From Basics of Deep Learning to Application in Biology

Jędrzej Jakub Szymański

@NAMlab www.szymanskilab.com

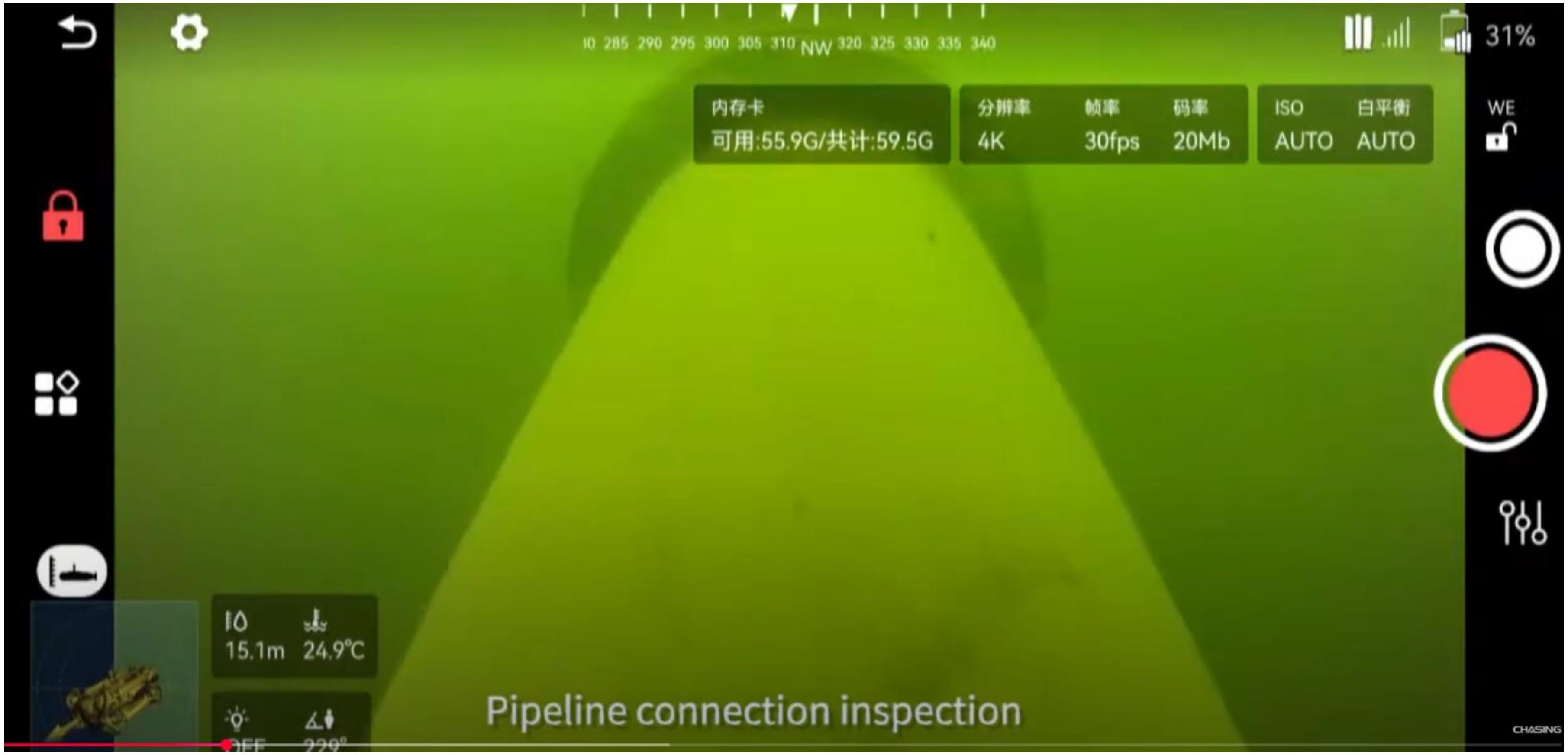
CEPLAS
Cluster of Excellence on Plant Sciences

Part 1: Gradient descent

@NAMlab www.szymanskilab.com

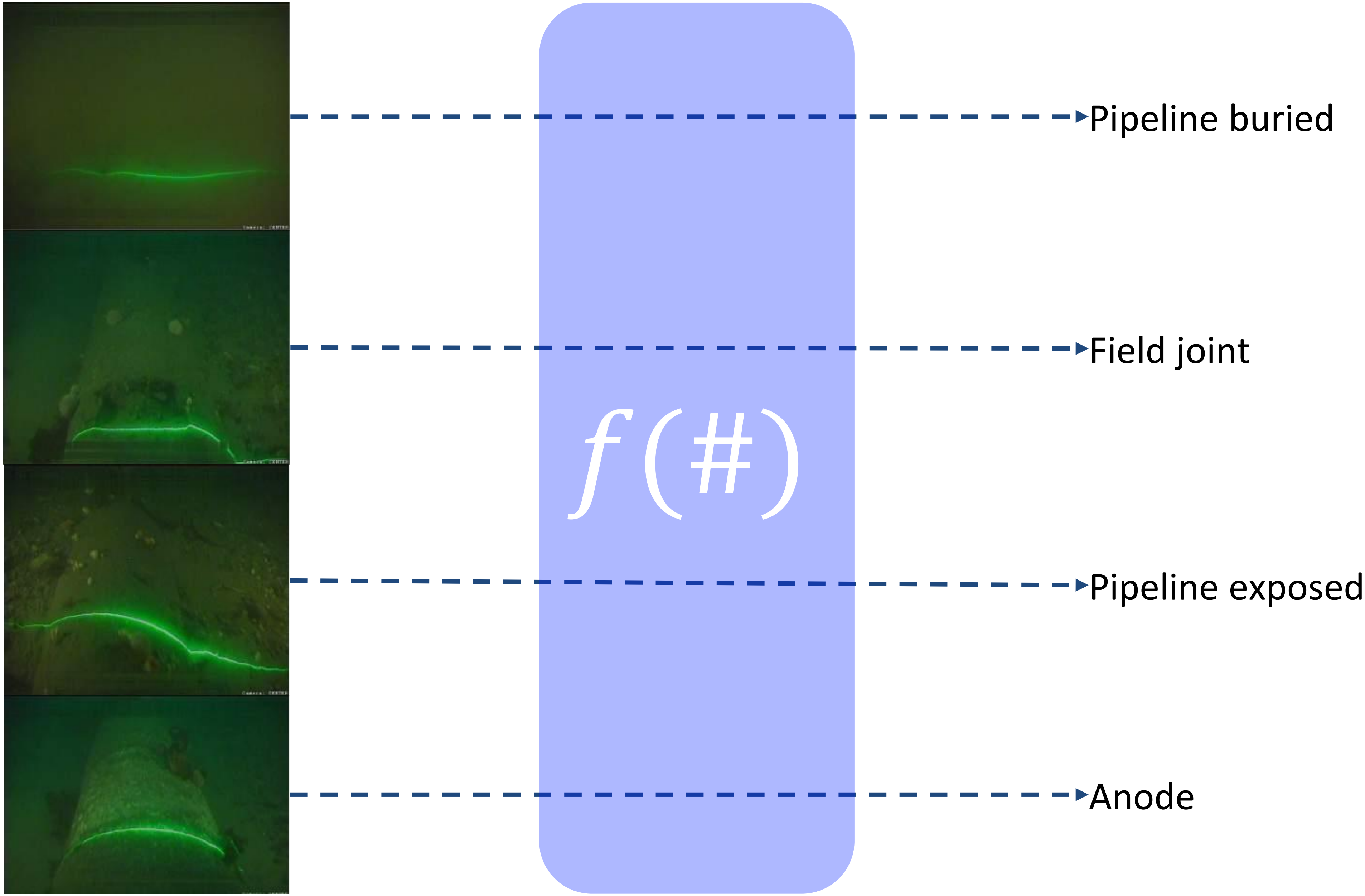
CEPLAS
Cluster of Excellence on Plant Sciences

Machine learning has a lot of great down-to-earth applications



31%
内存卡
可用:55.9G/共计:59.5G
分辨率
4K
帧率
30fps
码率
20Mb
ISO
AUTO
白平衡
AUTO
WE
15.1m
24.9℃
Pipeline connection inspection

Goal



Automatic detection of Cargo Carrying Units (CCU)

Automatic detection of Cargo Carrying Units (CCU)

Container

Basket

Frame/Rack

Tank

Skip

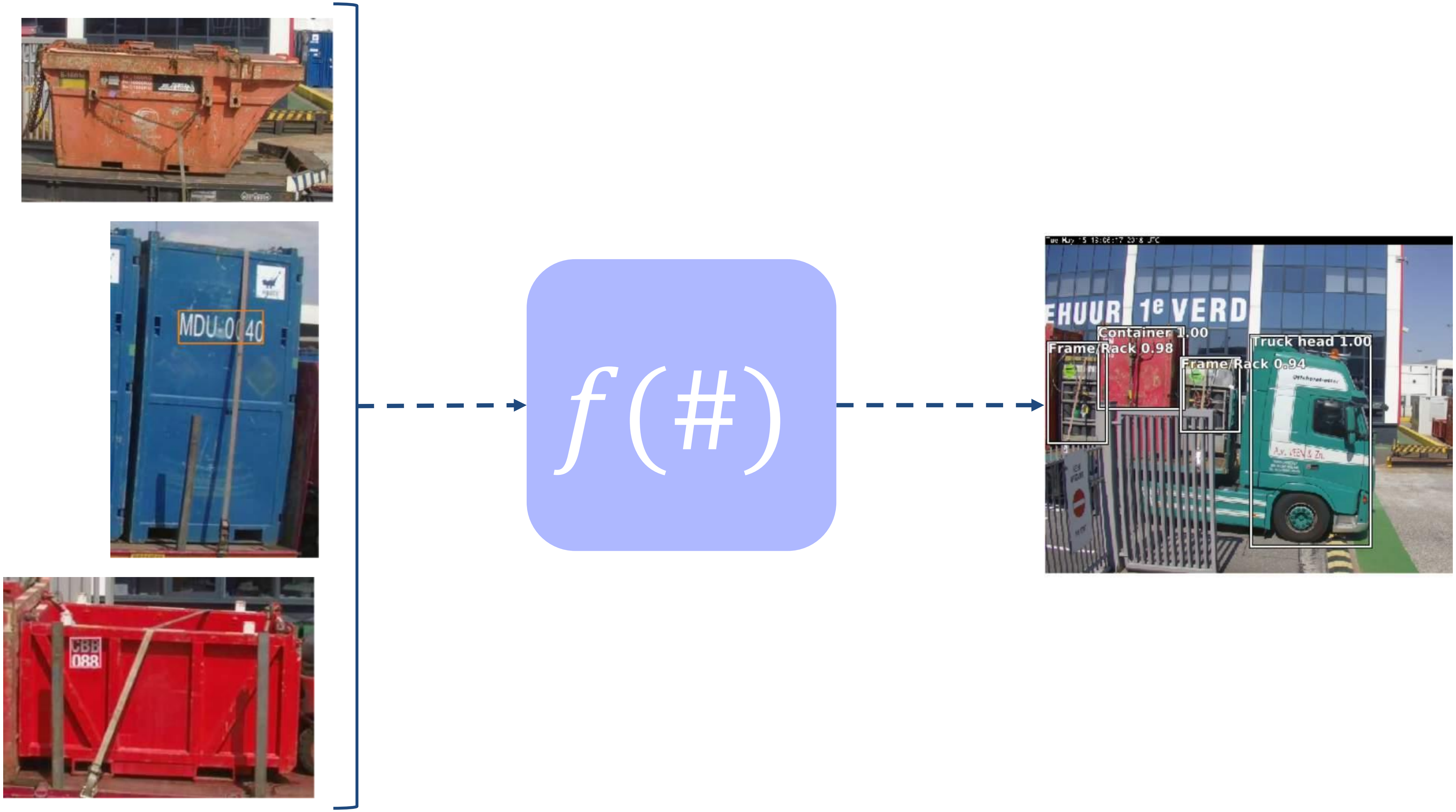
Bin

Other CCU

Other

Truck head

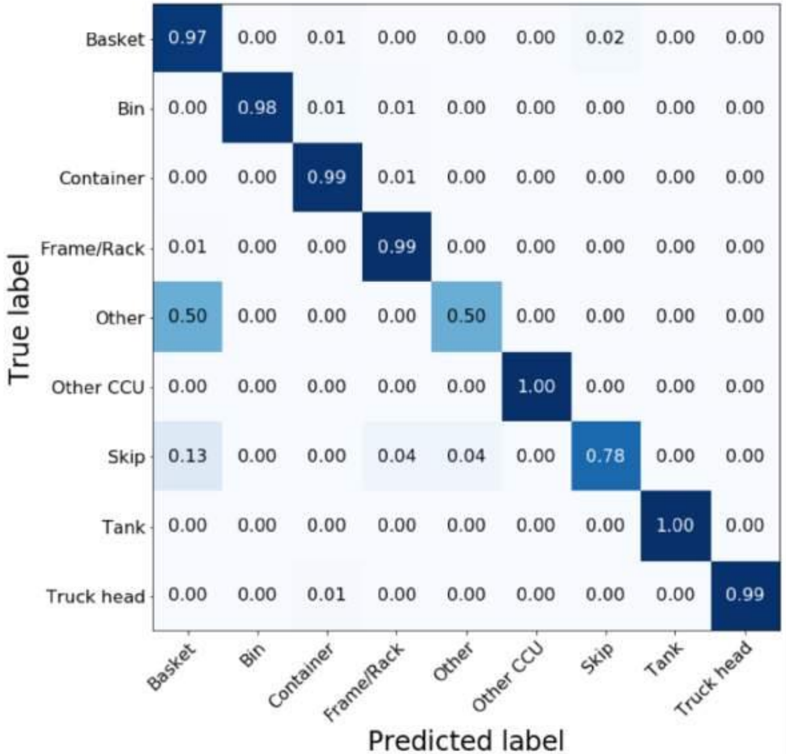
Automatic detection of Cargo Carrying Units (CCU)



Automatic detection of Cargo Carrying Units (CCU)

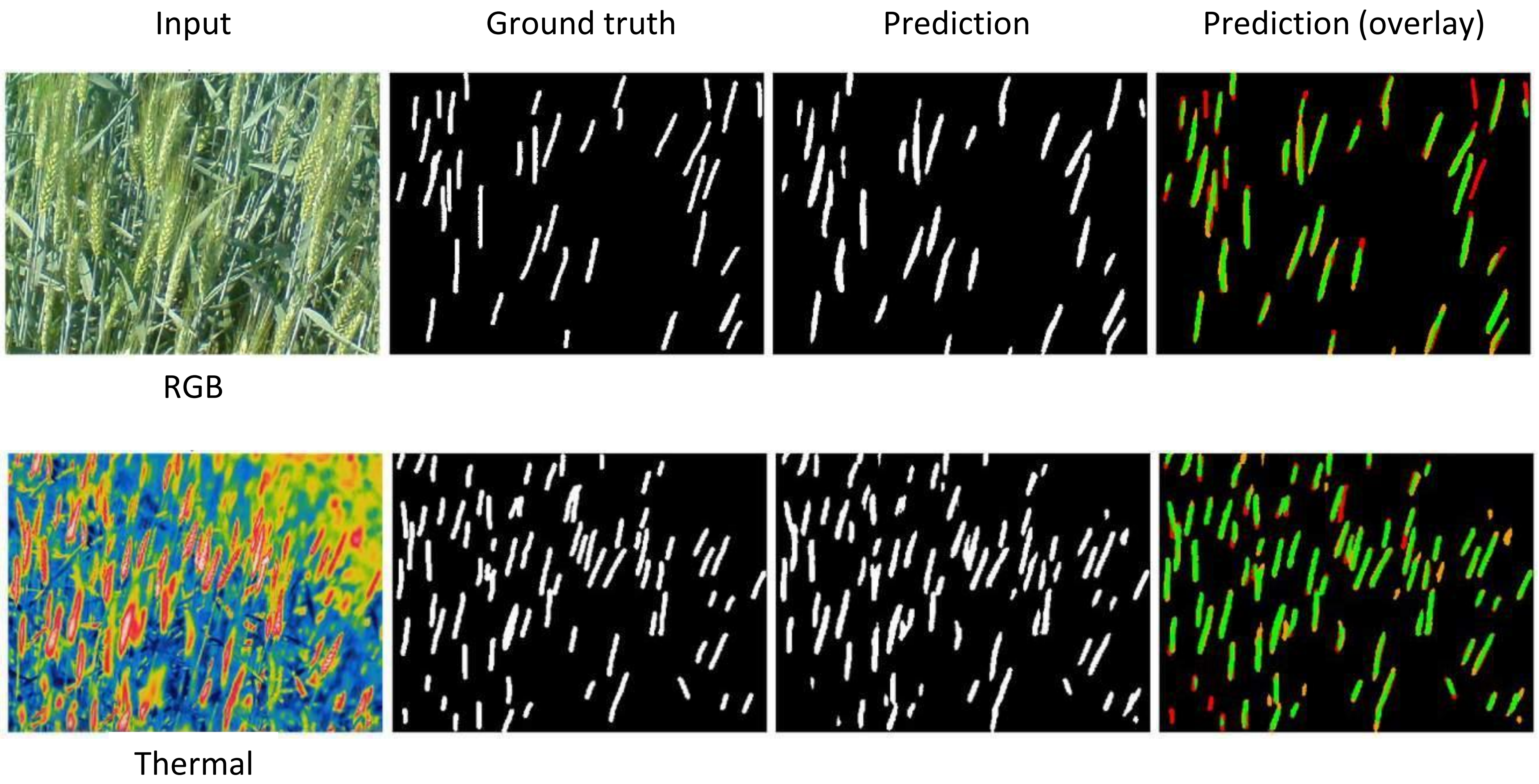
Prediction

Confusion matrix



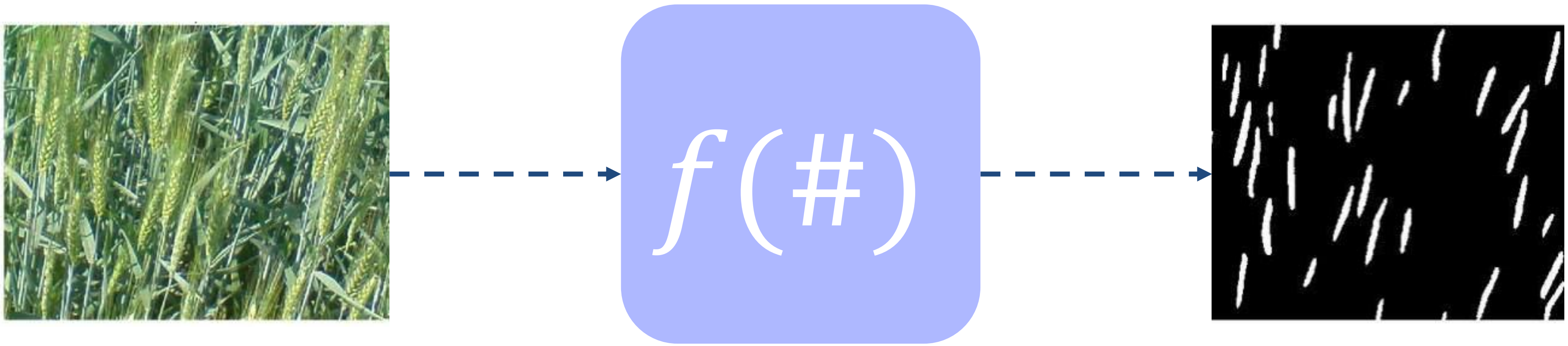
HPC-Enabled Precision Agriculture

CYBELE

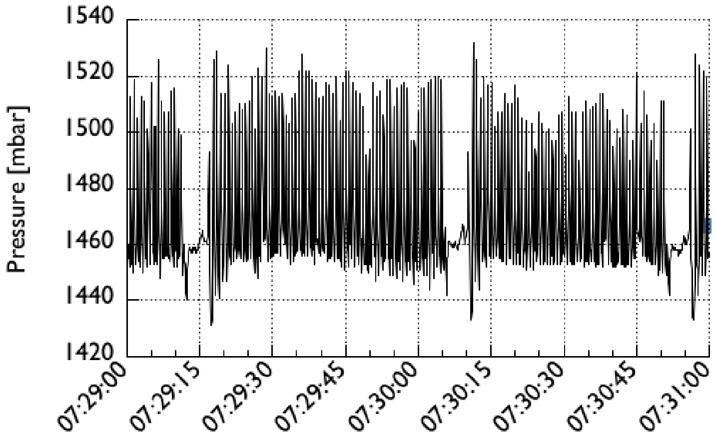


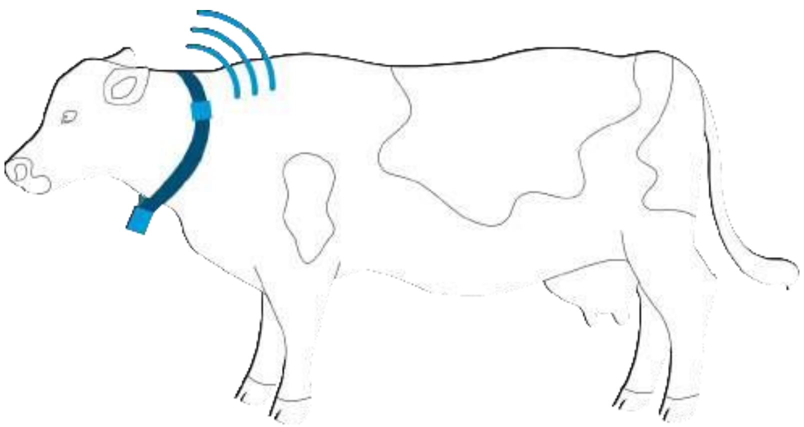
CYBELE

HPC-Enabled Precision Agriculture



What does the cow do?





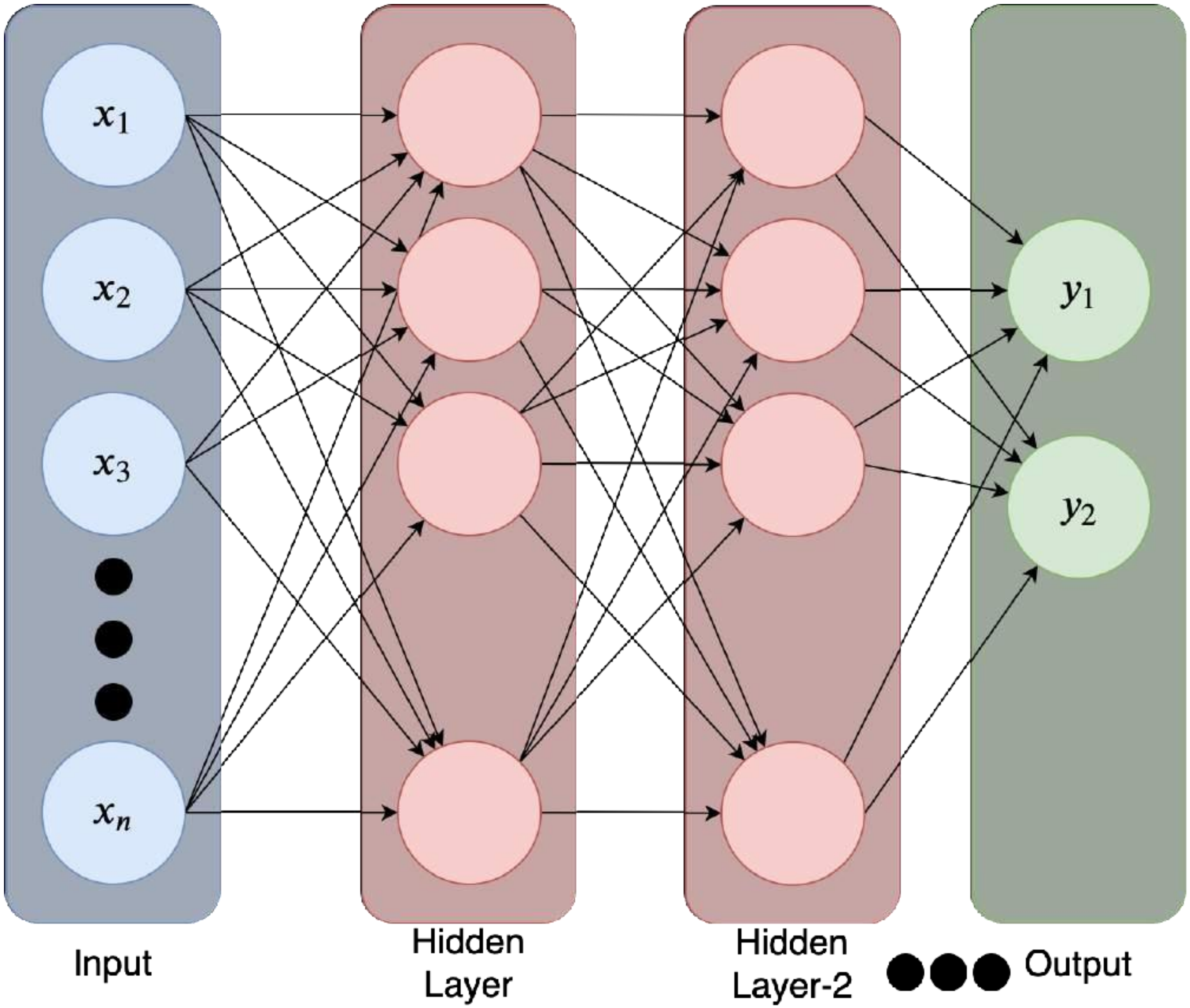
$f(\#)$

Rumination

Standing

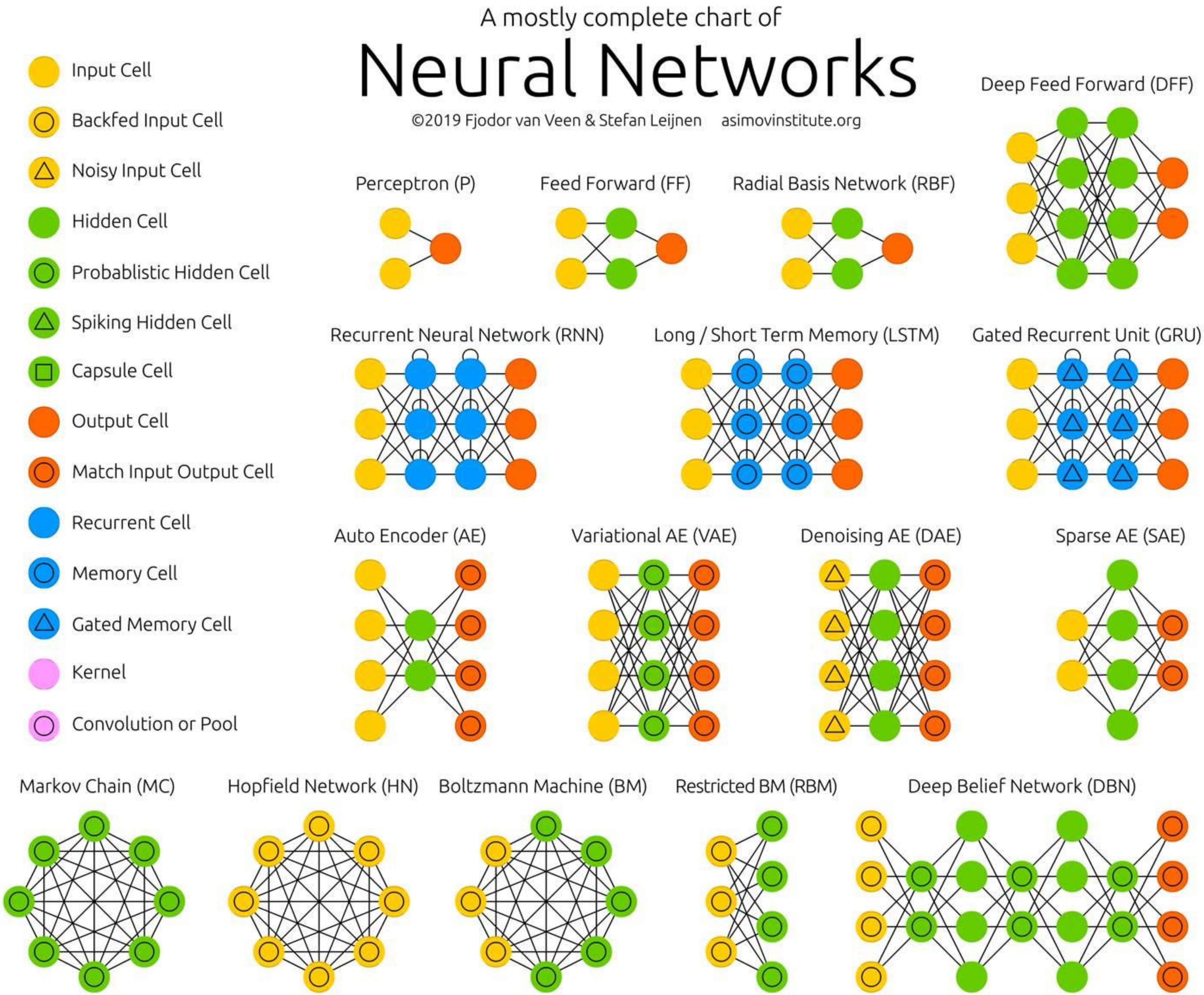
Eating

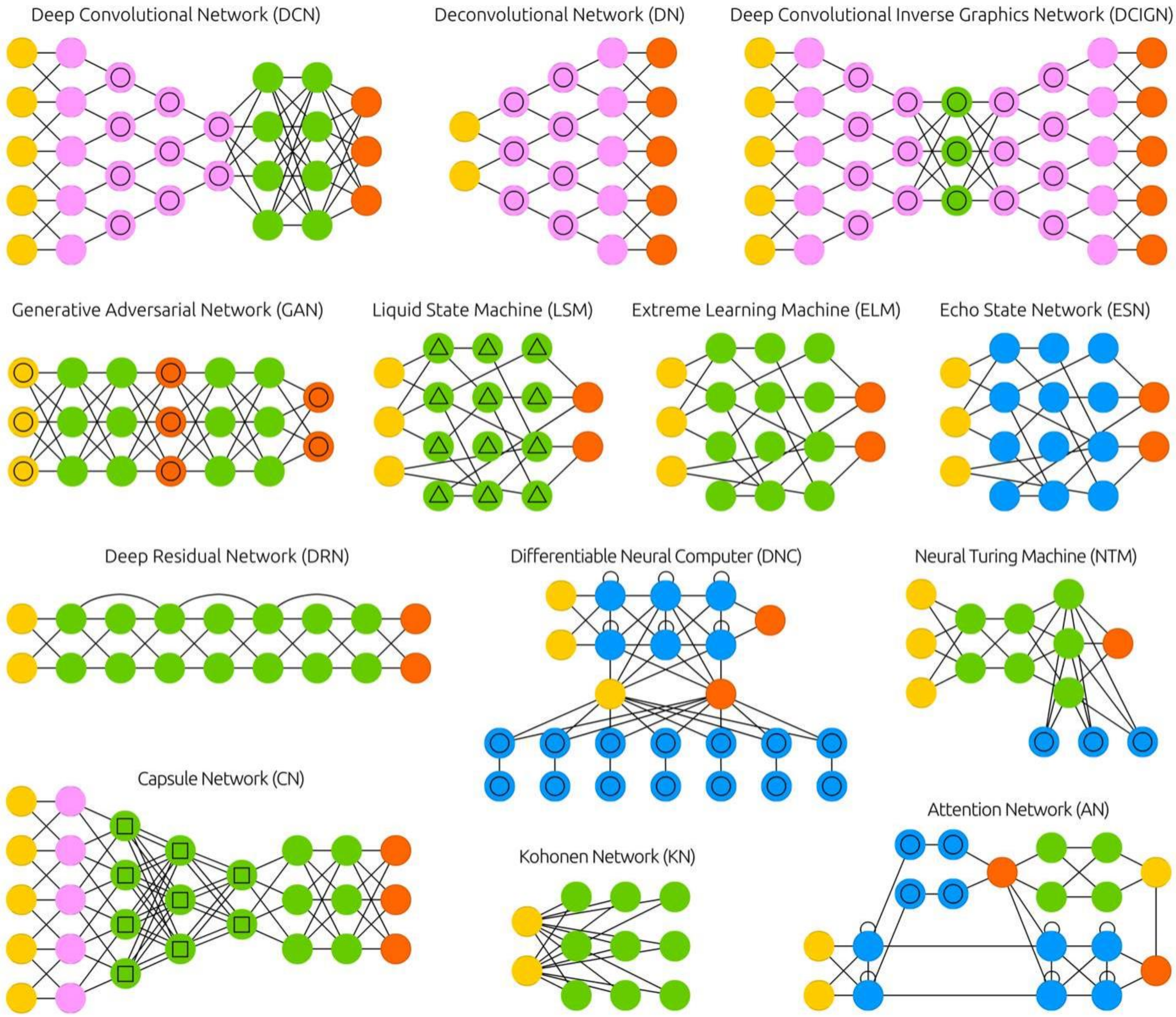
Artificial Neural Networks



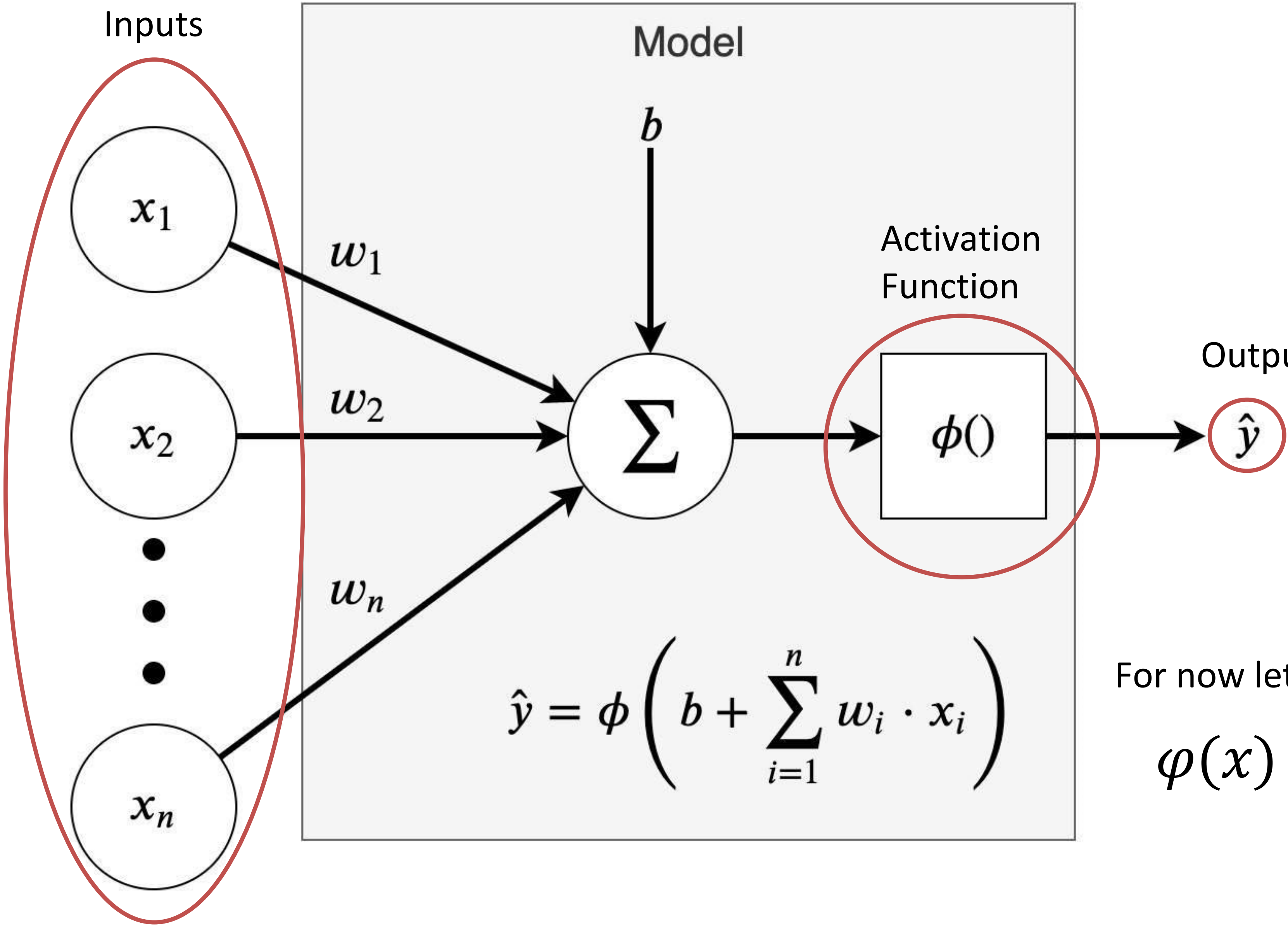
Decisions, Decisions, Decisions

- What are input parameters?
- What is the structure of the model?
- What is the required model complexity?
- How are we going to identify model parameters?


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen
asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated 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)
Denoising 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)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

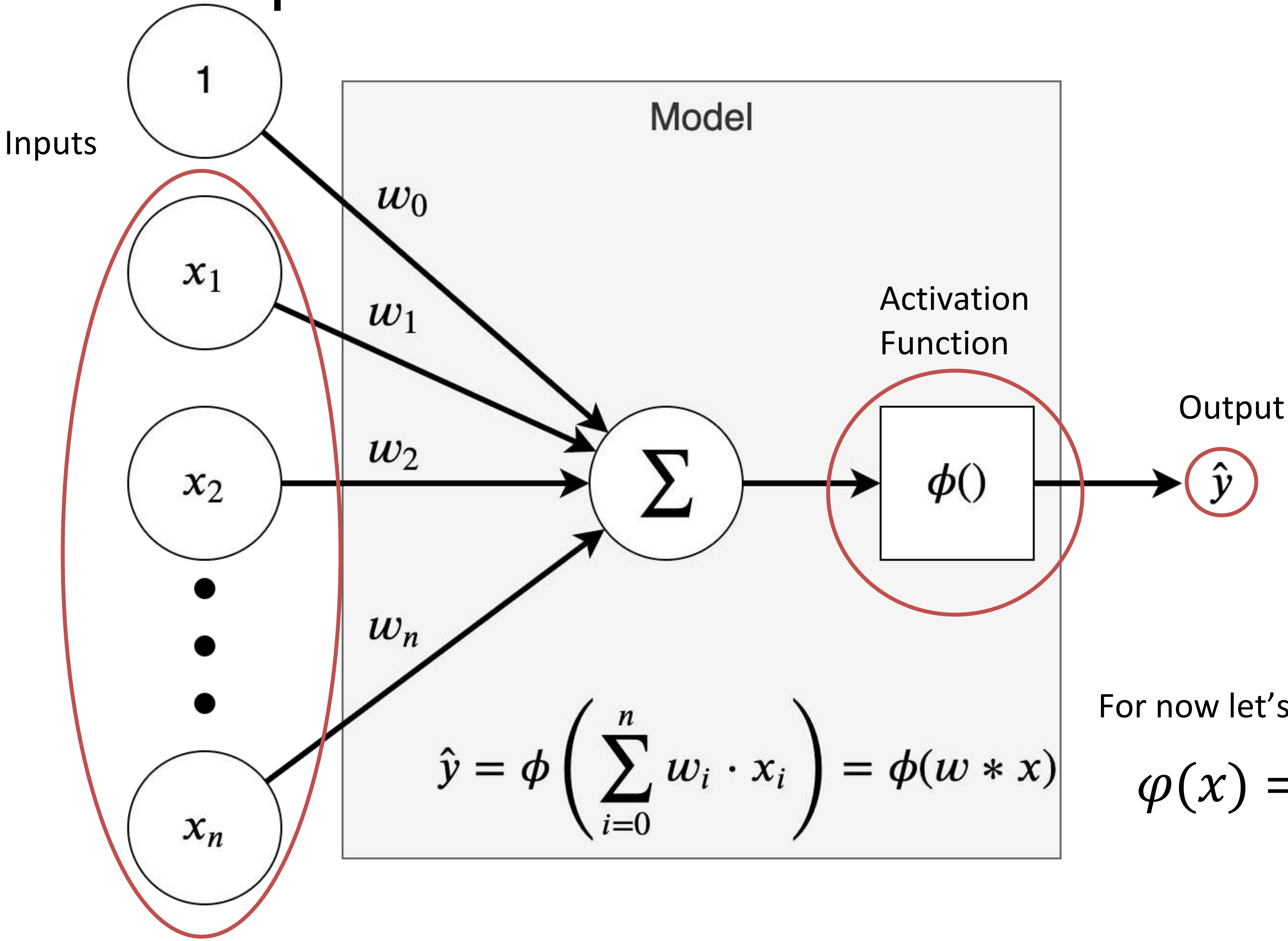
Perceptron



For now let's assume

$$\varphi(x) = x$$

Perceptron



For now let's assume

$$\varphi(x) = x$$

Data Set

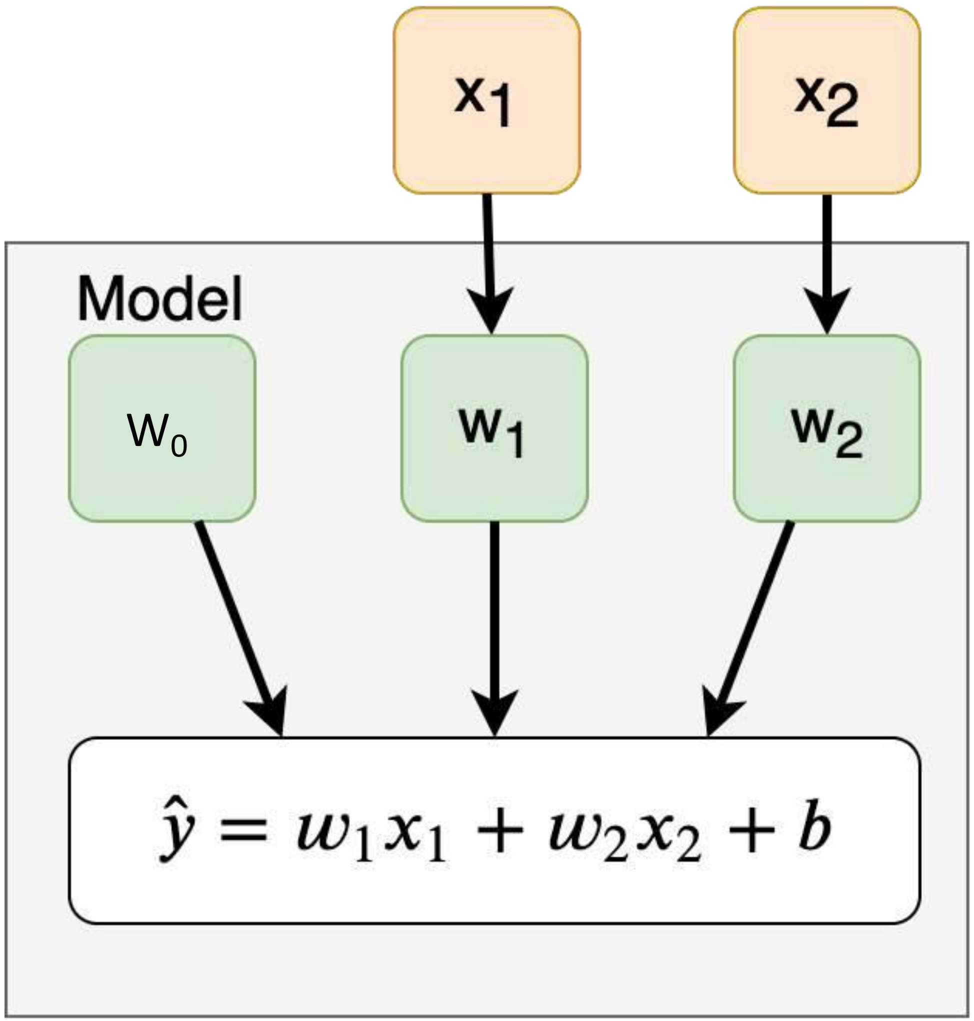
Observation #	x_1	x_2	y
1	4	7	-2.669
2	-4	0	-0.600
3	7	-5	7.230
4	5	7	-2.389
5	2	-1	2.410
6	-2	5	-3.490
7	-5	-1	-0.249
8	-6	2	-2.820
9	-1	1	-0.189
10	-4	-9	5.970

Synthetic Data Set created using:

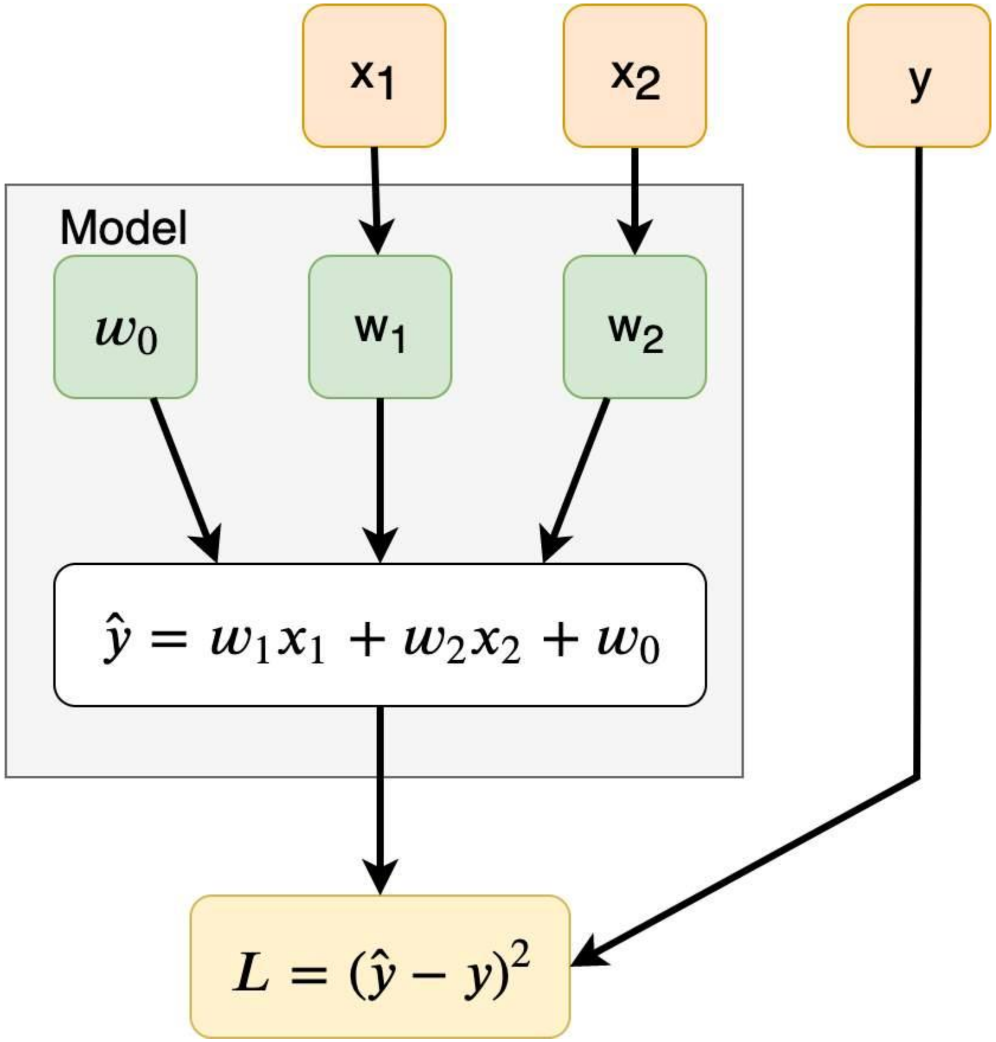
$$\hat{y} = w_1 x_1 + w_2 x + w_0$$

$$w_1 = 0.38,\ w_2 = -0.73,\ w_0 = 0.92$$

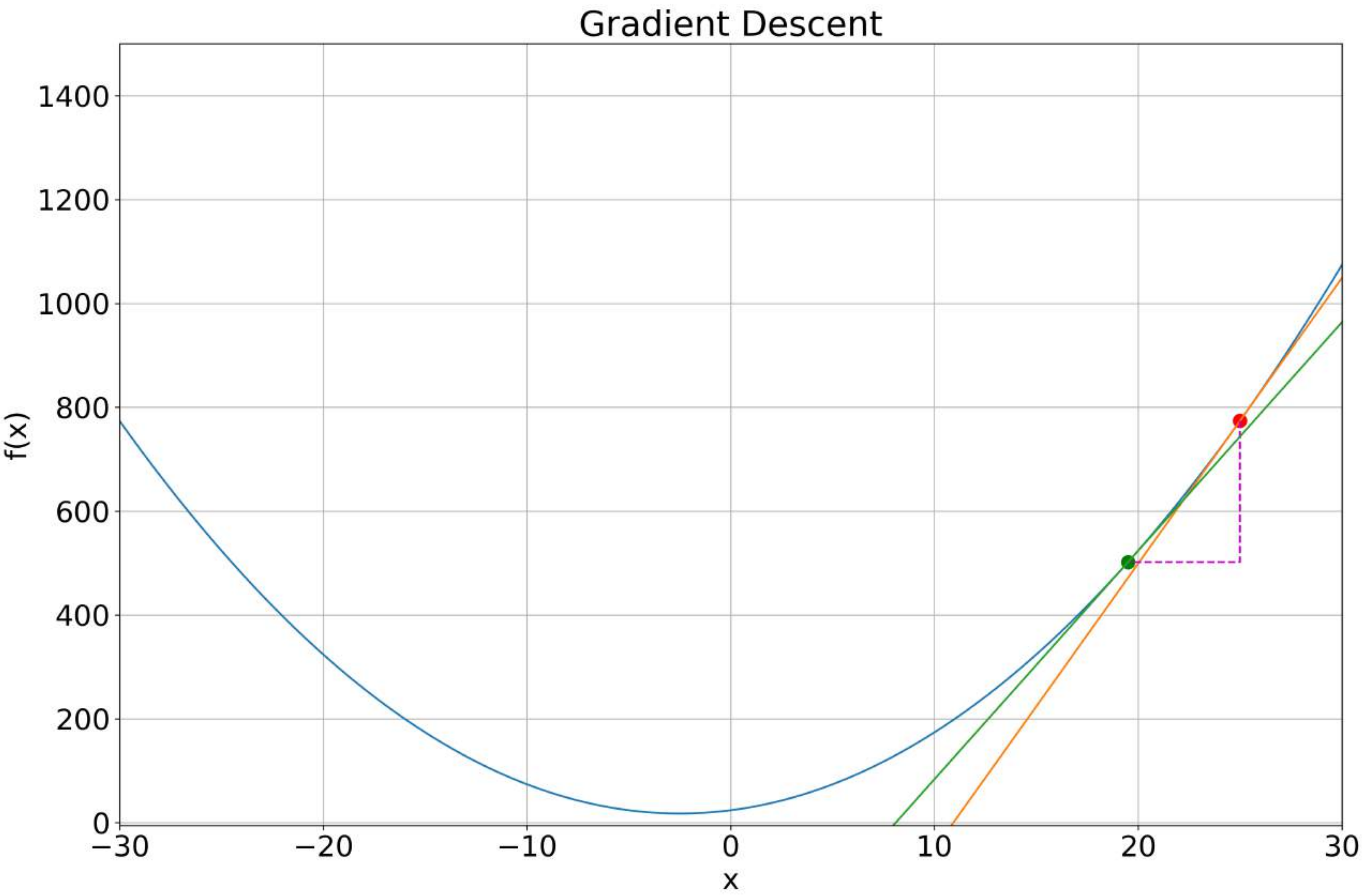
Linear Regression



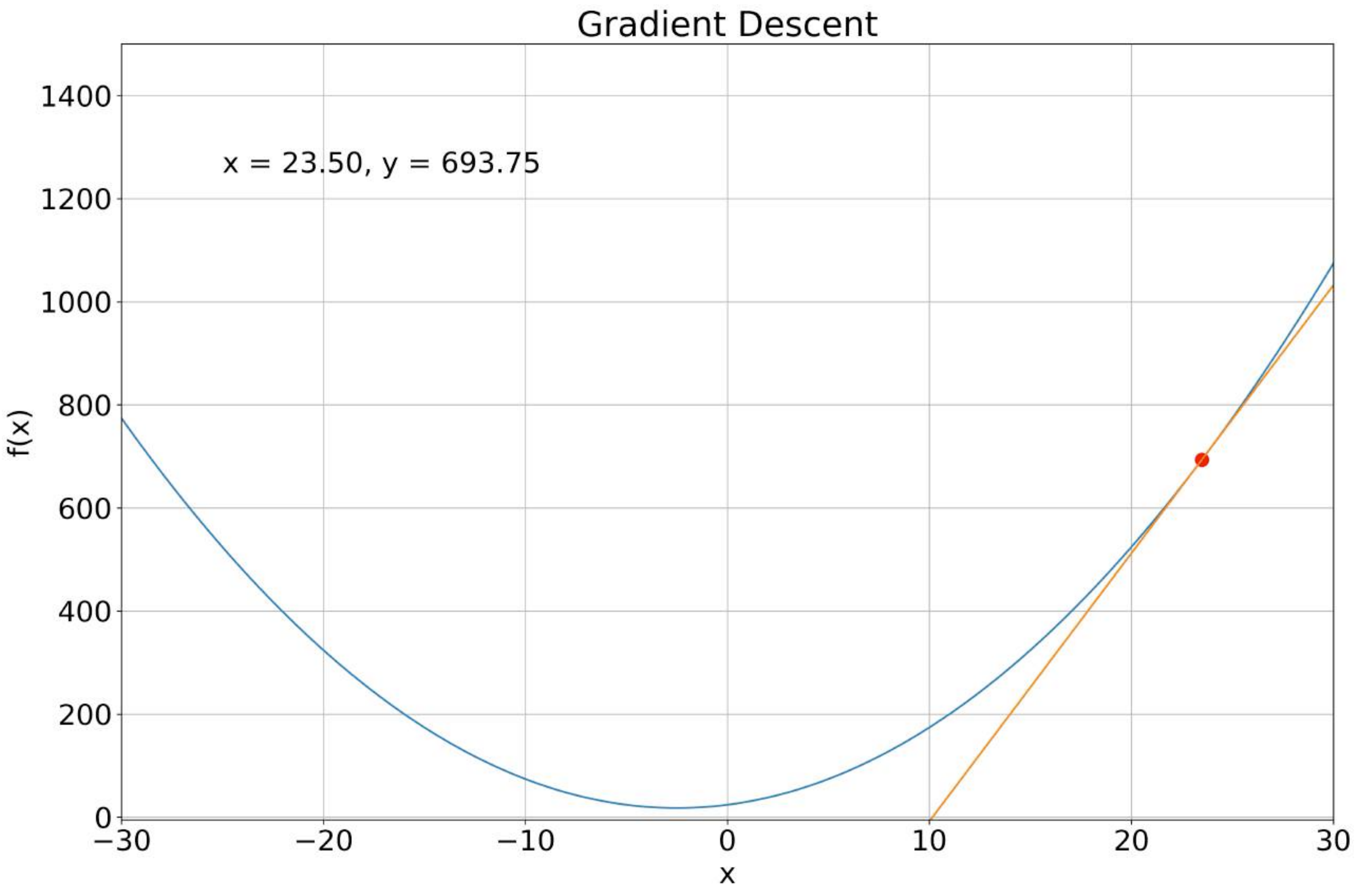
How to find parameters?



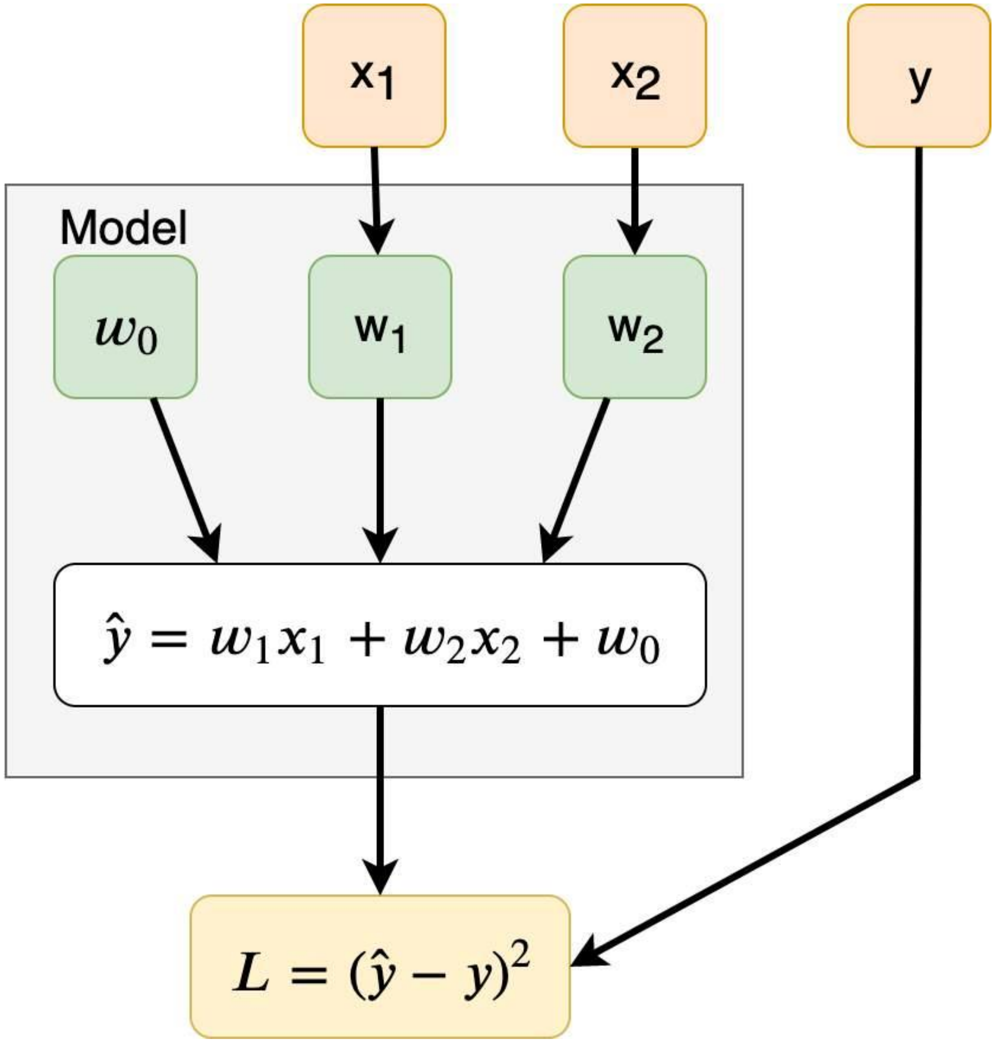
Gradient Descent



Gradient Descent



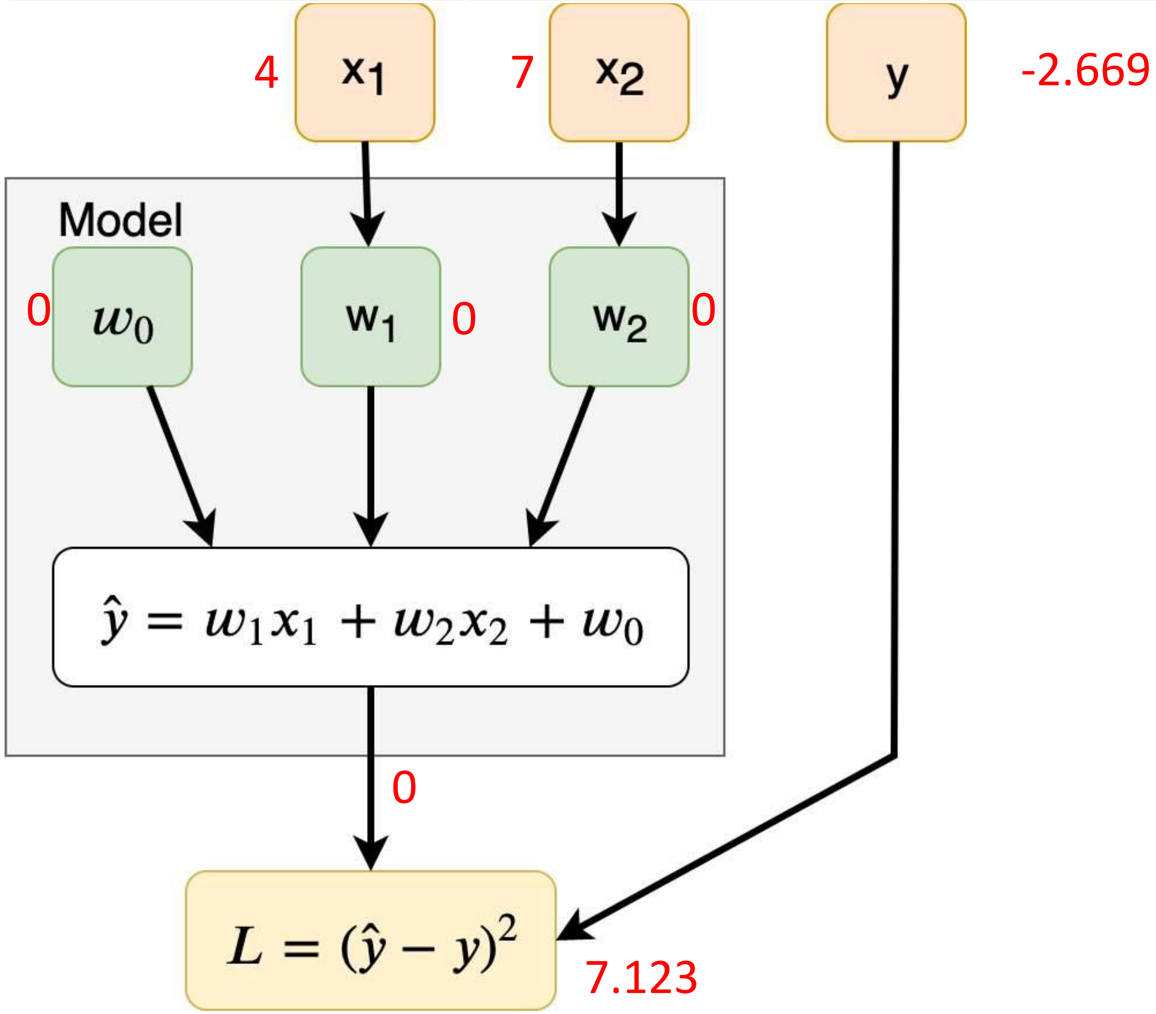
Gradient Decent Training



Guess values for w_0, w_1 w_2 - say all equal to zero

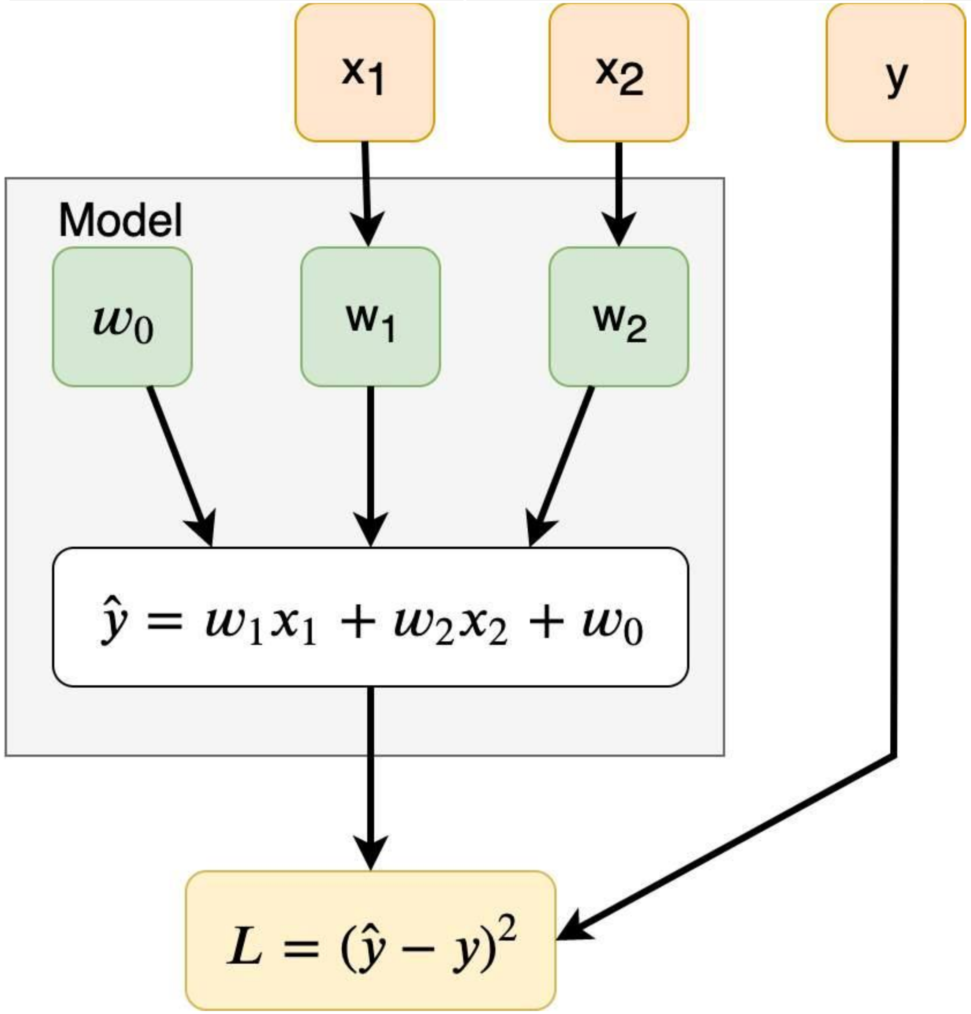
Forward Pass

Observation #	x_1	x_2	y
1	4	7	-2.669



Backward Pass – Update Weights

Observation #	x_1	x_2	y
1	4	7	-2.669



$$w_0' = w_0 - \eta \frac{\partial L}{\partial w_0}$$

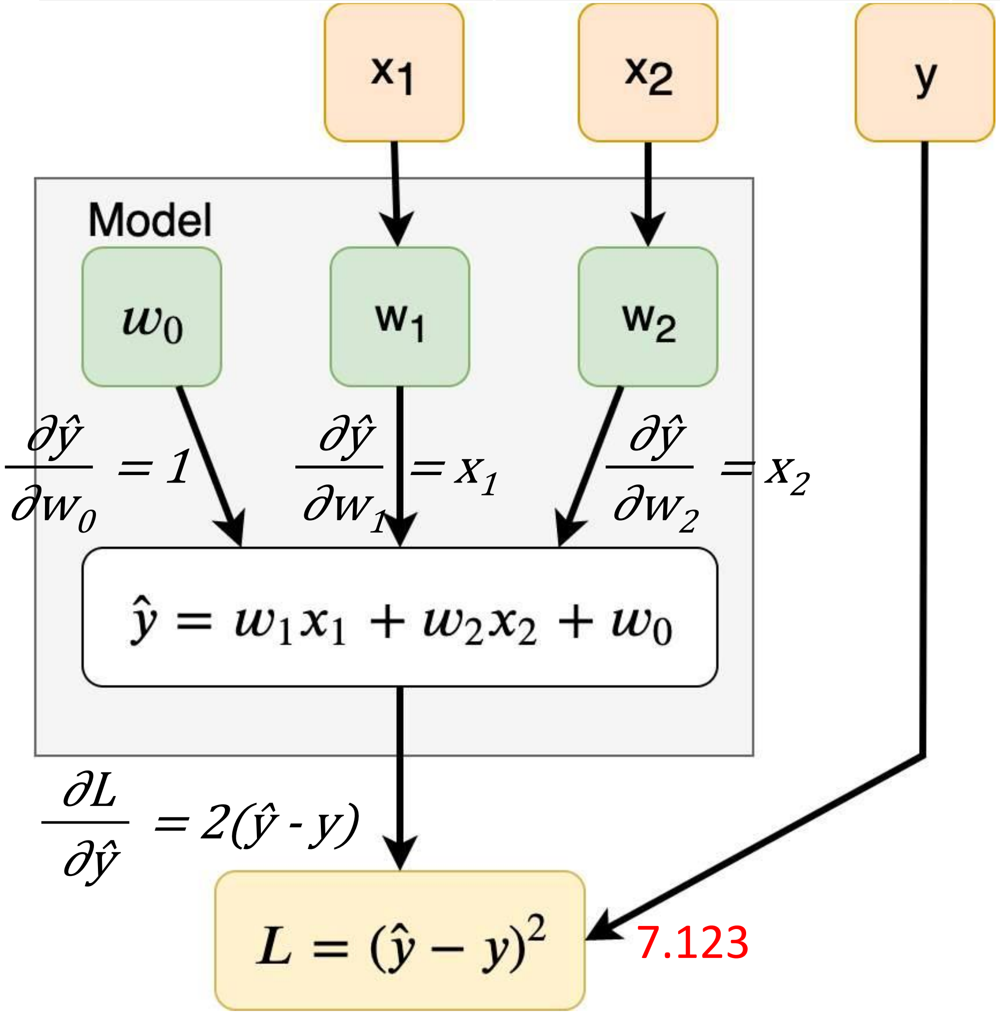
$$w_1' = w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$w_2' = w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$\eta = 0.1$$

Backward Pass – Compute Gradients

Observation #	x_1	x_2	y
1	4	7	-2.669



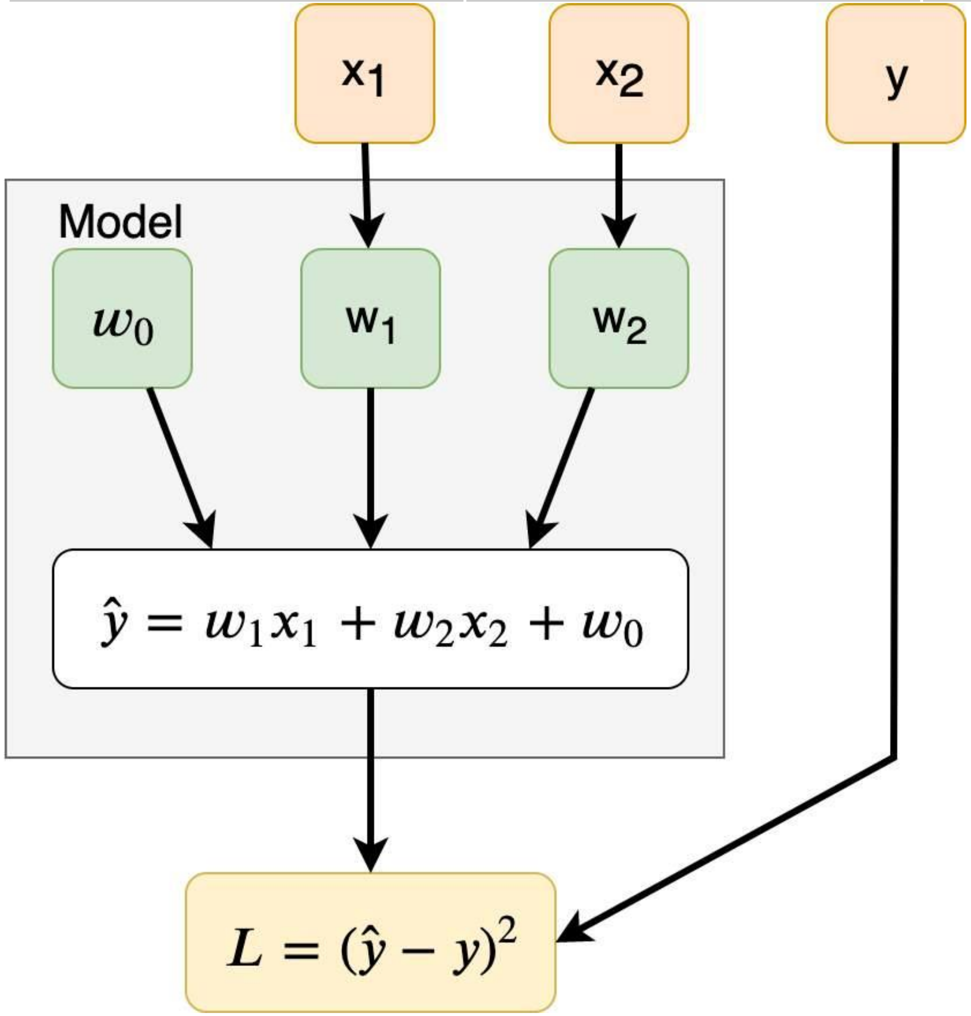
$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_0} = 2(\hat{y} - y) \cdot 1$$
$$= 2 \cdot (-2.669) = -5.338$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_1} = 2(\hat{y} - y) \cdot x_1$$
$$= 2 \cdot (-2.669) \cdot 4 = -21.352$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_2} = 2(\hat{y} - y) \cdot x_2$$
$$= 2 \cdot (-2.669) \cdot 7 = -37.366$$

Backward Pass – Update Weights

Observation #	x_1	x_2	y
1	4	7	-2.669



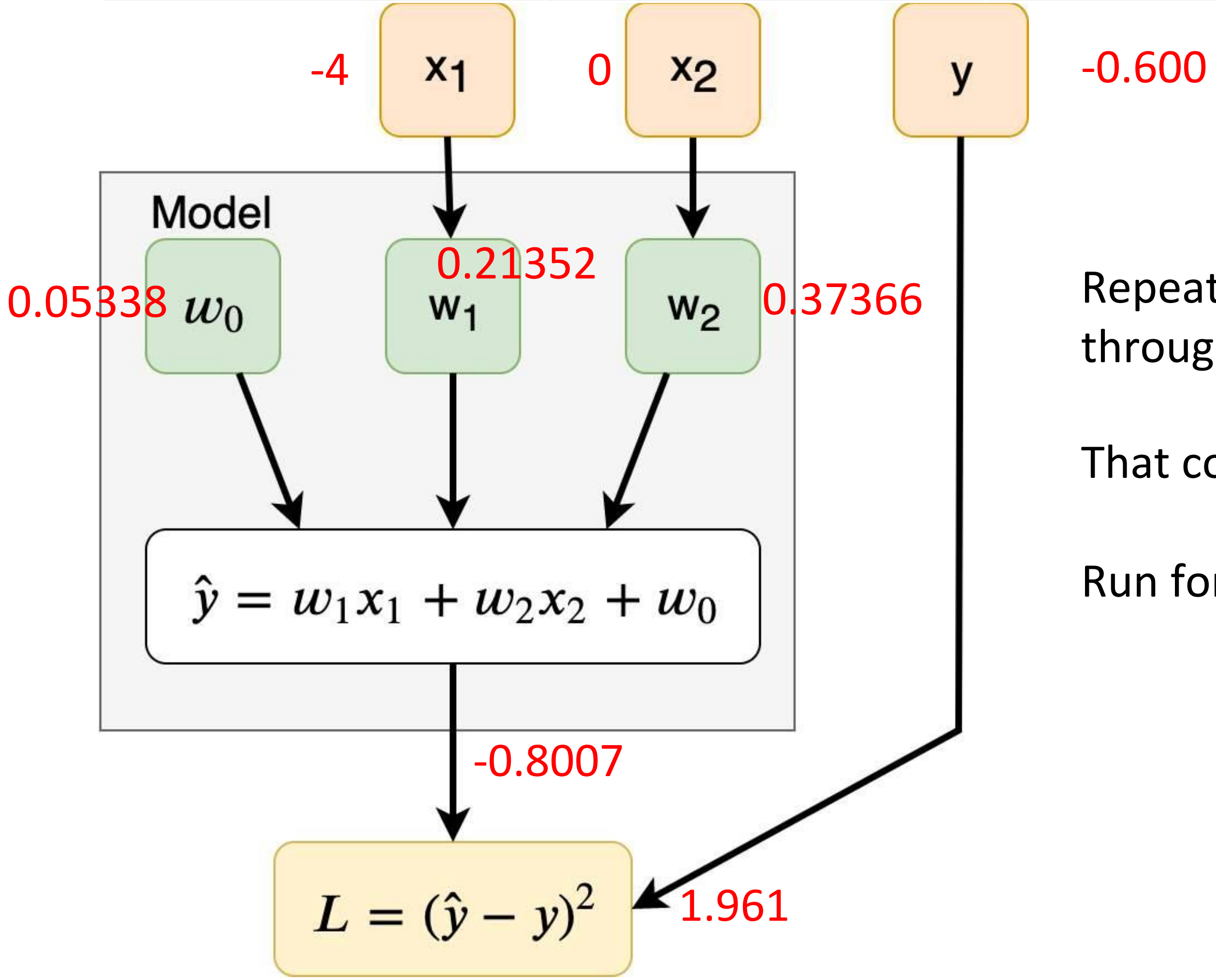
$$w_0' = w_0 - \eta \frac{\partial L}{\partial w_0} = 0.05338$$

$$w_1' = w_1 - \eta \frac{\partial L}{\partial w_1} = 0.21352$$

$$w_2' = w_2 - \eta \frac{\partial L}{\partial w_2} = 0.37366$$

Forward Pass

Observation #	x_1	x_2	y
2	-4	0	-0.600



Repeat until all data have been pushed through the network.

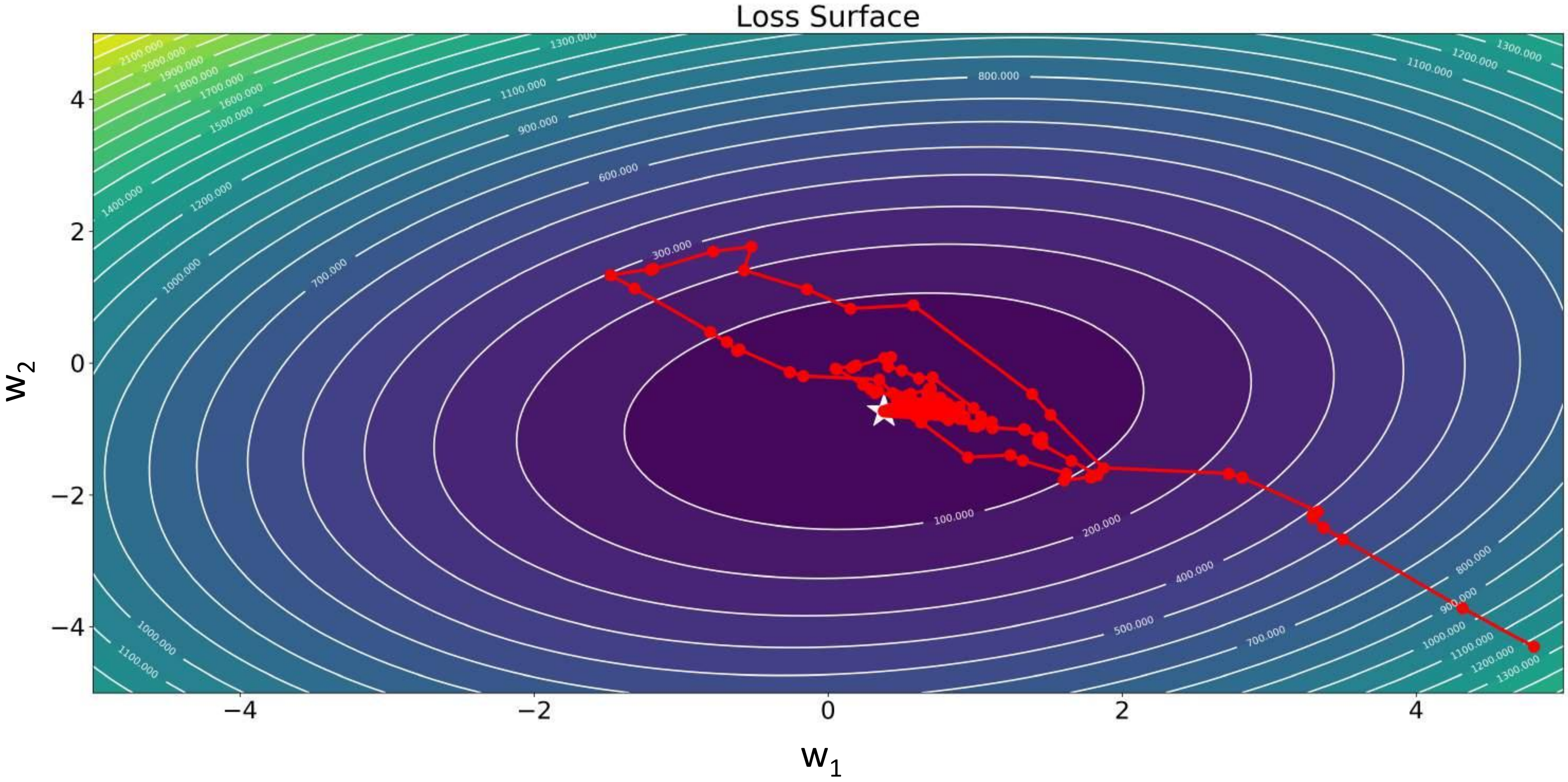
That constitutes 1 epoch

Run for multiple epochs

SGD Training Loop

```
for epoch in range(number_of_epochs):
    for sample in training_data:
        forward_pass(sample)
        compute_gradients_from_loss()
        update_model_parameters()
```

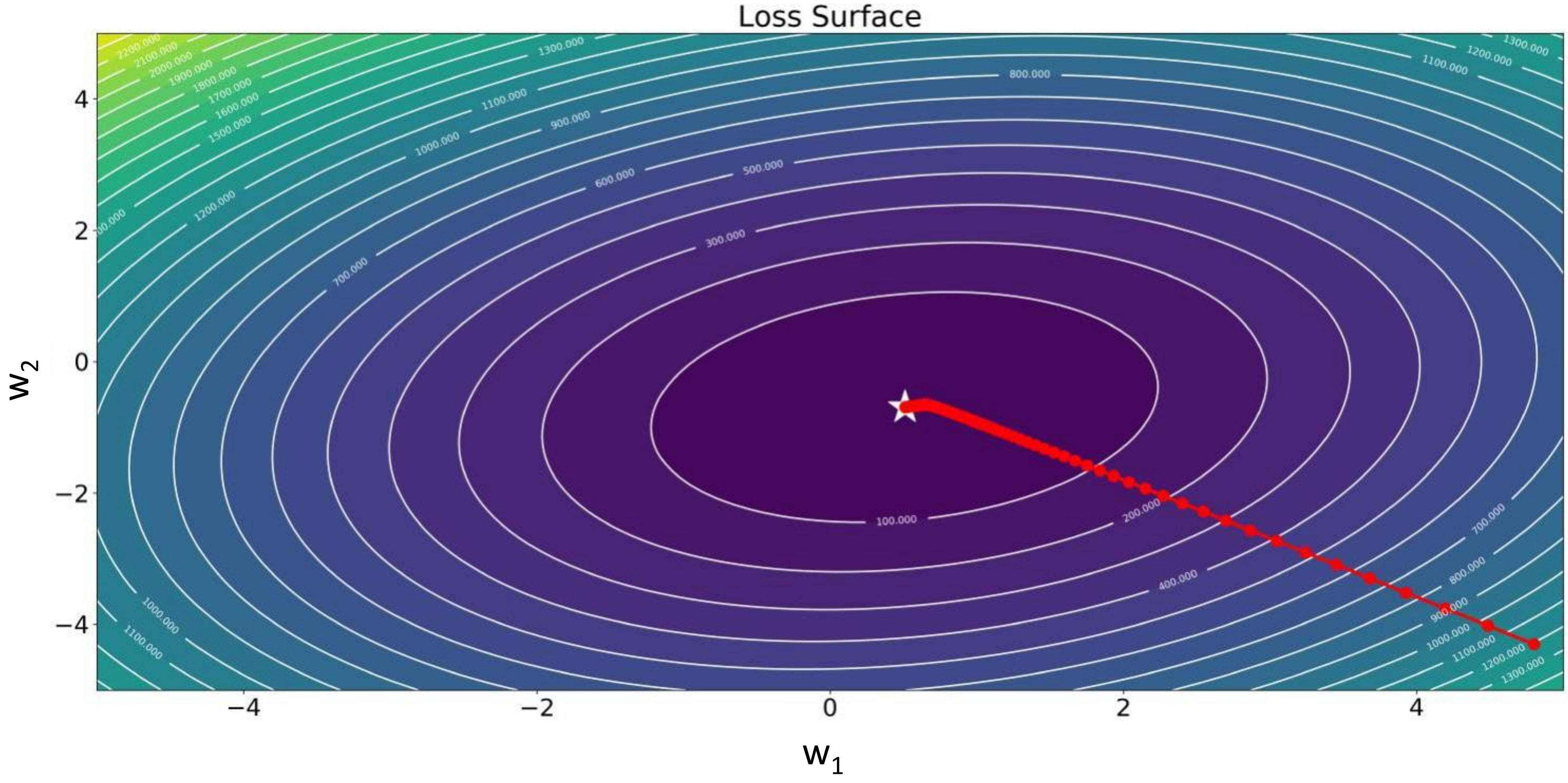
Loss Surface



Batch GD Training Loop

```
for epoch in range(number_of_epochs):
    for sample in training_data:
        forward_pass(sample)
        compute_gradients_from_loss()
        accumulate_average_grads()
    update_model_parameters()
```

Loss Surface



Mini-Batch GD Training Loop

```
for epoch in range(number_of_epochs):
  for batch in n_batches:
        for sample in batch:
          forward_pass(sample)
          compute_gradients_from_loss()
          accumulate_average_grads()
      update_model_parameters()
```

Gradient Descent Variants

- Stochastic Gradient Descent
- Batch Gradient Descent
- Mini-Batch Gradient Descent

Stochastic Gradient Descent (SGD)

- Push one data sample through the network
- Back propagate gradients – update weights

Pros:

- Fast computation per pass
- Frequent updates

Cons:

- Noisy gradient signal (jumping around the loss surface) and harder to settle
- Update frequency may end up requiring more computation

Batch Gradient Descent

- Push all data through the network
- Propagate average gradients

Pros:

- Optimal (true) gradient computed
- Stable Error

Cons:

- Slow computation (all data/pass)
- Requires whole dataset in memory

Mini-Batch SGD

- Push a portion of data through the network
- Propagate average gradients

Pros:

- Intermediate Update Frequency
- Higher computational efficiency

Cons:

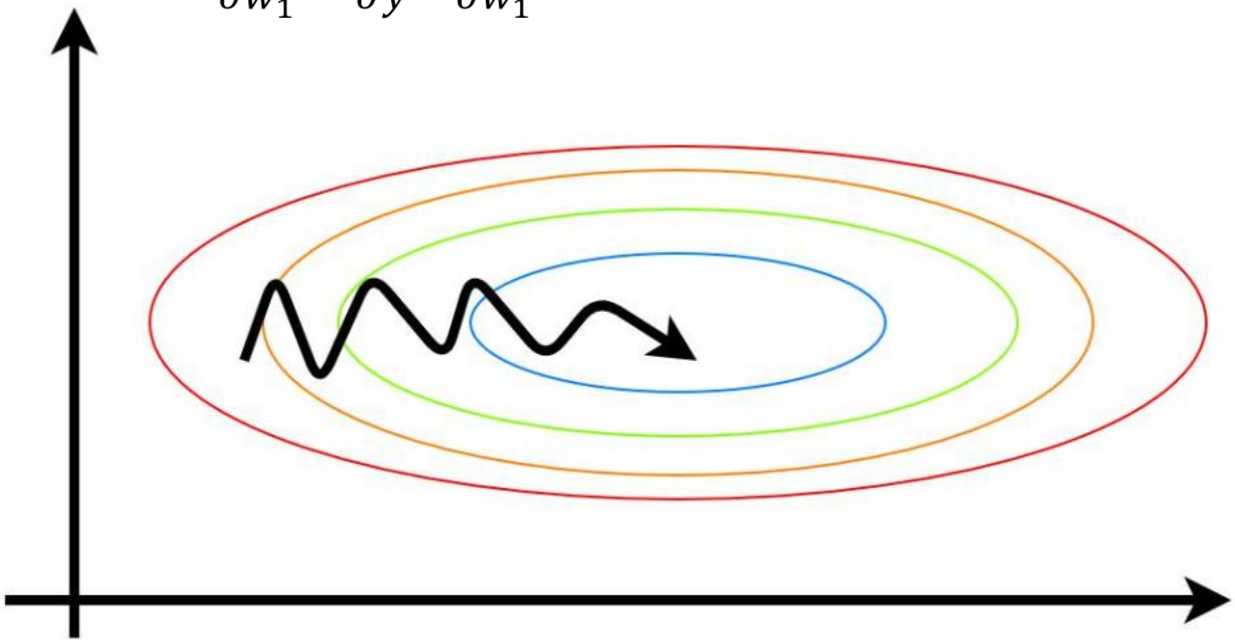
- Requires an additional parameter to configure
- Error information must be accumulated across mini-batches

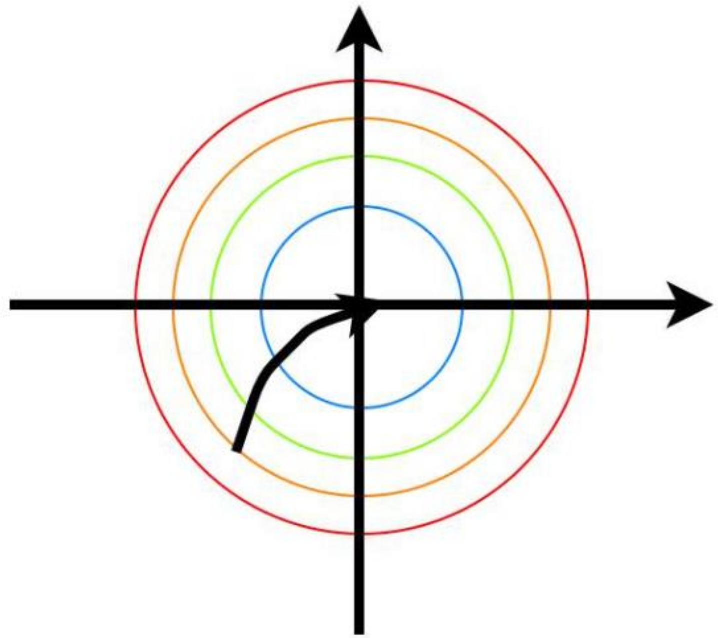
When do you stop training?

- After a set number of iterations (epochs)
- When the Loss for the training dataset falls below a threshold

Normalisation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_1}$$



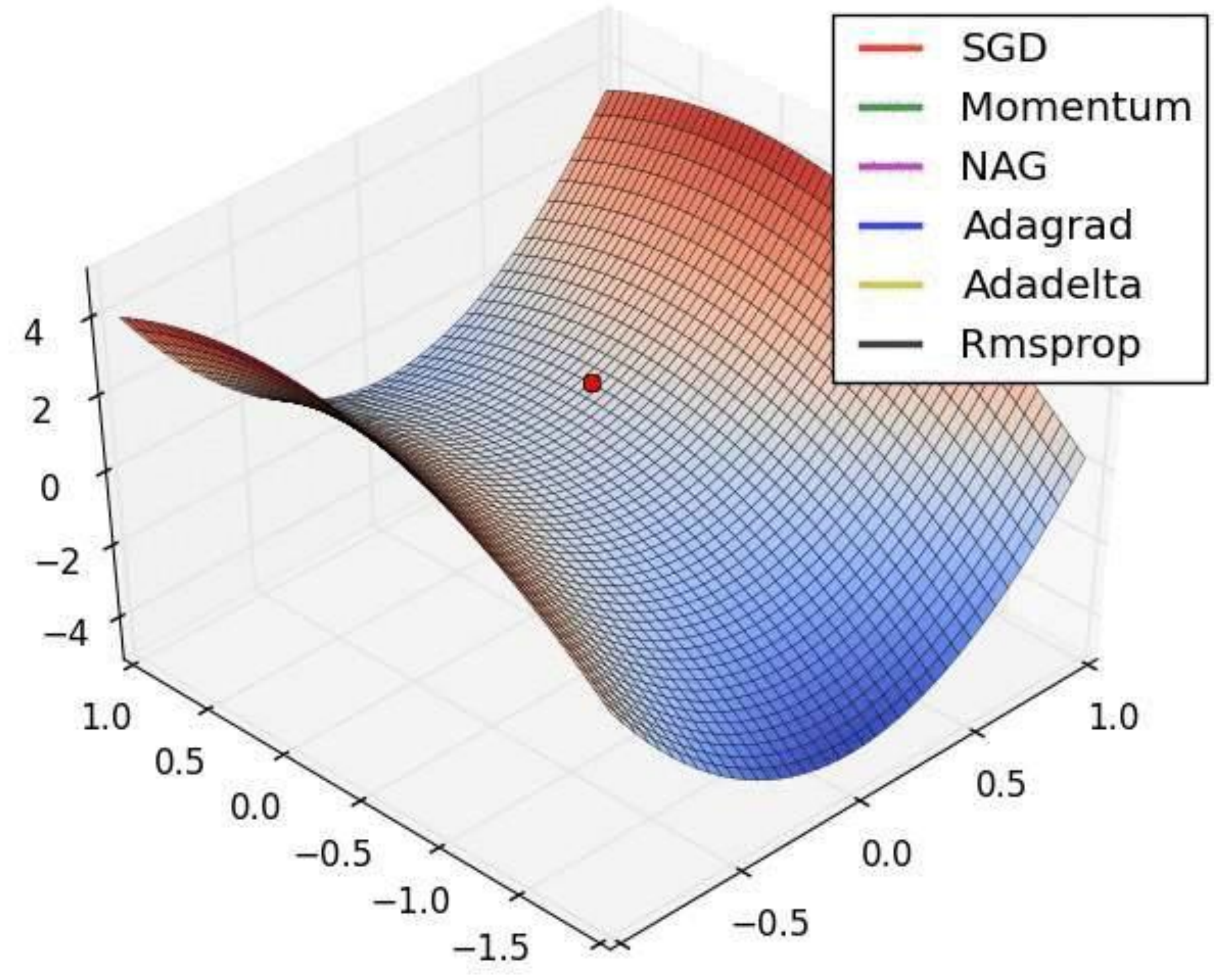


$$x' = \frac{x - \mu}{\sigma}$$

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{P}}$$

Fancy Optimisers



Fancy Optimisers