LECTURE 1: INTRODUCTION

University of Washington, Seattle

Fall 2025

OUTLINE

Part 1: Welcome to EEP 596

- Instruction team
- Goal of the course
- Instruction format
- Course materials
- Instruction components
- Schedule and Canvas page
- Syllabus and Grading

Part 2: Deep Learning and Neural Networks

- Definition(s) of Deep Learning
- Model and data
- Fixed vs Learned model
- Classical machine learning methods
- Artificial neural network and Brain

Part 3: Regression

- Regression problem
- Polynomial fit: Linear regression
- Improving linear regression

PART 1:

WELCOME TO EEP 596

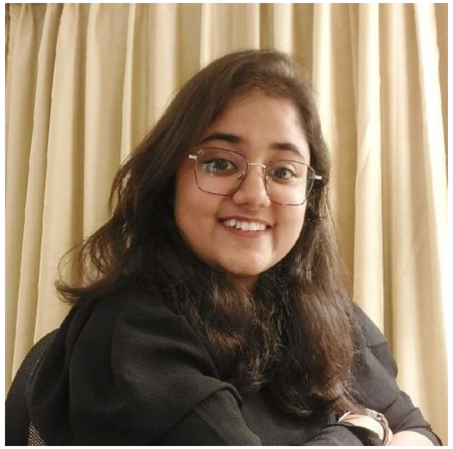
INSTRUCTION TEAM

Instructor:
Jimin Kim
UW NeuroAI Lab
jk55@uw.edu

Research Interests
Computational Neuroscience and AI

TA:
Yang Zheng
UW NeuroAI Lab
jk55@uw.edu

Research Interests
Neural network efficiency and Generative AI



TA:
Urvashi Taki
utaki@uw.edu

Research Interests
Deep learning for visual data and LLM

WHAT IS THIS COURSE ABOUT?

Fundamental concepts, skills and applications of deep learning.

Learn **fundamental principles** behind deep learning **architecture** and **training**.

Survey models leading up to current state of the art methods.

Apply models to **real-world problems and datasets**: Labeled, Visual, Time-series etc.

Comprehensive introductory course with **heavy emphasis on practical aspects.**

Uses **Python 3 and PyTorch** as main programming tools.

INSTRUCTION (Tentative)

(630pm – 640pm) Introduction and announcements

(640pm – 700pm) Canvas quiz discussion

(700pm – 820pm) Lecture: Theory

(820pm – 830pm) Break

(830pm – 910pm) Lecture: Practice (with live coding examples)

(910pm – 920pm) Lab assignment introduction

(920pm – 950pm) In-person Lab session

COURSE MATERIALS

Weekly Assignment page (Canvas)

1) Lecture slides/recording (.pdf, Panopto video)

2) Lab slides/recording (.pdf, Panopto video)

3) In-lab examples (.ipynb)

4) Lab report starter templates (.ipynb)

Additional Resources (Canvas)

Deep Learning (Ian Goodfellow)

Neural Networks and Deep Learning (Michael Nelson)

PyTorch Github Tutorials (https://github.com/yunjey/pytorch-tutorial)

INSTRUCTION COMPONENTS

1. Lecture (Theory)	→	Theoretical Concepts
2. Canvas quiz	→	Theoretical Concepts Feedback
3. Lecture (Lab)	→	Practical Concepts
4. Code examples	→	Code Implementations
5. Lab assignment	→	Real-world Applications

SCHEDULE

Weekly Instruction:

Mon 6:30 PM – 9:50 PM (ECE 037)

Office Hour

Jimin: TBA

Yang: TBA

Urvashi: TBA

CANVAS PAGE

EE P 596 Au 24: Practical Introduction to Deep Learning Applications and Theory

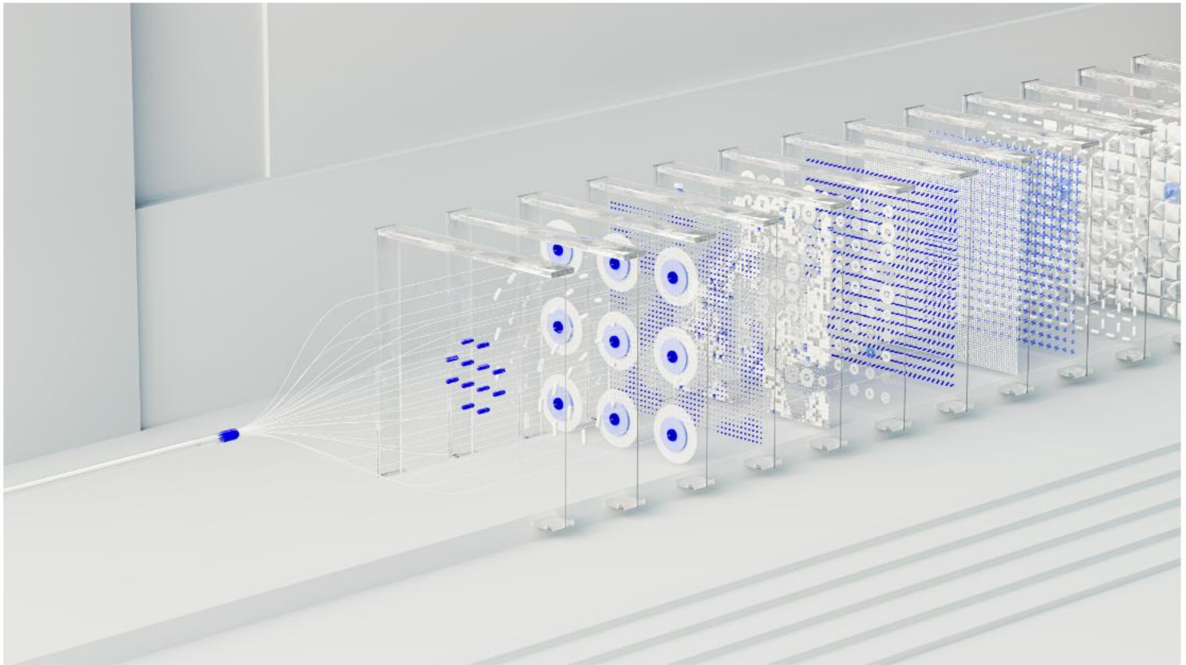


Image credit: Google DeepMind

Welcome to EEP 596!

Welcome to EEP 596 "Practical Introduction to Deep Learning Applications and Theory"! This is a graduate level course aiming to provide **fundamental skills, concepts, and applications of deep learning** and neural networks for the investigation of complex datasets with heavy emphasis on hands-on practices.

Home & Syllabus tabs for Detailed Course Info

Announcements tab for important course announcements

Discussions tab for student – student student – instructor discussions

Panopto tab for Post-lecture recordings

CANVAS ASSIGNMENT PAGE

Lab 1 Report

LAB 1: PYTHON FOUNDATION

Associated lectures

(Theoretical Concepts)
Lecture 1 - Introduction (Panopto recording, Slides)

(Practice)
Lab 1 - Python Foundation (Panopto recording, Slides)

Lab materials

Lab report guidelines (IMPORTANT: READ THIS FIRST!)

Lab 1 Examples notebook
ipynb file containing all the examples discussed in the lab videos. Use this to play with the examples yourself.

Lab 1 Report Template
Zip file containing lab report template ipynb + exercise image files (You need image files to load problems in ipynb).
Unzip the file using windows or 7-zip.

Read **lab report guidelines** before starting your first assignment

Your lab report = Filled in report template notebook **(.ipynb)**

SYLLABUS (Tentative)

Neural network fundamentals and feed-forward networks (09/29 – 10/13)

(W1) Intro to ANNs/Regression | Setting up Python environment/Regression
(W2) Intro to Deep Learning and MLP | PyTorch Introduction/Classification with MLP
(W3) Convolutional Neural Networks | CNNs in PyTorch

Sequence models (10/20 – 10/27)

(W4) Recurrent Neural Networks | RNNs in PyTorch
(W5) LSTM, GRU, Encoder-Decoder architectures | LSTM, GRU, Encoder-Decoder in PyTorch

Generative models (11/03 – 11/17)

(W6) Attention and Transformers | Text processing with Transformers
(W7) Advanced CNNs | Image classification with ResNet
(W8) Generative Adversarial Networks | Image generation with GAN

Advanced models (11/24)

(W9) Deep Reinforcement Learning | Control strategies with DeepRL

Final project (12/01 – 12/08)

(W10) Project week | Selected Projects
(Finals week) Project presentation

GRADING

(i): **Canvas Quiz (20%)** – individual, weekly

Evaluated on concept understanding and correctness of the responses

Goal: Fundamental concepts feedback

(ii): **Lab Assignment (40%)** – individual, weekly

Evaluated on code organization, documentation and task completion

Goal: Implementation and training of Deep Learning models on real datasets

(iii): **Final Project (40%)** – individual/team

Evaluated on project planning, presentation and code implementation

Goal: Solve an original, real-world project of your choice using deep learning

Q/A

PART 2:

WHAT IS DEEP LEARNING?

Definition(s) of Deep Learning

IBM: Subset of machine learning that uses multilayered neural networks, called deep neural networks, to **simulate** the complex decision-making power of the human brain.

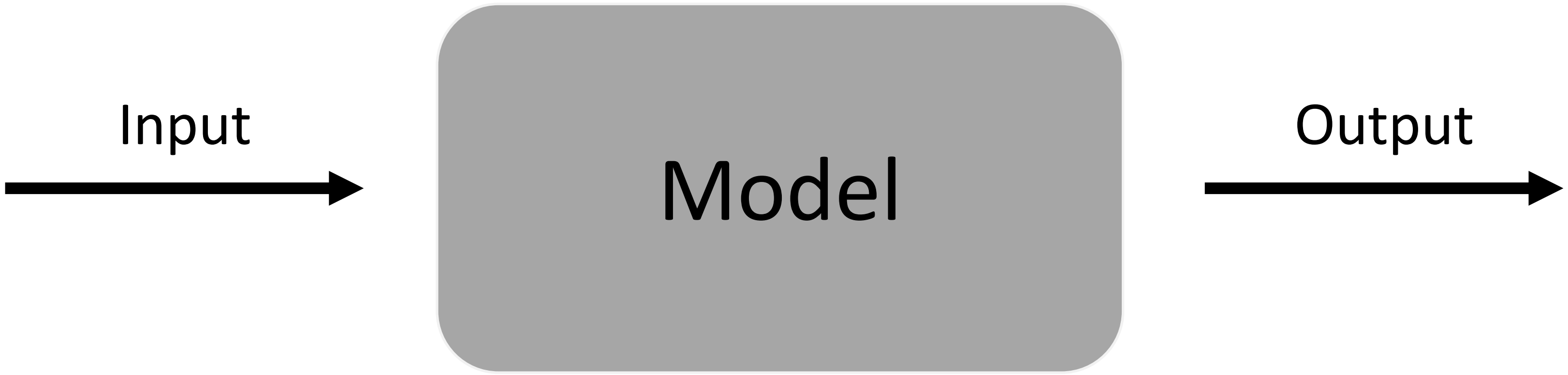
AWS: A method in artificial intelligence (AI) that teaches computers to **process** data in a way that is inspired by the human brain.

Ian Goodfellow: A form of machine learning that enables computers to **learn** from experience and understand the world in terms of a hierarchy of concepts.

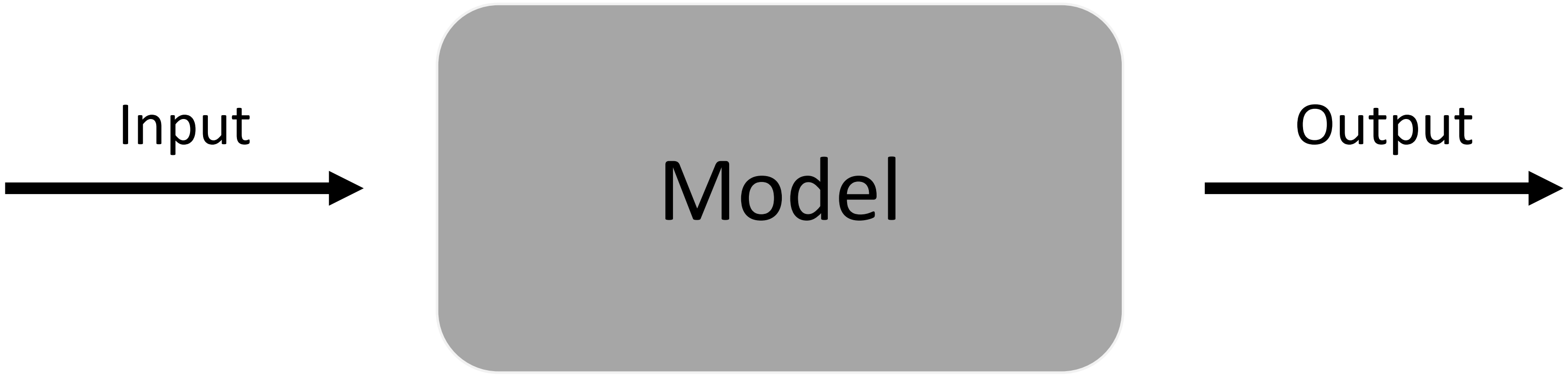
Yann LeCun: Constructing networks of parameterized functional modules & **training** them from examples using gradient-based optimization.

Geoffrey Hinton: An approach to machine learning that involves **computational models** with multiple processing layers to learn representations of data with multiple levels of abstraction

Deep Learning Foundation: Model and Data



Model and Data



System
Algorithms
Pipeline
⋮

Model and Data

Input → **Model** → Output

Input	Model	Output
Signal	System	Signal
Multi-signal	Algorithms	Multi-signal
Time dependent	Pipeline	Time dependent
Stationary	⋮	Stationary
⋮		Score
		Input characteristics
		⋮

Model and Data

Input

Signal
Multi-signal
Time dependent
Stationary
⋮

Model

System
Algorithms
Pipeline
⋮

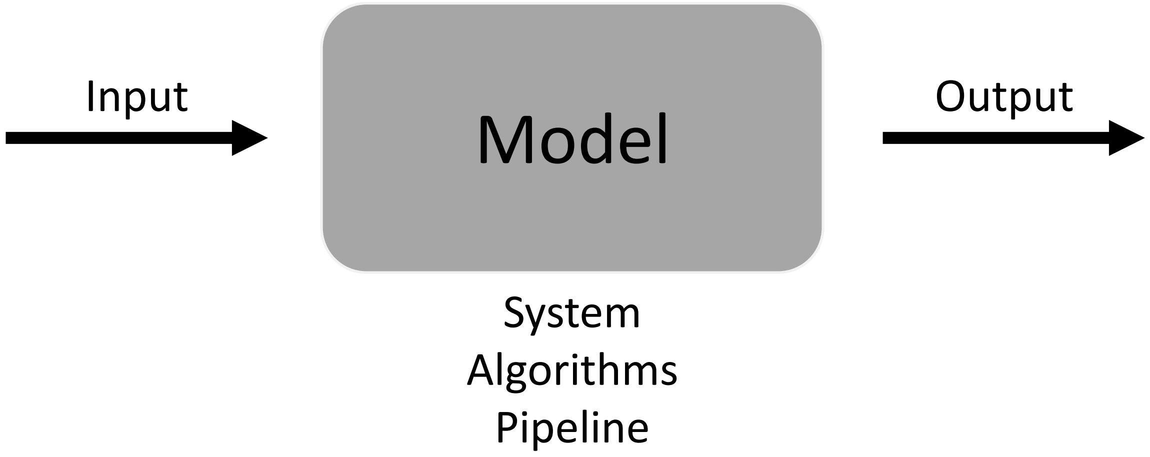
Output

Signal
Multi-signal
Time dependent
Stationary
Score
Input characteristics
⋮

Model = Function of the input

Model examples

- LTI systems
  - Convolution, Fourier, Z-transform, Laplace transform
- Non-LTI systems
  - Non-linear transform
- Data fitting models
  - Linear regression
  - Polynomial regression
- Scoring models
- Classification models



Fixed vs Learned model

Fixed model

$$x_n \longrightarrow \boxed{\sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}} \longrightarrow X_k$$

Discrete Fourier Transform
(All parameters are known)

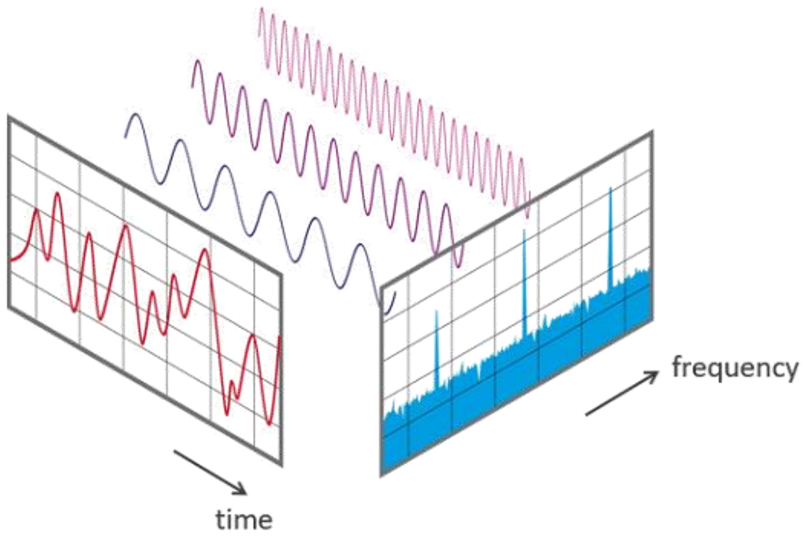


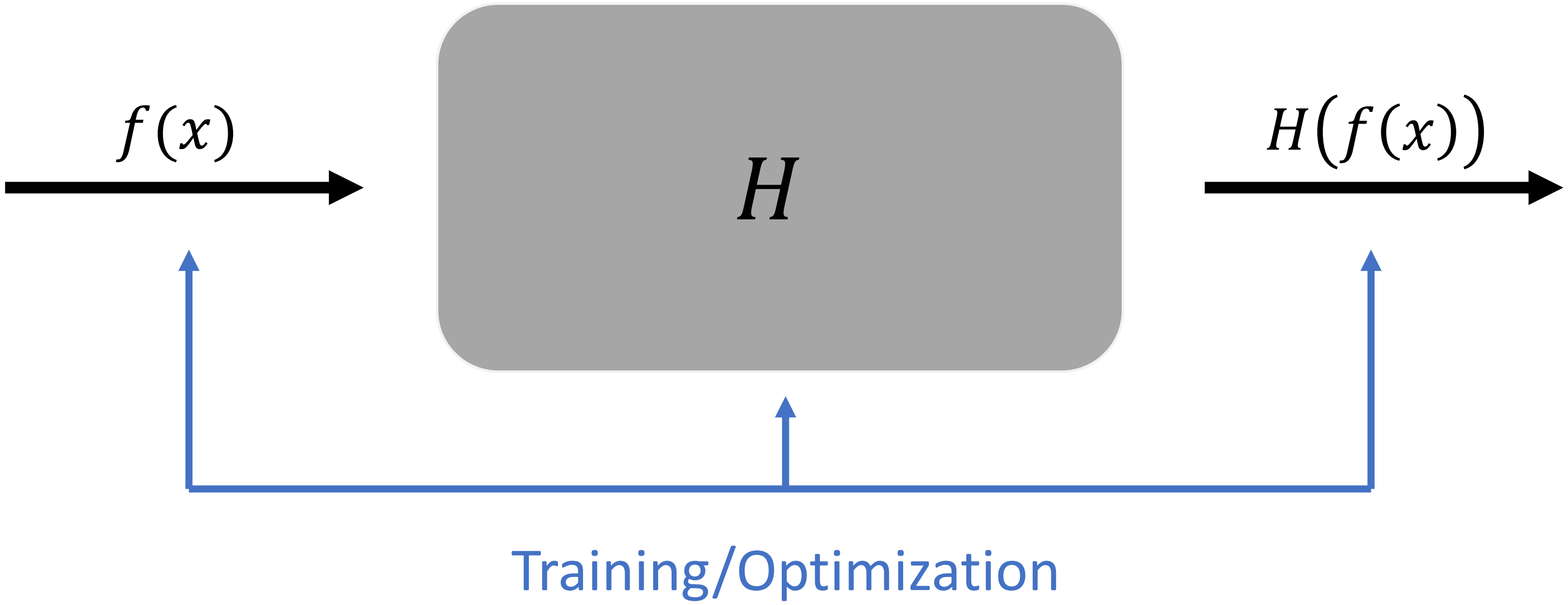
Image credit: All about circuits

Learned model

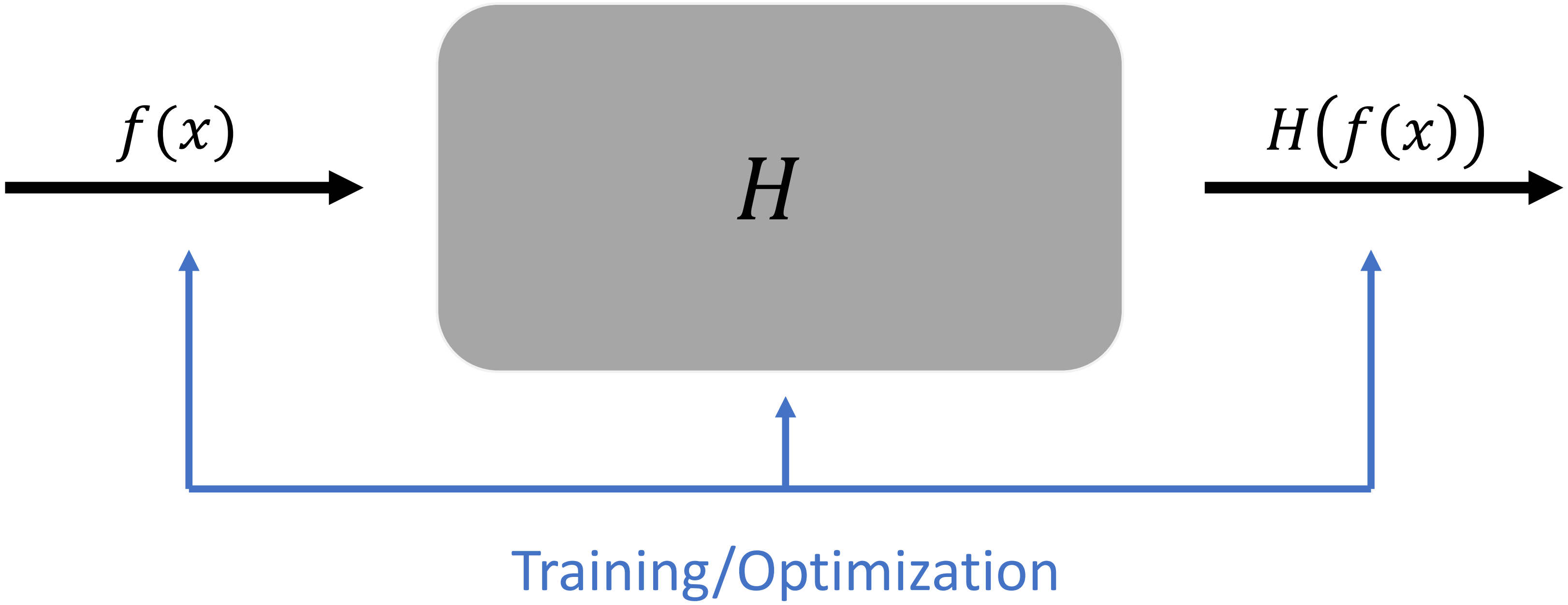
$$x_n \longrightarrow \boxed{\sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}} \longrightarrow X_k$$

Discrete Fourier Transform
(One of more parameters e.g., k needs to be learned given X_k and x_n)

Machine Learning =
Learning an Optimal Model for Data

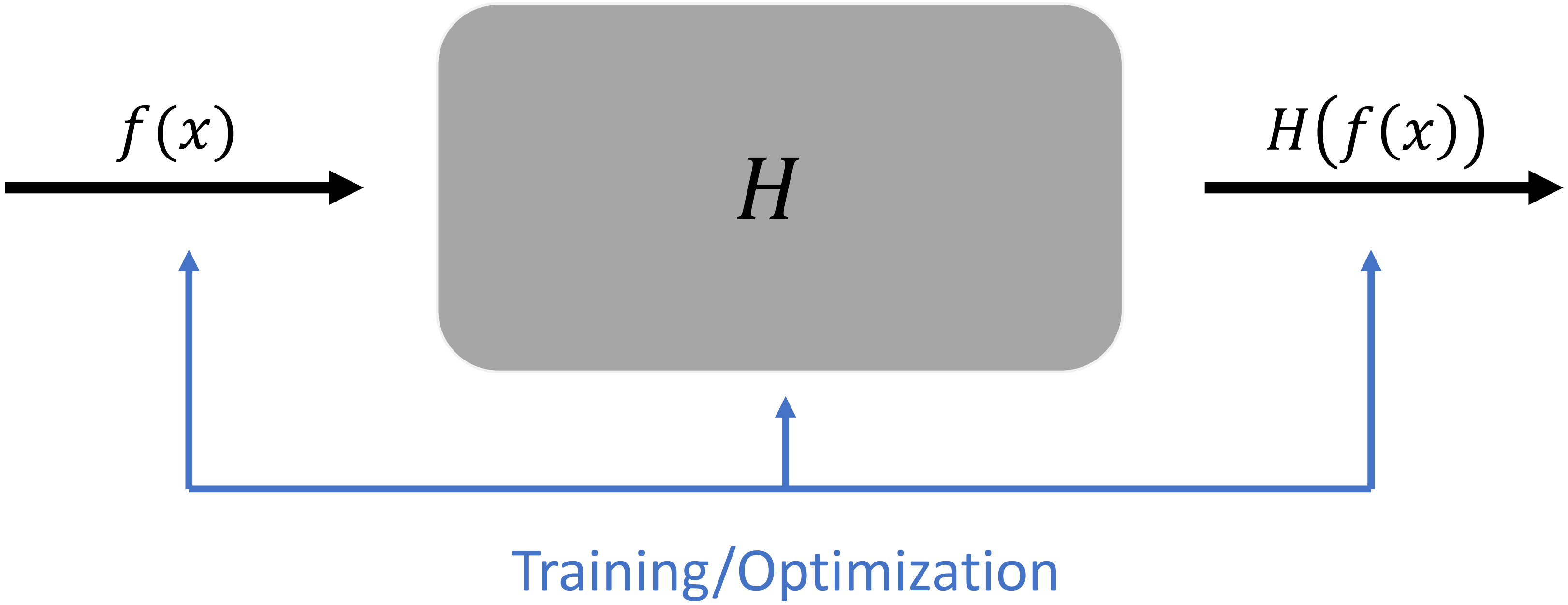


Machine Learning =
Learning an Optimal Model for Data



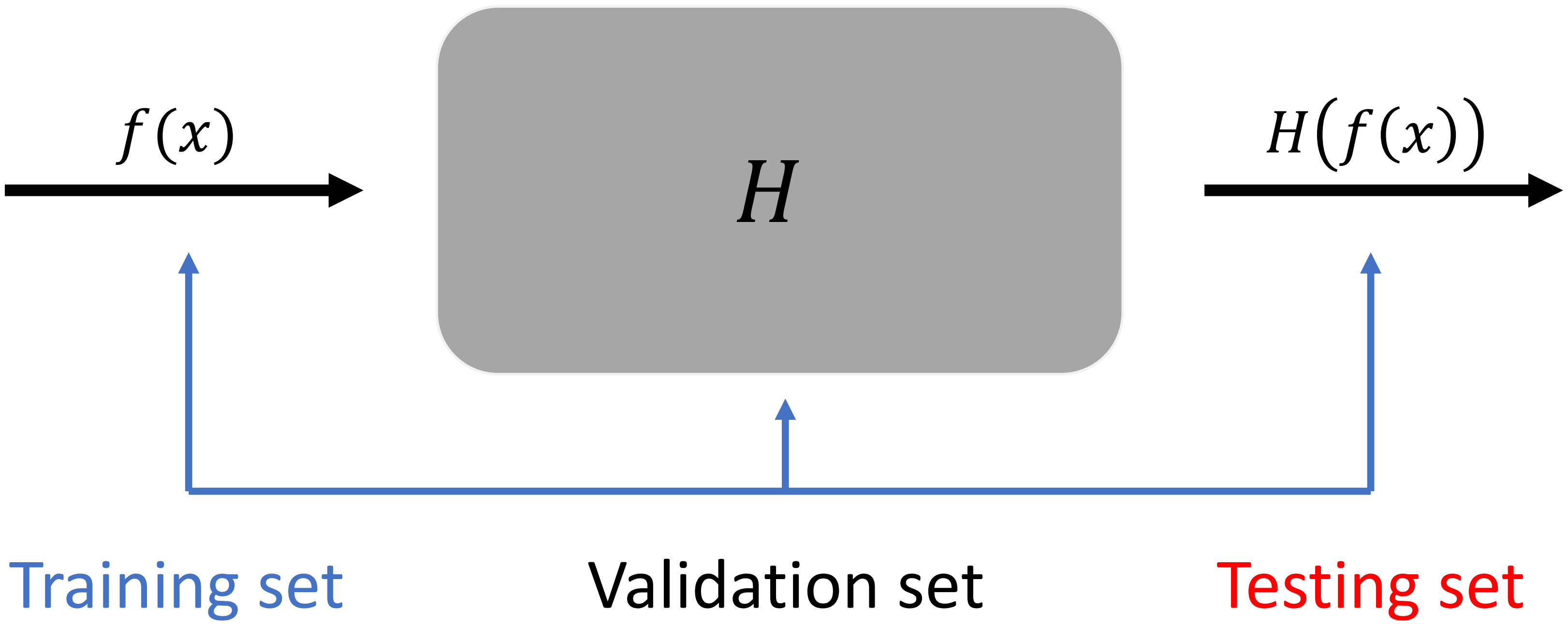
- H expression unknown(Hard)
- H expression fixed with known parameters (Easier)
- H expression can be iteratively updated through Machine Learning algorithms (optimization, training)
- **Deep Learning ⊂ Machine Learning**

Machine Learning =
Learning an Optimal Model for Data

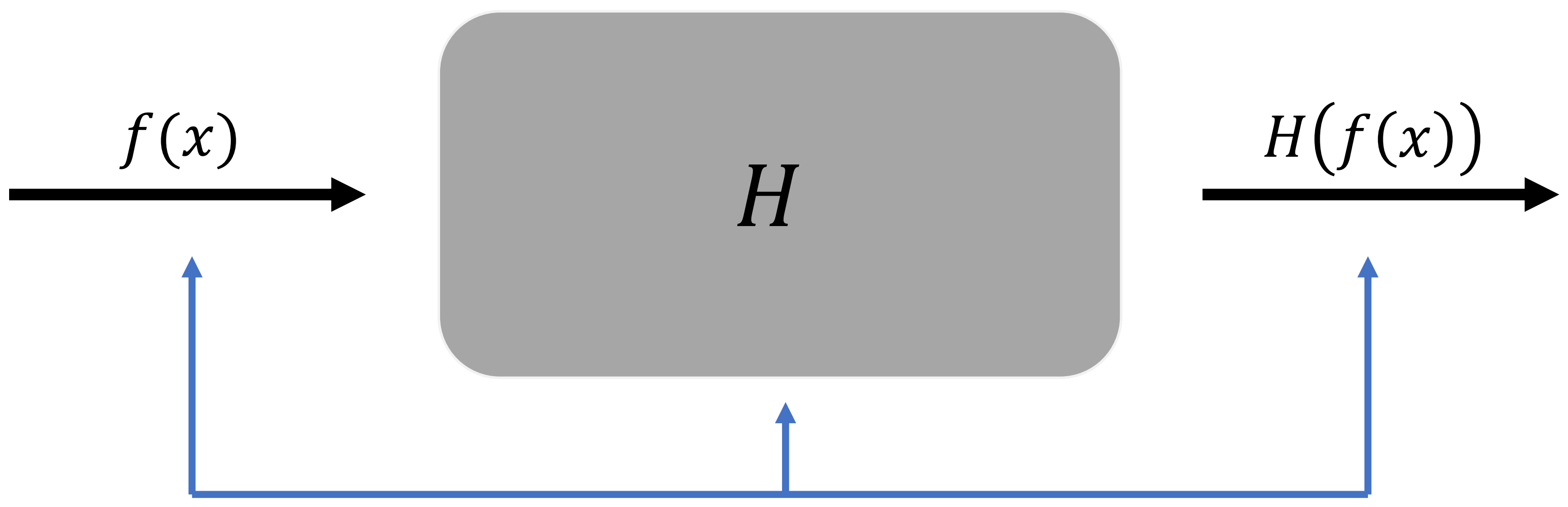


Model optimization is to convert a training set to a model which satisfies the training set – Ilya Sutskever

Three Pillars of Machine Learning Training



Three Pillars of Machine Learning Training



Training set (Seen data)

Used for optimization

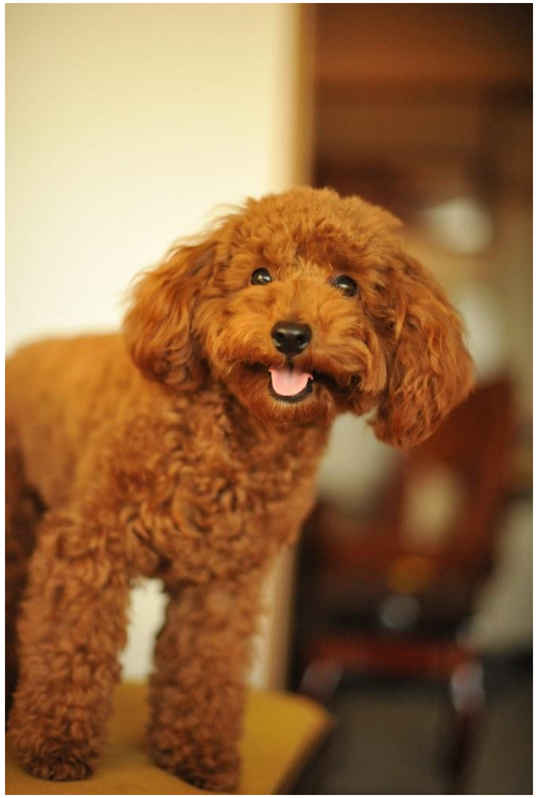
Validation set (Mid-train evaluation)

Used for checking training status/parameter tuning

Testing set (Unseen data)

Un-biased evaluation of the learned model

Classical Machine Learning

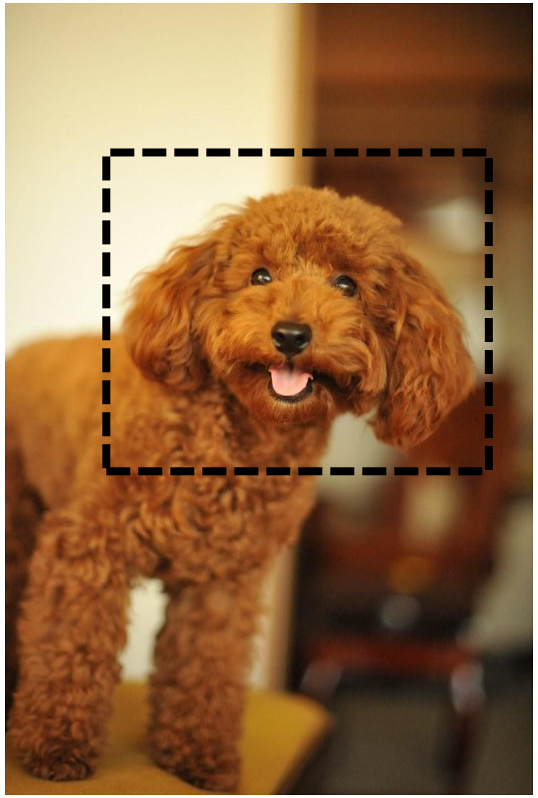


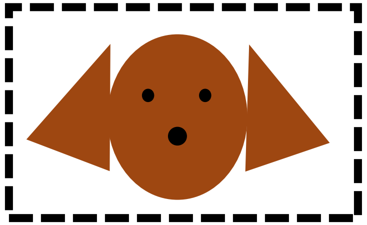
Model

1: Café Poodle

0: Not Café Poodle

Classical Machine Learning





Feature extractions

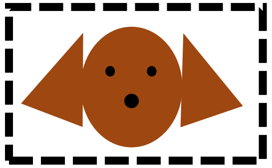
- Round face
- Black eyes and nose
- Triangular ears

Model

1: Café Poodle

0: Not Café Poodle

Limitations of Classical Machine Learning



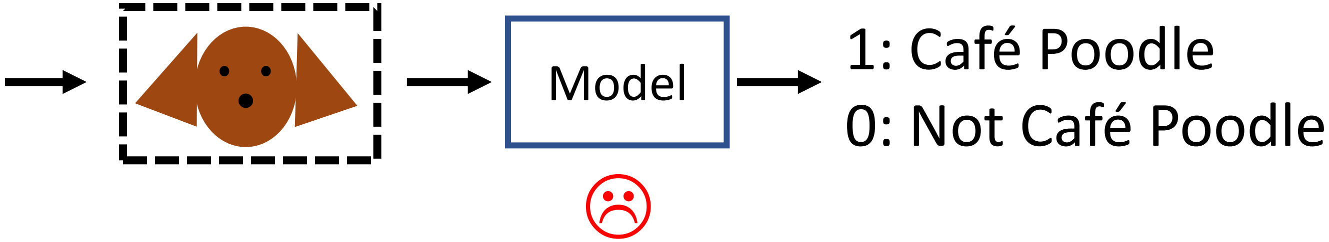
Model

☹

1: Café Poodle

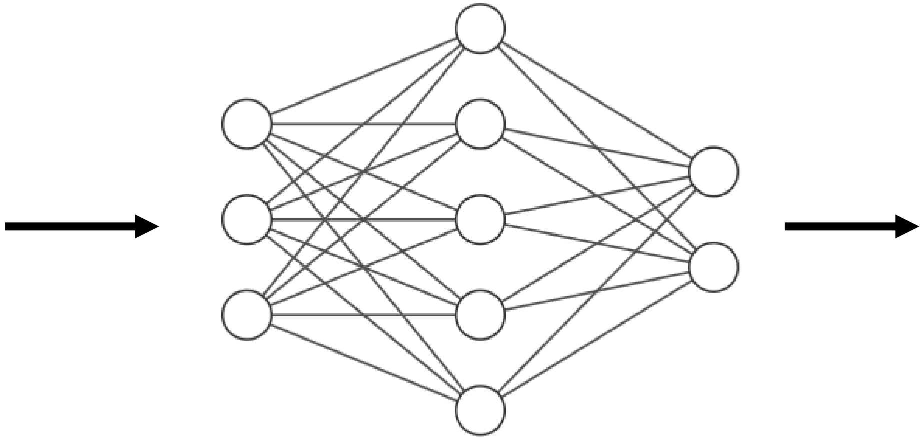
0: Not Café Poodle

Limitations of Classical Machine Learning



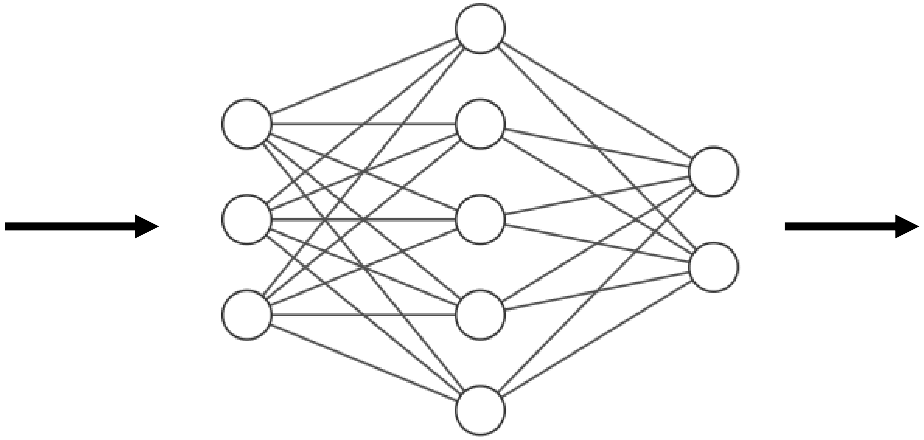
- Feature extractions are often done manually
- Not robust with data
- Hard to scale

Neural Network Model



1: Café Poodle
0: Not Café Poodle

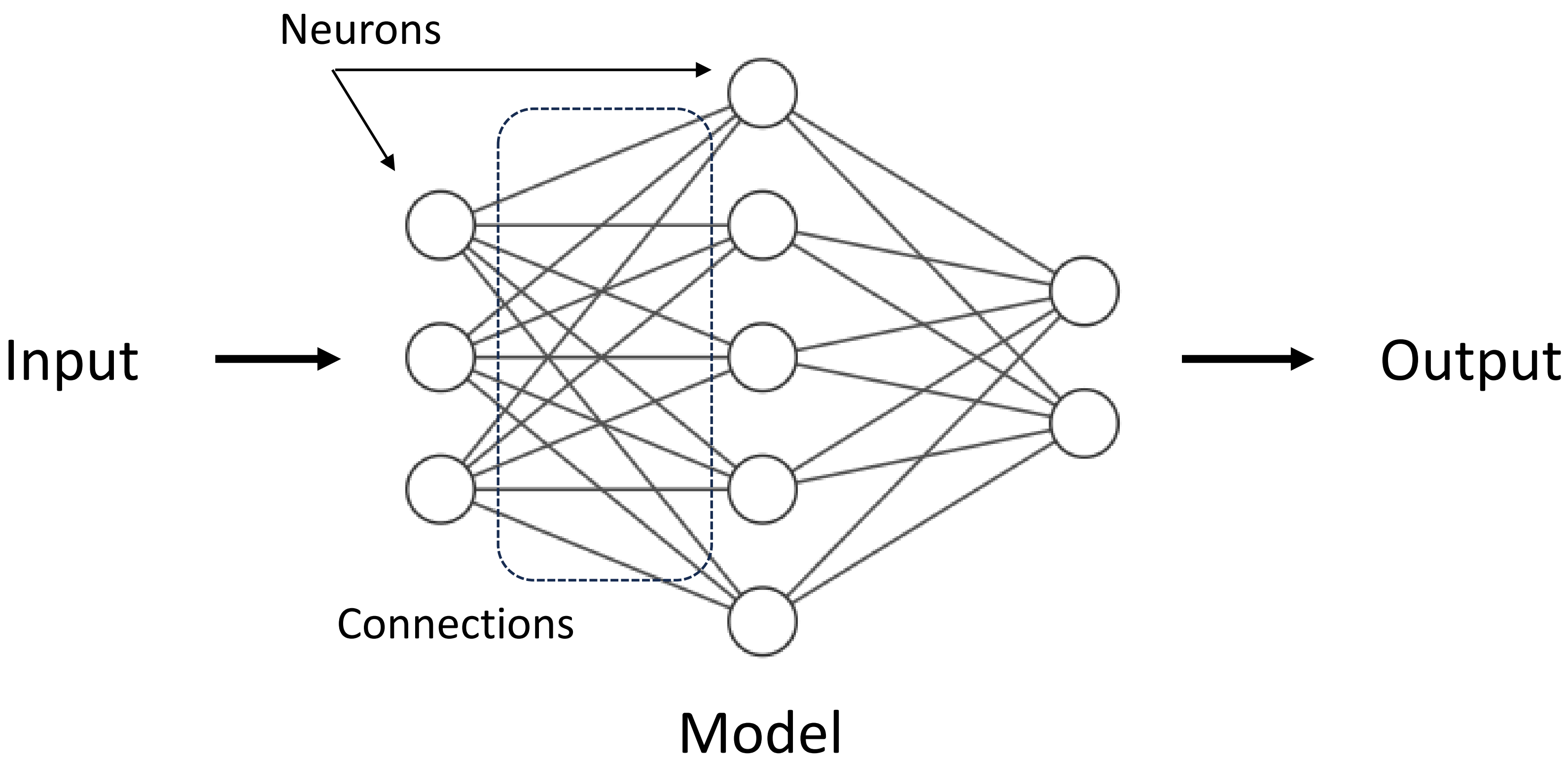
Neural Network Model



1: Café Poodle
0: Not Café Poodle

- Handles (learns) feature extractions
- Robust with data
- Scalable

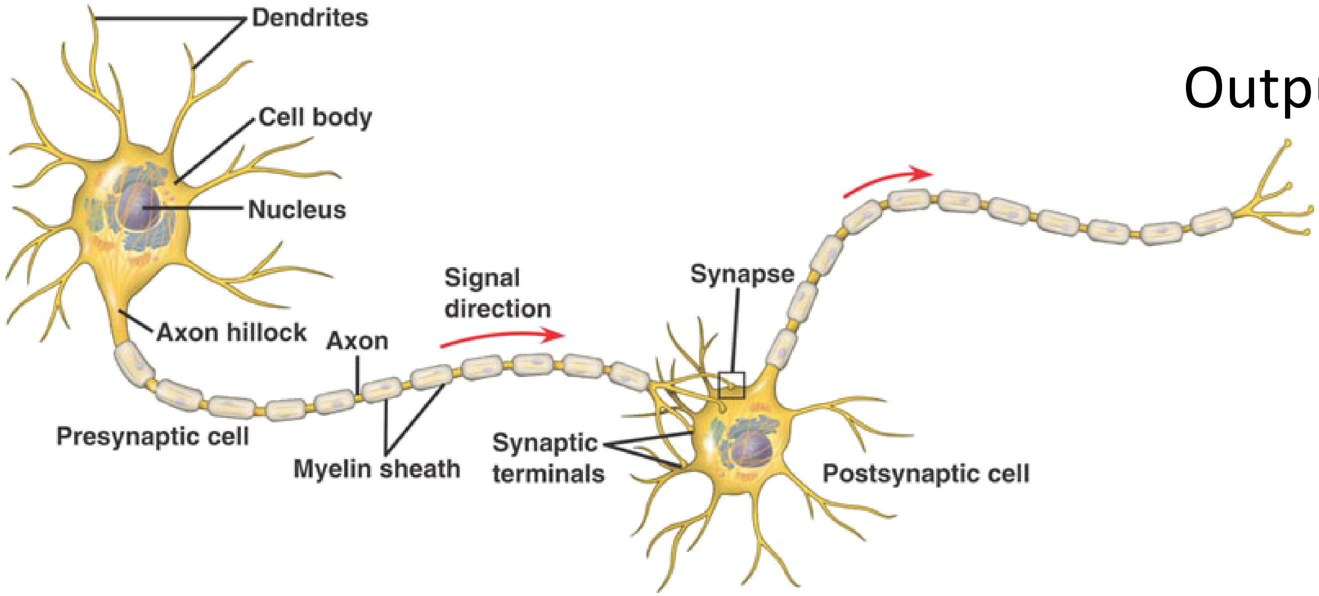
Neural Network Model

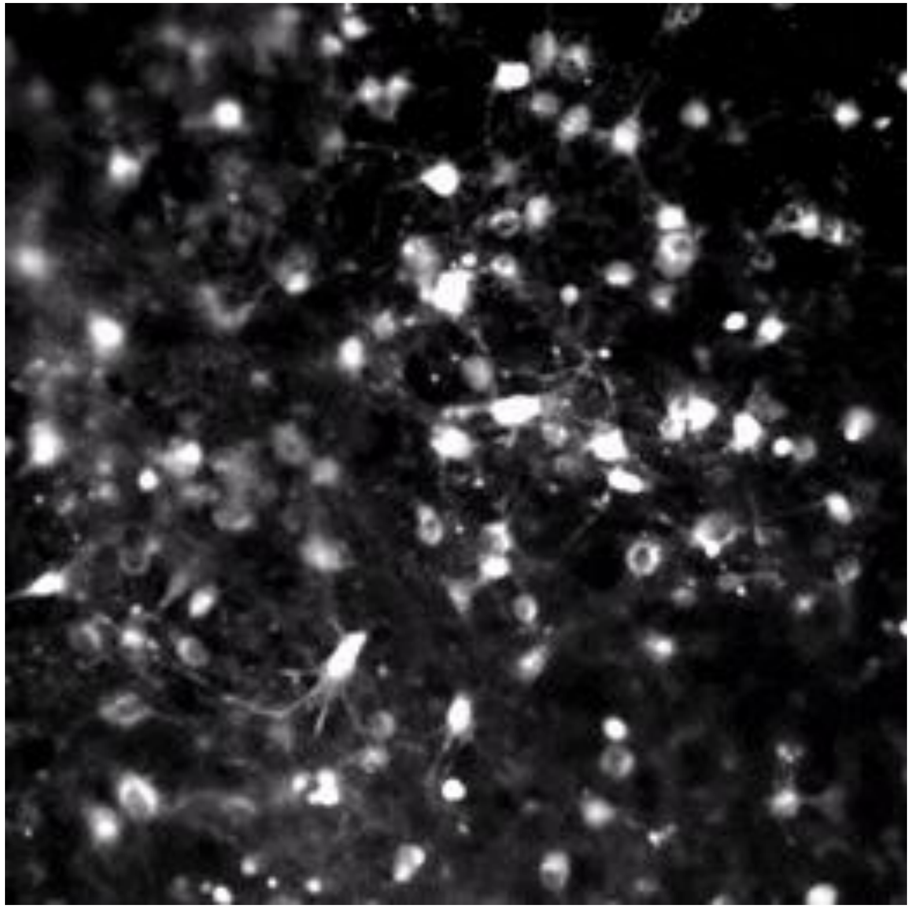


Lessons from brain

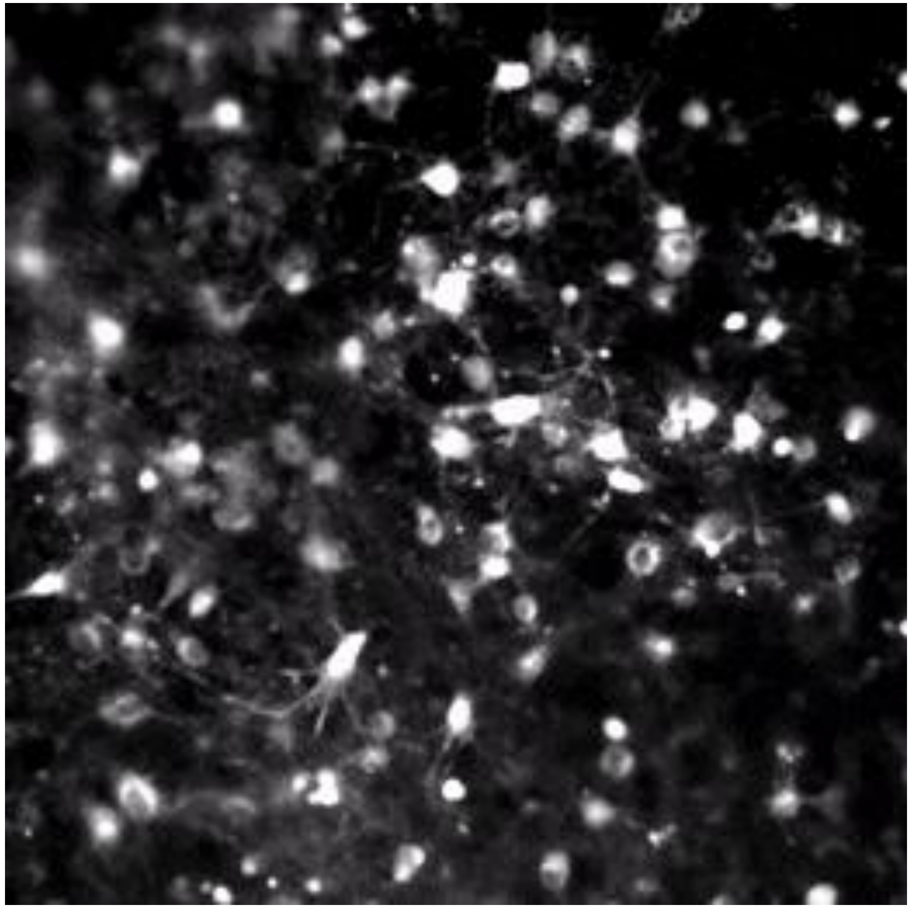
Input

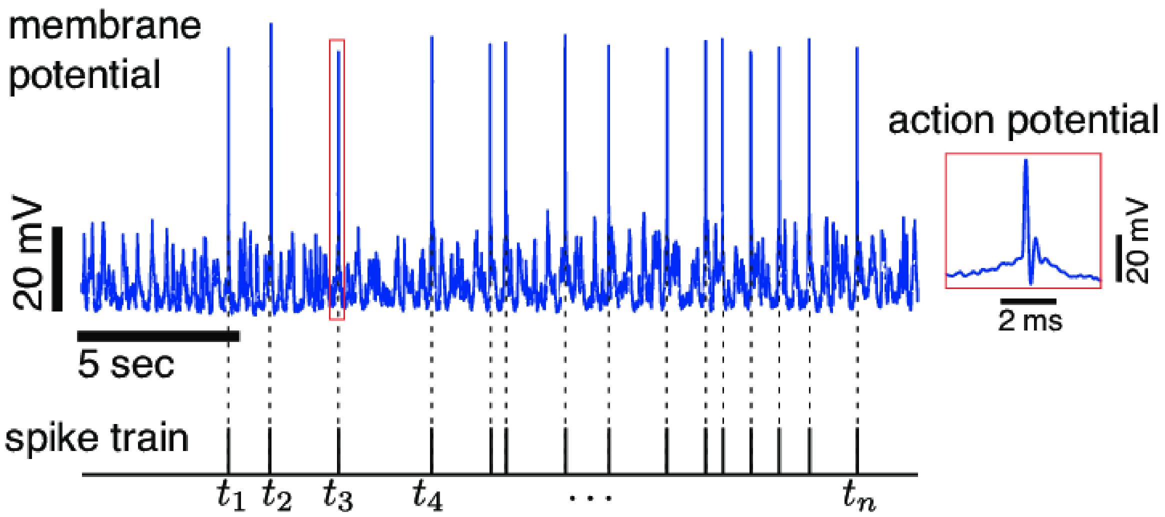
Output



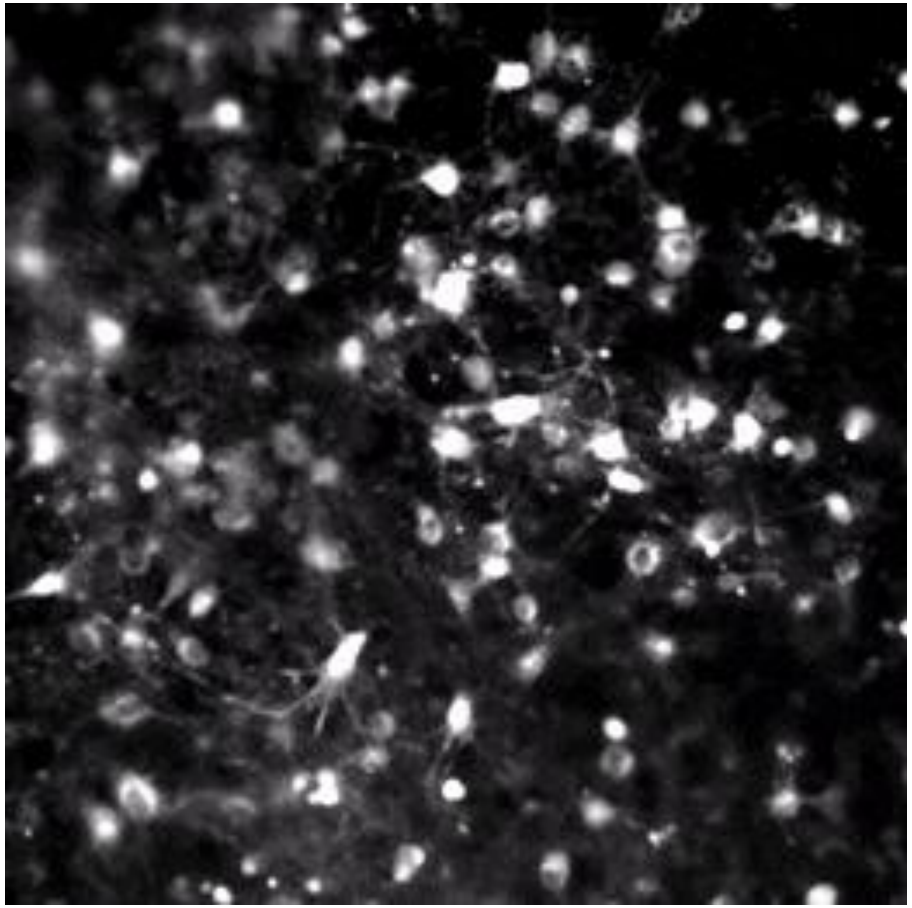


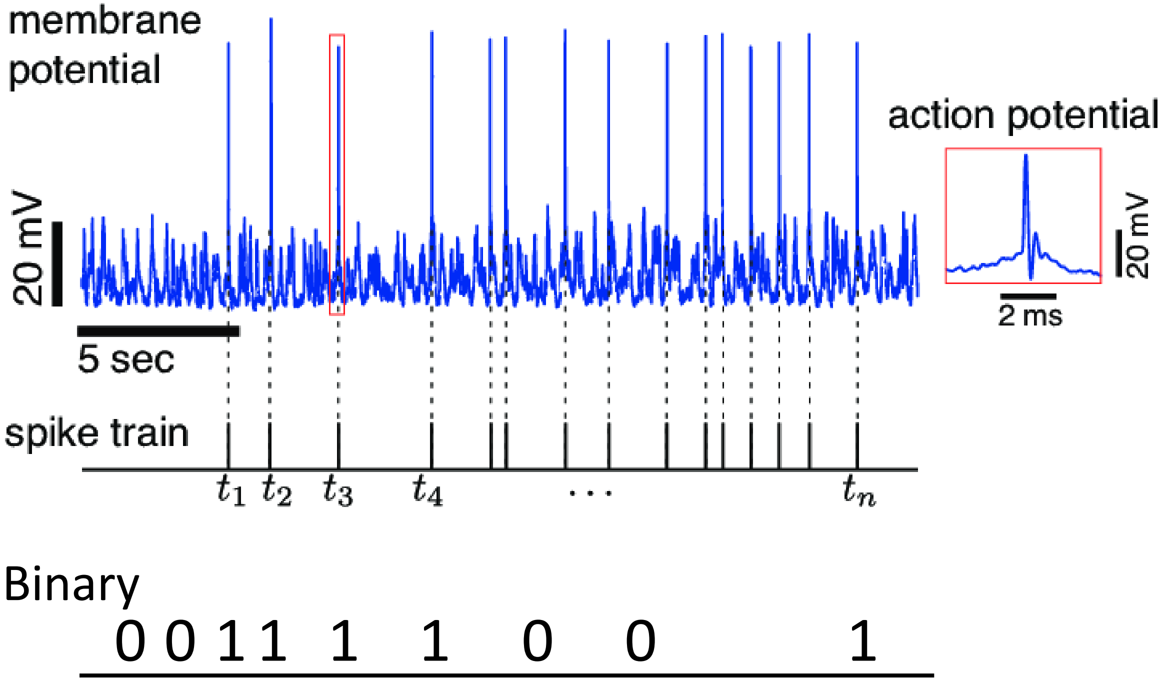
Lessons from brain



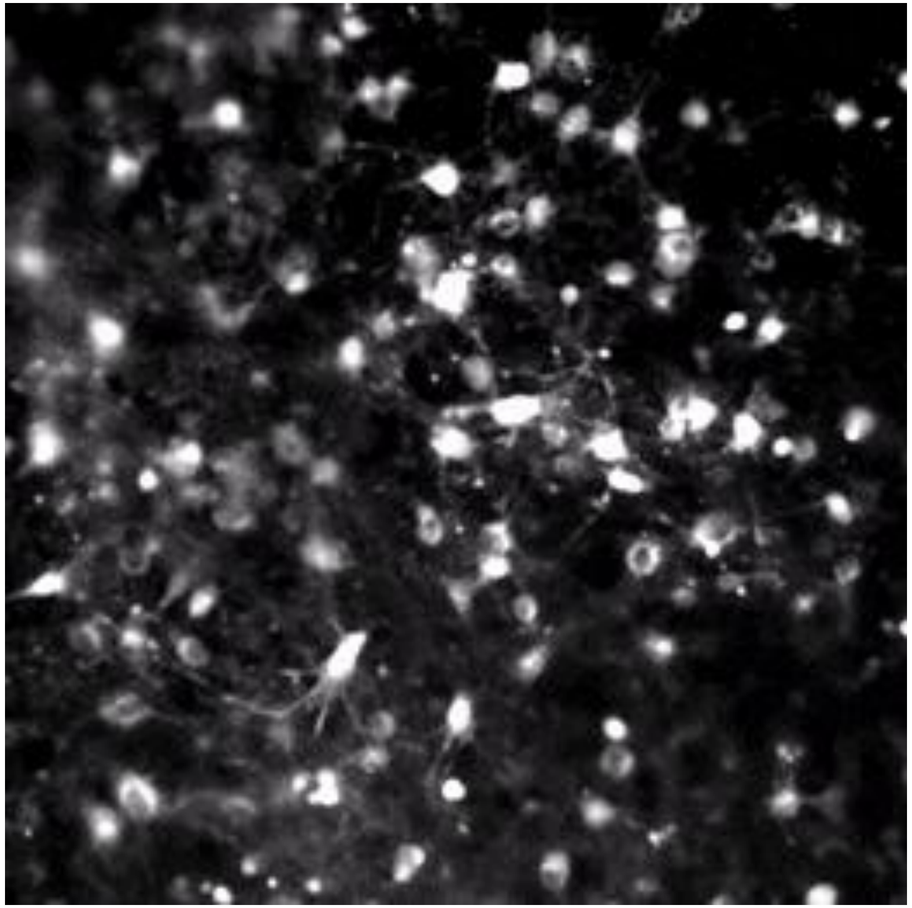


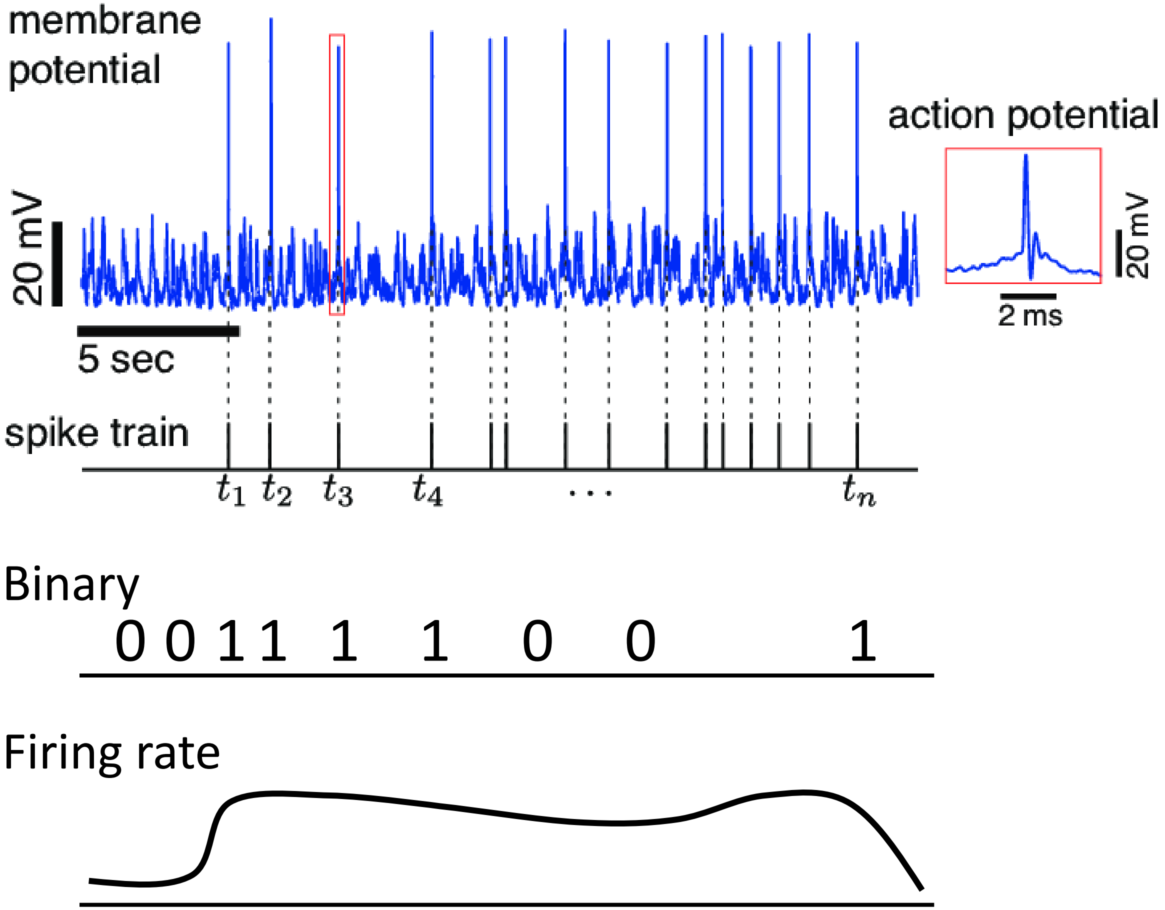
Lessons from brain



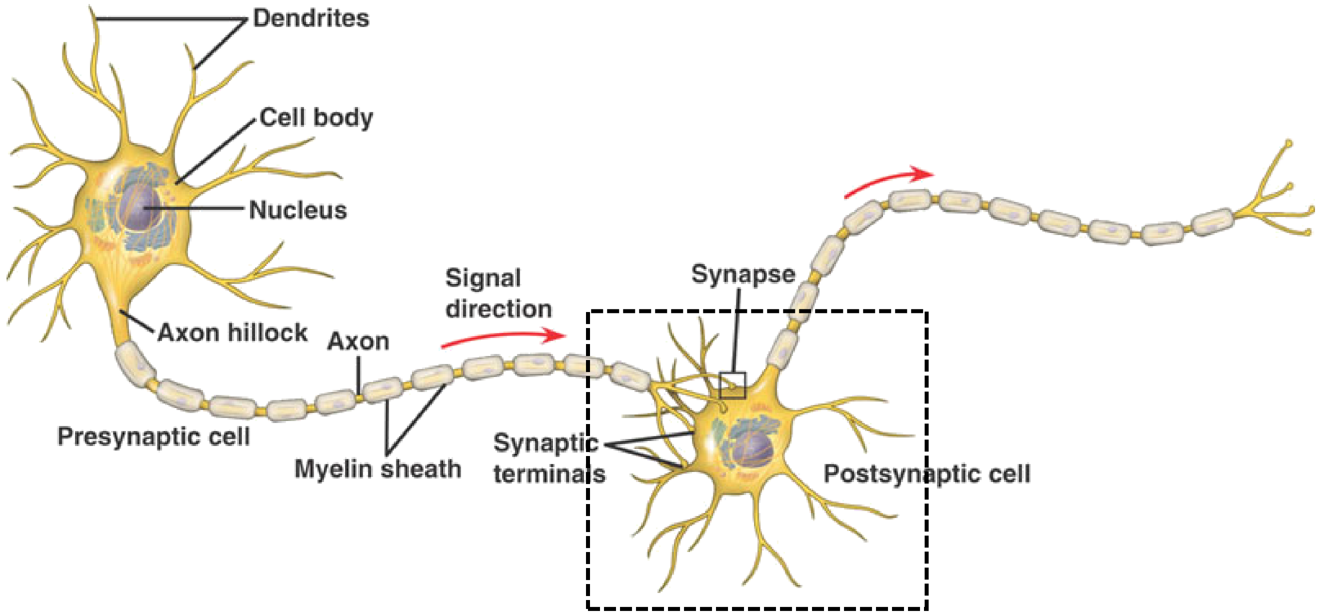


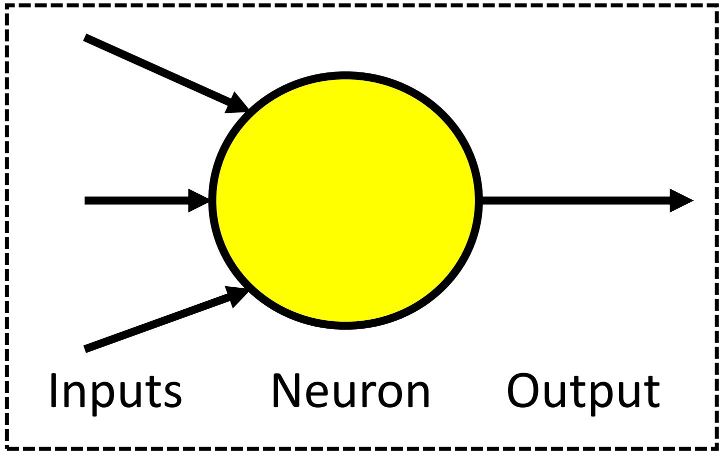
Lessons from brain



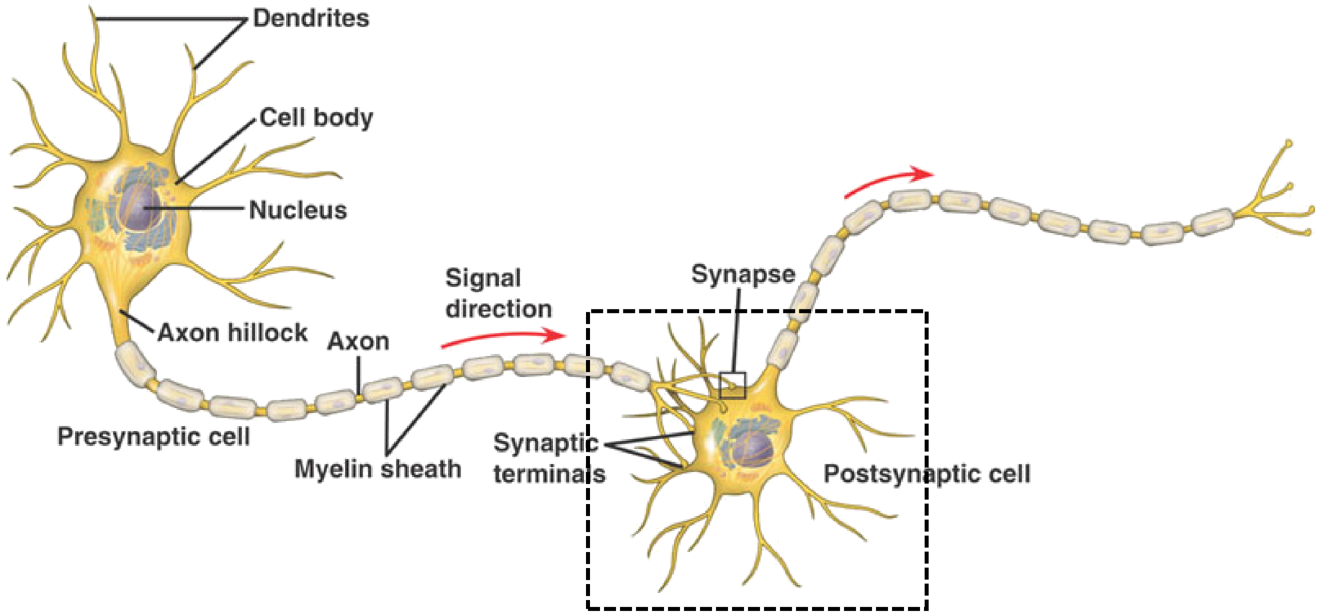


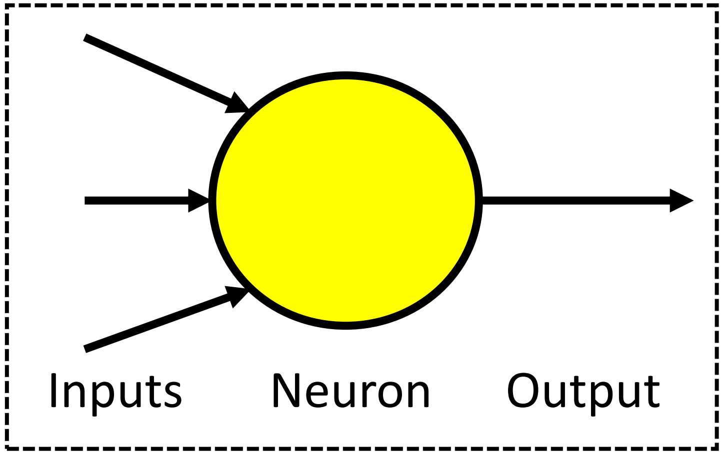
Lessons from brain

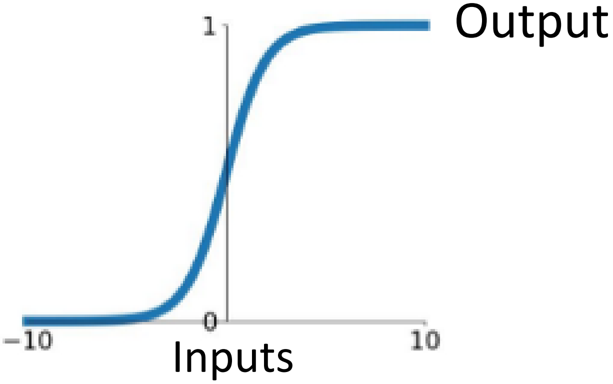




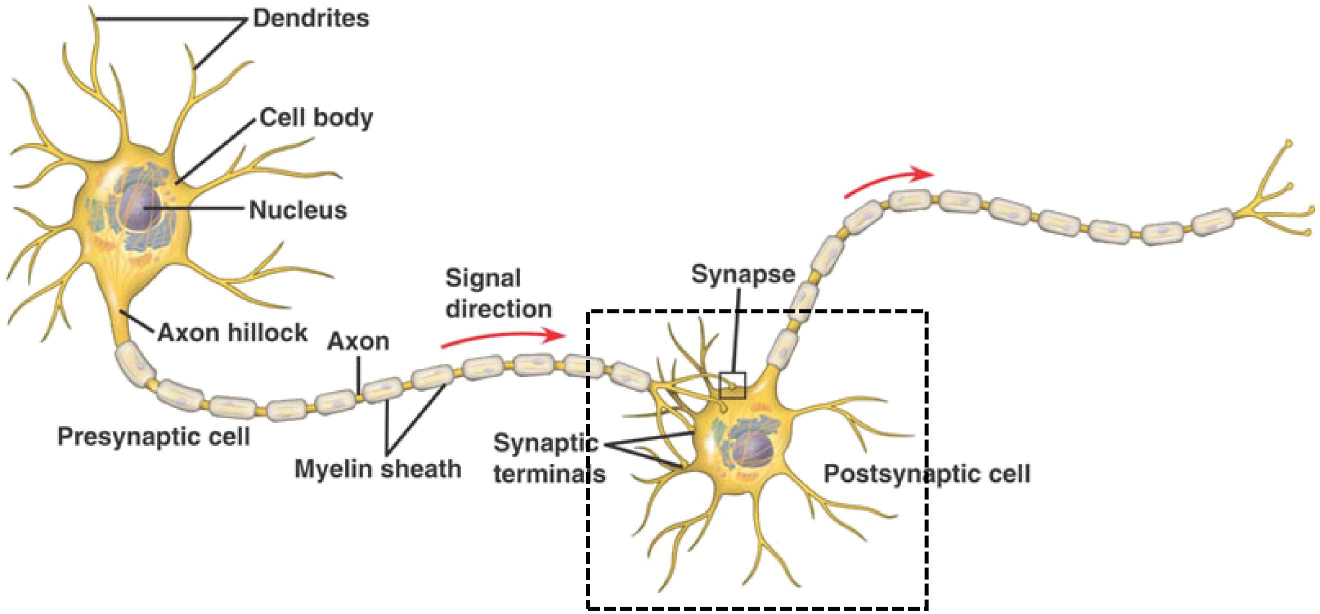
Lessons from brain

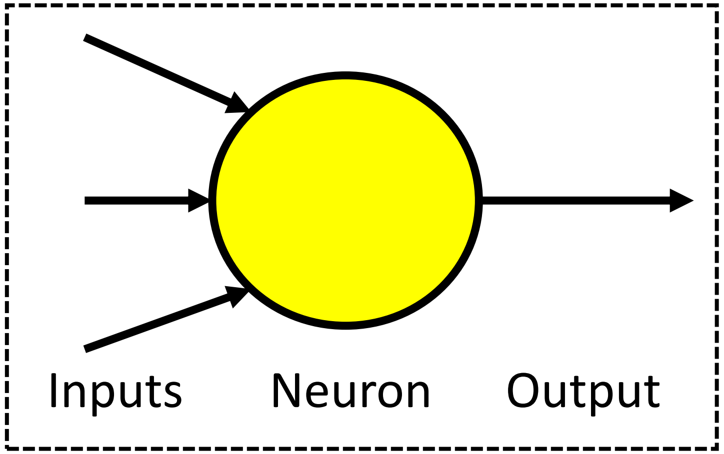


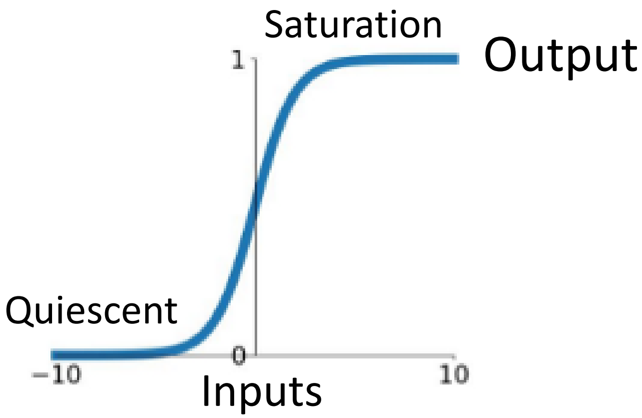




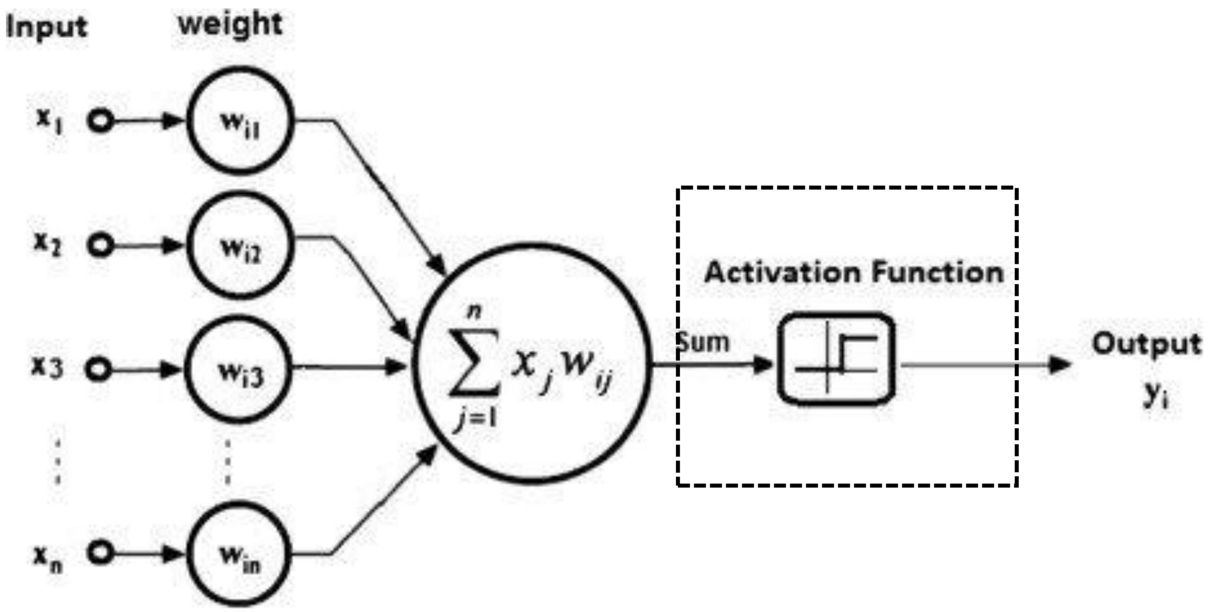
Lessons from brain







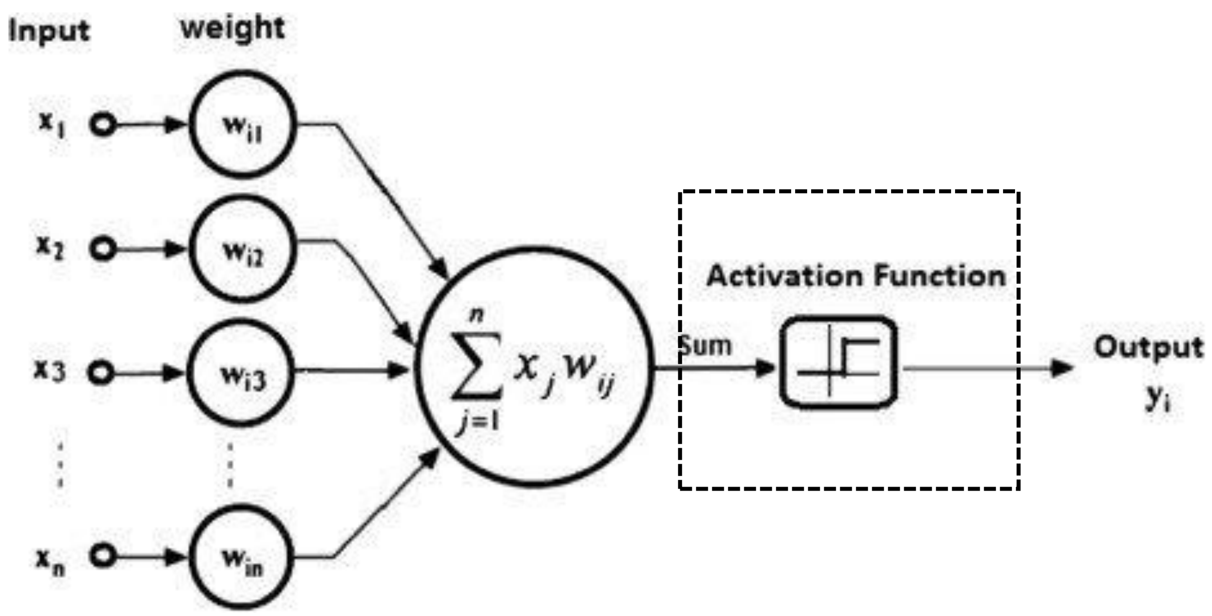
Mathematical Description of a Neuron



$I = x_1 w_1 + \cdots + x_n w_n$ $f(I)$ $x = f(I)$

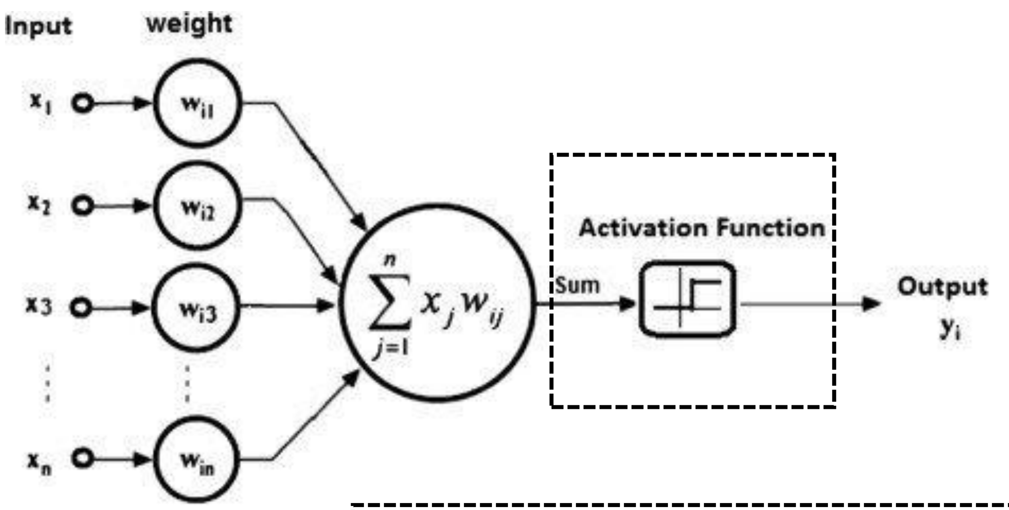
Integration Activation Output to other neurons

Activation Functions in Neural Network

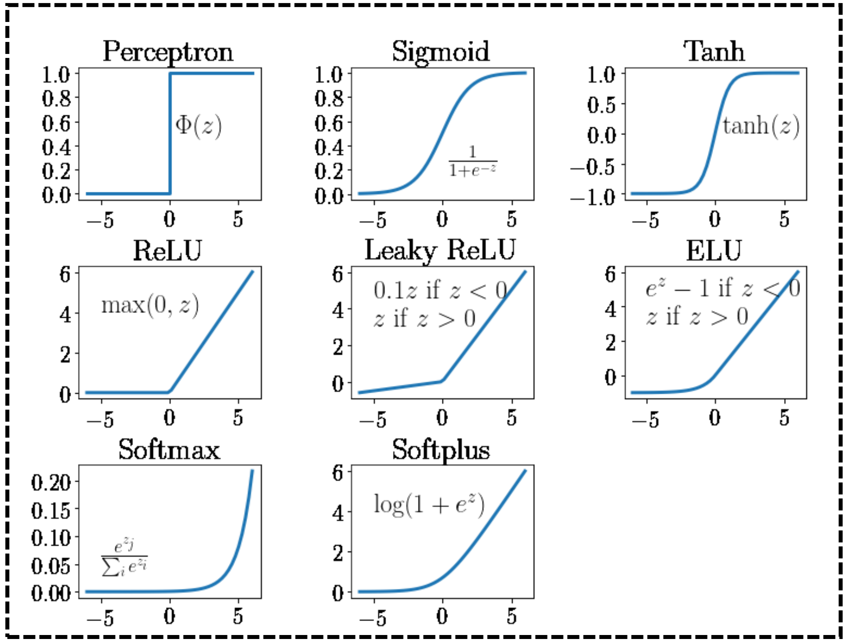


$I = x_1 w_1 + \cdots + x_n w_n$	$f(I)$	$y = f(I)$
Integration	Activation	Output to other neurons
$\sum_{i=1}^{n} x_i w_i + b$	$f\left(\sum_{i=1}^{n} x_i w_i + b\right)$	$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$

Mathematical Description of a Neuron

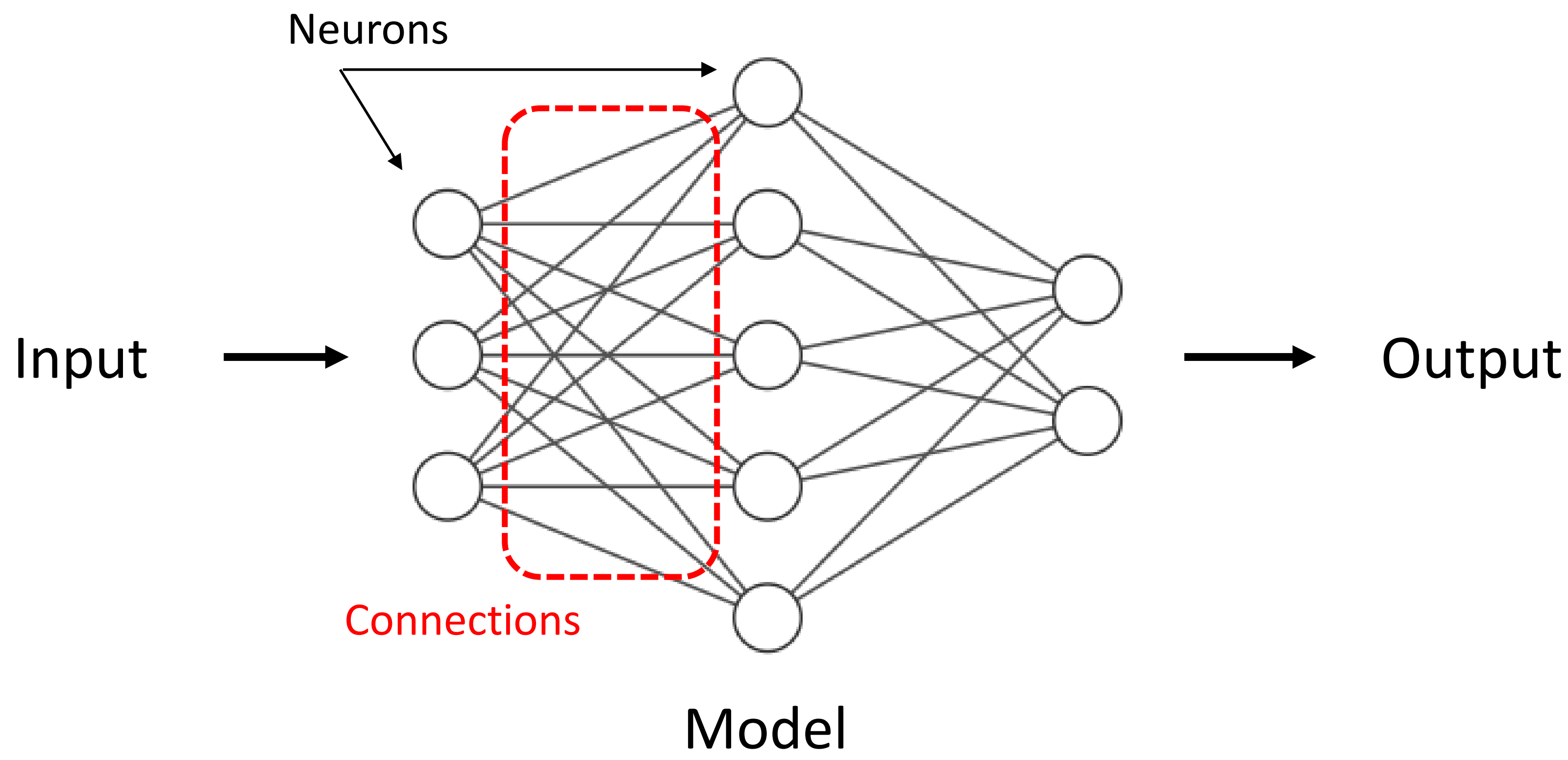


Without non-linear activation, neural network becomes a **linear model**



Activation function introduces **non-linearity** to the model

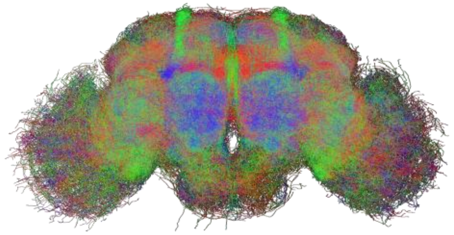
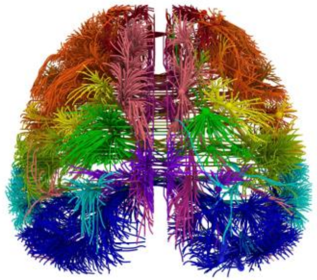
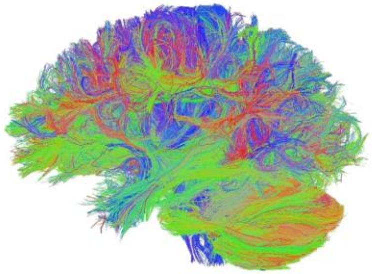
Neural Network Model



$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

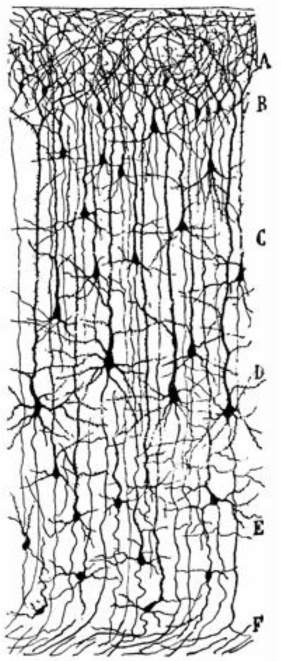
Learn the connection weights and biases that optimizes the training set

Biological Neural Networks are large and complex

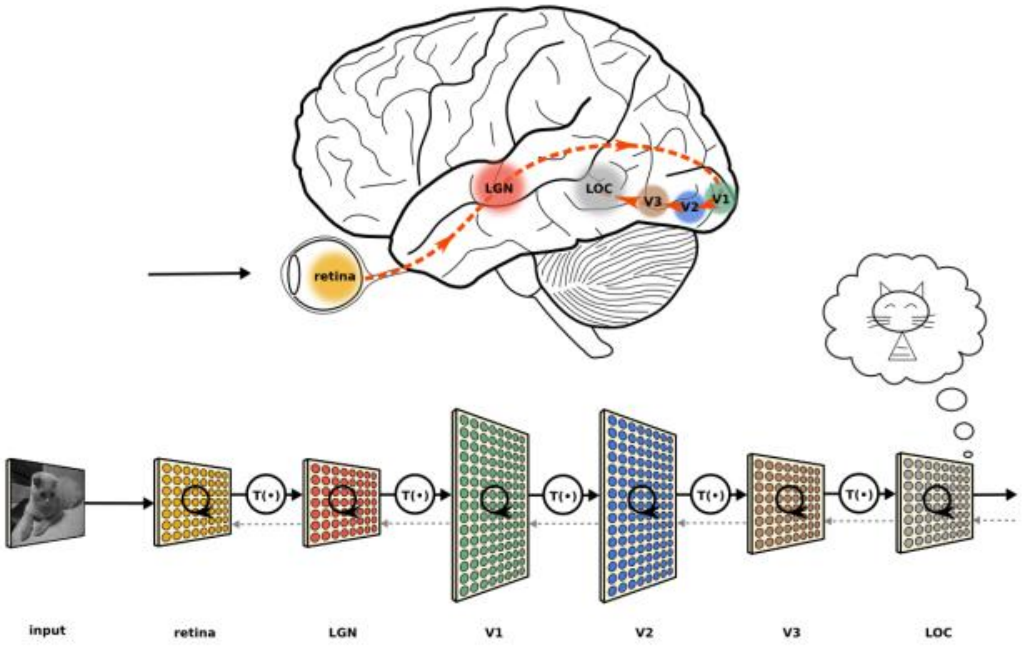
Nematode *C. elegans*	Fruit fly	Mouse	Human
			
302 neurons	~150k neurons	~70mil neurons	~10^11 neurons
~7000 synapses	*~70mil synapses*	*~7 x 10^8 synapses*	*~10^14 synapses*

Biological Neural Networks are large and complex

Cerebral cortex

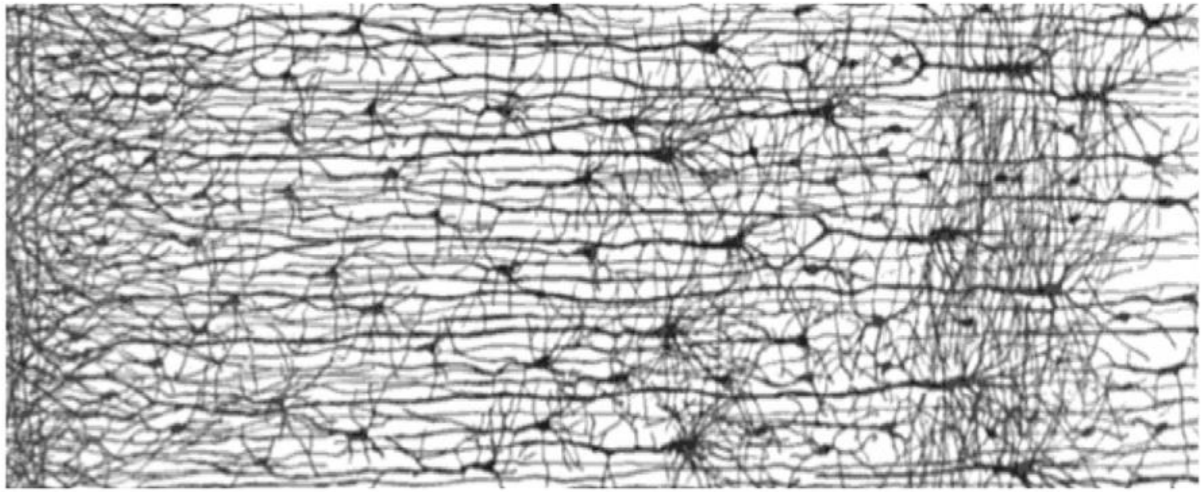


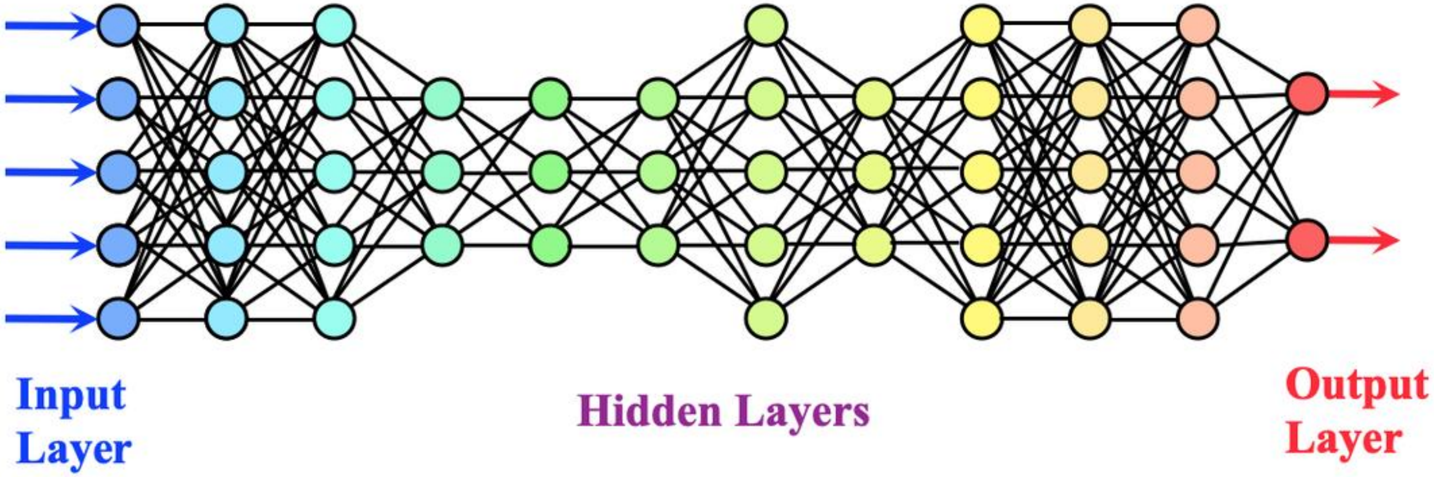
Visual system



Biological neural networks are both Hierarchical and Recurrent (parallelism)

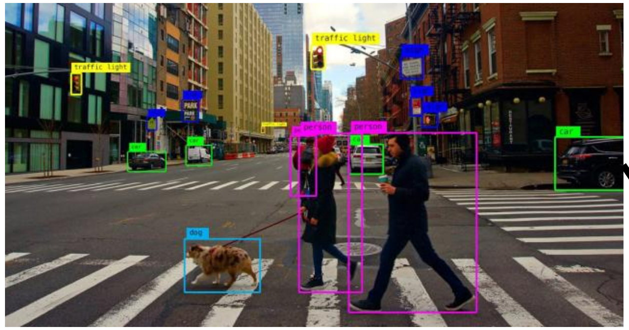
Deep Neural Network as Brain Analogue

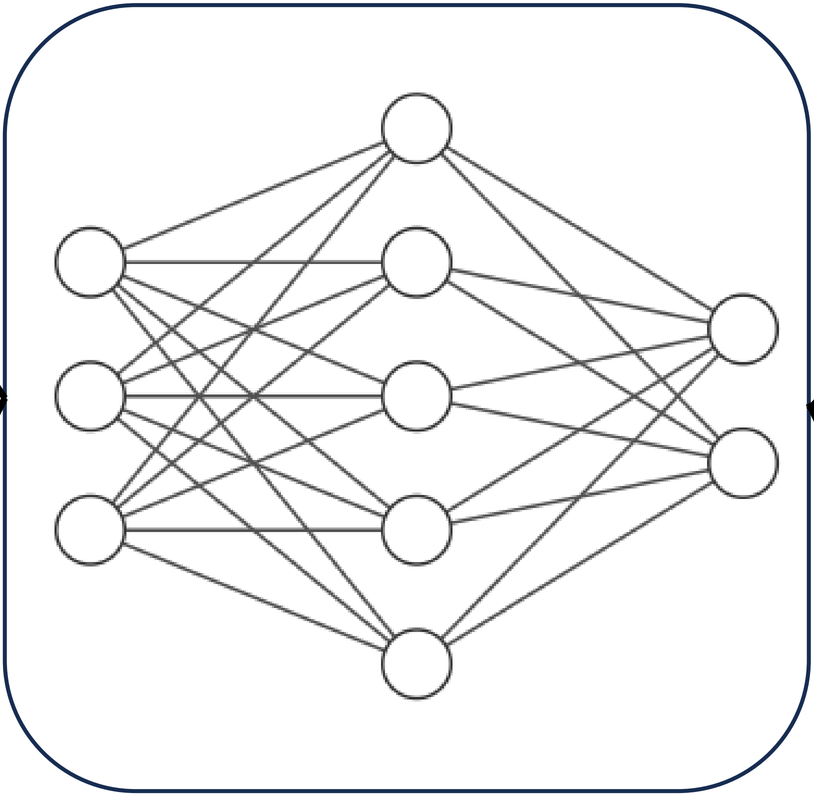




Deep Learning Applications

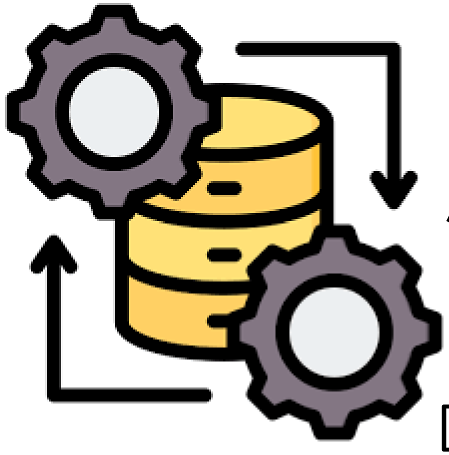
Computer vision





Text/speech generation

Image/video generation



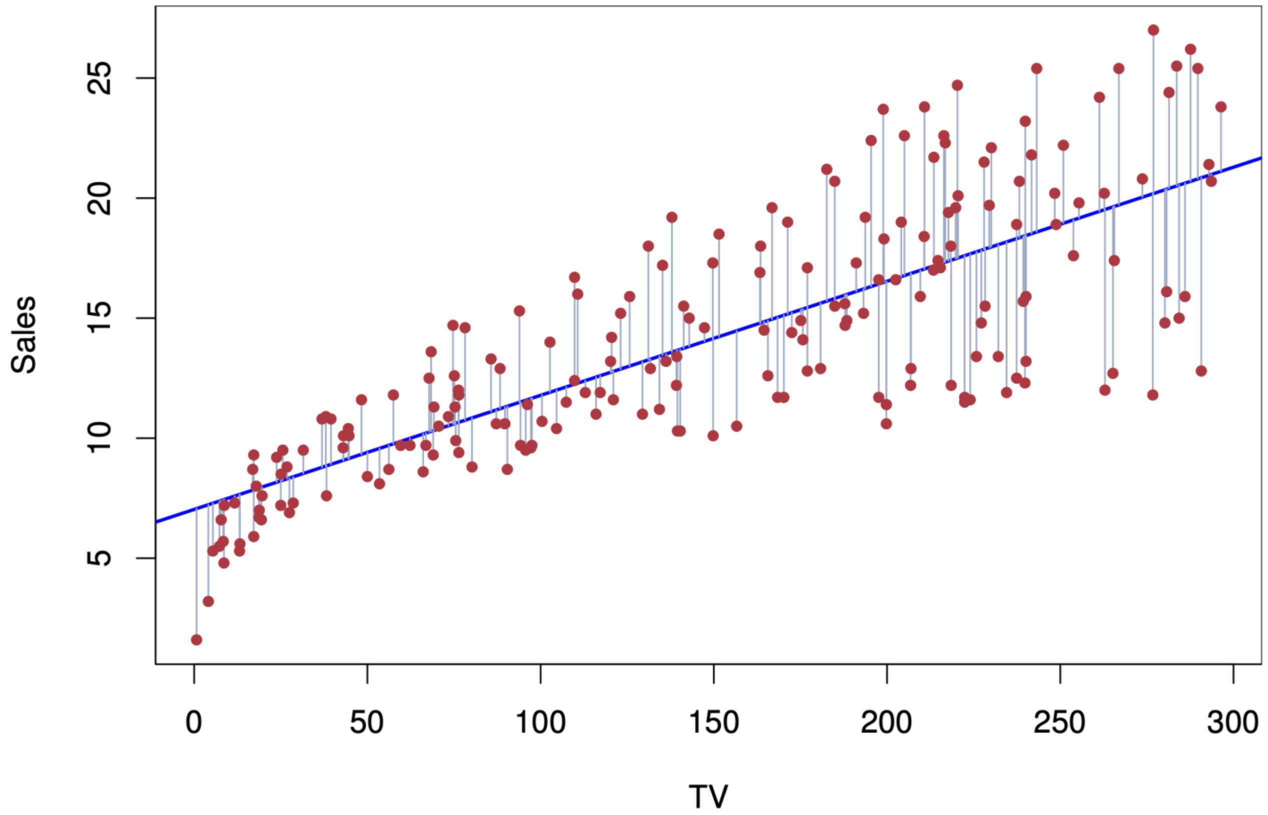
Data processing

PART 3:

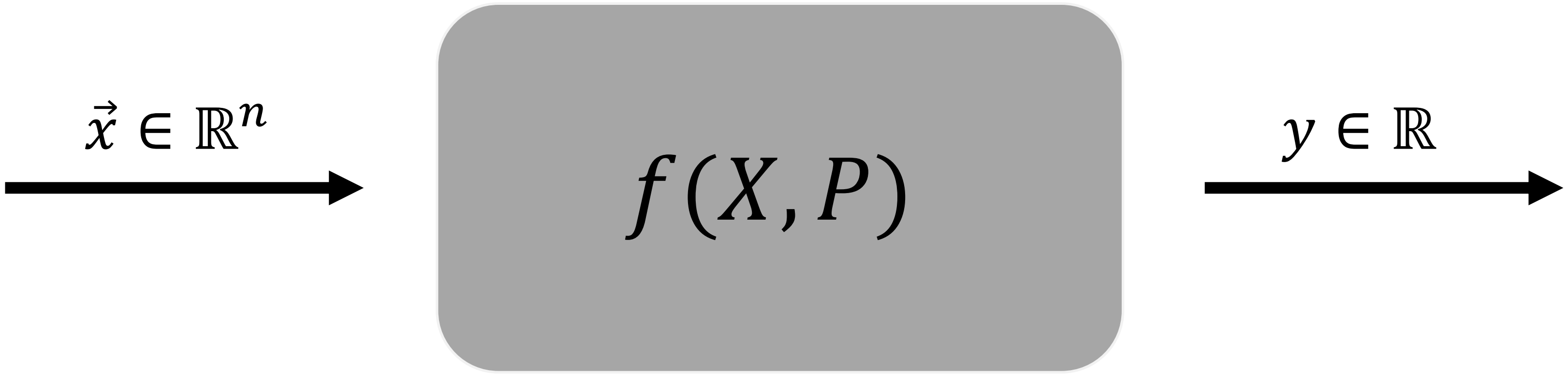
Regression

Regression problem

$$Y = f(X, W)$$



Regression problem

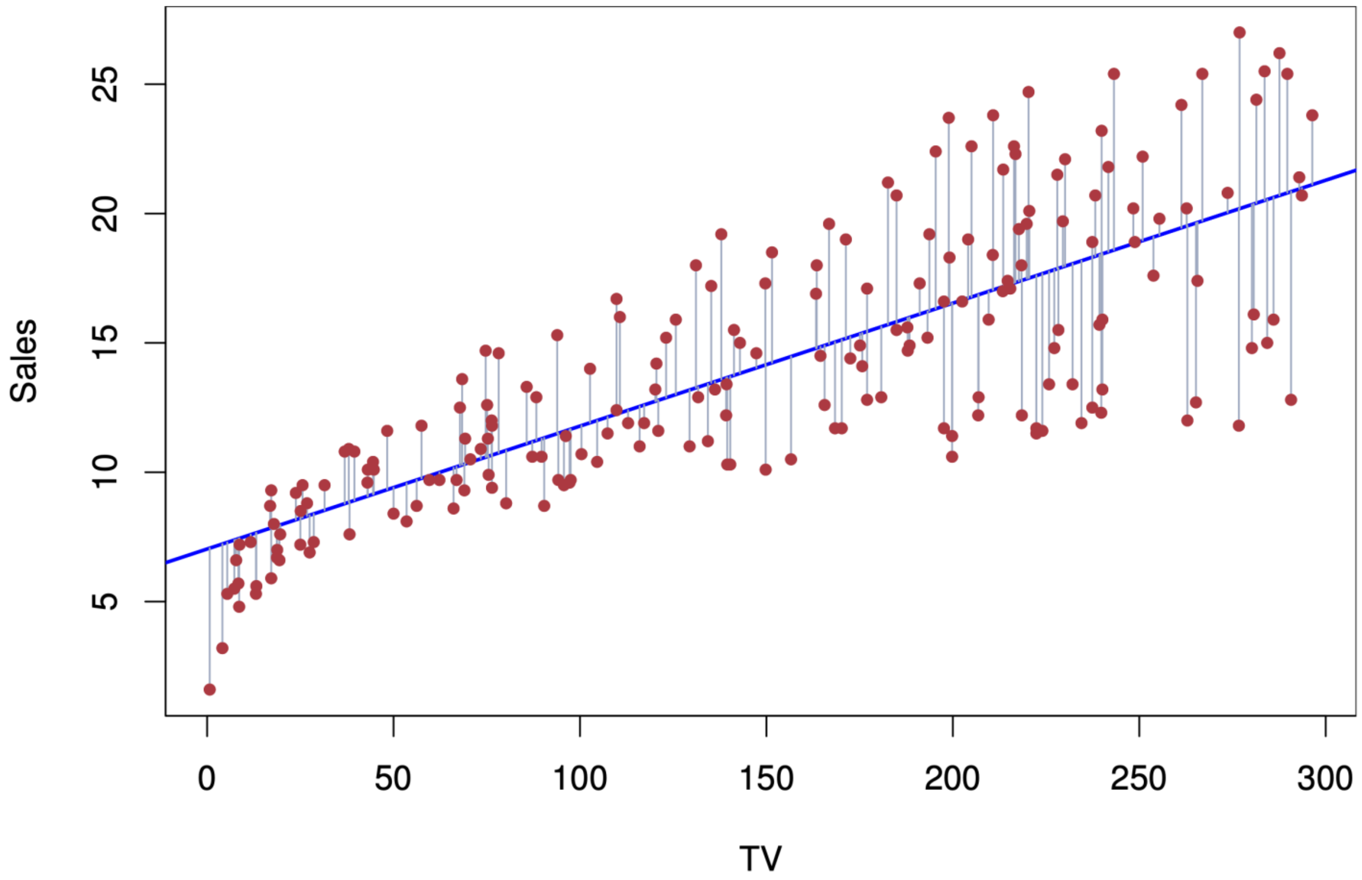


Regression model

Linear regression (polynomial fit)

$$y(x) = p_0 + p_1 x + \cdots + p_m x^m \qquad\qquad y = \vec{p} \cdot \vec{x}$$

Polynomial fit – Linear regression



$x = input\ data$

$y = output\ data$

$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$

Goal: Minimize $e = y - \hat{y}$

Polynomial fit – Linear regression

$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$ Goal: Minimize $e = y - \hat{y}$

Polynomial fit – Linear regression

$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$ Goal: Minimize $e = y - \hat{y}$

$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$

n-given points

$$X = \begin{array}{cc} \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{pmatrix} & \begin{array}{c} \vec{\tilde{x}}_0 \\ \\ \\ \vec{\tilde{x}}_n \end{array} \end{array} \qquad P = \begin{pmatrix} p_0 \\ p_1 \\ . \\ . \\ . \\ p_m \end{pmatrix}$$

Polynomial fit – Linear regression

$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$ Goal: Minimize $e = y - \hat{y}$

$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$

n-given points

$$X = \begin{array}{cc} \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{pmatrix} & \begin{array}{c} \vec{\tilde{x}}_0 \\ \\ \\ \vec{\tilde{x}}_n \end{array} \end{array} \qquad P = \begin{pmatrix} p_0 \\ p_1 \\ . \\ . \\ . \\ p_n \end{pmatrix}$$

Fit ("Learn") the model:

Least squares error

Cost $J = E_2 = \sum_{i=1}^{n} (\overbrace{\vec{p} \cdot \vec{\tilde{x}}_i}^{\hat{y}_i} - y_i)^2$

Total error

Polynomial fit – Linear regression

Find the parameters minimizing the error

$$J = E_2 = \sum_{i=1}^{n} (\vec{p} \cdot \vec{\tilde{x}}_i - y_i)^2$$

$$\vec{p}^* = \arg \min_{\vec{p}} J\left(\vec{p}\right)$$

Polynomial fit – Linear regression

Find the parameters minimizing the error

$$J = E_2 = \sum_{i=1}^{n} (\vec{p} \cdot \vec{\tilde{x}}_i - y_i)^2$$

$$\vec{p}^* = \arg\min_{\vec{p}} J(\vec{p})$$

$$\forall i : \quad \frac{\partial J}{\partial p_j} = 0 \qquad \frac{\partial J}{\partial p_j} = \sum_{i=1}^{n} 2(\vec{p} \cdot \vec{\tilde{x}}_i - y_i)\vec{\tilde{x}}_i$$

$$\vec{p}^* = \left(X^\top X\right)^{-1} X^\top \vec{y}$$

Polynomial fit – Linear regression

Find the parameters minimizing the error

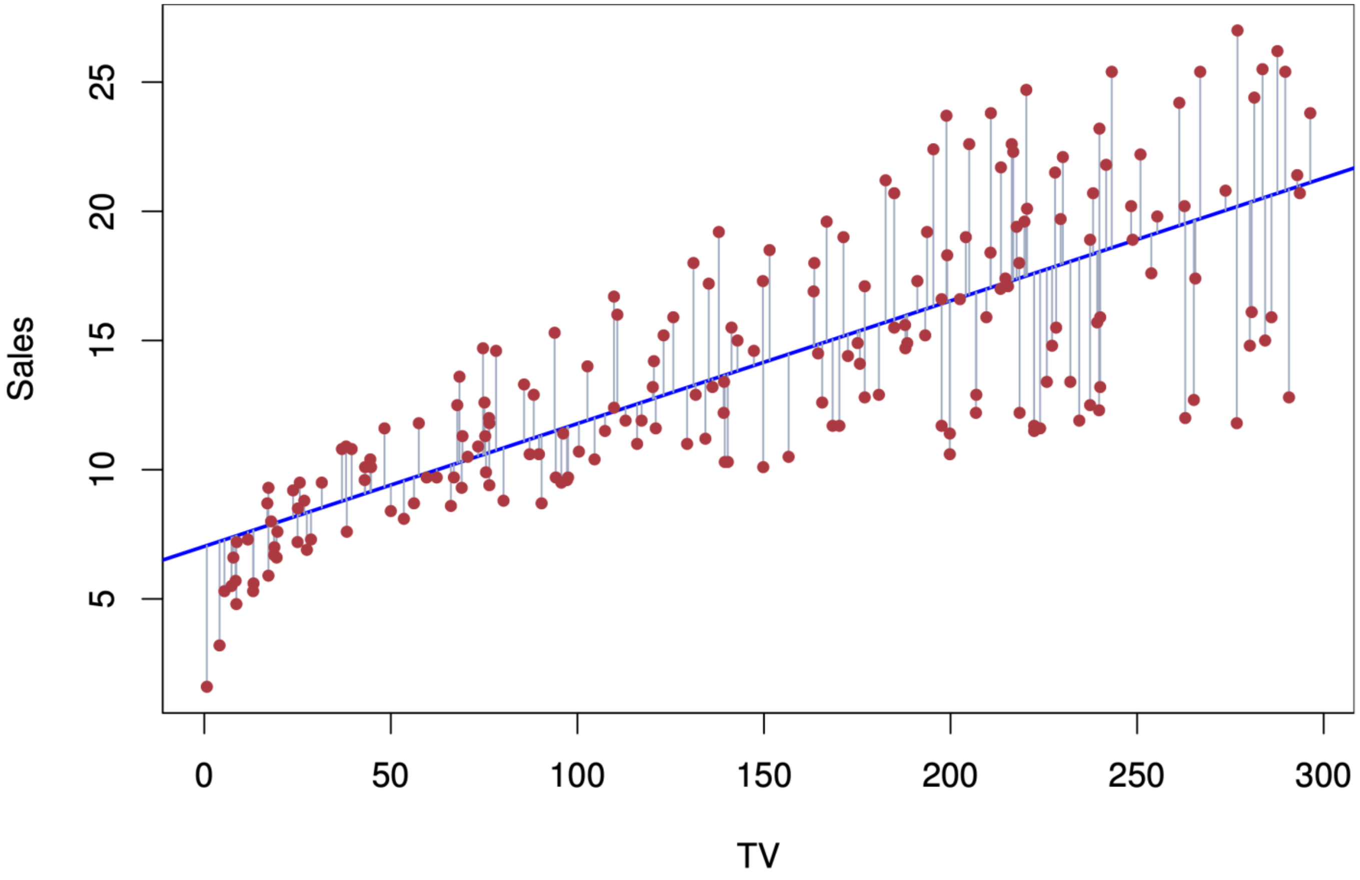
$$J = E_2 = \sum_{i=1}^{n} (\vec{p} \cdot \vec{\tilde{x}}_i - y_i)^2$$

$$\vec{p}^* = \arg \min_{\vec{p}} J(\vec{p})$$

$$\forall i : \quad \frac{\partial J}{\partial p_j} = 0 \qquad \frac{\partial J}{\partial p_j} = \sum_{i=1}^{n} 2(\vec{p} \cdot \vec{\tilde{x}}_i - y_i)\vec{\tilde{x}}_i$$

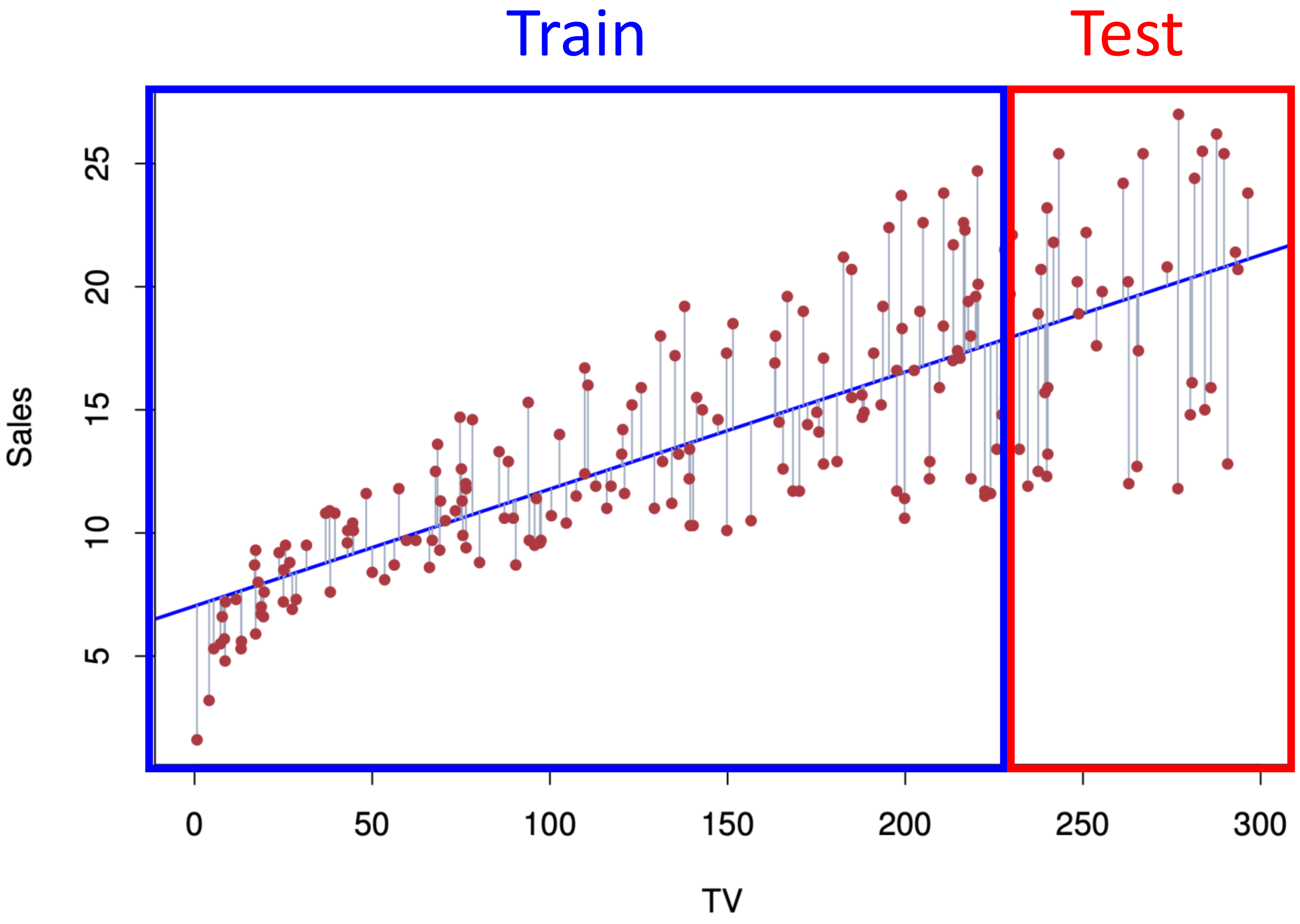
$$\vec{p}^* = \left(X^\top X\right)^{-1} X^\top \vec{y}$$ Linear least square formula

Linear regression is not perfect



$$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$$

Linear regression is not perfect



$$\hat{y}(x) = p_0 + p_1 x + \cdots + p_m x^m$$

Model might perform worse in testing in the presence of outliers

Improving Linear Regression

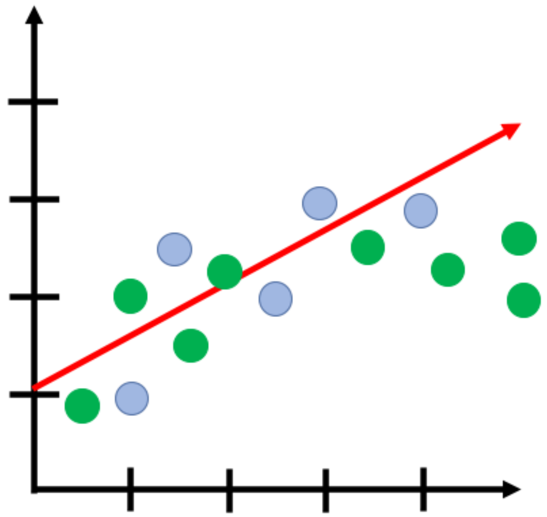
$$W^* = \arg\min_{W} L\left(X, W\right)$$

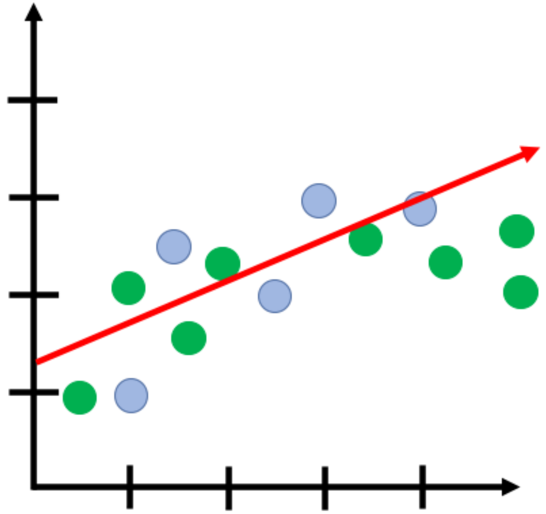
$$J_2 = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(\tilde{x}_i, W\right) - y_i\right)^2$$ Mean squared error

$$J_1 = \frac{1}{n}\sum_{i=1}^{n}\left|f\left(\tilde{x}_i, W\right) - y_i\right|$$ Mean absolute error

$$J_\infty = \max_{1<i<n}\left|f\left(\tilde{x}_i, W\right) - y_i\right|$$ Max error

Improving Linear Regression

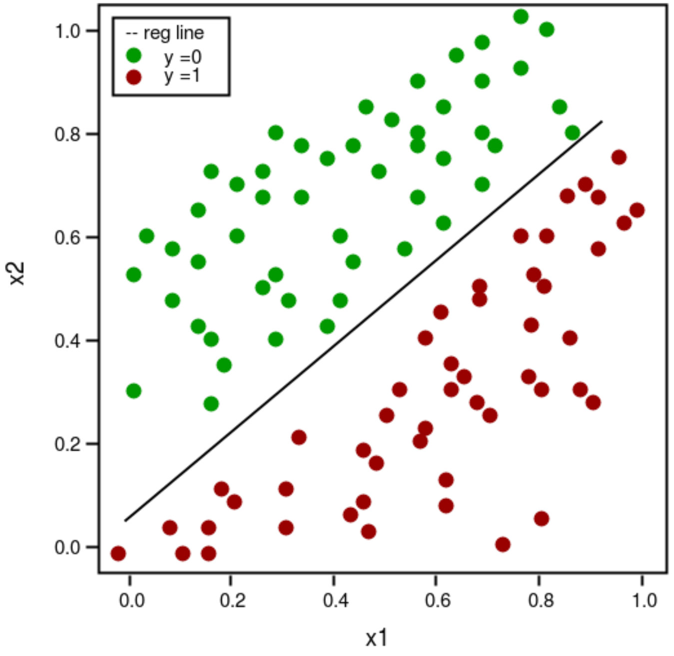




	Standard	Ridge
Minimizes	$(y - Xp)^T(y - Xp)$	$(y - Xp)^T(y - Xp) + \lambda \lvert p \rvert^2$
Solution	$p = (X^TX)^{-1}X^Ty$	$p = (X^TX + \lambda I)^{-1}X^Ty$
	Unbiased	*Biased*
	High variance	*Low variance*

Next episode in EE P 596…

$$y = \sigma(\vec{w}^T \vec{x} + b)$$



Classification

$$\nabla_{\vec{w}} L(\hat{y}, y) = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \nabla_{\vec{w}} z$$

FWD $\quad z = \vec{w}^T \vec{x} + b \longrightarrow \hat{y} = \sigma(z) \longrightarrow L(\hat{y}, y)$

BWD $\quad \nabla_{\vec{w}} z \longleftarrow \frac{\partial \hat{y}}{\partial z}(z) \longleftarrow \frac{\partial L}{\partial \hat{y}}(\hat{y}, y)$

Optimizations in Deep Learning using Back-propagation and Gradient descent