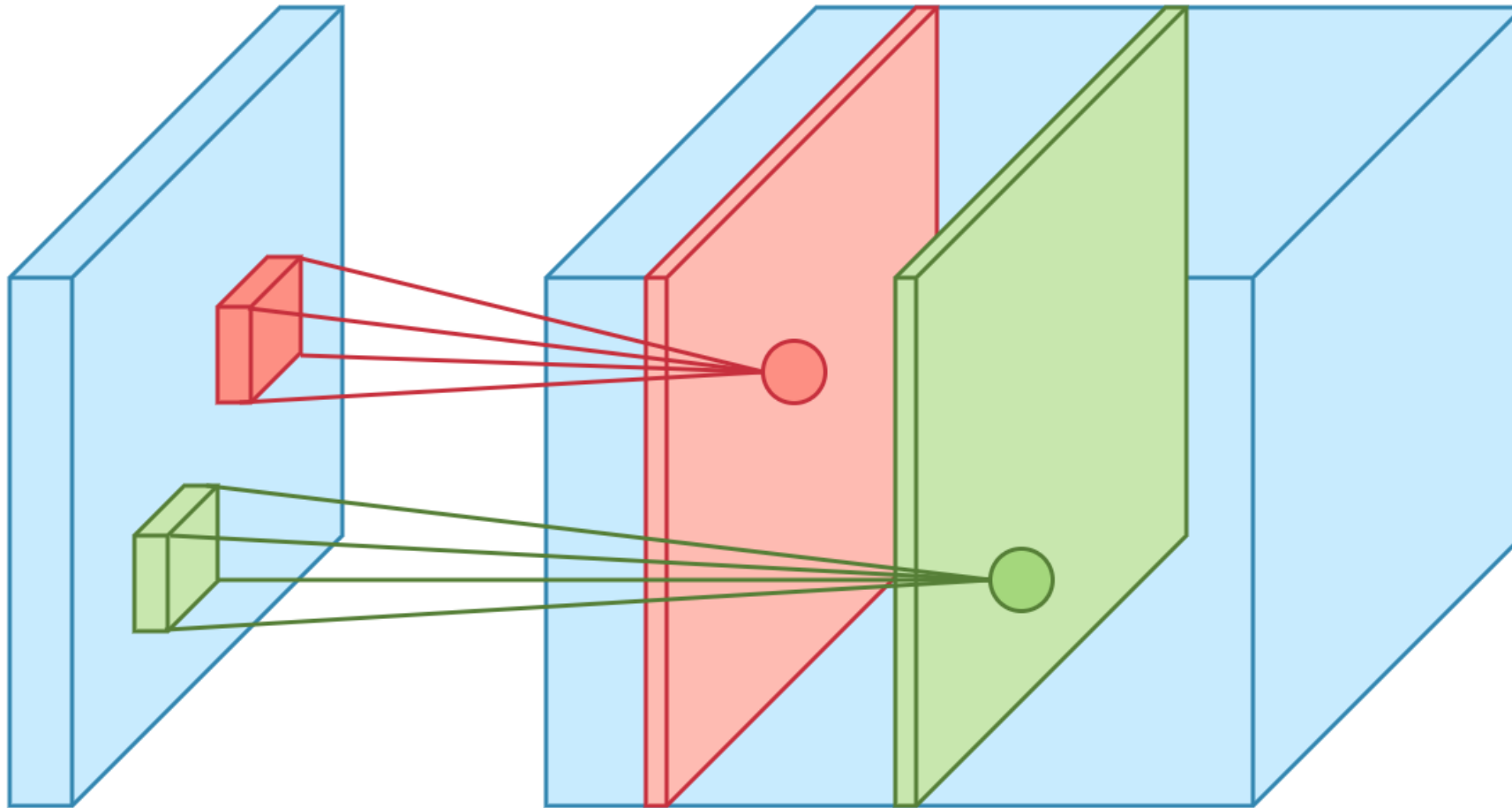
CNN Convolution Layer

Sliding operation at 4 locations



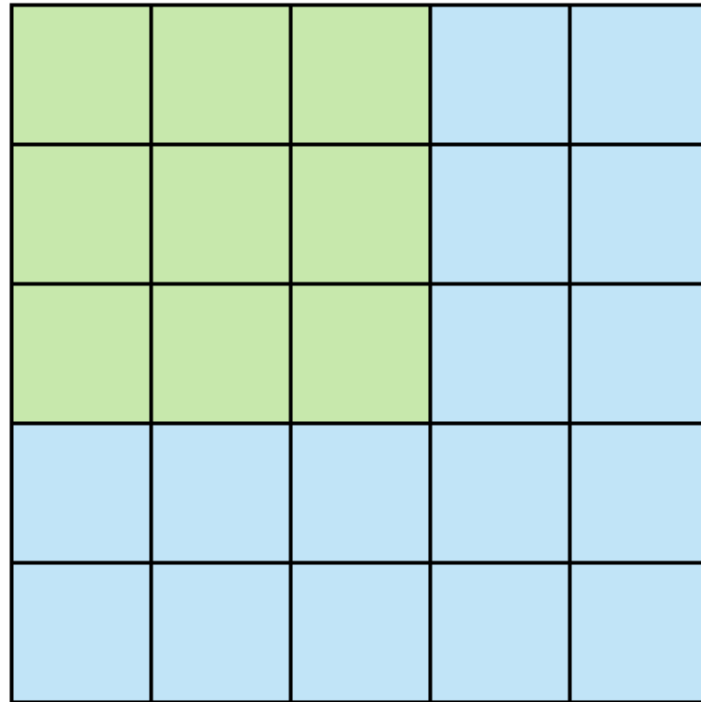
CNN Convolution Layer

two feature maps

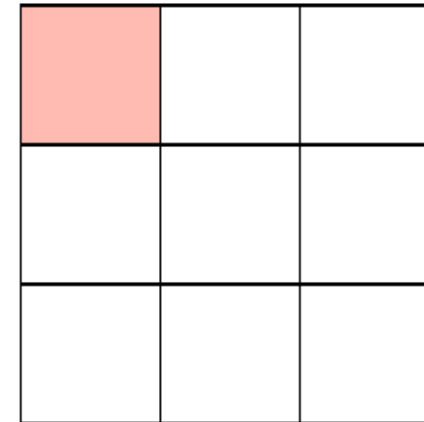


CNN Convolution Layer

Stride specifies how much we move the convolution filter at each step



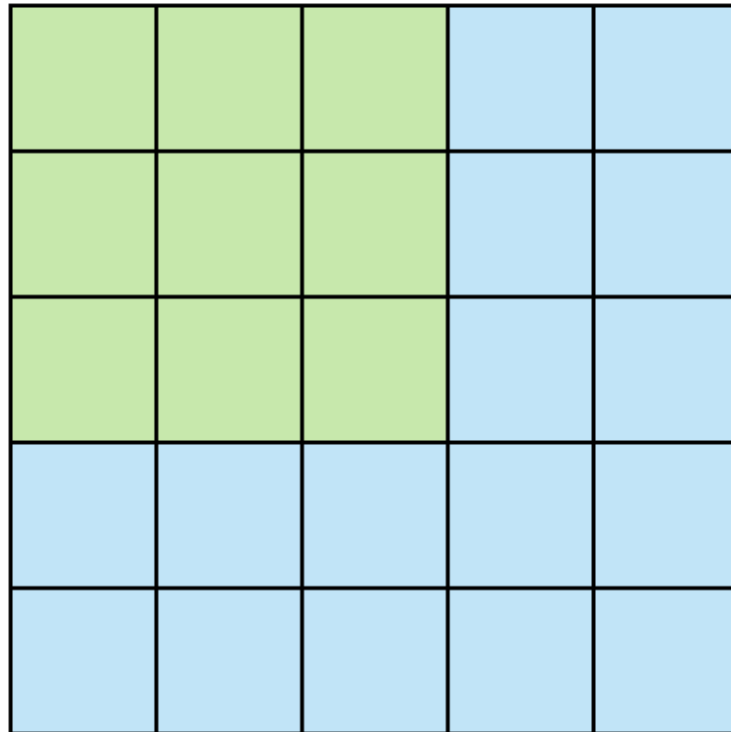
Stride 1



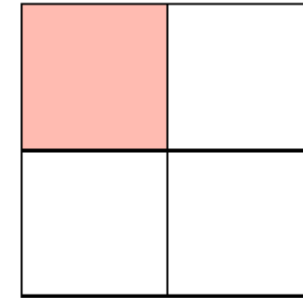
Feature Map

CNN Convolution Layer

Stride specifies how much we move the convolution filter at each step



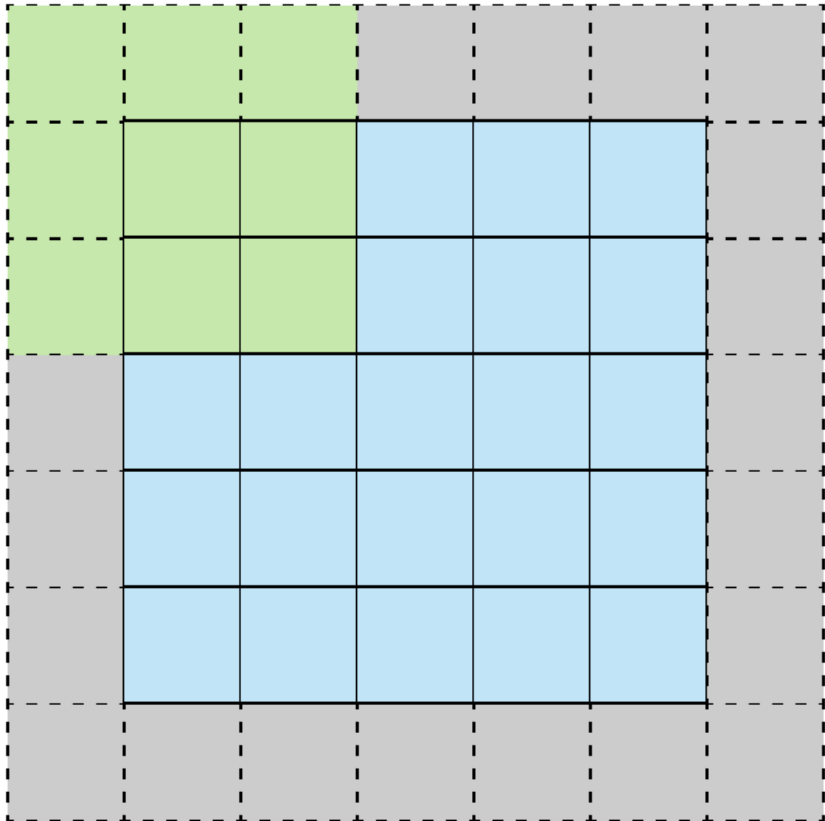
Stride 2



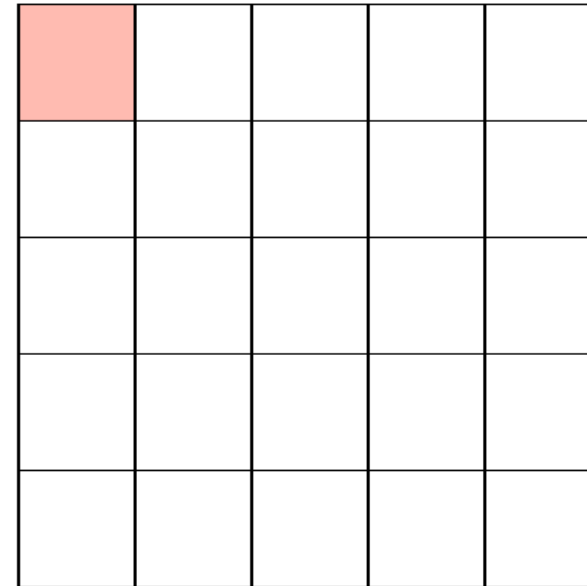
Feature Map

CNN Convolution Layer

Stride 1 with Padding



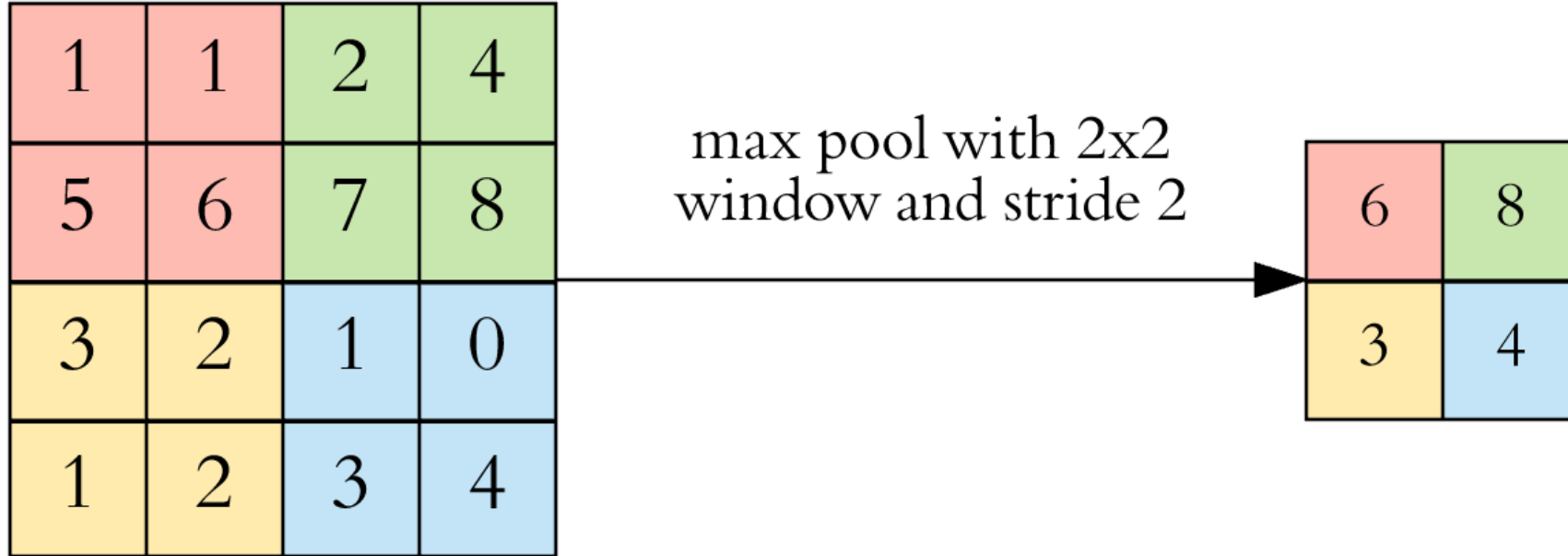
Stride 1 with Padding



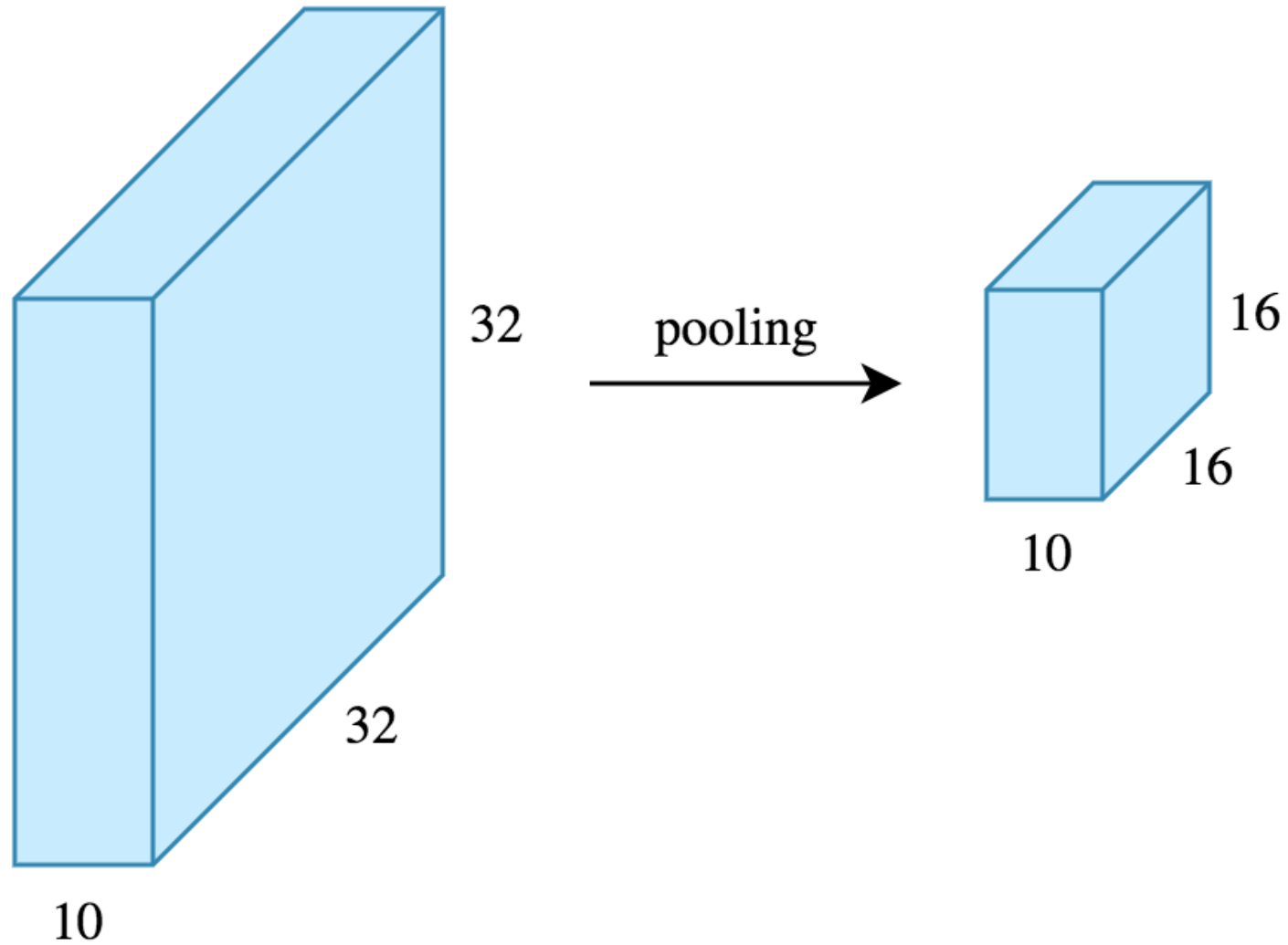
Feature Map

CNN Pooling Layer

Max Pooling

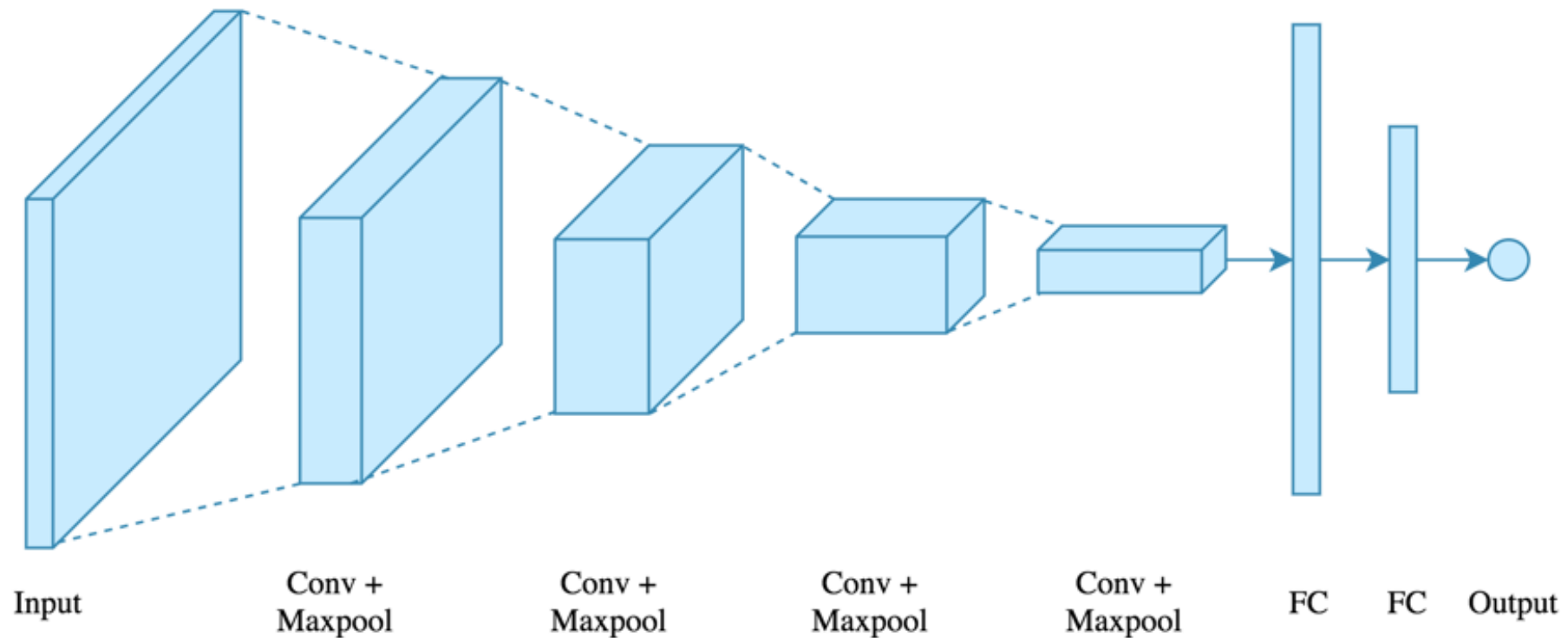


CNN Pooling Layer



CNN Architecture

4 convolution + pooling layers, followed by 2 fully connected layers



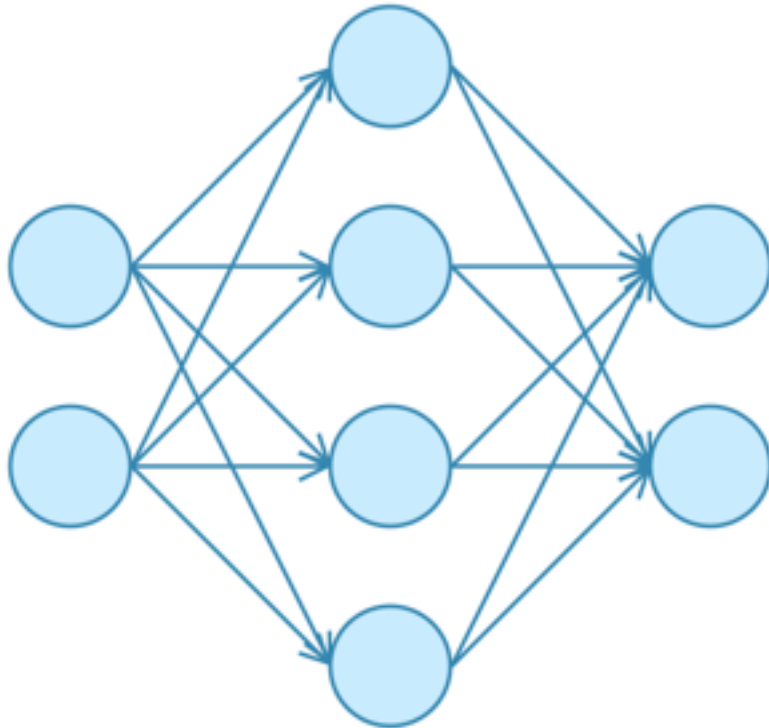
CNN Architecture

4 convolution + pooling layers, followed by 2 fully connected layers

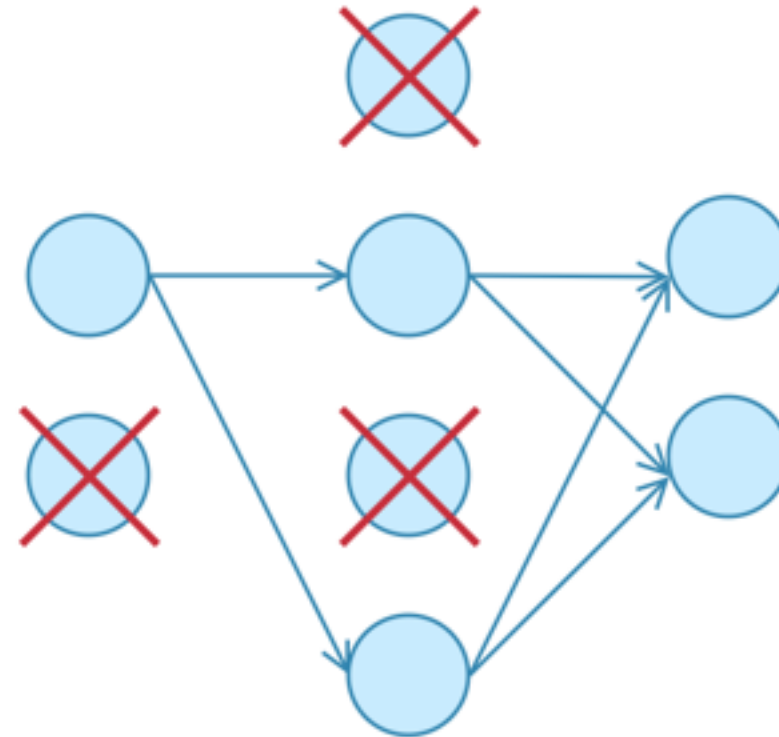
<https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3>

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
```

Dropout

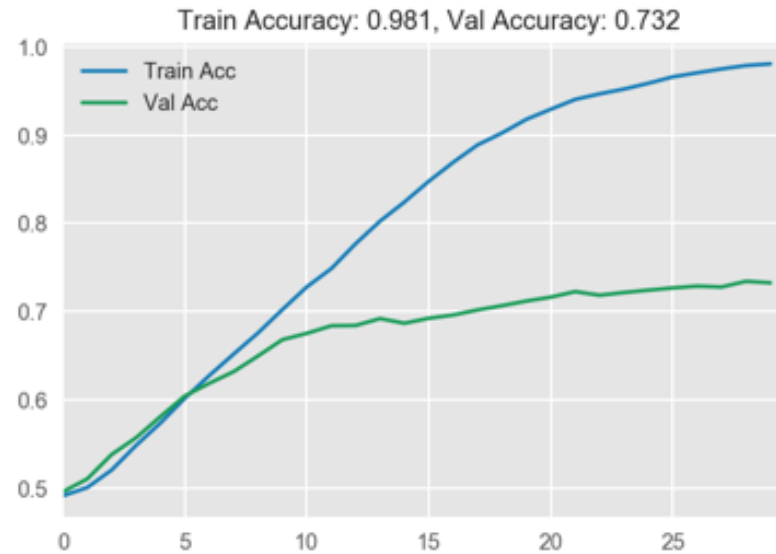
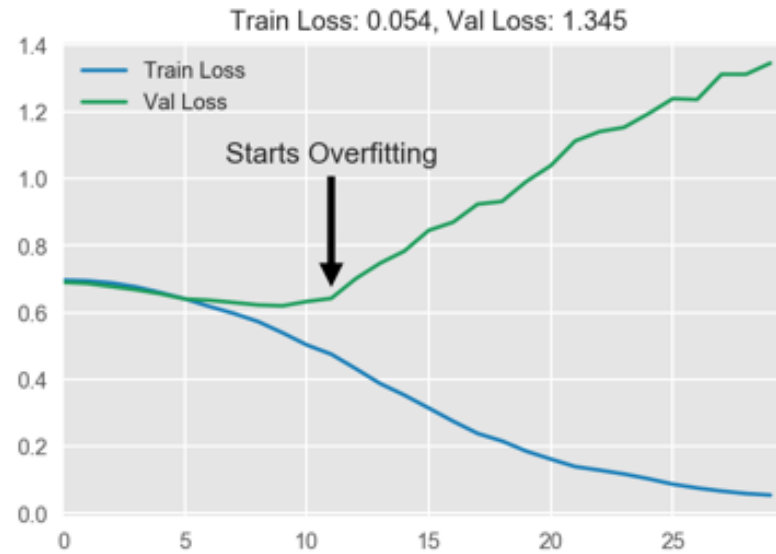


No Dropout



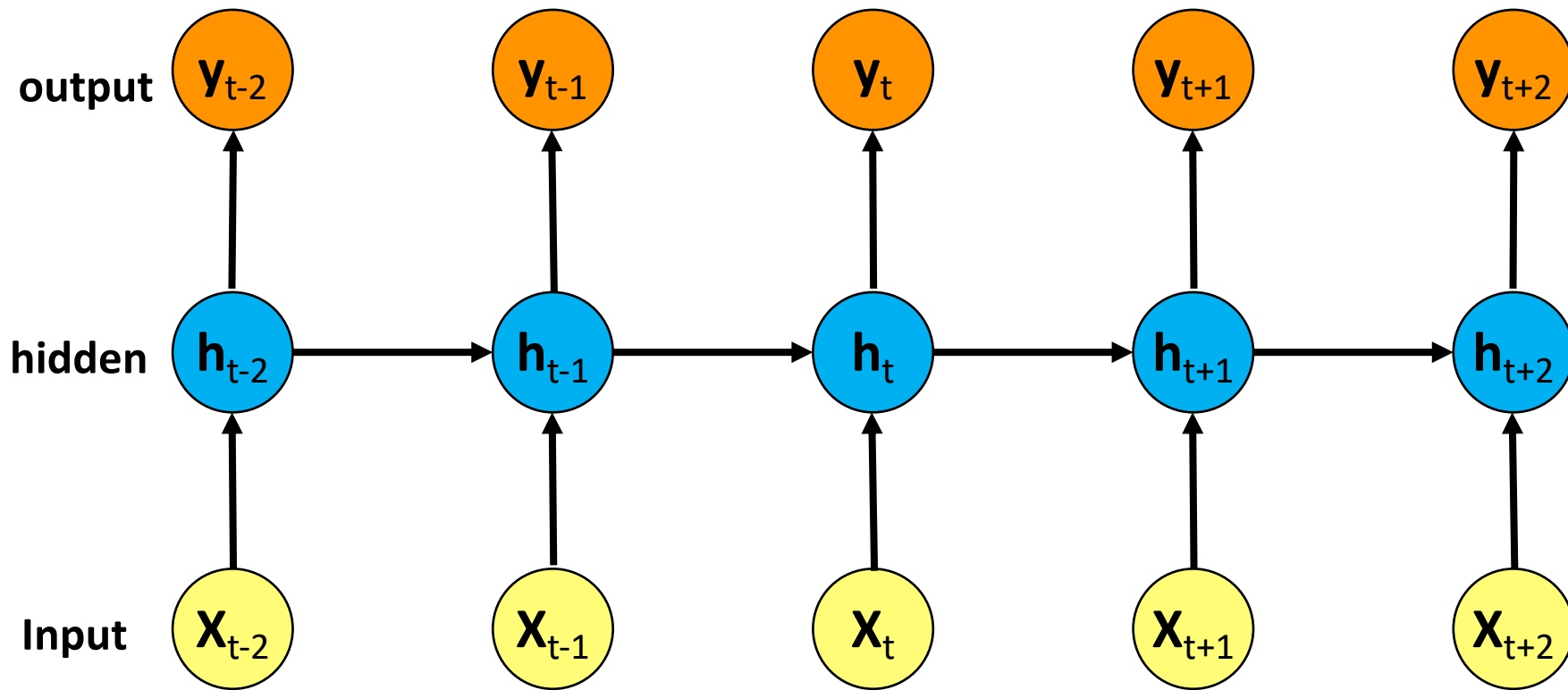
With Dropout

Model Performance



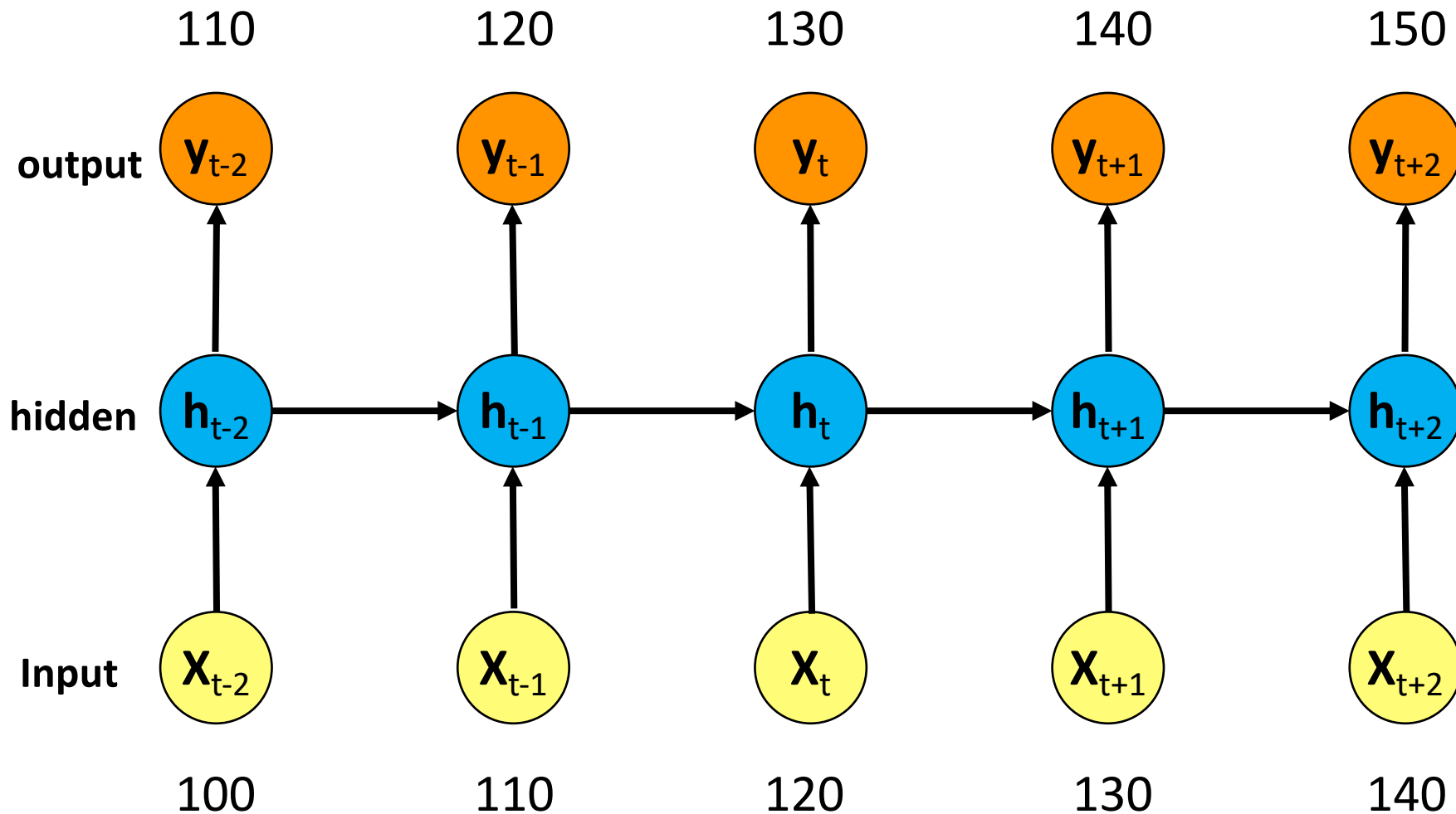
Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN)

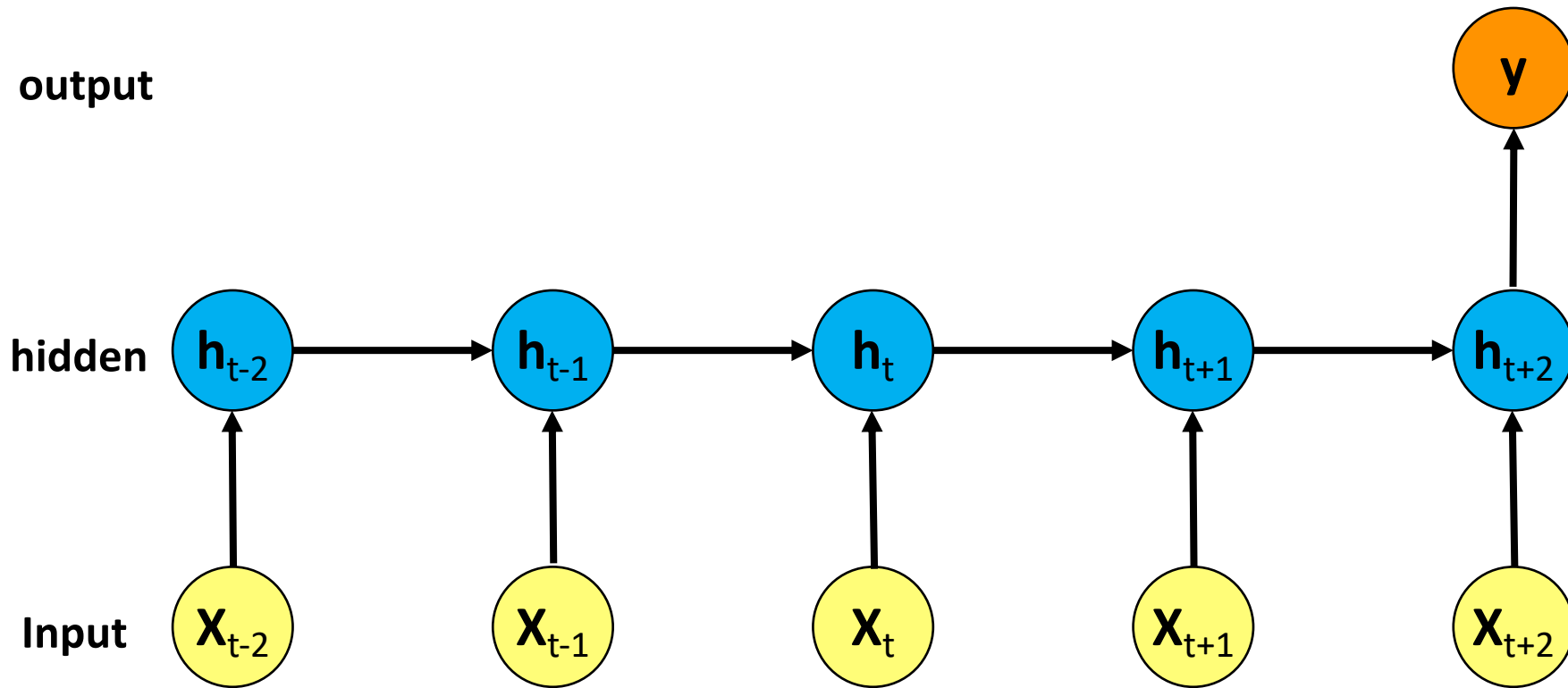


Recurrent Neural Networks (RNN)

Time Series Forecasting

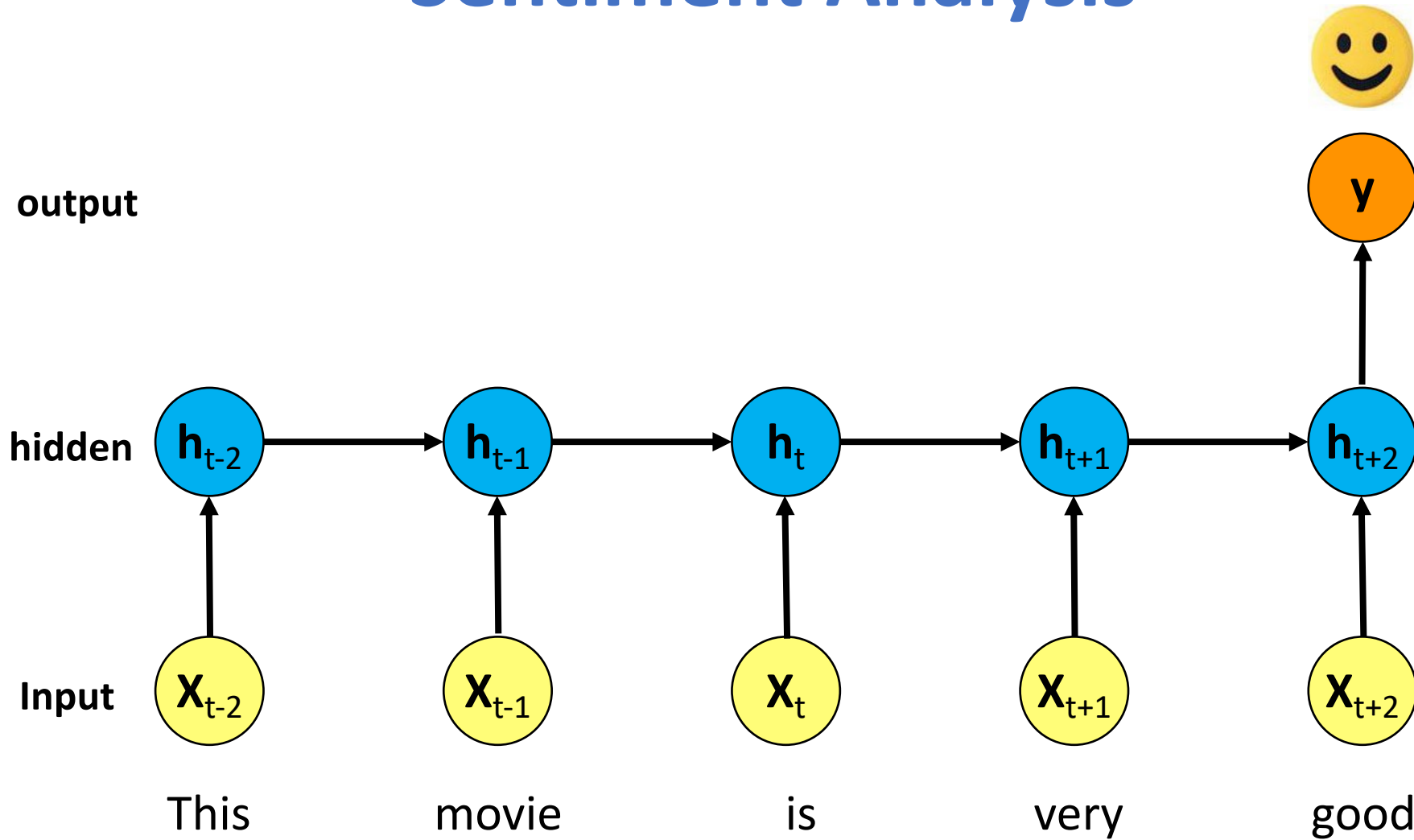


Recurrent Neural Networks (RNN)



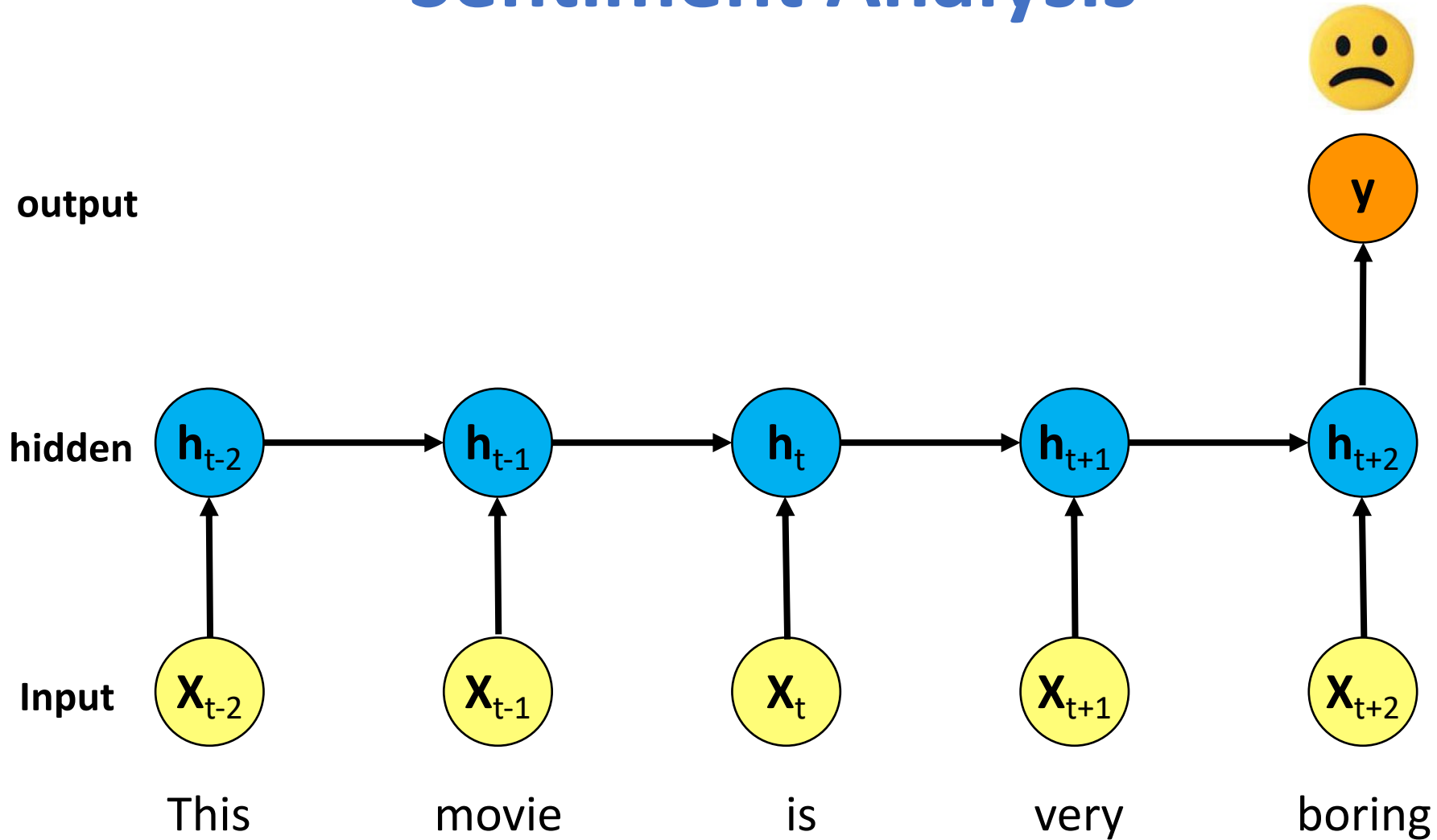
Recurrent Neural Networks (RNN)

Sentiment Analysis

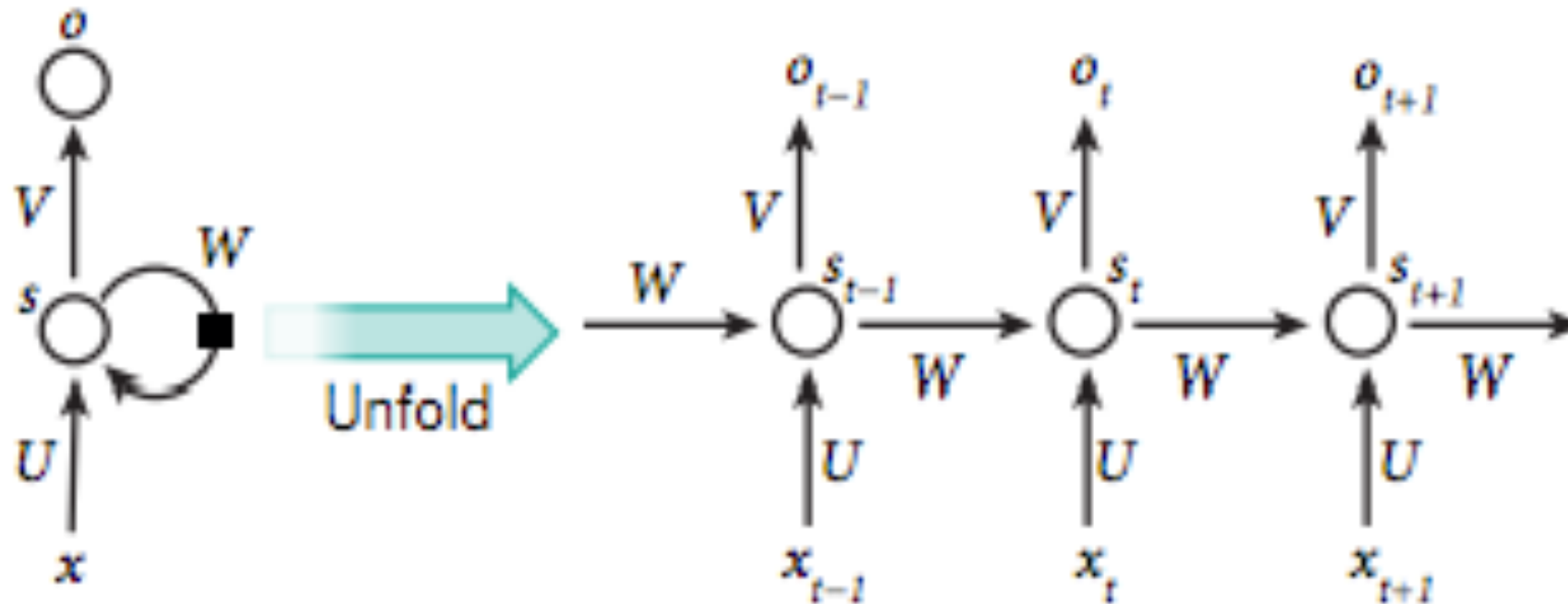


Recurrent Neural Networks (RNN)

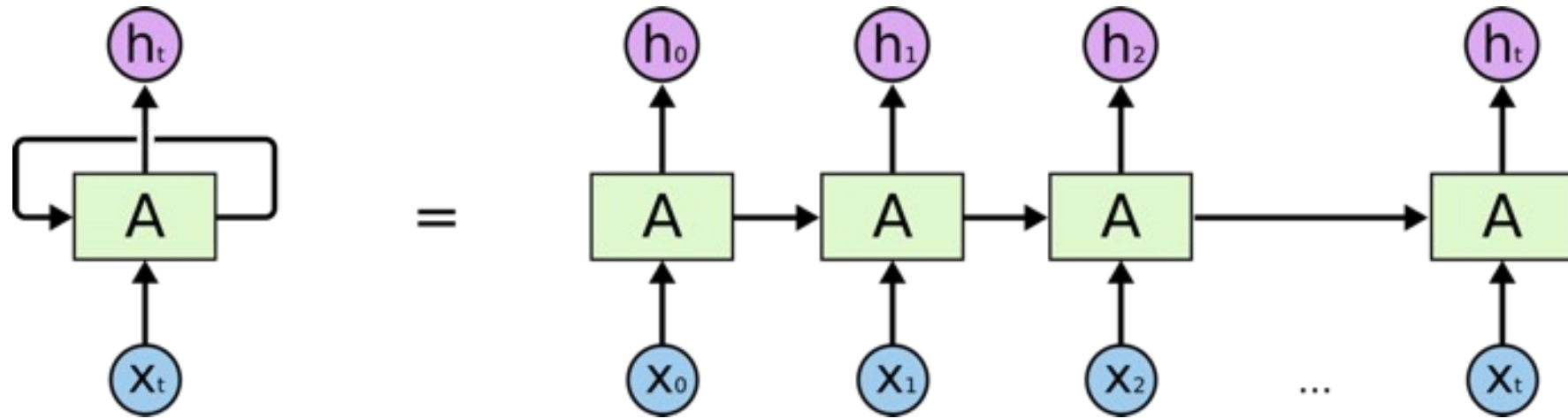
Sentiment Analysis



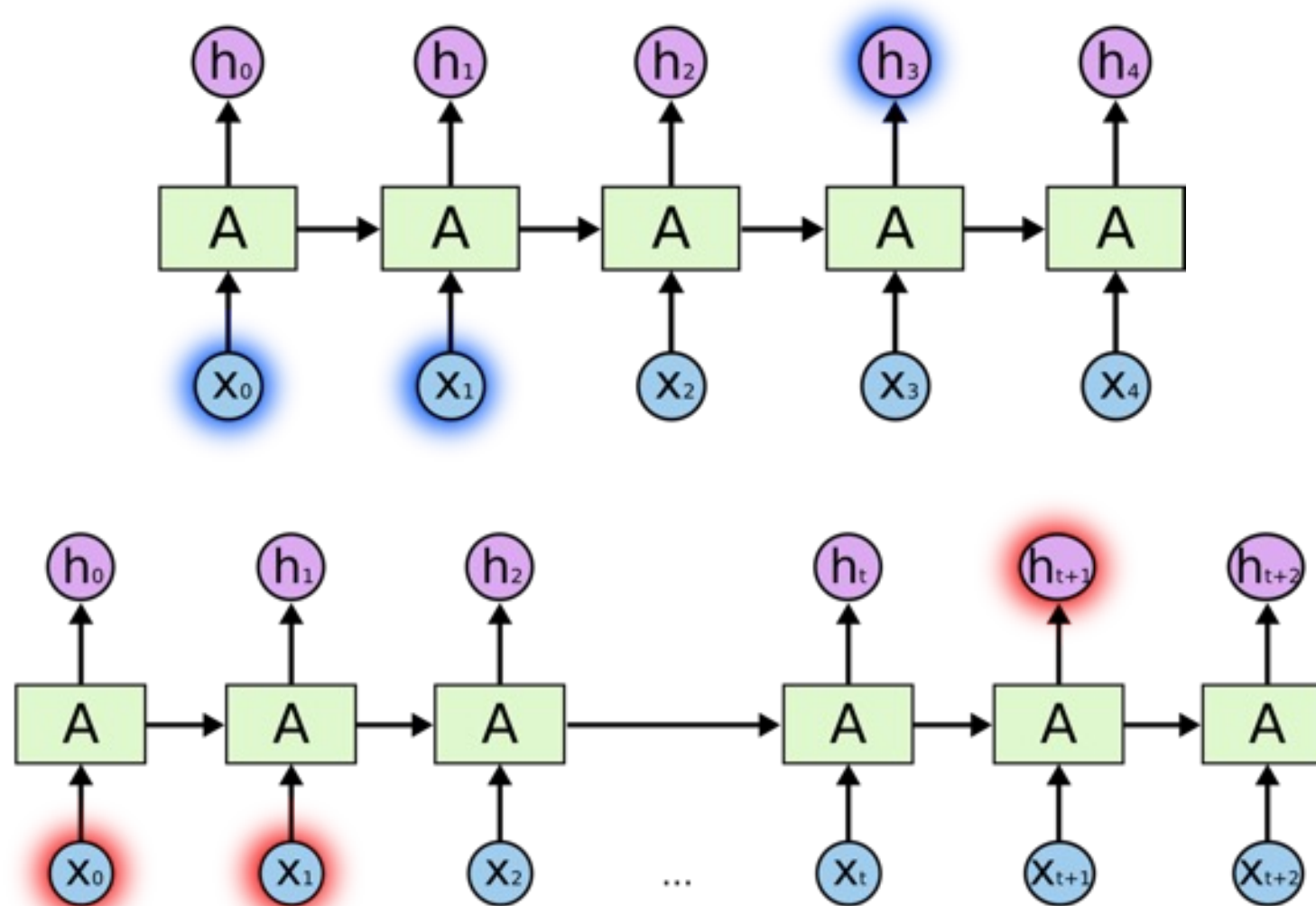
Recurrent Neural Network (RNN)



RNN



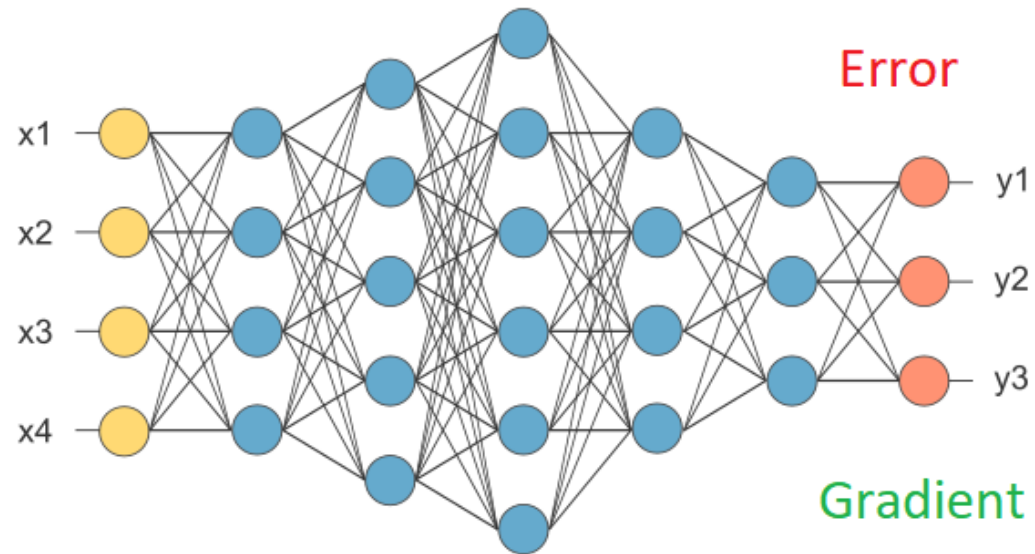
RNN long-term dependencies



I grew up in France... I speak fluent French.

Vanishing Gradient

Exploding Gradient

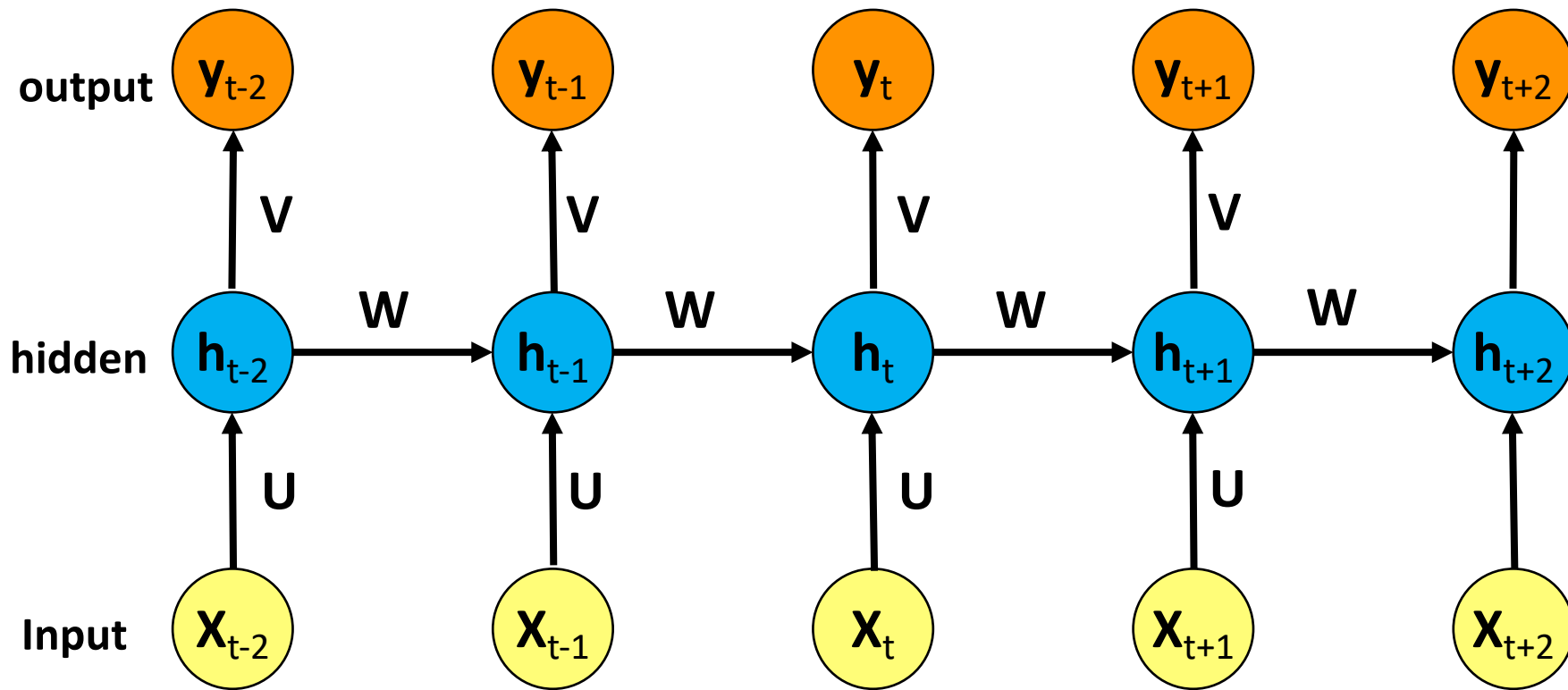


Vanishing Gradient



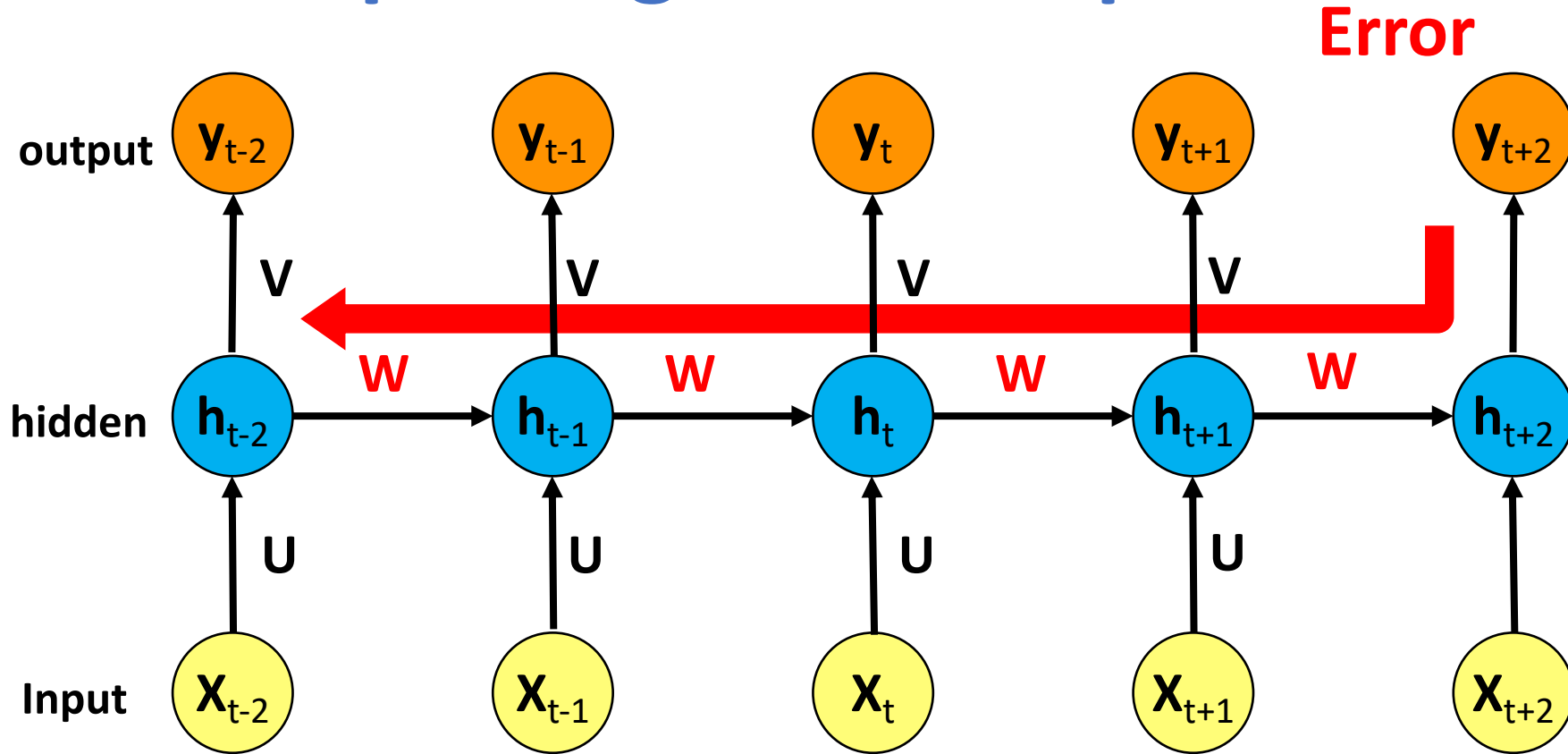
Exploding Gradient

Recurrent Neural Networks (RNN)



RNN

Vanishing Gradient problem Exploding Gradient problem

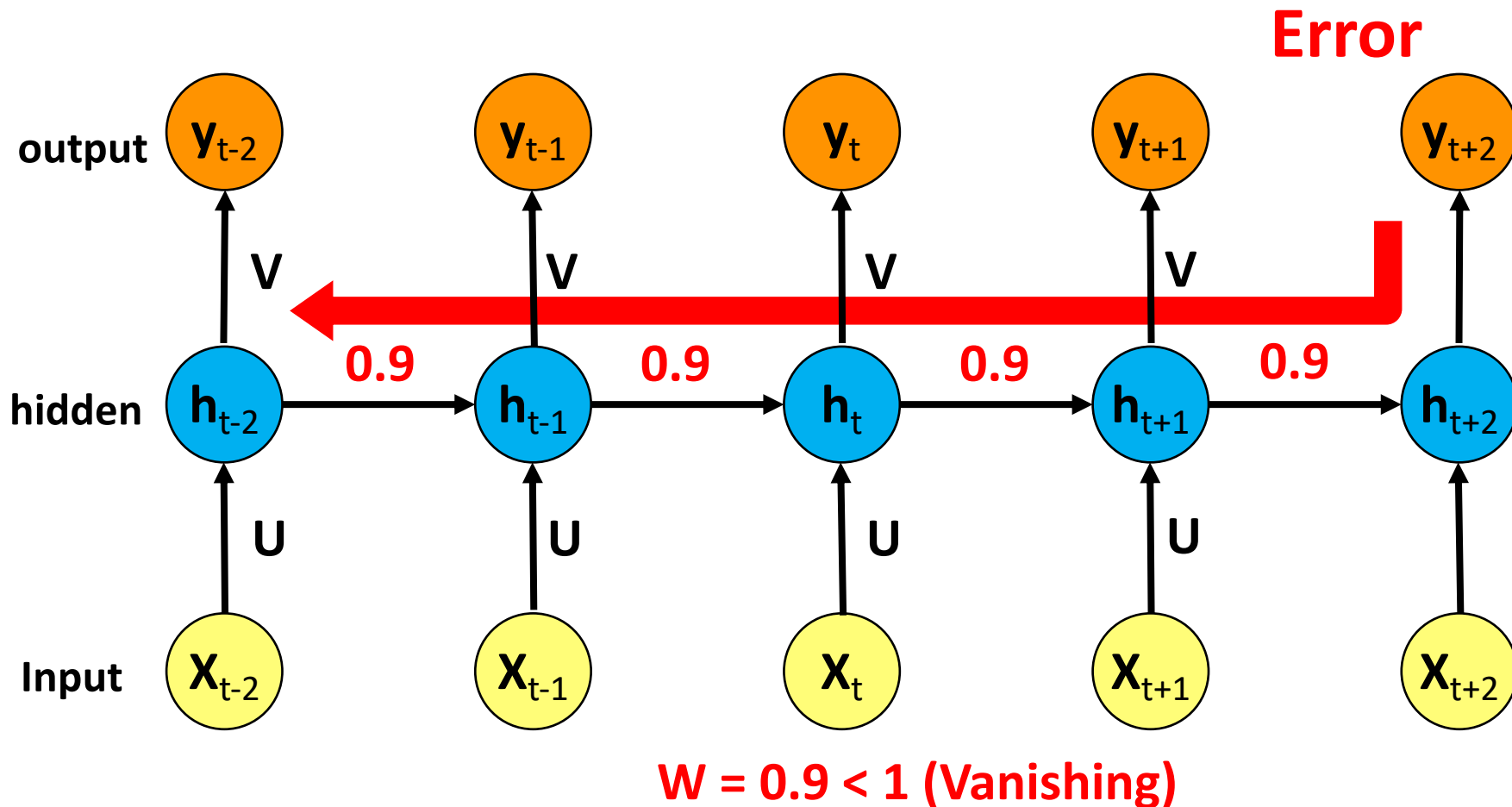


if $|W| < 1$ (Vanishing)

if $|W| > 1$ (Exploding)

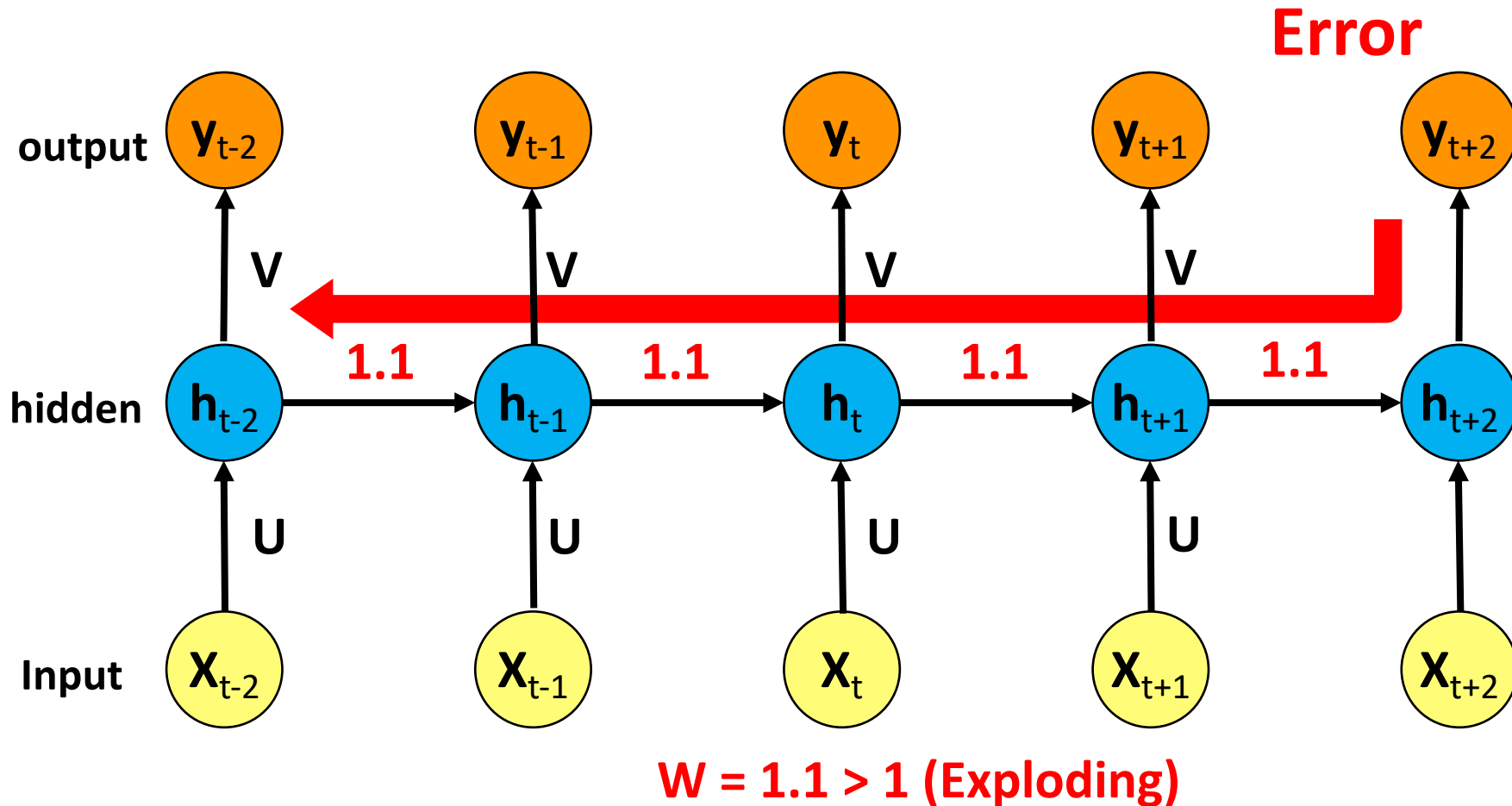
RNN

Vanishing Gradient problem

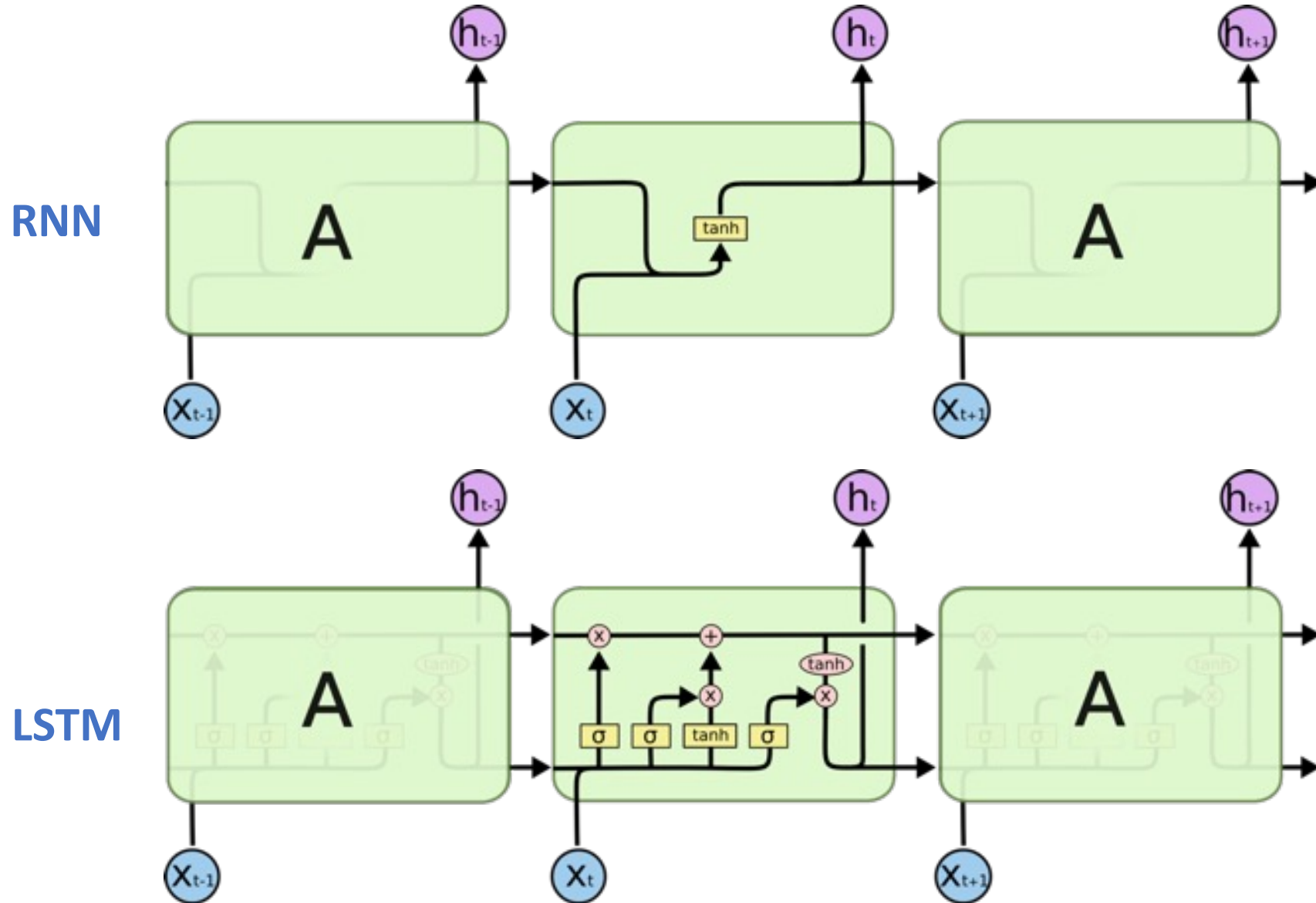


RNN

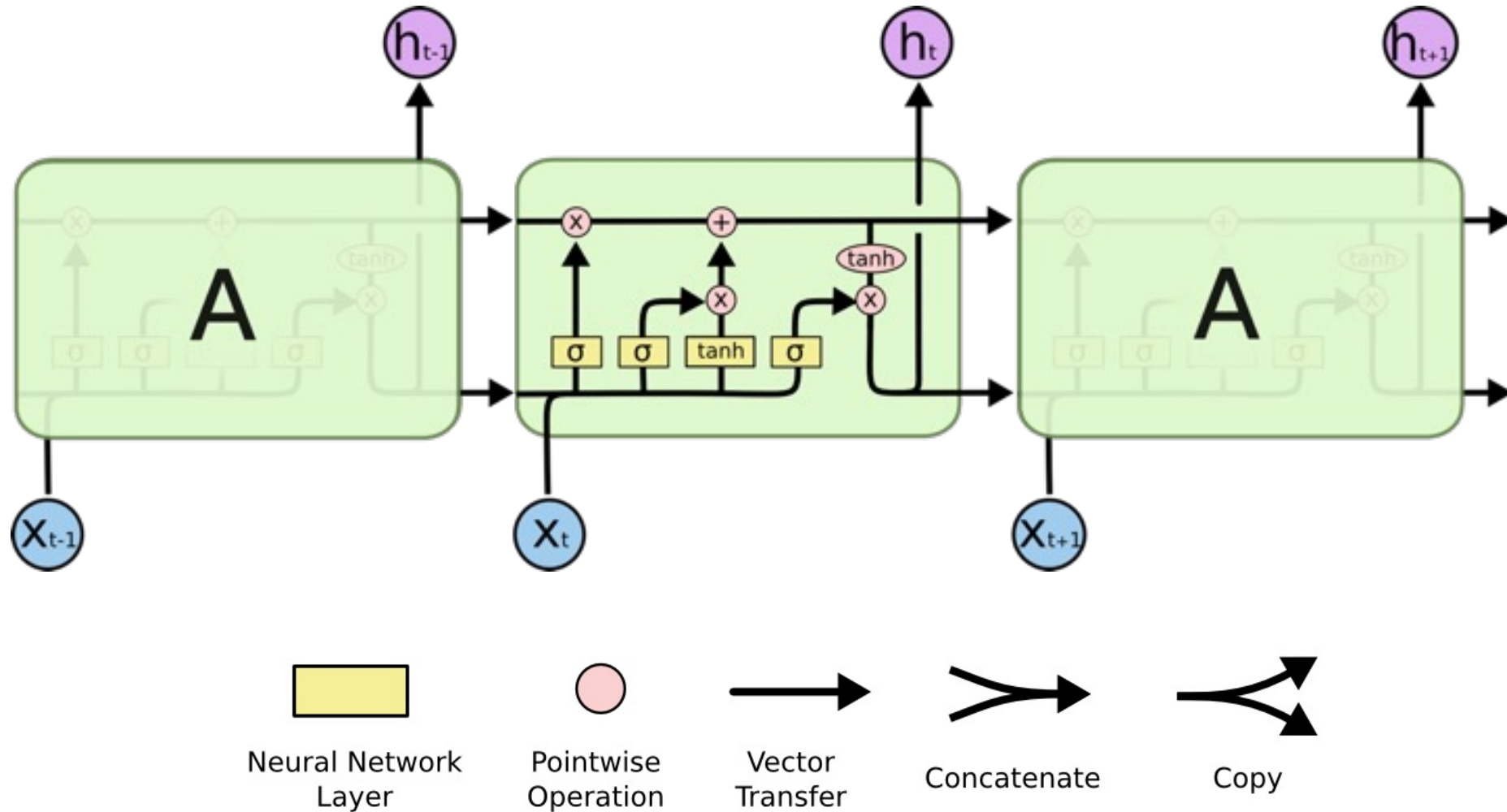
Exploding Gradient problem



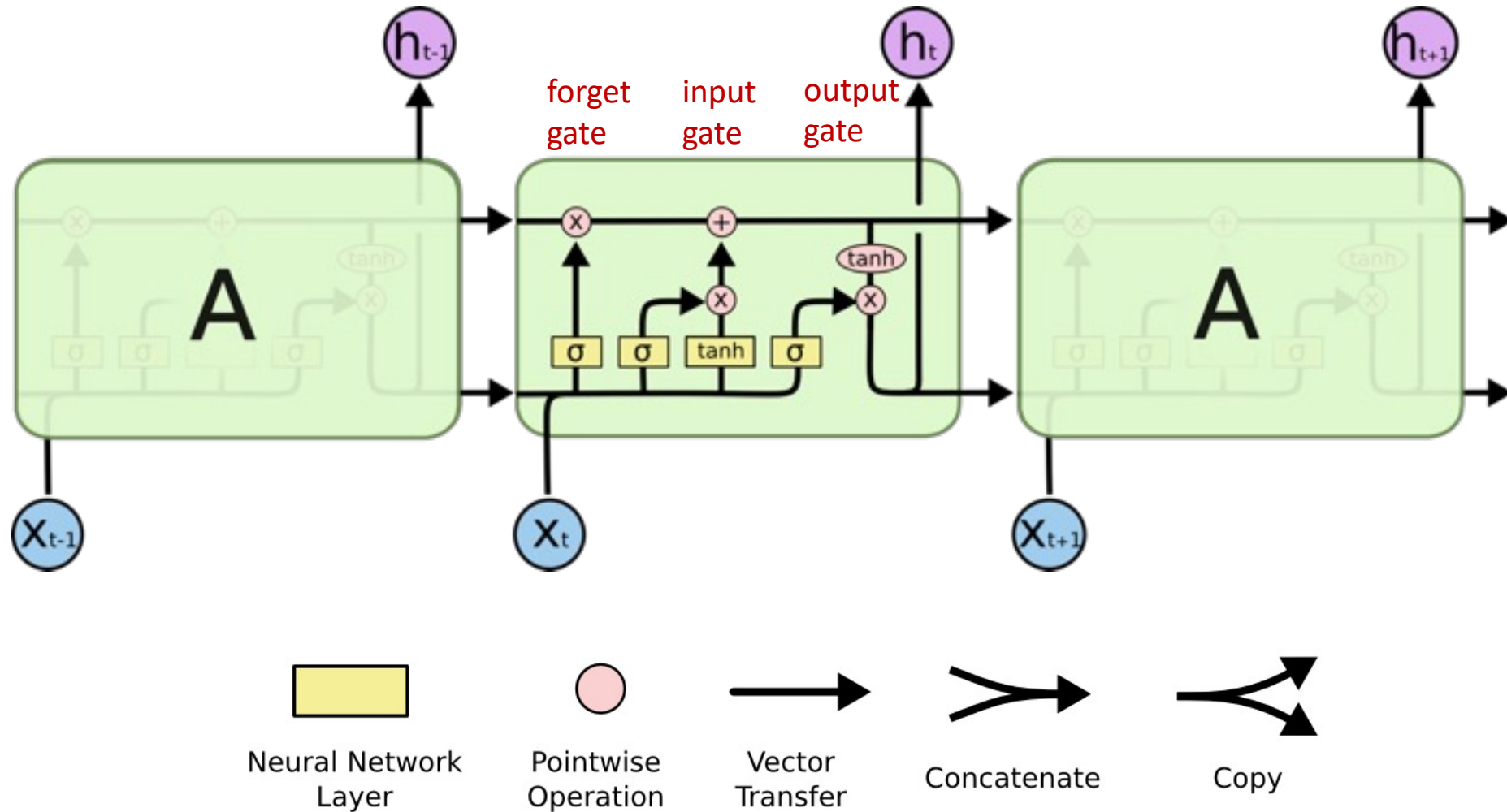
RNN LSTM



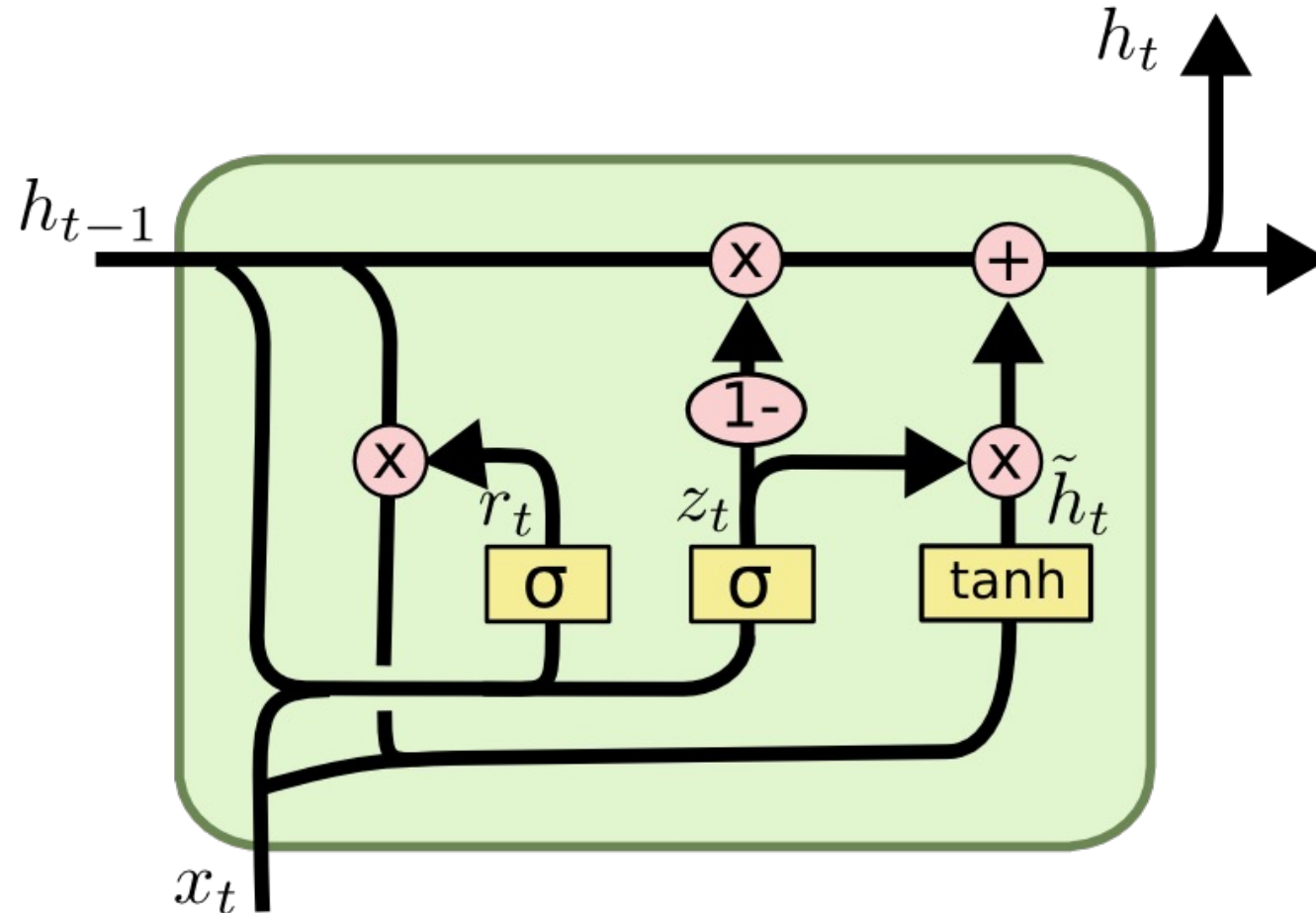
Long Short Term Memory (LSTM)



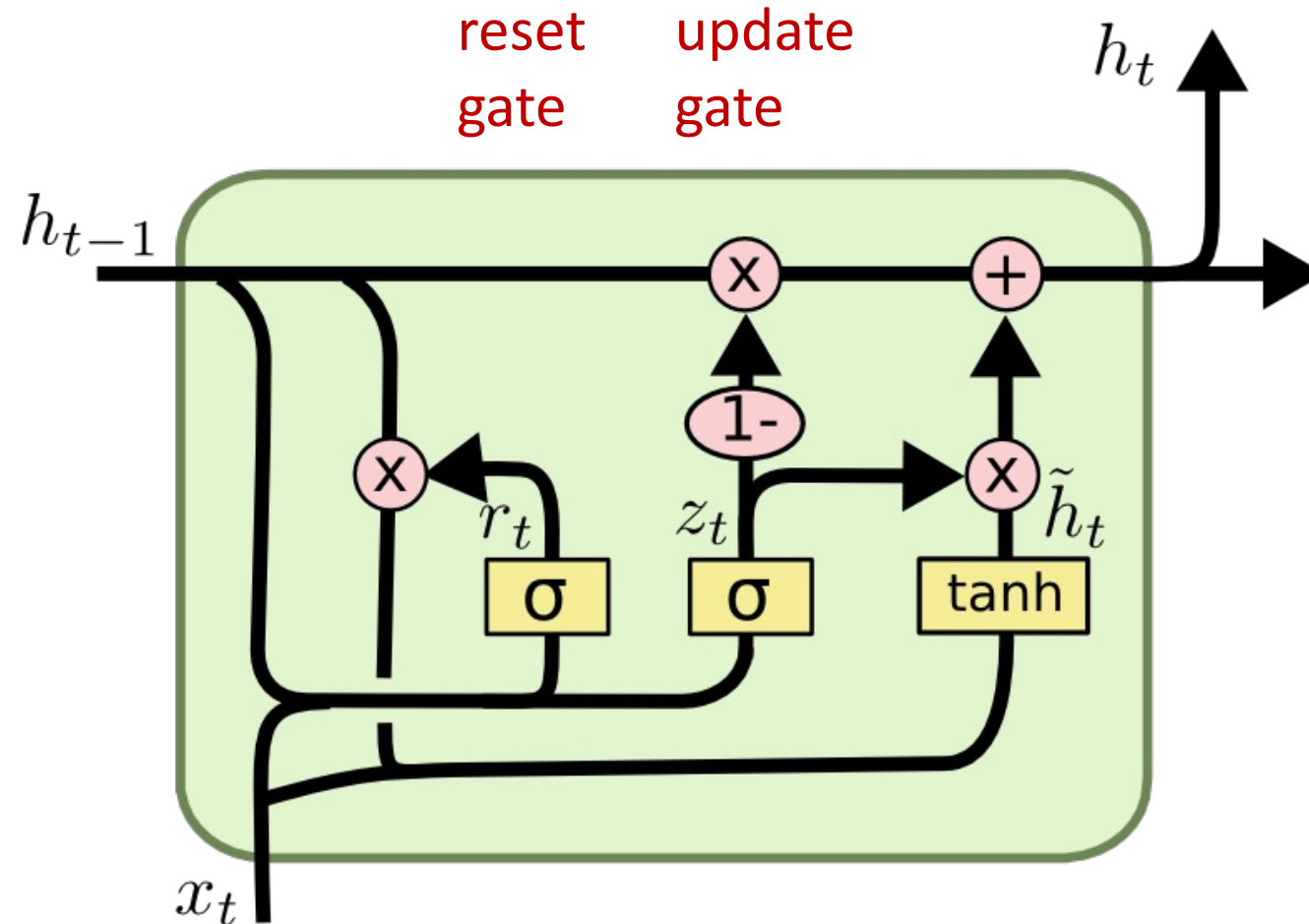
Long Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)

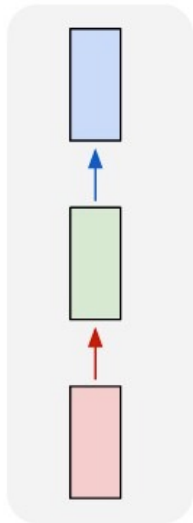


Gated Recurrent Unit (GRU)



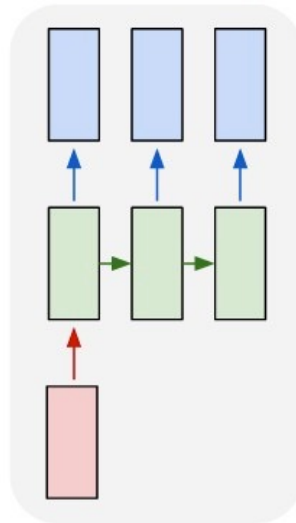
LSTM Recurrent Neural Network

one to one



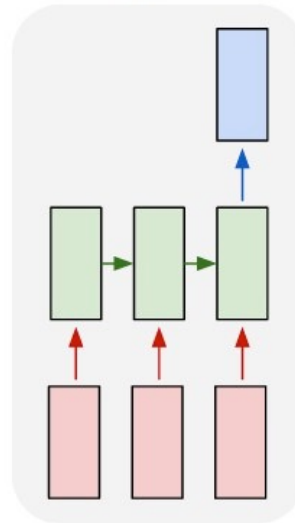
**Traditional
Neural
Network**

one to many



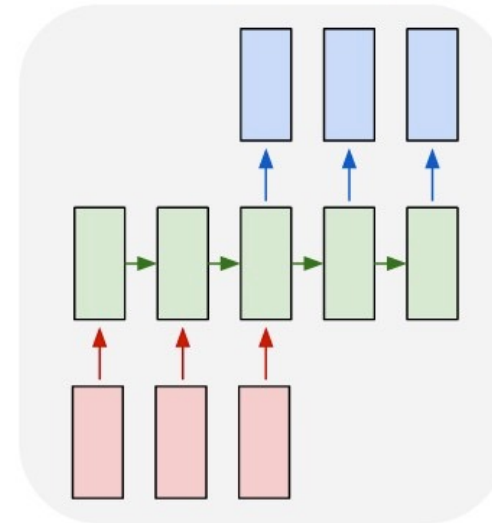
**Music
Generation**

many to one



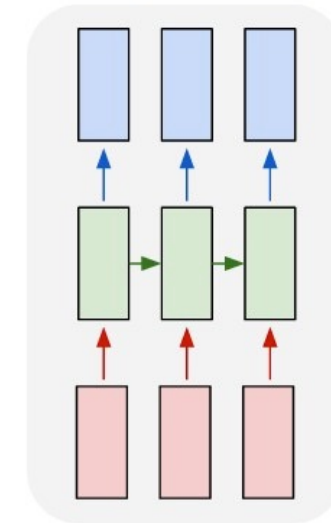
**Sentiment
Classification**

many to many



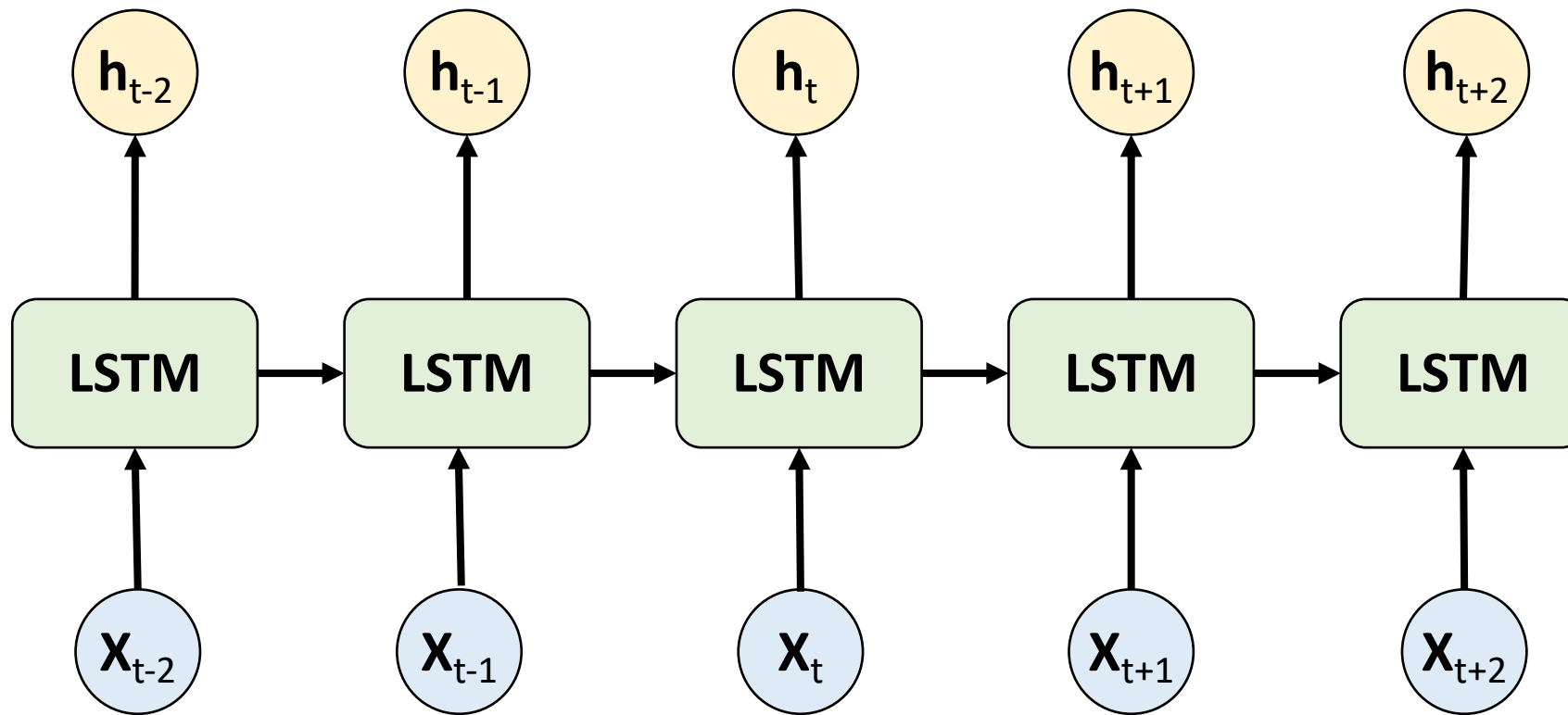
**Name
Entity
Recognition**

many to many



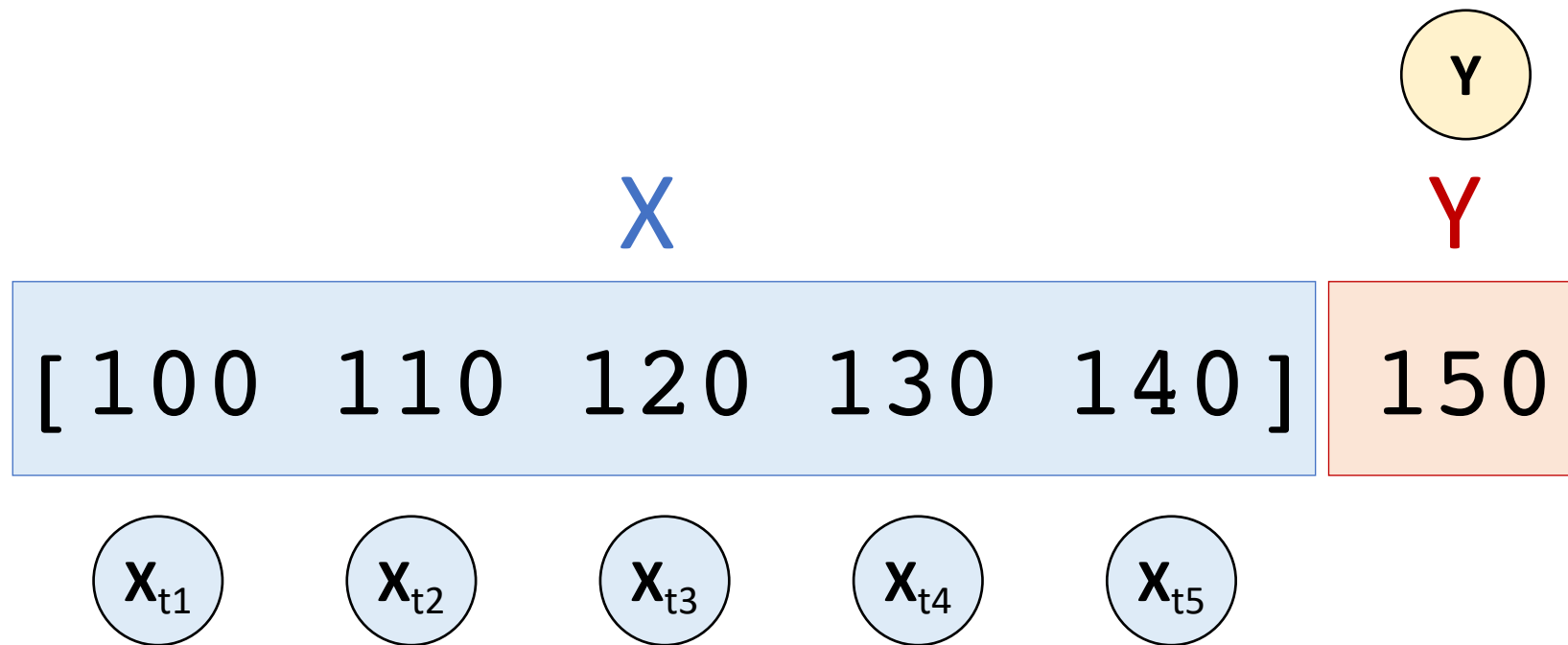
**Machine
Translation**

Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

[100, 110, 120, 130, 140, 150]



Time Series Data

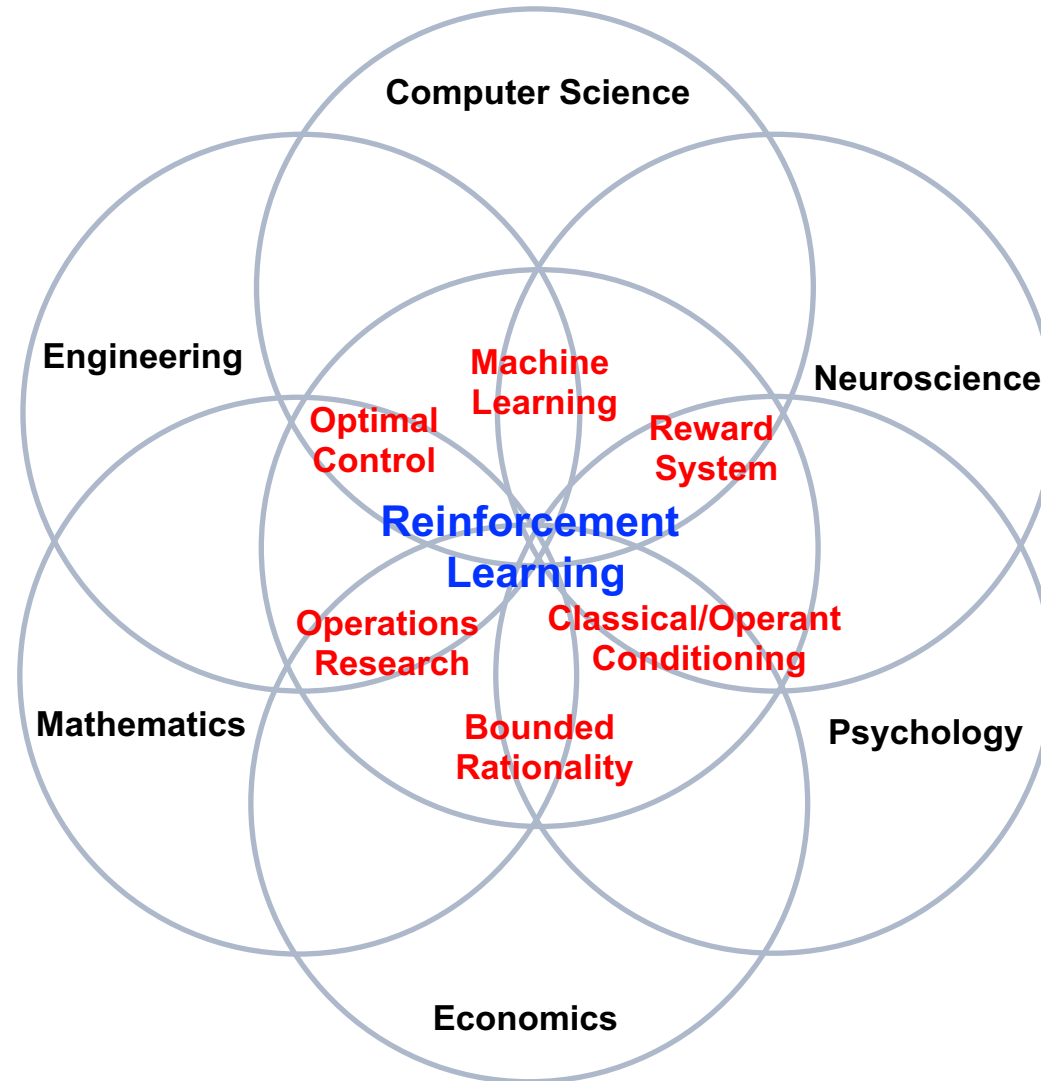
[10, 20, 30, 40, 50, 60, 70, 80, 90]

X

Y

[10	20	30]	40
[20	30	40]	50
[30	40	50]	60
[40	50	60]	70
[50	60	70]	80
[60	70	80]	90

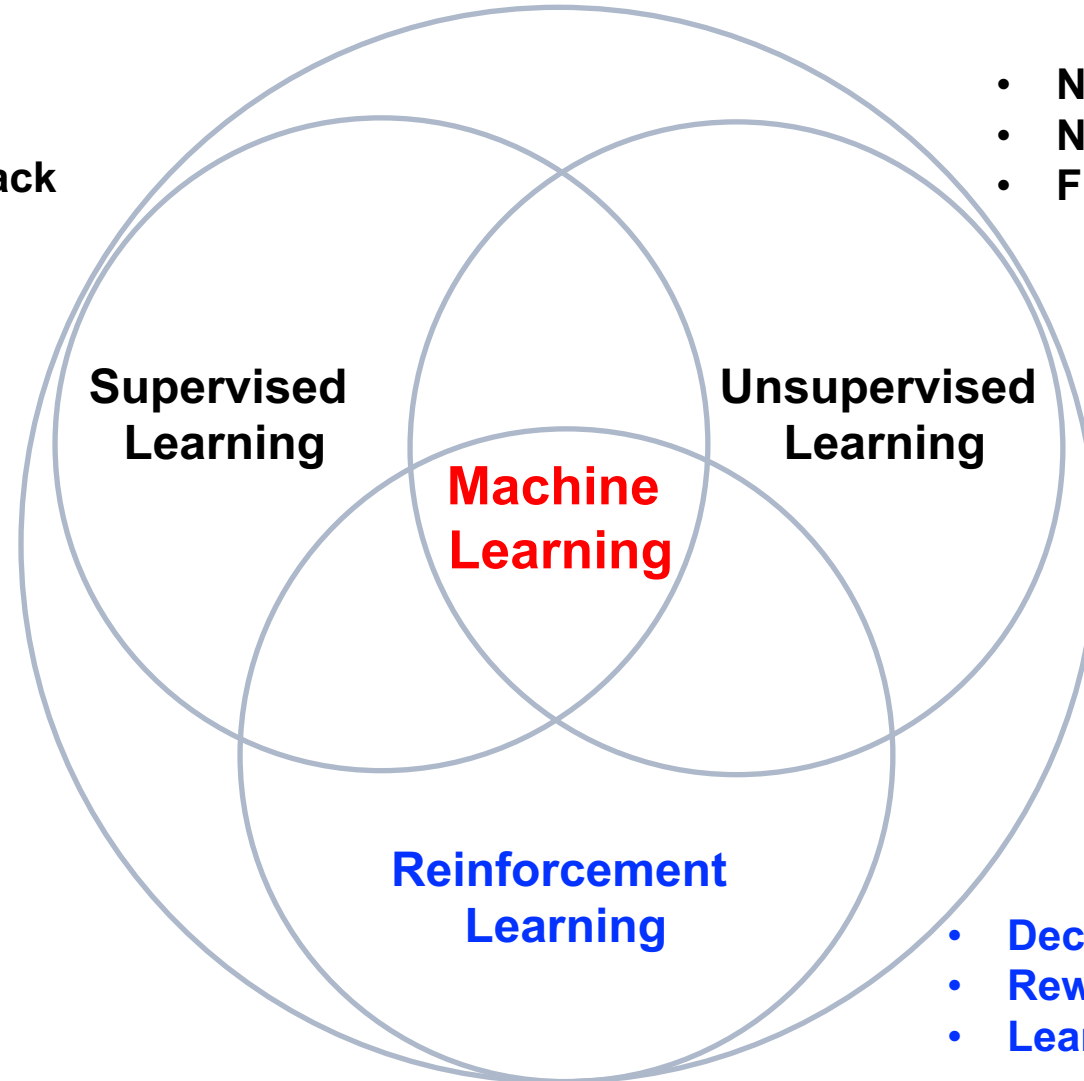
Reinforcement Learning (RL)



Branches of Machine Learning (ML)

Reinforcement Learning (RL)

- Labeled data
- Direct feedback
- Predict



- No Labels
- No feedback
- Find hidden structure

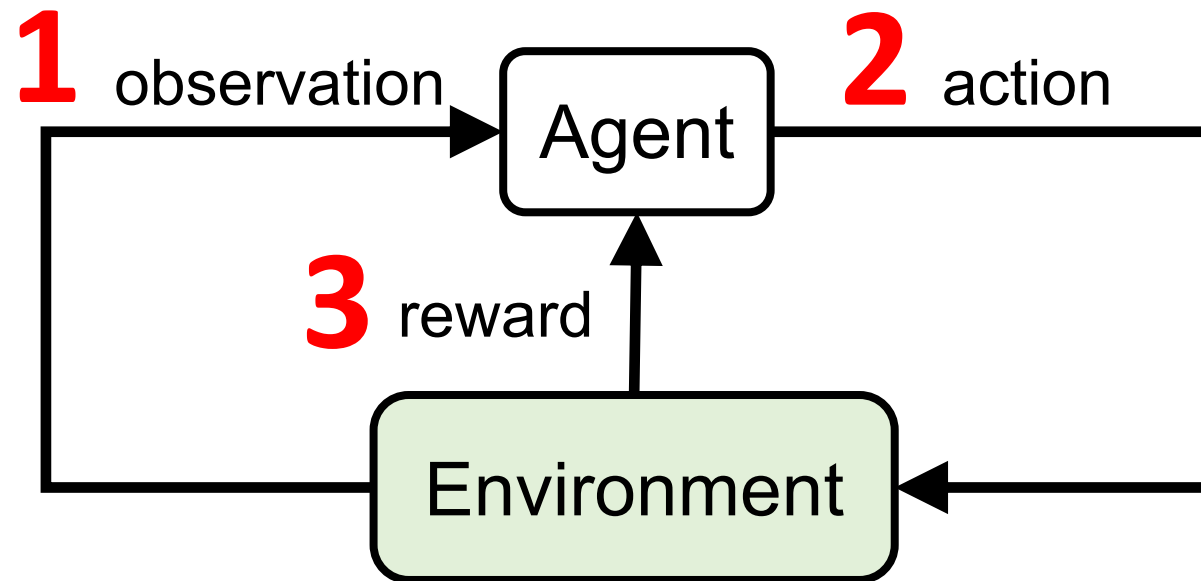
- Decision process
- Reward system
- Learn series of actions

Reinforcement Learning (DL)

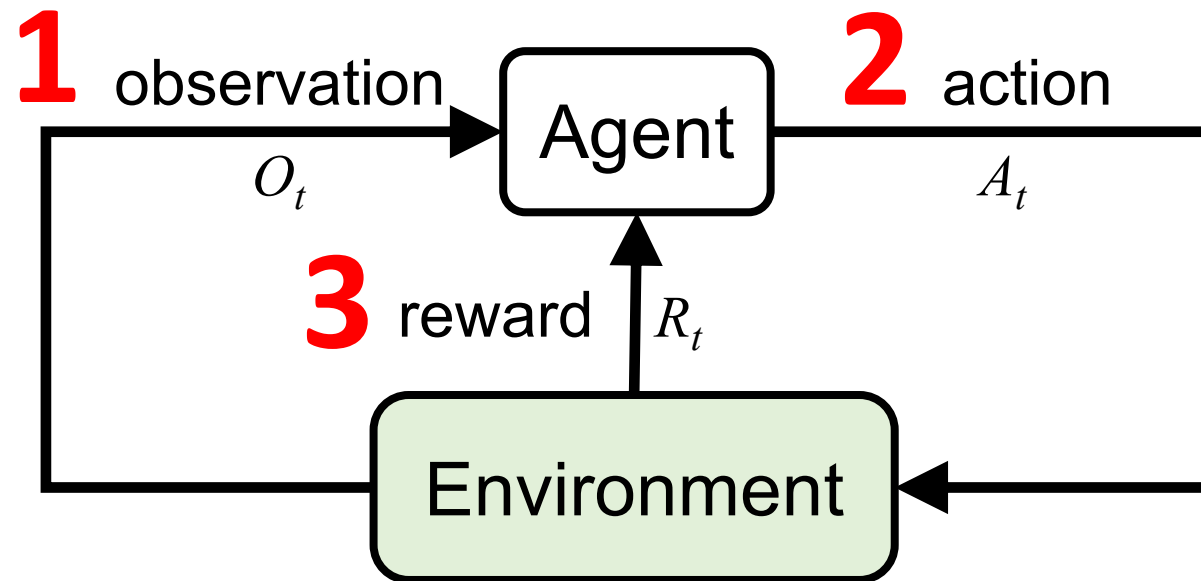
Agent

Environment

Reinforcement Learning (DL)

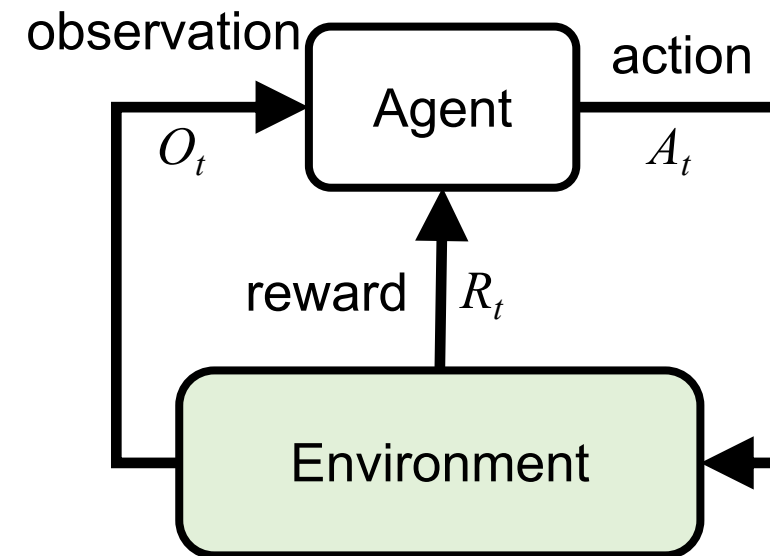


Reinforcement Learning (DL)



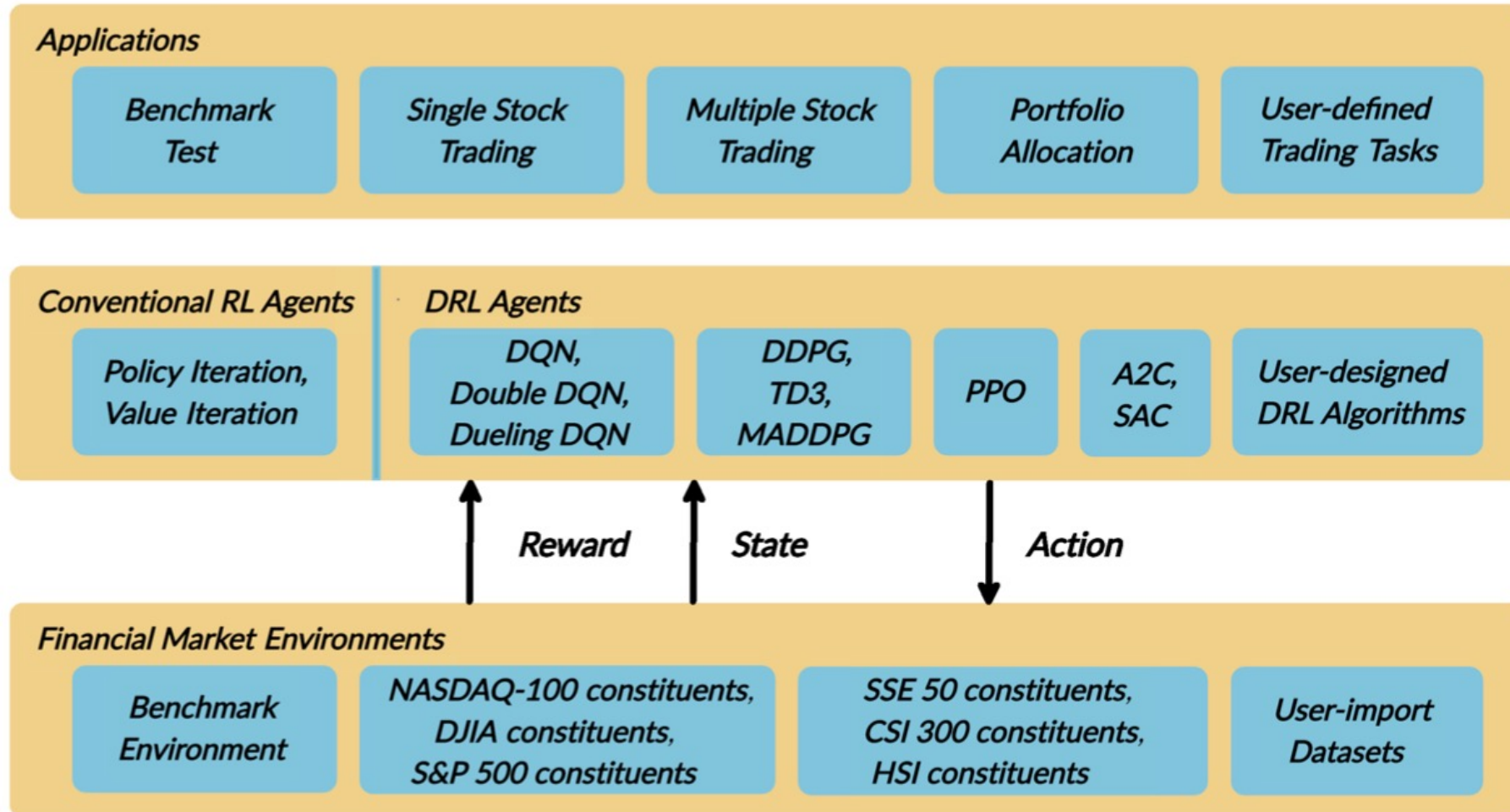
Agent and Environment

- At each step t the agent:
 - Executes **action** A_t
 - Receives **observation** O_t
 - Receives scalar **reward** R_t
- The environment:
 - Receives action A_t
 - Emits observation O_{t+1}
 - Emits scalar reward R_{t+1}
- t increments at env. step



FinRL:

A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance



FinRL

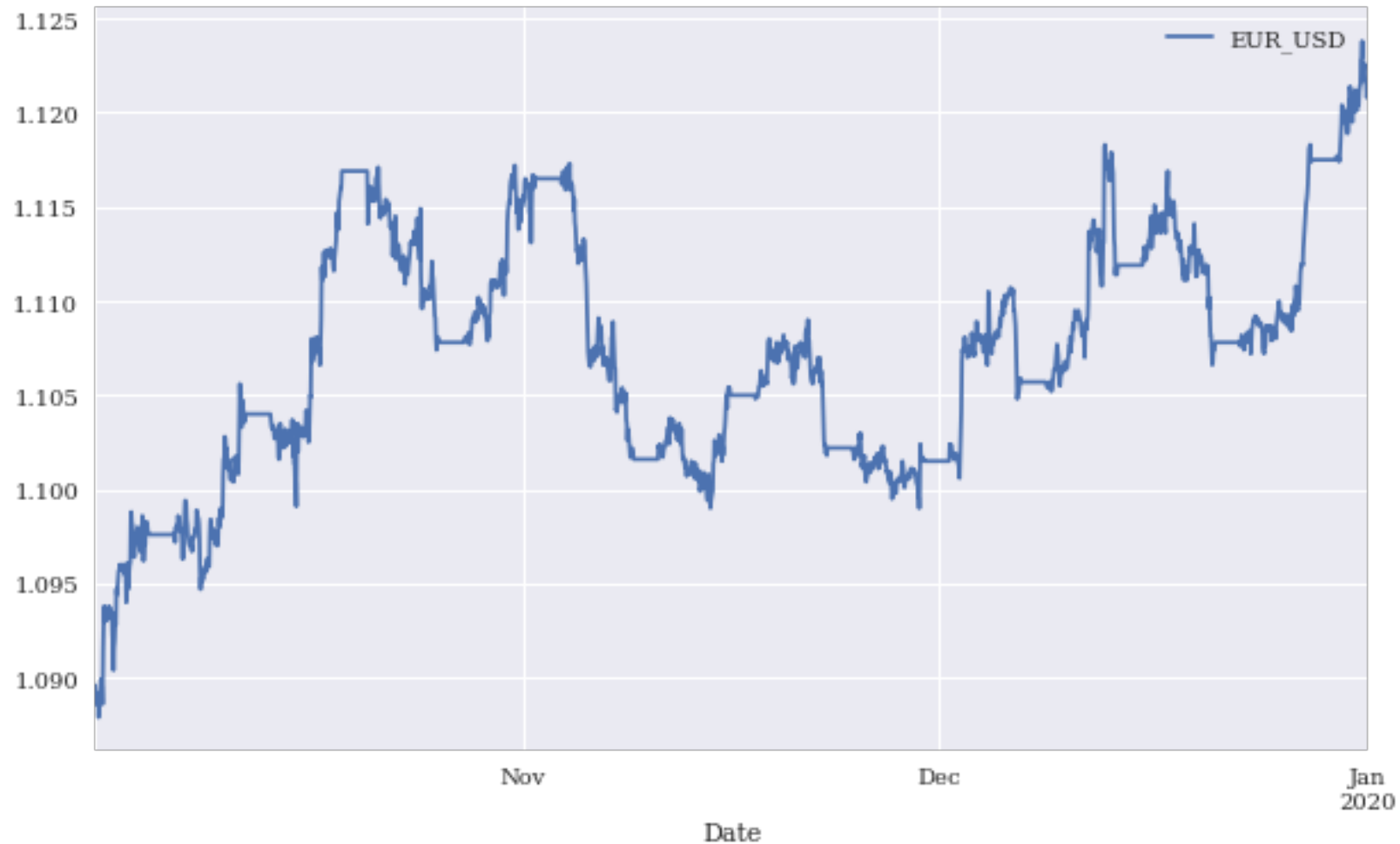
Deep Reinforcement Learning Algorithms

Algorithms	Input	Output	Type	State-action spaces support	Finance use cases support	Features and Improvements	Advantages
DQN	States	Q-value	Value based	Discrete only	Single stock trading	Target network, experience replay	Simple and easy to use
Double DQN	States	Q-value	Value based	Discrete only	Single stock trading	Use two identical neural network models to learn	Reduce overestimations
Dueling DQN	States	Q-value	Value based	Discrete only	Single stock trading	Add a specialized dueling Q head	Better differentiate actions, improves the learning
DDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Being deep Q-learning for continuous action spaces	Better at handling high-dimensional continuous action spaces
A2C	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Advantage function, parallel gradients updating	Stable, cost-effective, faster and works better with large batch sizes
PPO	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Clipped surrogate objective function	Improve stability, less variance, simply to implement
SAC	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Entropy regularization, exploration-exploitation trade-off	Improve stability
TD3	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Clipped double Q-Learning, delayed policy update, target policy smoothing.	Improve DDPG performance
MADDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Handle multi-agent RL problem	Improve stability and performance

```
import os
import numpy as np
import pandas as pd
from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['font.family'] = 'serif'
pd.set_option('precision', 4)
np.set_printoptions(suppress=True, precision=4)
os.environ['PYTHONHASHSEED'] = '0'
```

```
url = 'http://hilpisch.com/aiif_eikon_id_eur_usd.csv'
symbol = 'EUR_USD'
raw = pd.read_csv(url, index_col=0, parse_dates=True)
raw.head()
```

Mid-closing prices for EUR/USD (intraday)




```
optimizer = Adam(lr=0.001)

def create_model(hl=1, hu=128, optimizer=optimizer):
    model = Sequential()
    model.add(Dense(hu, input_dim=len(cols),
                    activation='relu'))
    for _ in range(hl):
        model.add(Dense(hu, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])

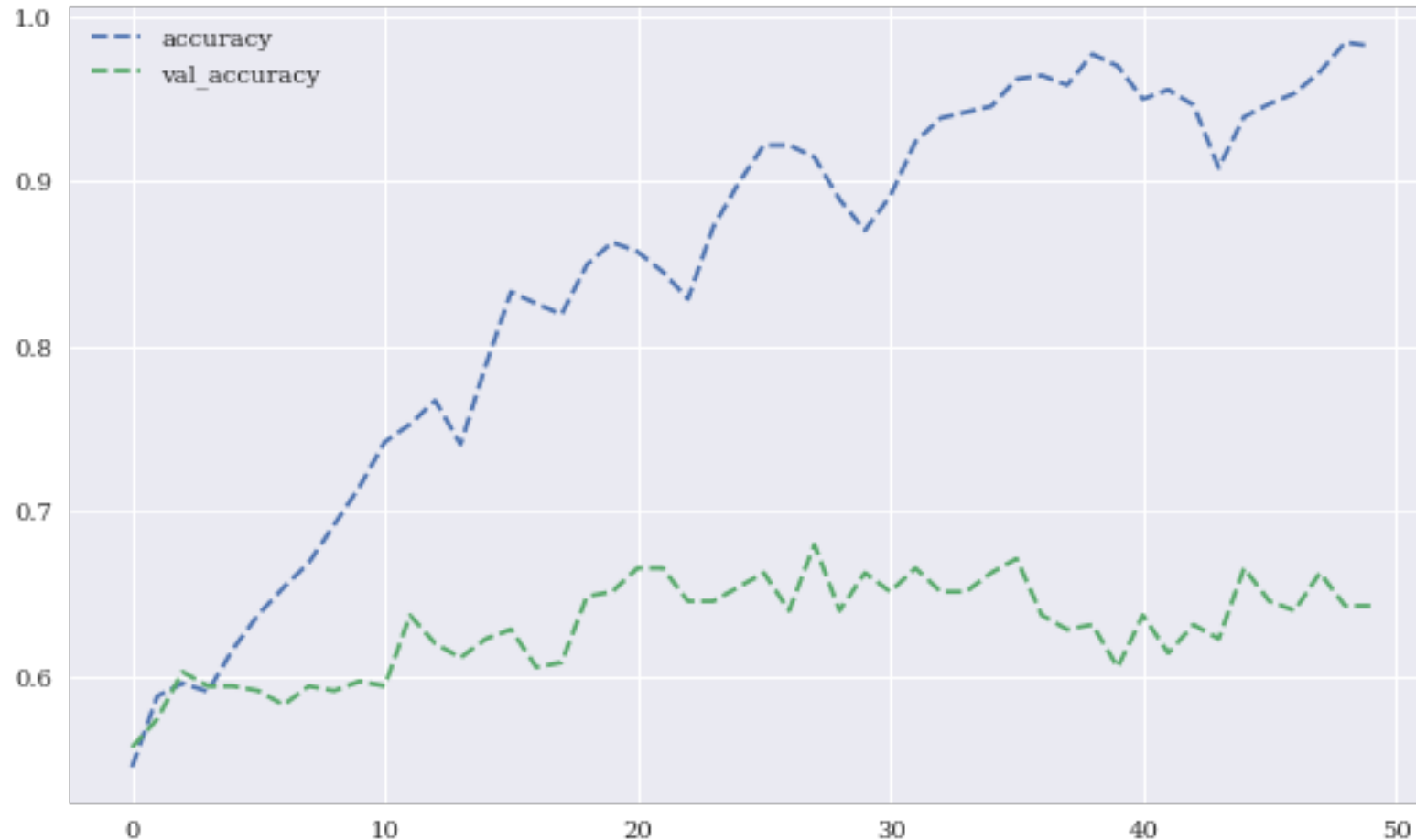
    return model

set_seeds()
model = create_model(hl=1, hu=128)
model.summary()
```

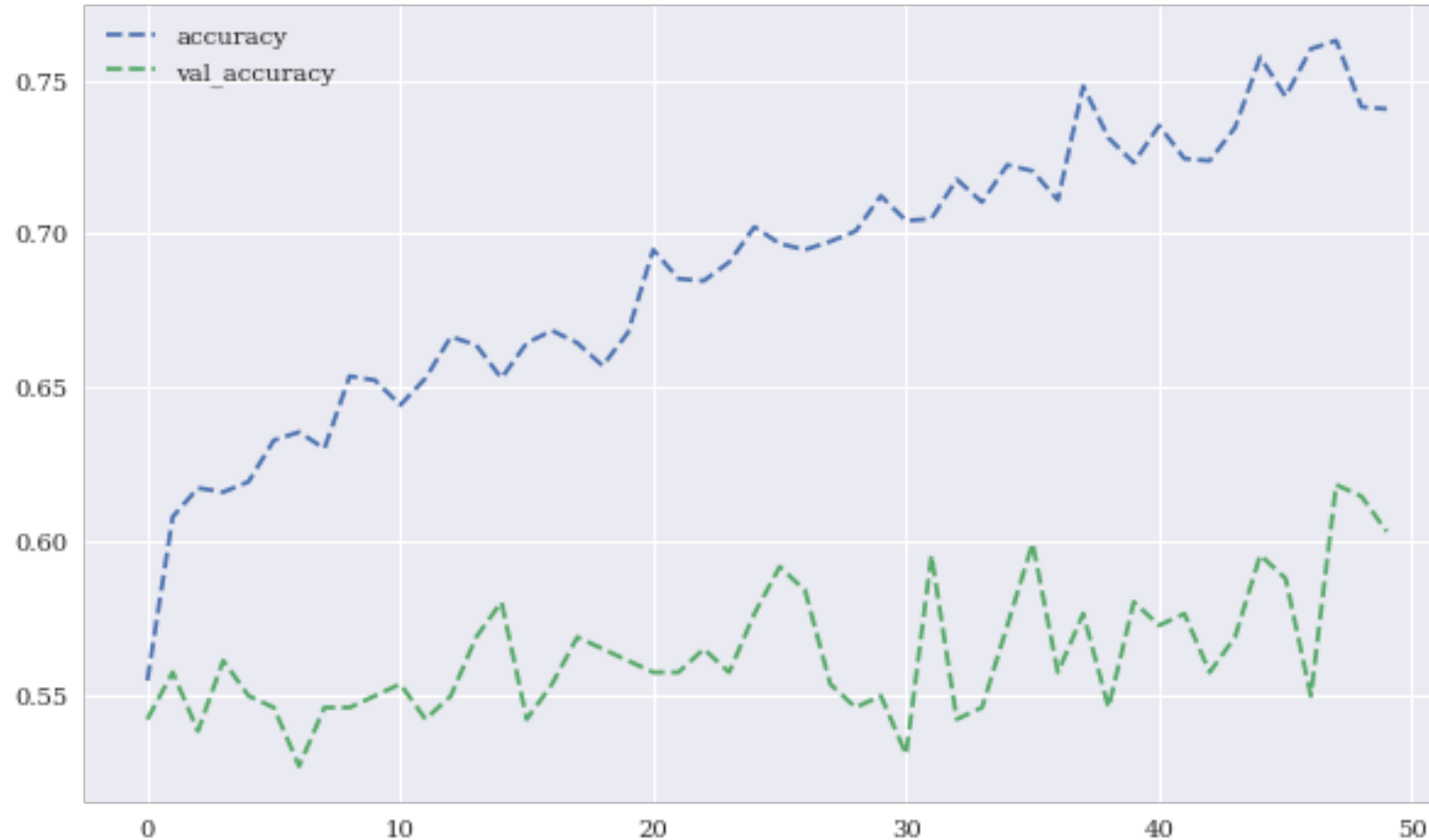
Training and validation accuracy values



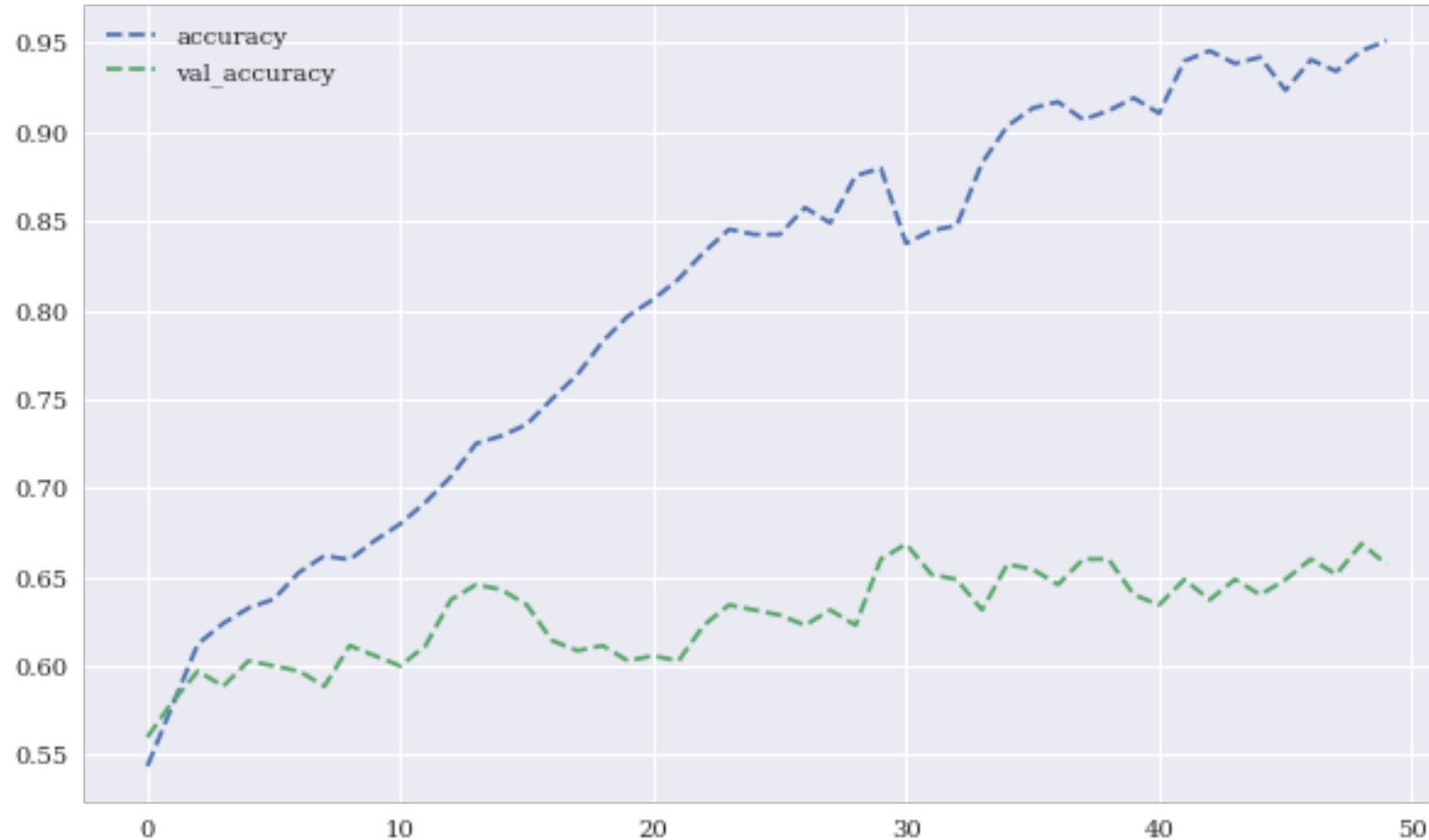
Training and validation accuracy values (normalized features data)



Training and validation accuracy values (with dropout)



Training and validation accuracy values (with regularization)



Training and validation accuracy values (with dropout and regularization)



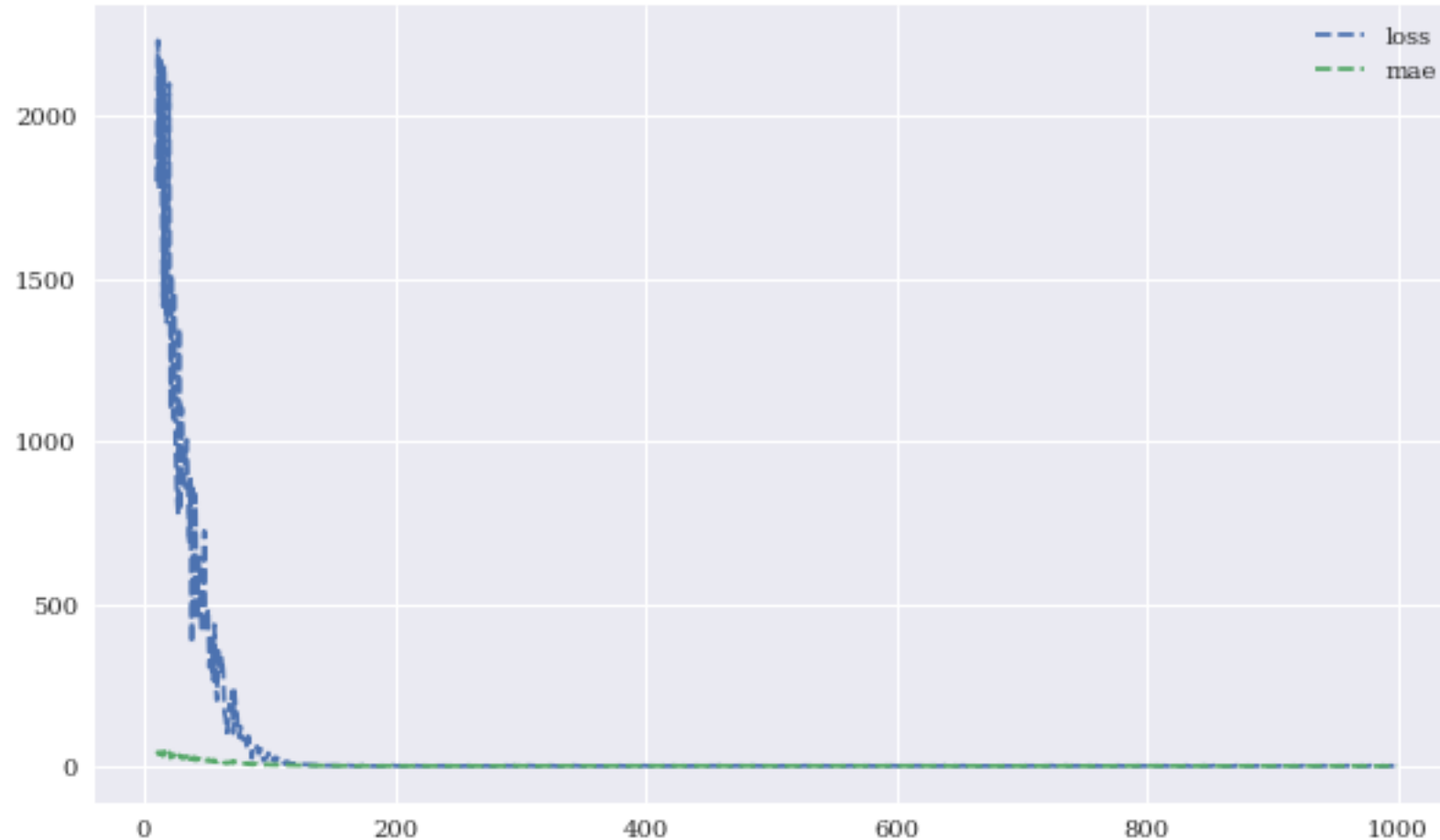
```
from keras.models import Sequential
from keras.layers import SimpleRNN, LSTM, Dense
```

```
model = Sequential()
model.add(SimpleRNN(100, activation='relu',
                    input_shape=(lags, 1)))
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adagrad', loss='mse',
              metrics=['mae'])
```

```
model.summary()
```

```
model.fit(g, epochs=1000, steps_per_epoch=5, verbose=False)
```

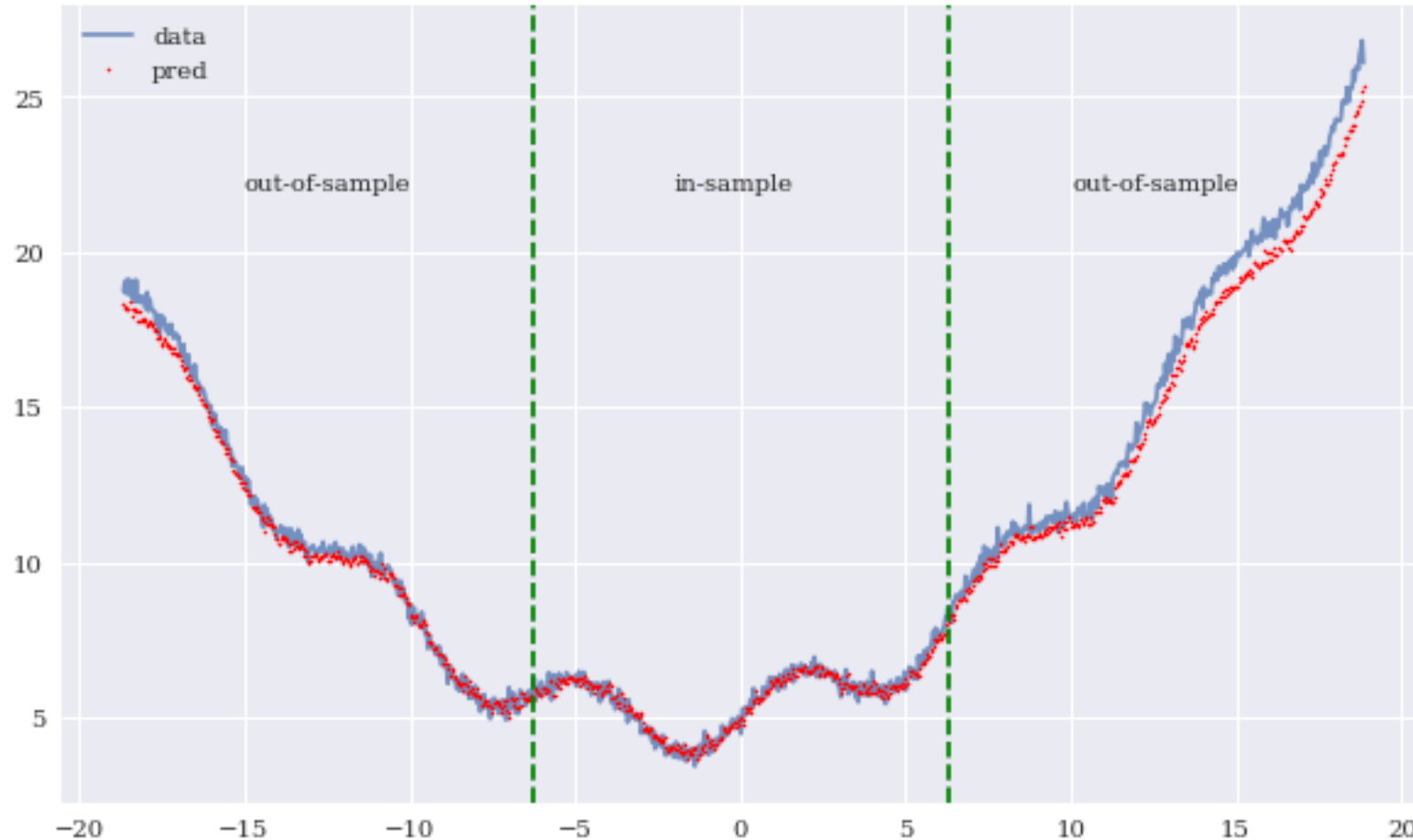
Performance metrics during RNN training



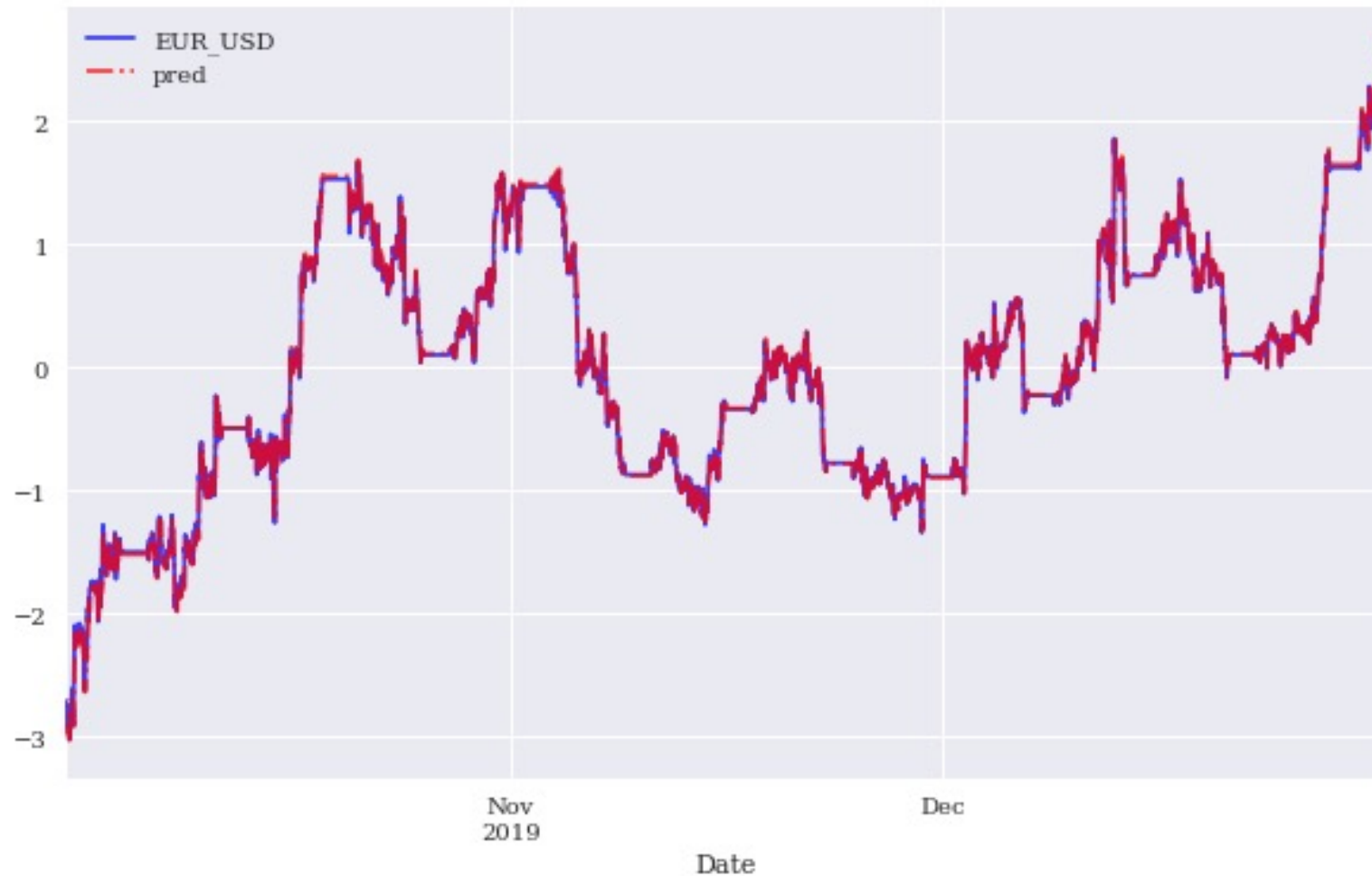
Sample sequence data



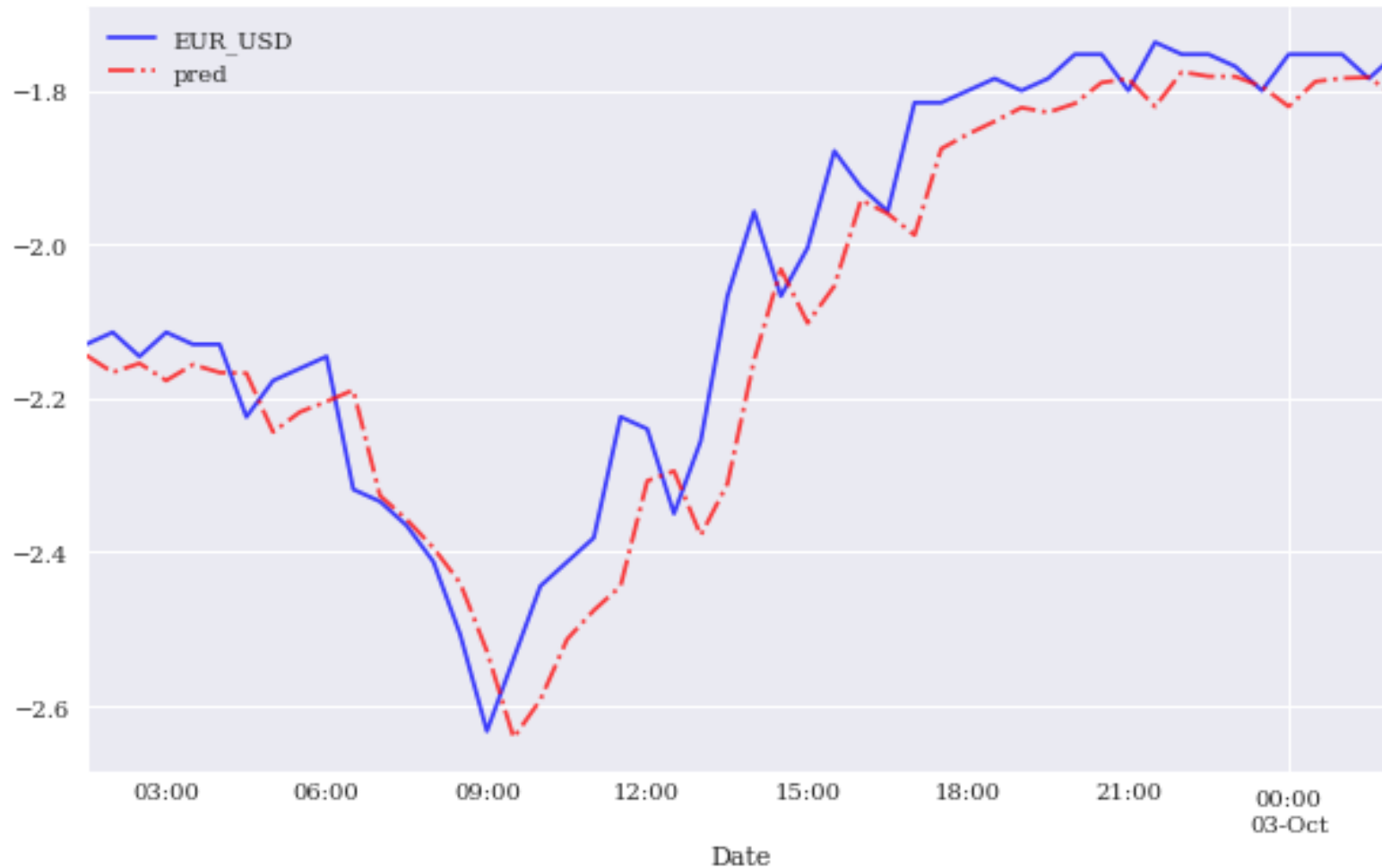
in-sample and out-of-sample predictions of the RNN



In-sample prediction for financial price series by the RNN (whole data set)



In-sample prediction for financial price series by the RNN (data sub-set)



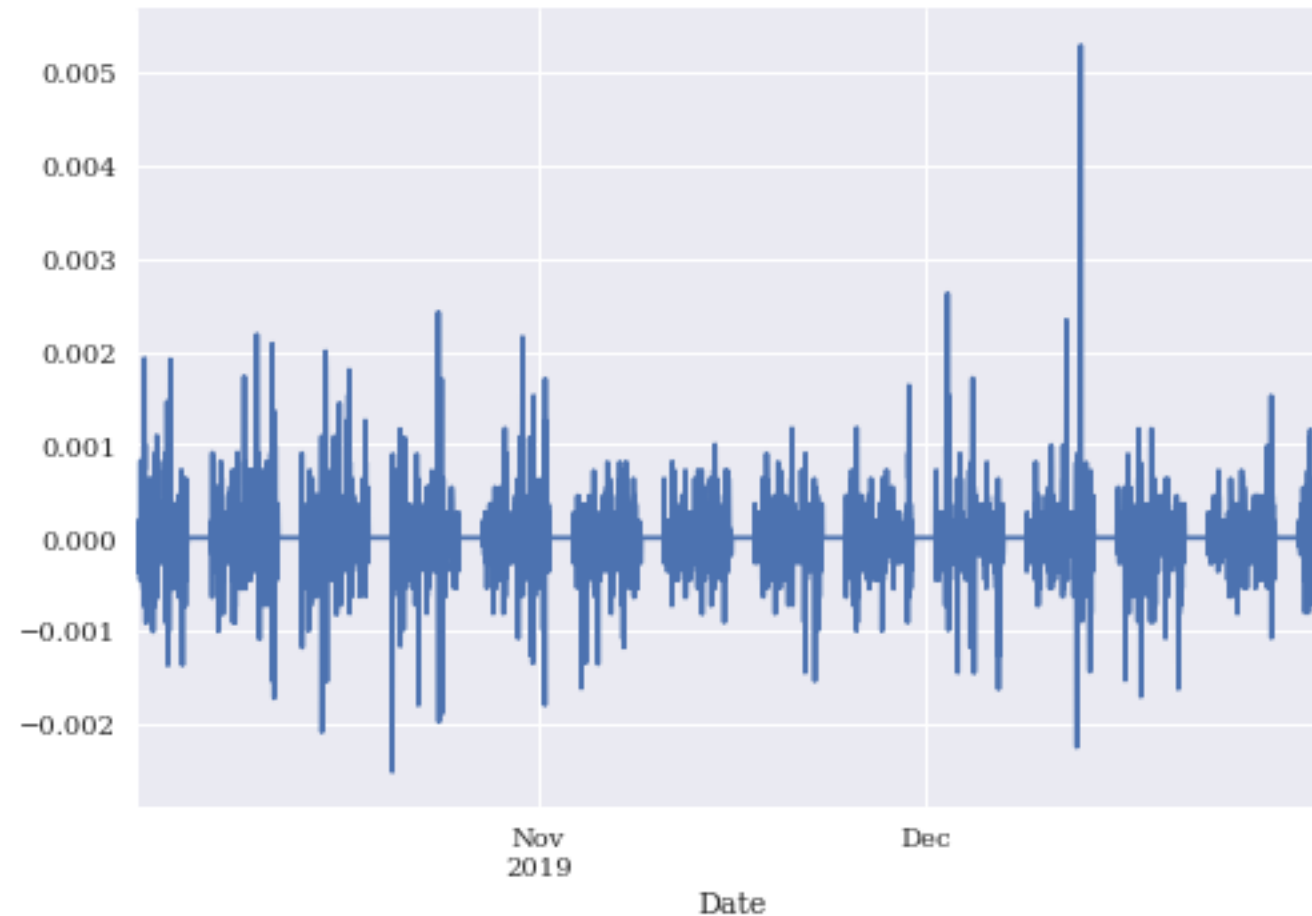
Financial Price Series

```
data = generate_data()  
data.plot()
```



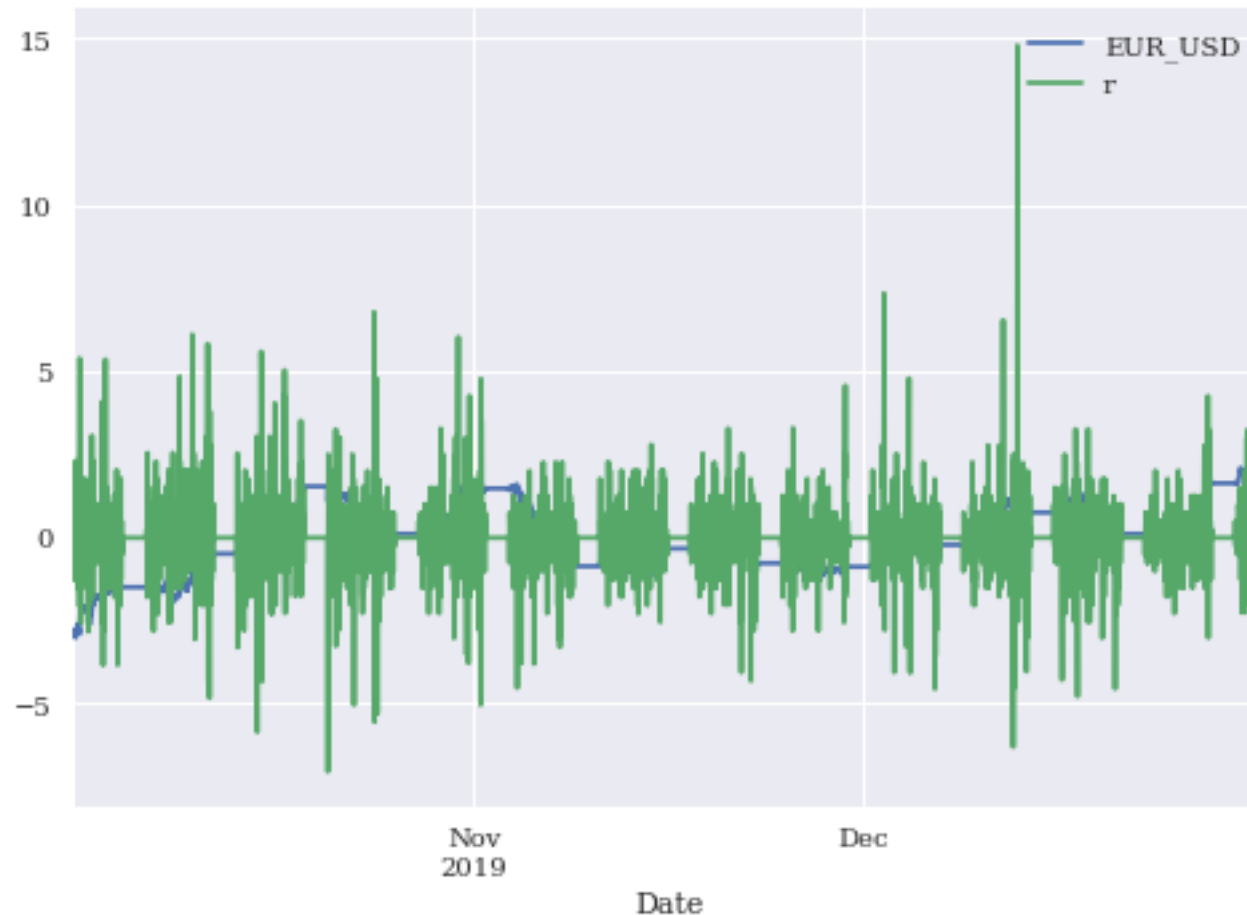
Financial Return Series

```
data['r'] = np.log(data / data.shift(1))  
data['r'].plot()
```

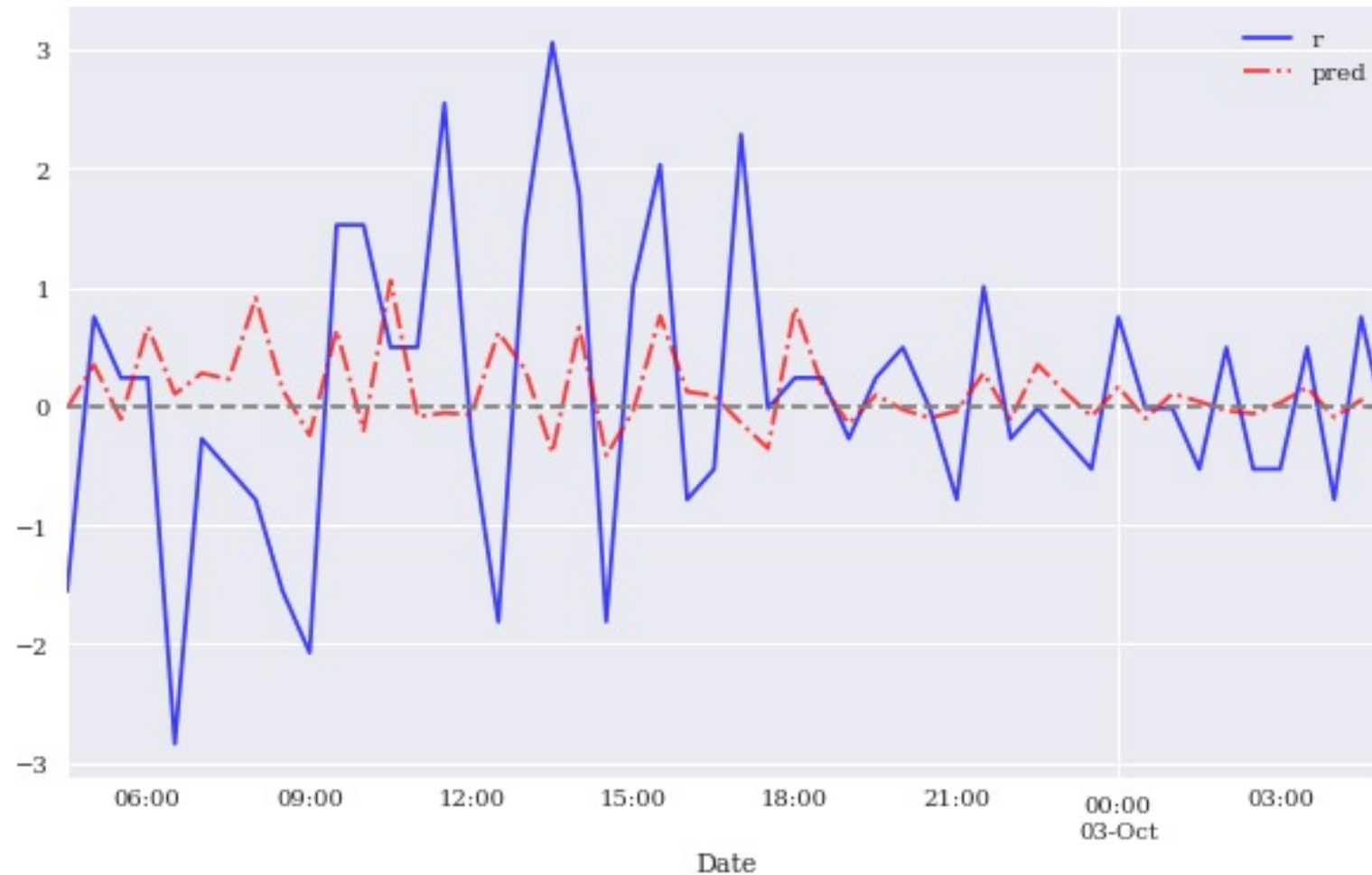


Financial Price and Return Normalization Series

```
data.dropna(inplace=True)
data = (data - data.mean()) / data.std()
data.plot()
```



In-sample prediction for financial return series by the RNN (data sub-set)




```
model = Sequential()
model.add(Conv1D(filters=96, kernel_size=5,
                 activation='relu',
                 input_shape=(len(cols), 1)))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()

model.fit(np.atleast_3d(train[cols]), train['d'],
         epochs=60, batch_size=48, verbose=False,
         validation_split=0.15, shuffle=False)
```

Performance metrics for the training and validation of the CNN



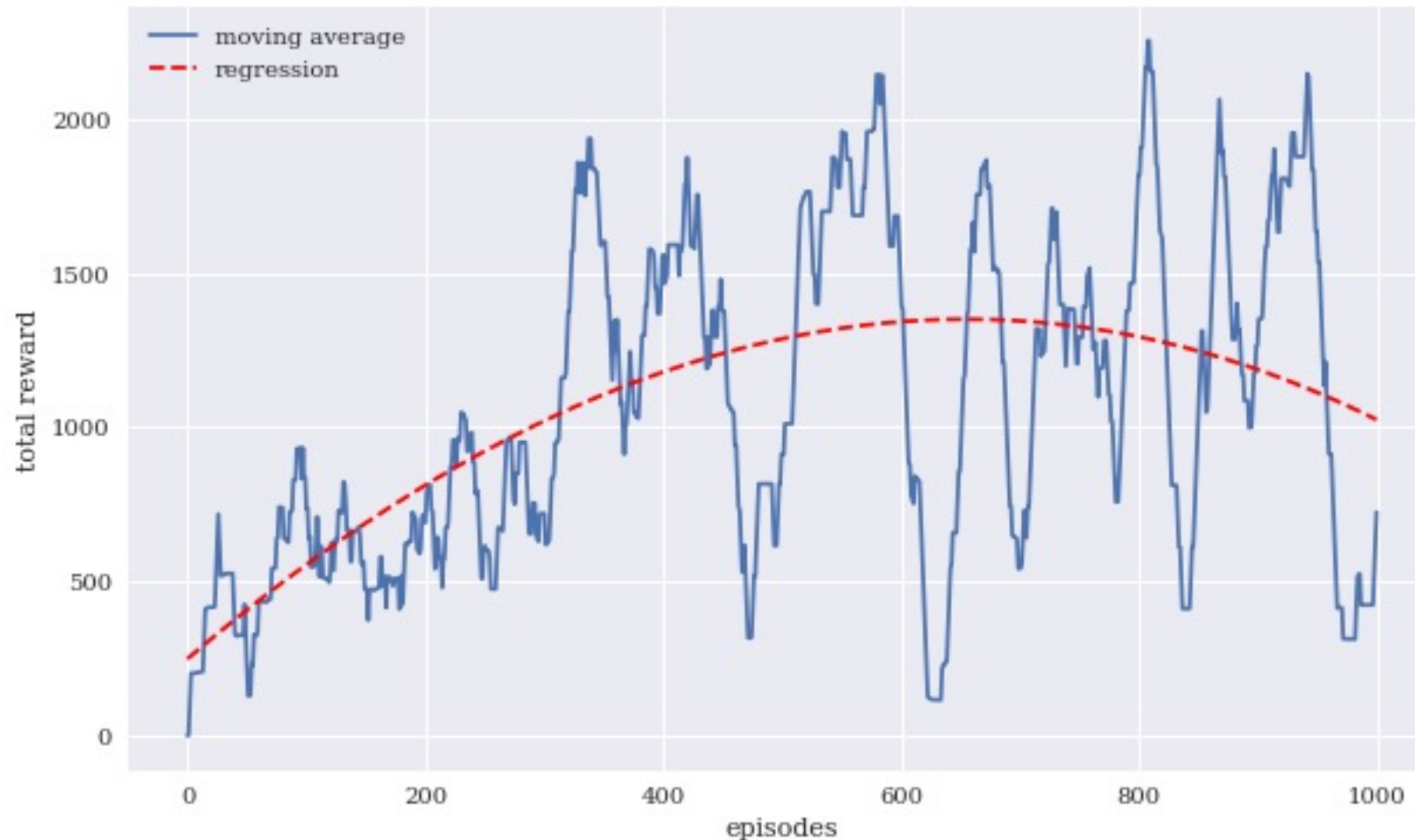
Gross performance of passive benchmark investment and CNN strategy (before/after transaction costs)



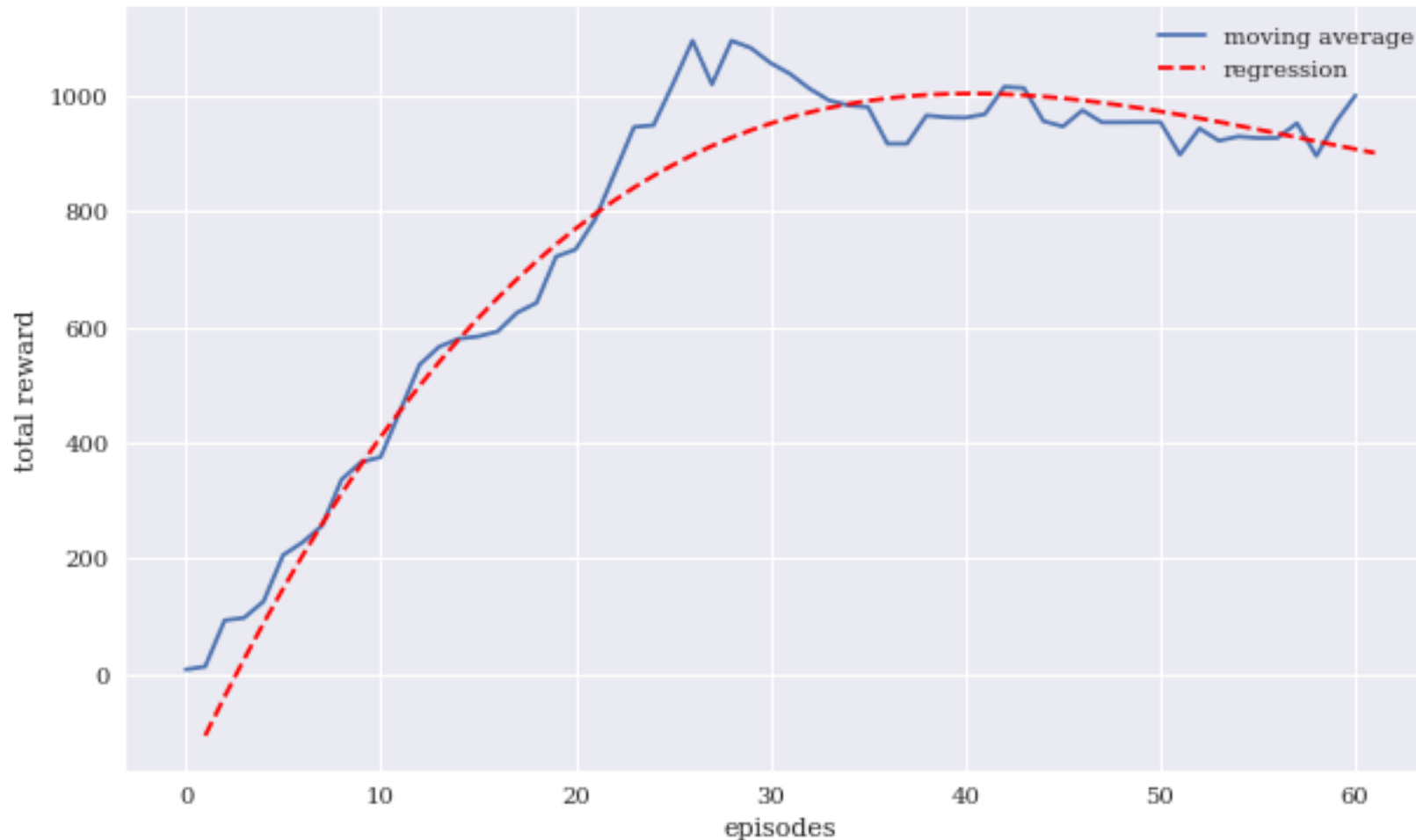
Reinforcement Learning in Finance

- **Simple Learning**
- **DNN Learning**
- **Q Learning**
- **Finance Environment**
- **Improved Finance Environment**
- **Improved Financial QL Agent**

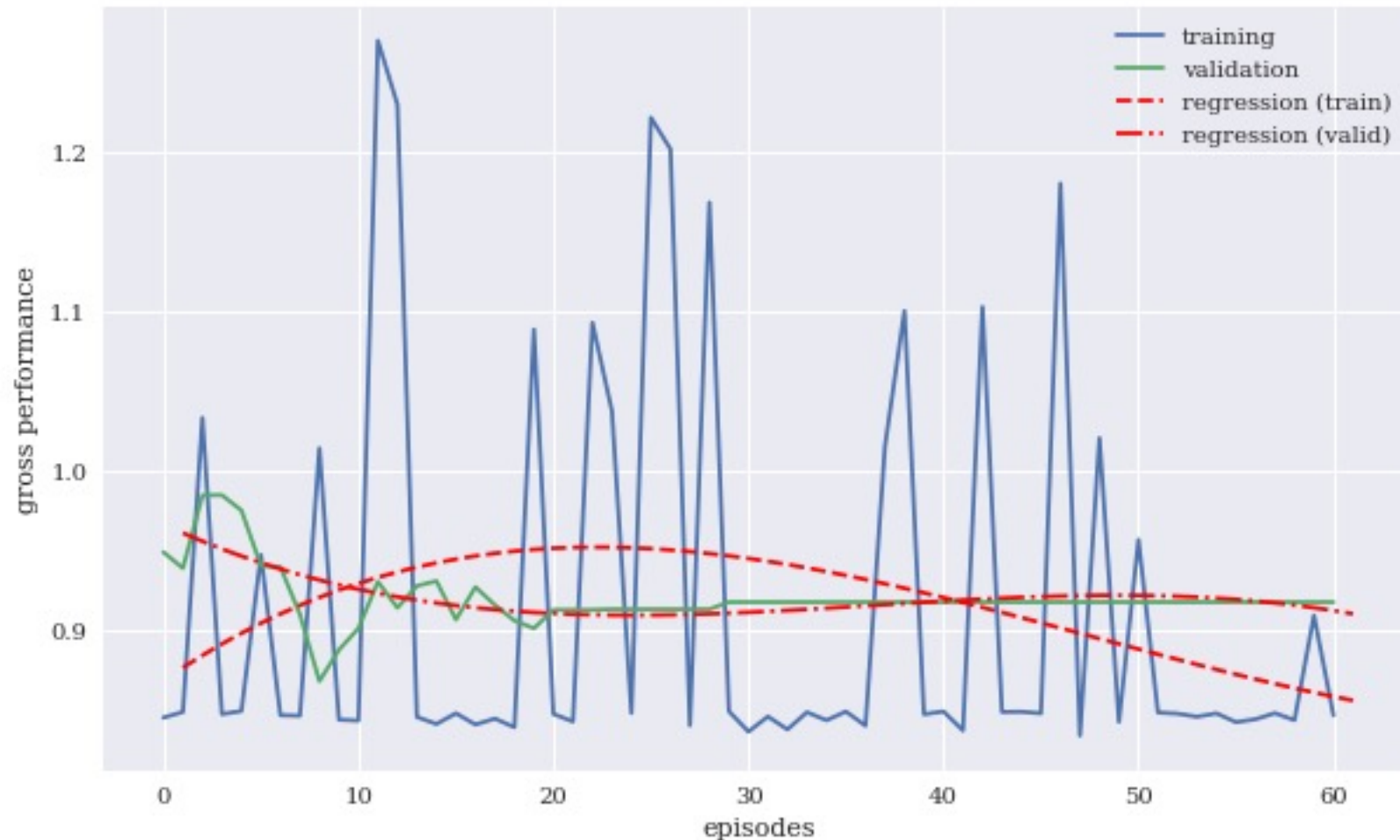
Average total rewards of DQLAgent for CartPole



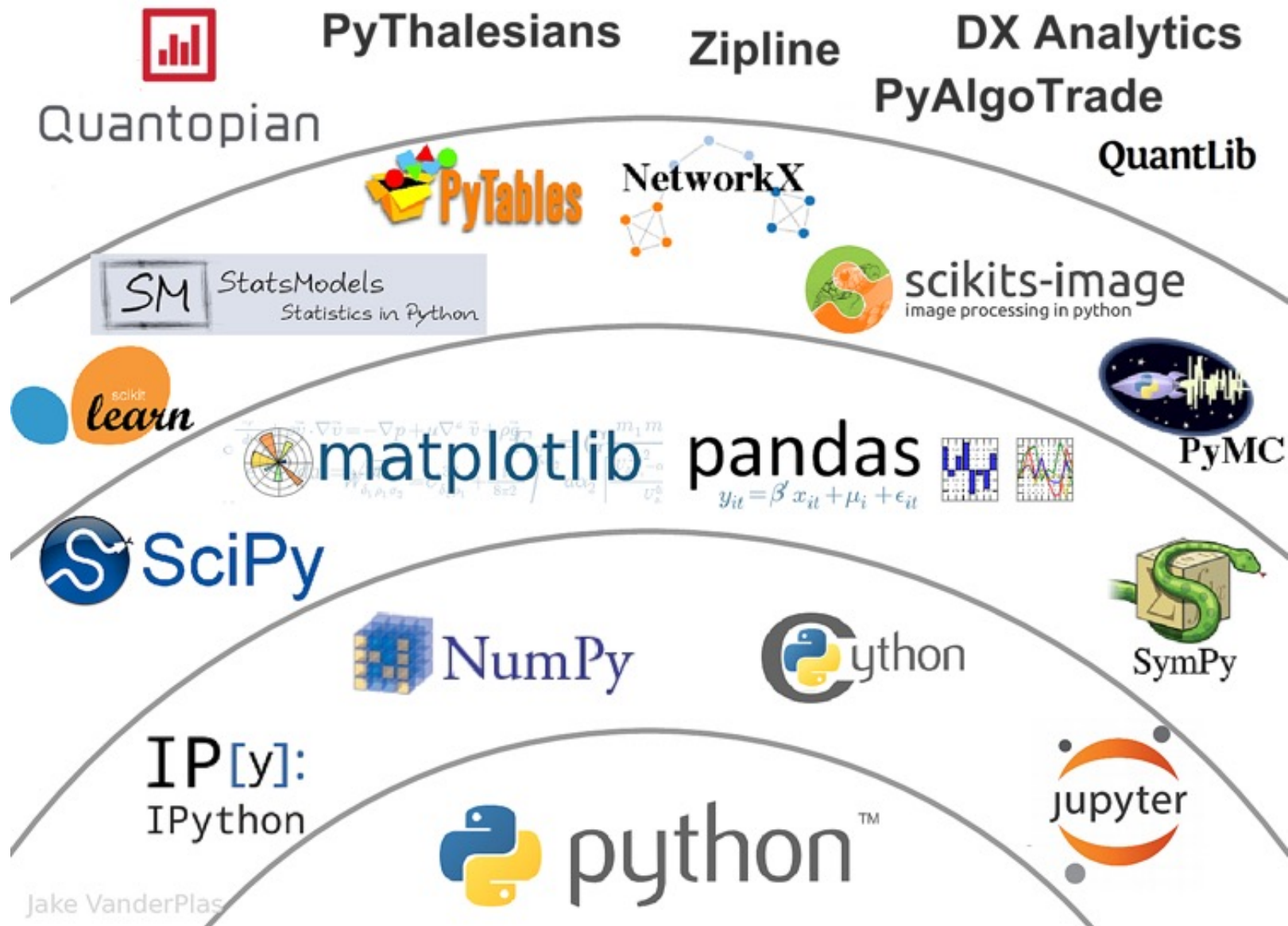
Average total rewards of DQLAgent for Finance



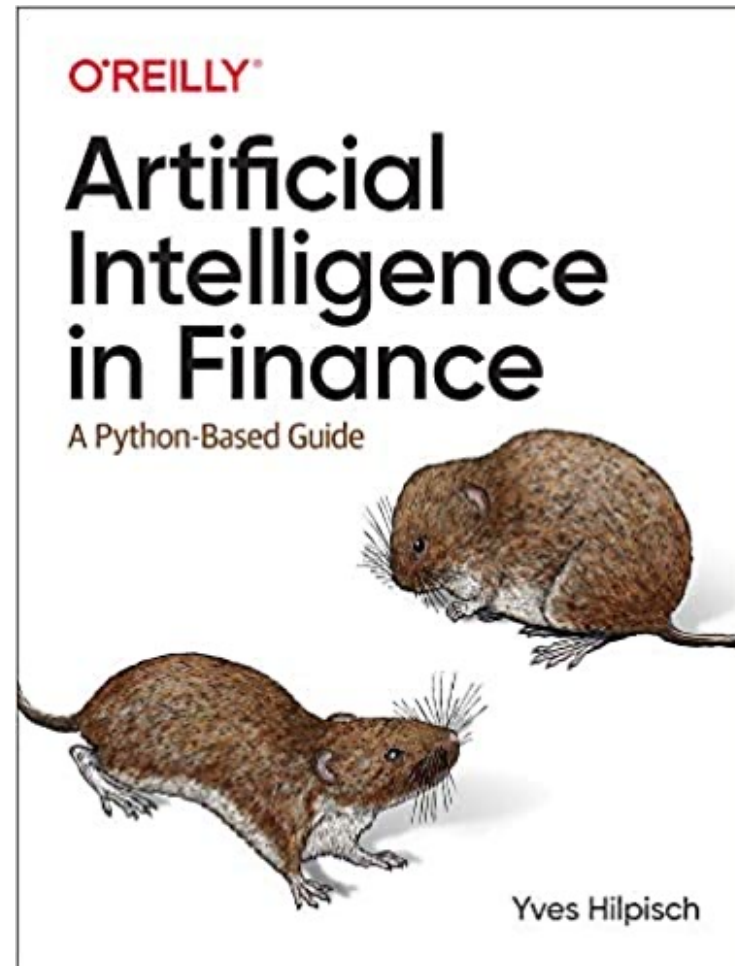
Training and validation performance of the FQLAgent per episode



The Quant Finance PyData Stack



Yves Hilpisch (2020),
Artificial Intelligence in Finance:
A Python-Based Guide,
O'Reilly



Yves Hilpisch (2020), **Artificial Intelligence in Finance: A Python-Based Guide**, O'Reilly

The screenshot shows the GitHub repository page for 'yhilpisch / aiif'. The repository is public and has 98 stars and 77 forks. The main branch is 'main'. The repository contains a 'code' directory, a '.gitignore' file, a 'LICENSE.txt' file, and a 'README.md' file. The README.md file is open, showing the title 'Artificial Intelligence in Finance' and a section 'About this Repository' which states that the repository provides Python code and Jupyter Notebooks for the book 'Artificial Intelligence in Finance' published by O'Reilly. The O'Reilly logo is visible at the bottom left of the README content. On the right side of the repository page, there is an 'About' section with a link to 'home.tpq.io/books/aiif', a 'Readme' link, and a 'View license' link. Below the 'About' section, there are sections for 'Releases' (no releases published), 'Packages' (no packages published), and 'Languages' (Jupyter Notebook 97.4%, Python 2.6%). A book cover for 'Artificial Intelligence in Finance: A Python-Based Guide' by Yves Hilpisch is displayed on the right side of the repository page. The cover features the O'Reilly logo, the title, the subtitle 'A Python-Based Guide', and an illustration of two mice.

https://github.com/yhilpisch/aiif

Notifications Star 98 Fork 77

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags

Go to file Code

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 4 commits

code	Code updates for TF 2.3.	11 months ago
.gitignore	Code updates for TF 2.3.	11 months ago
LICENSE.txt	Code updates.	11 months ago
README.md	Code updates.	11 months ago

README.md

Artificial Intelligence in Finance

About this Repository

This repository provides Python code and Jupyter Notebooks accompanying the **Artificial Intelligence in Finance** book published by [O'Reilly](#).

O'REILLY

About

Jupyter Notebooks and code for the book **Artificial Intelligence in Finance** (O'Reilly) by Yves Hilpisch.

home.tpq.io/books/aiif

Readme

View license

Releases

No releases published

Packages

No packages published

Languages

- Jupyter Notebook 97.4%
- Python 2.6%

O'REILLY

Artificial Intelligence in Finance
A Python-Based Guide

Yves Hilpisch

Yves Hilpisch (2020), **Artificial Intelligence in Finance: A Python-Based Guide**, O'Reilly

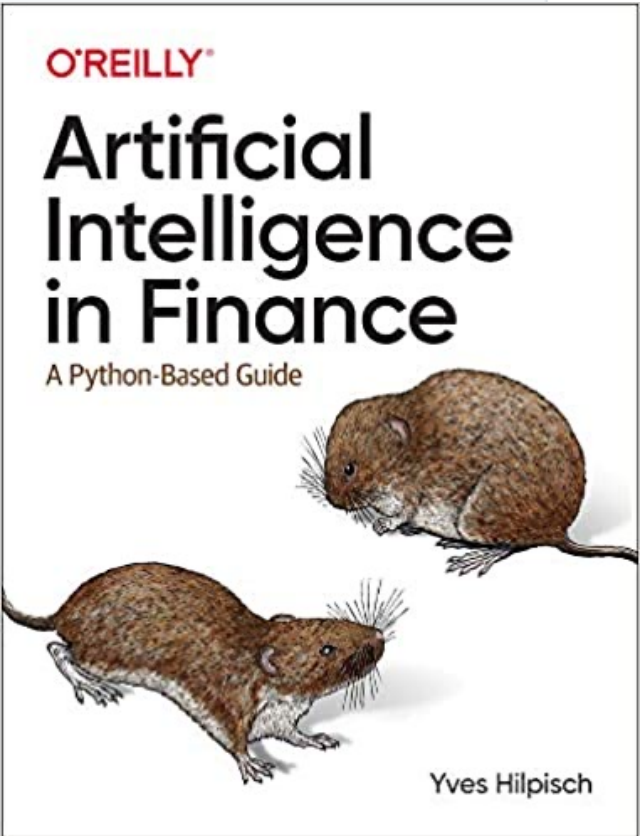
yhilpisch / aiif Public Notifications Star 98 Fork 77

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main aiif / code / <https://github.com/yhilpisch/aiif/tree/main/code> Go to file

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 History

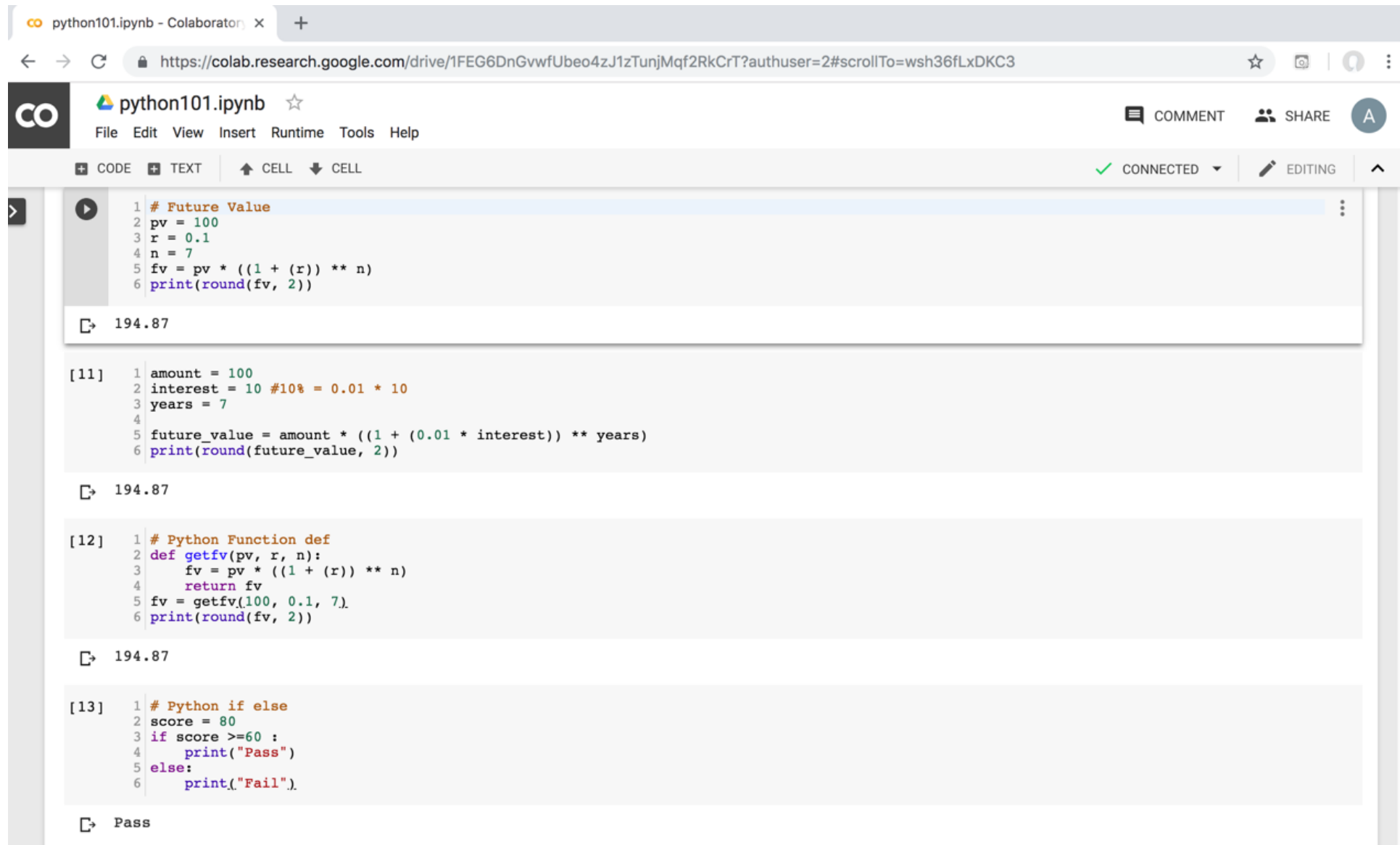
..	
oanda	Code updates for TF 2.3.
01_artificial_intelligence.ipynb	Code updates for TF 2.3.
02_superintelligence.ipynb	Code updates for TF 2.3.
03_normative_finance.ipynb	Code updates for TF 2.3.
04_data_driven_finance_a.ipynb	Initial commit.
04_data_driven_finance_b.ipynb	Initial commit.
05_machine_learning.ipynb	Code updates for TF 2.3.
06_ai_first_finance.ipynb	Code updates for TF 2.3.
07_dense_networks.ipynb	Code updates for TF 2.3.
08_recurrent_networks.ipynb	Code updates for TF 2.3.
09_reinforcement_learning_a.ipynb	Code updates.
09_reinforcement_learning_b.ipynb	Code updates for TF 2.3.



Source: <https://github.com/yhilpisch/aiif/tree/main/code>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options for CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells:

- Cell 1:** A code cell with the following Python code:

```
1 # Future Value
2 pv = 100
3 r = 0.1
4 n = 7
5 fv = pv * ((1 + (r)) ** n)
6 print(round(fv, 2))
```

The output is "194.87".
- Cell [11]:** A code cell with the following Python code:

```
1 amount = 100
2 interest = 10 #10% = 0.01 * 10
3 years = 7
4
5 future_value = amount * ((1 + (0.01 * interest)) ** years)
6 print(round(future_value, 2))
```

The output is "194.87".
- Cell [12]:** A code cell with the following Python code:

```
1 # Python Function def
2 def getfv(pv, r, n):
3     fv = pv * ((1 + (r)) ** n)
4     return fv
5 fv = getfv(100, 0.1, 7)
6 print(round(fv, 2))
```

The output is "194.87".
- Cell [13]:** A code cell with the following Python code:

```
1 # Python if else
2 score = 80
3 if score >=60 :
4     print("Pass")
5 else:
6     print("Fail").
```

The output is "Pass".

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a "Comment" button, a "Share" button, and a settings gear. A "Table of contents" sidebar on the left lists various topics under "AI in Finance", with "Uncertainty and Risk" selected. The main content area displays a table of contents with expandable sections: "AI in Finance", "Normative Finance and Financial Theories", and "Uncertainty and Risk". Below the table of contents, a code cell is visible, containing Python code that imports numpy and defines variables for stock and bond prices and payoffs.

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ A

RAM Disk Editing

Table of contents

- AI in Finance
 - Normative Finance and Financial Theories
 - Uncertainty and Risk**
 - Expected Utility Theory (EUT)
 - Mean-Variance Portfolio Theory (MVPT)
 - Capital Asset Pricing Model (CAPM)
 - Arbitrage Pricing Theory (APT)
 - Deep Learning for Financial Time Series Forecasting
 - Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python
 - Investment Portfolio Optimization

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing ^

Table of contents

- Data Driven Finance
 - Financial Econometrics and Regression**
 - Data Availability
 - Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
 - Debunking Central Assumptions
 - Normality
 - Sample Data Sets
 - Real Financial Returns
 - Linear Relationships
- Deep Learning for Financial Time Series Forecasting
- Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python
 - Investment Portfolio Optimization

Data Driven Finance

Financial Econometrics and Regression

```
[18] 1 import numpy as np
      2
      3 def f(x):
      4     return 2 + 1 / 2 * x
      5
      6 x = np.arange(-4, 5)
      7 x

array([-4, -3, -2, -1, 0, 1, 2, 3, 4])
```

```
1 y = f(x)
2 y

array([ 0.00,  0.50,  1.00,  1.50,  2.00,  2.50,  3.00,  3.50,  4.00])
```

```
1 print('x', x)
2
3 print('y', y)
4
5 beta = np.cov(x, y, ddof=0)[0, 1] / x.var()
6 print('beta', beta)
```

Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled 'python101.ipynb' and has a star icon. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status indicator 'All changes saved'. On the right, there are icons for 'Comment', 'Share', and a user profile 'A'. Below the menu, there are RAM and Disk usage indicators, and a status 'Editing'. A 'Table of contents' sidebar is on the left, listing various topics, with 'Machine Learning' selected. The main area shows a code cell with the following Python code:

```
1 import numpy as np
2 import pandas as pd
3 from pylab import plt, mpl
4 np.random.seed(100)
5 plt.style.use('seaborn')
6 mpl.rcParams['savefig.dpi'] = 300
7 mpl.rcParams['font.family'] = 'serif'
8
9 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
10
11 raw = pd.read_csv(url, index_col=0, parse_dates=True)['EUR=']
12 raw.head()
```

The output of the code cell is a DataFrame with the following data:

Date	EUR=
2010-01-01	1.4323
2010-01-04	1.4411
2010-01-05	1.4368
2010-01-06	1.4412
2010-01-07	1.4318

Below the output, there is a code cell with the following code:

```
[2] 1 raw.tail()
```

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

Table of contents


- Mean-Variance Portfolio Theory
- Capital Asset Pricing Model
- Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
- Sample Data Sets
- Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
- Machine Learning
- Data
- Success
- Capacity
- Evaluation
- Bias & Variance
- Cross-Validation
- AI-First Finance
 - Efficient Markets**
 - Market Prediction Based on Returns Data
 - Market Prediction With More Features
 - Market Prediction Intraday

+ Code + Text

Efficient Markets

```
1 import numpy as np
2 import pandas as pd
3 from pylab import plt, mpl
4 plt.style.use('seaborn')
5 mpl.rcParams['savefig.dpi'] = 300
6 mpl.rcParams['font.family'] = 'serif'
7 pd.set_option('precision', 4)
8 np.set_printoptions(suppress=True, precision=4)
9
10 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
11 data = pd.read_csv(url, index_col=0, parse_dates=True).dropna()
12 (data / data.iloc[0]).plot(figsize=(10, 6), cmap='coolwarm')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f29f972f210>



14
12
10
8
6
4
2
0

2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020

Legend:
— AAPL.O
— MSFT.O
— INTC.O
— AMZN.O
— GS.N
— SPY
— .SPX
— .VIX
— EUR=
— XAU=
— GDZ
— GLD

Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share



Table of contents

Deep Learning (DL) in Finance

Dense Neural Networks (DNN)

Baseline Prediction

Normalization

Dropout

Regularization

Bagging

Optimizers

Recurrent Neural Networks (RNN)

First Example

Second Example

Financial Price Series

Financial Return Series

Financial Features

Deep RNNs

Convolutional Neural Networks (CNN)

Reinforcement Learning (RL) in Finance

+ Code + Text

Connect

Editing

Deep Learning (DL) in Finance

- Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.
- Github: <https://github.com/yhilpisch/aiif/>

Dense Neural Networks (DNN)

```
1 import os
2 import numpy as np
3 import pandas as pd
4 from pylab import plt, mpl
5 plt.style.use('seaborn')
6 mpl.rcParams['savefig.dpi'] = 300
7 mpl.rcParams['font.family'] = 'serif'
8 pd.set_option('precision', 4)
9 np.set_printoptions(suppress=True, precision=4)
10 os.environ['PYTHONHASHSEED'] = '0'
```

```
[ ] 1 url = 'http://hilpisch.com/aiif_eikon_id_eur_usd.csv'
    2 symbol = 'EUR_USD'
    3 raw = pd.read_csv(url, index_col=0, parse_dates=True)
    4 raw.head()
```

HIGH LOW OPEN CLOSE

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

- Table of contents
- Deep Learning (DL) in Finance
 - Dense Neural Networks (DNN)
 - Baseline Prediction
 - Normalization
 - Dropout
 - Regularization**
 - Bagging
 - Optimizers
 - Recurrent Neural Networks (RNN)
 - First Example
 - Second Example
 - Financial Price Series
 - Financial Return Series
 - Financial Features
 - Deep RNNs
 - Convolutional Neural Networks (CNN)
 - Reinforcement Learning (RL) in Finance

+ Code + Text Connect Editing

```
[0.35268253087997437, 0.8991981744766235]
```

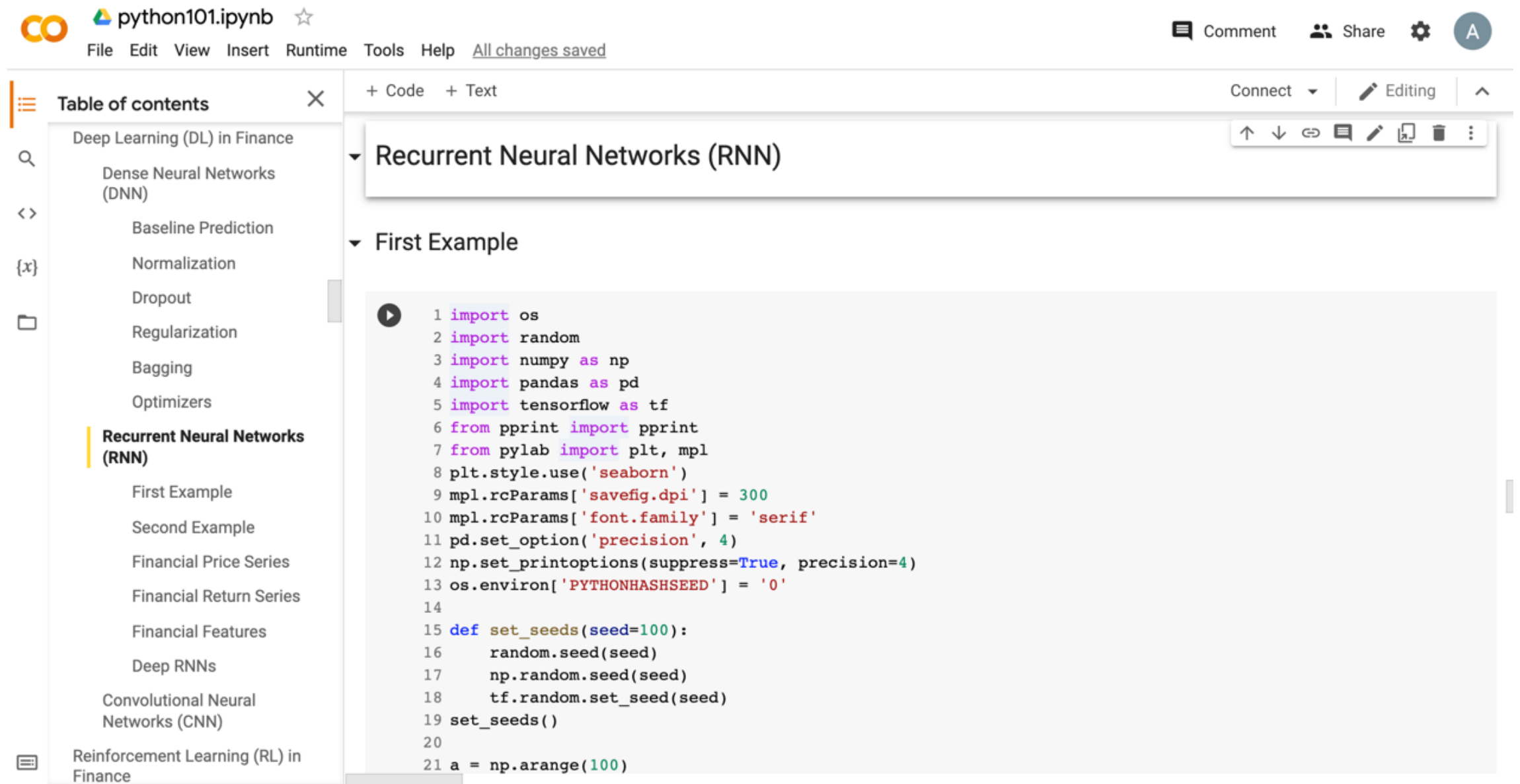
```
[ ] 1 model.evaluate(test_cols, test['d'])
```

```
14/14 [=====] - 0s 8ms/step - loss: 0.9098 - accuracy: 0.6705  
[0.9098052382469177, 0.6704805493354797]
```

```
1 res = pd.DataFrame(h.history)  
2 res[['accuracy', 'val_accuracy']].plot(figsize=(10, 6), style='--');
```



Python in Google Colab (Python101)



The image shows a Google Colab notebook interface. At the top left is the Colab logo and the notebook title 'python101.ipynb'. Below the title is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and 'All changes saved'. On the top right, there are icons for Comment, Share, and a user profile icon labeled 'A'. Below the menu bar, there is a toolbar with '+ Code' and '+ Text' buttons, and a 'Connect' dropdown menu. The main area is divided into a left sidebar and a main content area. The sidebar contains a 'Table of contents' panel with a search icon and a list of topics: Deep Learning (DL) in Finance, Dense Neural Networks (DNN), Baseline Prediction, Normalization, Dropout, Regularization, Bagging, Optimizers, Recurrent Neural Networks (RNN) (highlighted), First Example, Second Example, Financial Price Series, Financial Return Series, Financial Features, Deep RNNs, Convolutional Neural Networks (CNN), and Reinforcement Learning (RL) in Finance. The main content area shows a code cell with a play button icon and the following Python code:

```
1 import os
2 import random
3 import numpy as np
4 import pandas as pd
5 import tensorflow as tf
6 from pprint import pprint
7 from pylab import plt, mpl
8 plt.style.use('seaborn')
9 mpl.rcParams['savefig.dpi'] = 300
10 mpl.rcParams['font.family'] = 'serif'
11 pd.set_option('precision', 4)
12 np.set_printoptions(suppress=True, precision=4)
13 os.environ['PYTHONHASHSEED'] = '0'
14
15 def set_seeds(seed=100):
16     random.seed(seed)
17     np.random.seed(seed)
18     tf.random.set_seed(seed)
19 set_seeds()
20
21 a = np.arange(100)
```

Python in Google Colab (Python101)

CO python101.ipynb ☆

File Edit View Insert Runtime Tools Help Saving...

Comment Share Settings A

Connect Editing

Table of contents

- Deep Learning (DL) in Finance
 - Dense Neural Networks (DNN)
 - Baseline Prediction
 - Normalization
 - Dropout
 - Regularization
 - Bagging
 - Optimizers
 - Recurrent Neural Networks (RNN)
 - First Example
 - Second Example
 - Financial Price Series
 - Financial Return Series
 - Financial Features
 - Deep RNNs
 - Convolutional Neural Networks (CNN)**
 - Reinforcement Learning (RL) in Finance

+ Code + Text

Convolutional Neural Networks (CNN)

```
1 import os
2 import math
3 import numpy as np
4 import pandas as pd
5 from pylab import plt, mpl
6 plt.style.use('seaborn')
7 mpl.rcParams['savefig.dpi'] = 300
8 mpl.rcParams['font.family'] = 'serif'
9 os.environ['PYTHONHASHSEED'] = '0'
10
11 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
12 symbol = 'EUR='
13 data = pd.DataFrame(pd.read_csv(url, index_col=0,
14                               parse_dates=True).dropna()[symbol])
15 data.info()
16 lags = 5
17 features = [symbol, 'r', 'd', 'sma', 'min', 'max', 'mom', 'vol']
18
19 def add_lags(data, symbol, lags, window=20, features=features):
20     cols = []
21     df = data.copy()
22     df.dropna(inplace=True)
23     df['r'] = np.log(df / df.shift(1))
24     df['sma'] = df[symbol].rolling(window).mean()
25     df['min'] = df[symbol].rolling(window).min()
```

Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share



A

Table of contents

- Financial Features
- Deep RNNs
- Convolutional Neural Networks (CNN)
- Reinforcement Learning (RL) in Finance**
- Reinforcement Learning (RL)
 - CartPole Environment
 - Dimensionality Reduction
 - Action Rule
 - Total Reward per Episode
 - Simple Learning
 - Testing the Results
 - DNN Learning
 - Q Learning
 - Finance Environment
 - Improved Finance Environment
 - Improved Financial QL Agent

+ Code + Text

Connect

Editing

Reinforcement Learning (RL) in Finance

- Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.
- Github: <https://github.com/yhilpisch/aiif/>

Reinforcement Learning (RL)

```
1 import os
2 import math
3 import random
4 import numpy as np
5 import pandas as pd
6 from pylab import plt, mpl
7 plt.style.use('seaborn')
8 mpl.rcParams['savefig.dpi'] = 300
9 mpl.rcParams['font.family'] = 'serif'
10 np.set_printoptions(precision=4, suppress=True)
11 os.environ['PYTHONHASHSEED'] = '0'
```

CartPole Environment

```
[ ] 1 import gym
     2
```

Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook interface. At the top left, the notebook is titled 'python101.ipynb' with a star icon. Below the title is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by the text 'All changes saved'. On the top right, there are icons for 'Comment', 'Share', a settings gear, and a user profile icon labeled 'A'.

On the left side, there is a 'Table of contents' panel with a search icon and a close button. The table of contents lists various topics, with 'Improved Financial QL Agent' highlighted at the bottom. Other items include 'Deep RNNs', 'Convolutional Neural Networks (CNN)', 'Reinforcement Learning (RL) in Finance', 'Reinforcement Learning (RL)', 'CartPole Environment', 'Dimensionality Reduction', 'Action Rule', 'Total Reward per Episode', 'Simple Learning', 'Testing the Results', 'DNN Learning', 'Q Learning', 'Finance Environment', and 'Improved Finance Environment'.

The main area of the notebook shows a code cell titled '+ Code + Text' with a play button icon. The code defines an 'FQLAgent' class with an '__init__' method and a '_build_model' method. The code is as follows:

```
1 from collections import deque
2
3 class FQLAgent:
4     def __init__(self, hidden_units, learning_rate, learn_env, valid_env):
5         self.learn_env = learn_env
6         self.valid_env = valid_env
7         self.epsilon = 1.0
8         self.epsilon_min = 0.1
9         self.epsilon_decay = 0.98
10        self.learning_rate = learning_rate
11        self.gamma = 0.95
12        self.batch_size = 128
13        self.max_treward = 0
14        self.trewards = list()
15        self.averages = list()
16        self.performances = list()
17        self.aperformances = list()
18        self.vperformances = list()
19        self.memory = deque(maxlen=2000)
20        self.model = self._build_model(hidden_units, learning_rate)
21
22    def _build_model(self, hu, lr):
23        model = Sequential()
24        model.add(Dense(hu, input_shape=(
25            self.learn_env.lags, self.learn_env.n_features),
26            activation='relu'))
```

Summary

- **Deep Learning (DL) in Finance**
 - **Dense Neural Networks (DNN)**
 - **Recurrent Neural Networks (RNN)**
 - **Convolutional Neural Networks (CNN)**
- **Reinforcement Learning (RL) in Finance**
 - **Q Learning (QL)**
 - **Improved Finance Environment**
 - **Improved Financial QL Agent**

References

- Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media, <https://github.com/yhilpisch/aiif> .
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.
- Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.
- Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (2020), "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019." Applied Soft Computing 90 (2020): 106181.
- Min-Yuh Day (2022), Python 101, <https://tinyurl.com/aintpuppython101>