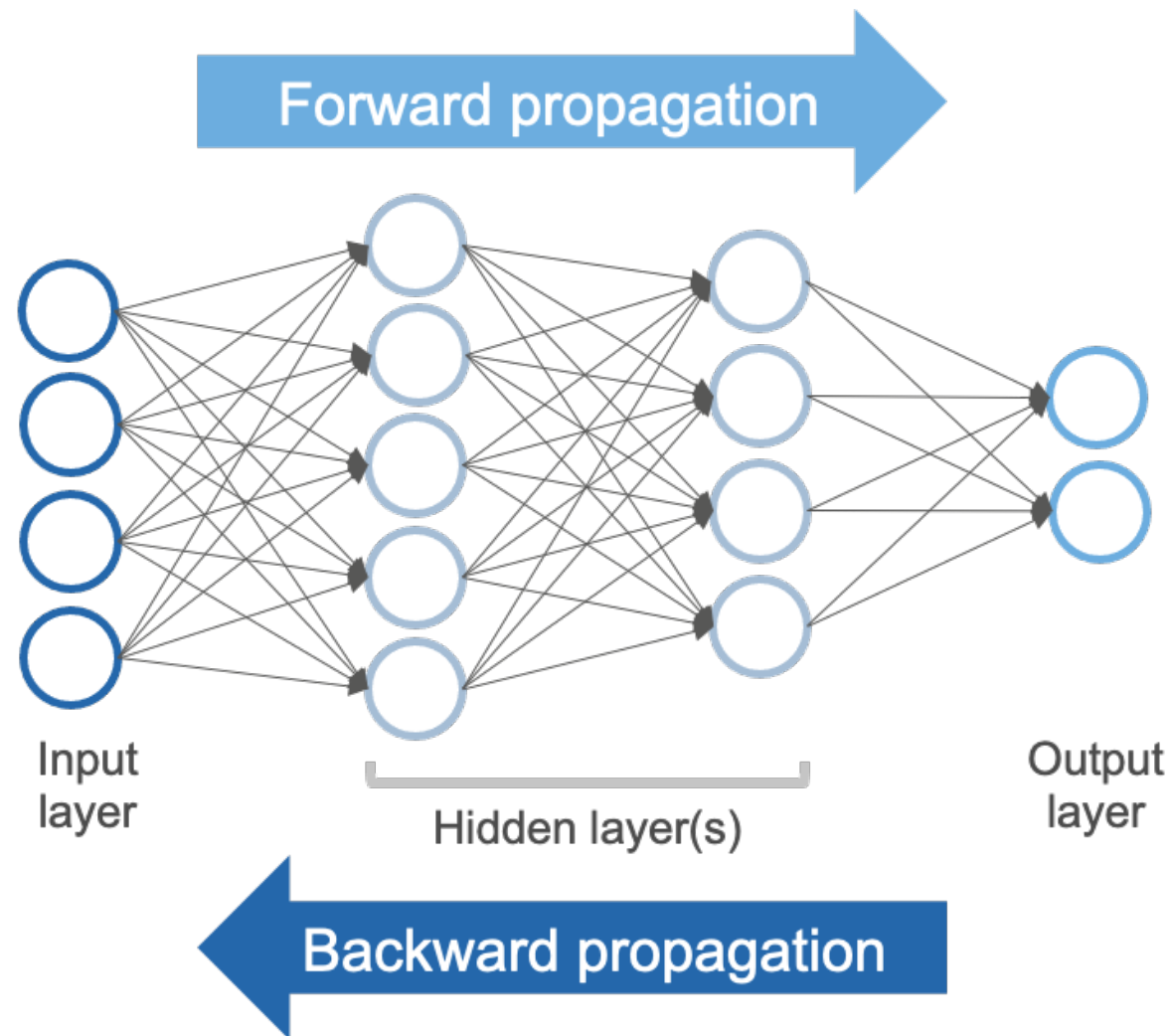
A close-up photograph of a hand holding a string figure. The string is red and is looped around the fingers in a complex, interconnected pattern. The background is a soft, out-of-focus grey. The overall image serves as a metaphor for neural networks and their intricate connections.

Introduction to Neural Networks Operations and Distributed Training

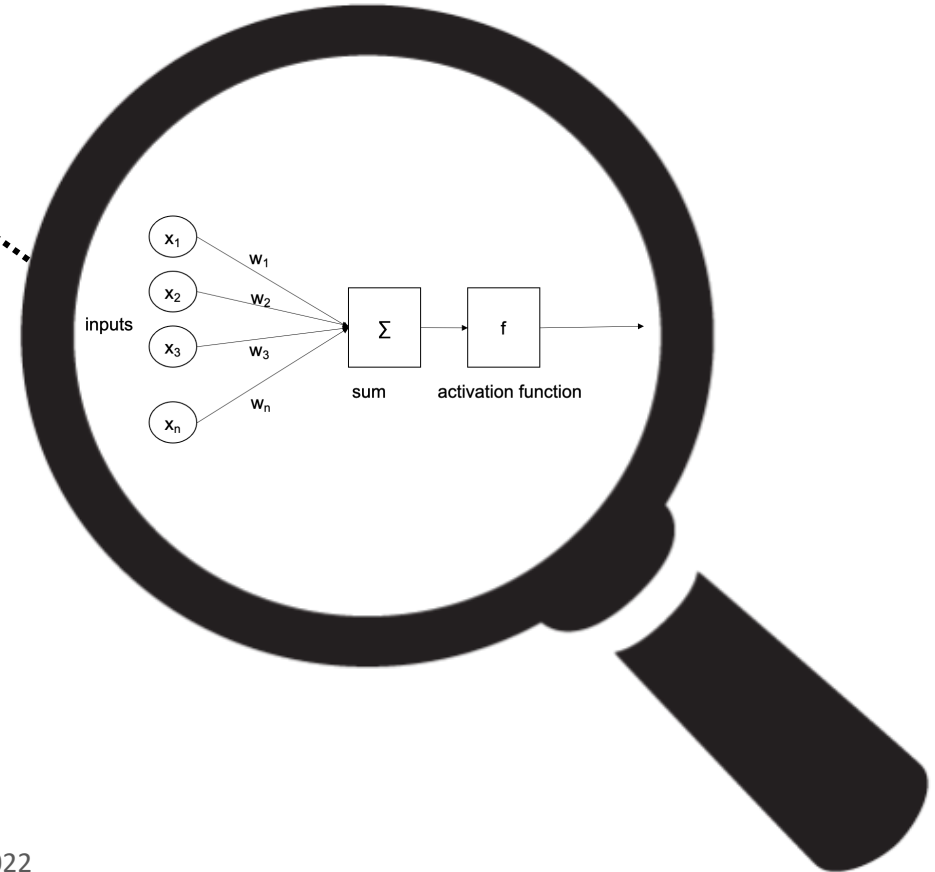
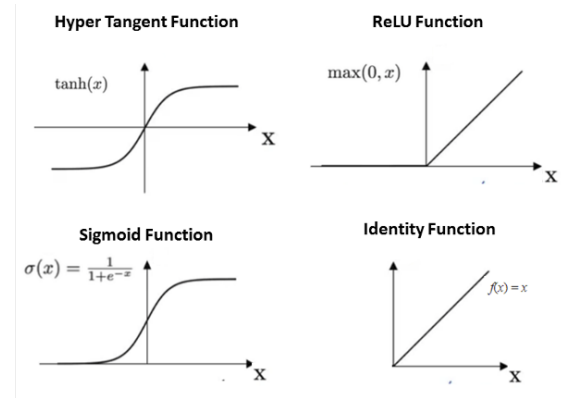
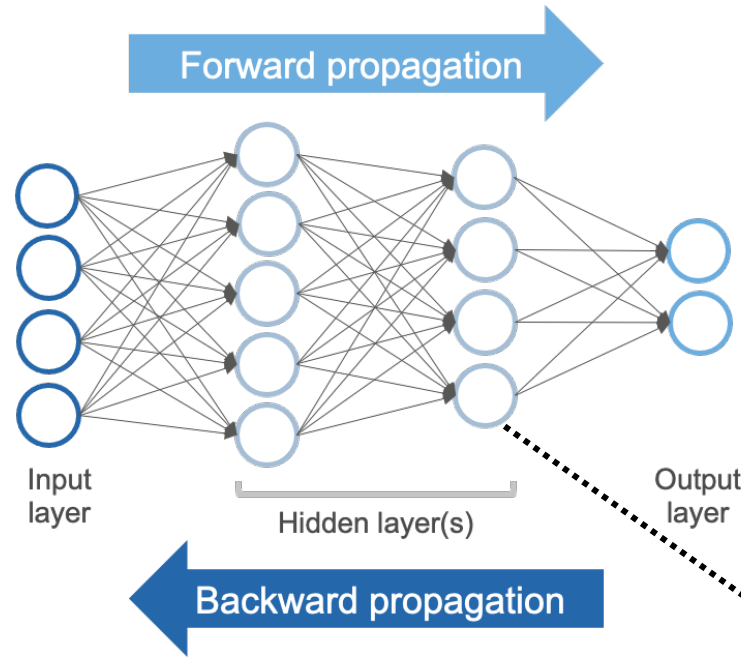
PD. Dr. Juan J. Durillo

Deep Learning and GPU Programming
Workshop @ CSC 10.5-13.5.2022

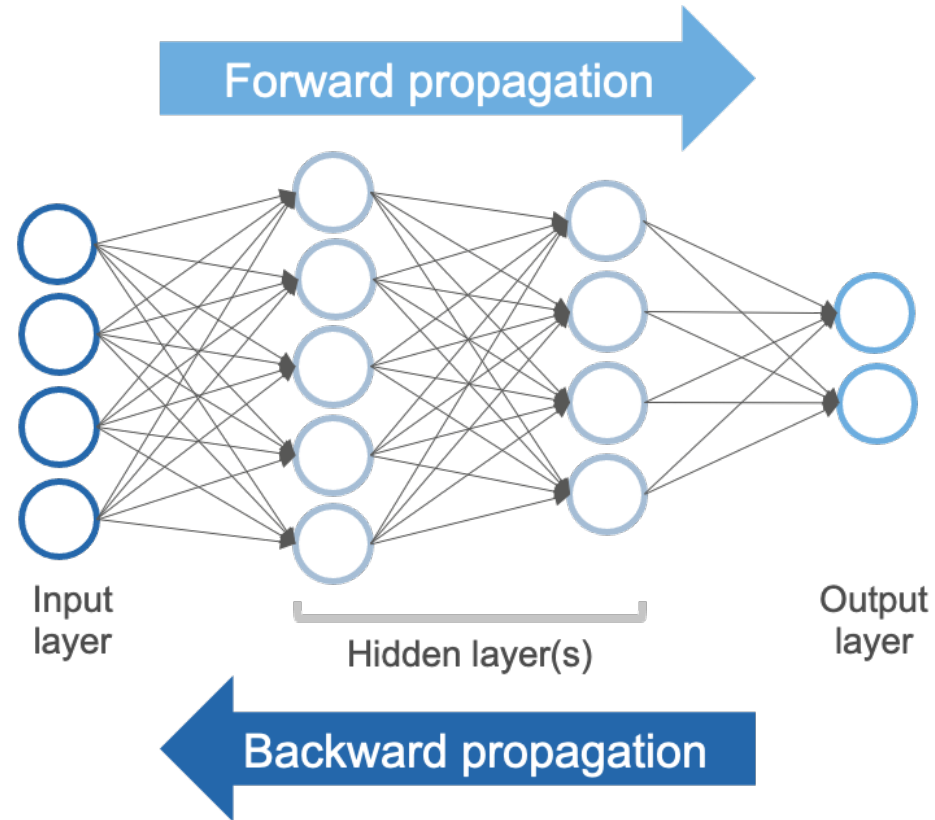
Machine Learning with (deep) Neural Networks



Machine Learning with (deep) Neural Networks



Machine Learning with (deep) Neural Networks



Loss function

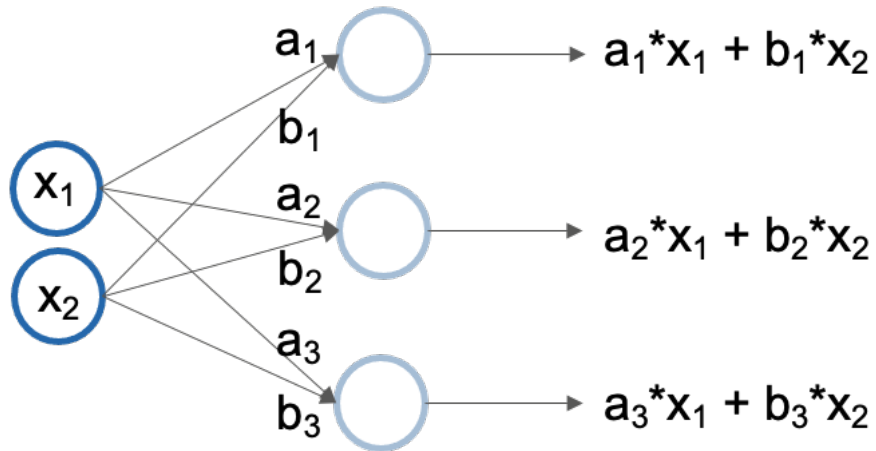
$$l = MSE = \frac{\sum_{i=1}^n (y_i - y_i^p)^2}{n}$$

Optimizer SGD

$$\theta_t \leftarrow \theta_{t-1} - n_t * g(\theta_t; B_t)$$

$$g(\theta_t; B_t) = \frac{1}{|B_t|} \sum_{z \in B_t} \nabla l(\theta_t; z)$$

Forward Operations



Matrix Multiplication Operation

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a_1 x_1 + b_1 x_2 \\ a_2 x_1 + b_2 x_2 \\ a_3 x_1 + b_3 x_2 \end{pmatrix}$$

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix} = \begin{pmatrix} a_1 x_1 + b_1 x_2 & a_1 y_1 + b_1 y_2 \\ a_2 x_1 + b_2 x_2 & a_2 y_1 + b_2 y_2 \\ a_3 x_1 + b_3 x_2 & a_3 y_1 + b_3 y_2 \end{pmatrix}$$

batch of two inputs

Forward Operations

Convolutions as Matrix Multiplication

1	2	3
4	5	6
7	8	9

a	b	c	j						
d	e	f	k						
g	h	i	l						

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

a	b
b	c
c	j
d	e
e	f
f	k
g	h
h	i
i	l

Duplication of elements
Built of special matrices causes overhead

q_{11}	q_{12}	...	q_{1d}
q_{21}	q_{22}	...	q_{2d}
...
q_{n1}	q_{n2}	...	q_{nd}

query matrix Q

k_{11}	k_{21}	...	k_{n1}
k_{12}	k_{22}	...	k_{n2}
...
k_{1d}	k_{2d}	...	k_{nd}

key matrix K^T

$Q \times K^T$

s_{11}	s_{12}	...	s_{1n}
s_{21}	s_{22}	...	s_{2n}
...
s_{n1}	s_{n2}	...	s_{nn}

score matrix S
(similarity (dot product) between queries and keys)

Attention Layers are also Matrix Multiplications

s_{11}	s_{12}	...	s_{1n}
s_{21}	s_{22}	...	s_{2n}
...
s_{n1}	s_{n2}	...	s_{nn}

score matrix S
(similarity (dot product) between queries and keys)

Soft Normalization

s'_{11}	s'_{12}	...	s'_{1n}
s'_{21}	s'_{22}	...	s'_{2n}
...
s'_{n1}	s'_{n2}	...	s'_{nn}

normalized score matrix S'
(possibly sparse)

v_{11}	v_{12}	...	v_{1d}
v_{21}	v_{22}	...	v_{2d}
...
v_{n1}	v_{n2}	...	v_{nd}

value matrix V

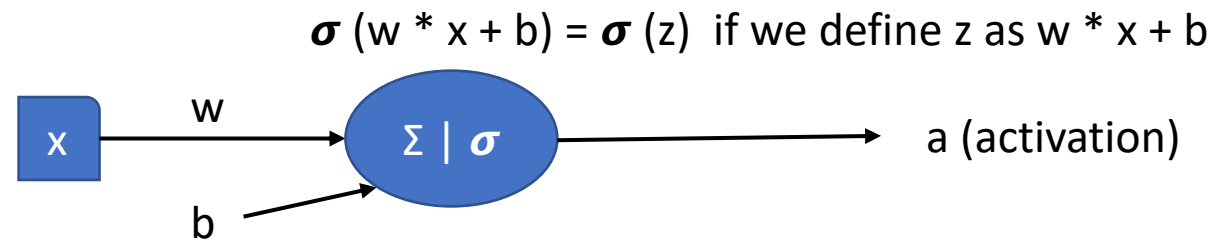
$S' \times V$

o_{11}	o_{12}	...	o_{1d}
o_{21}	o_{22}	...	o_{2d}
...
o_{n1}	o_{n2}	...	o_{nd}

Output matrix V

Back Propagation and Gradient Descent

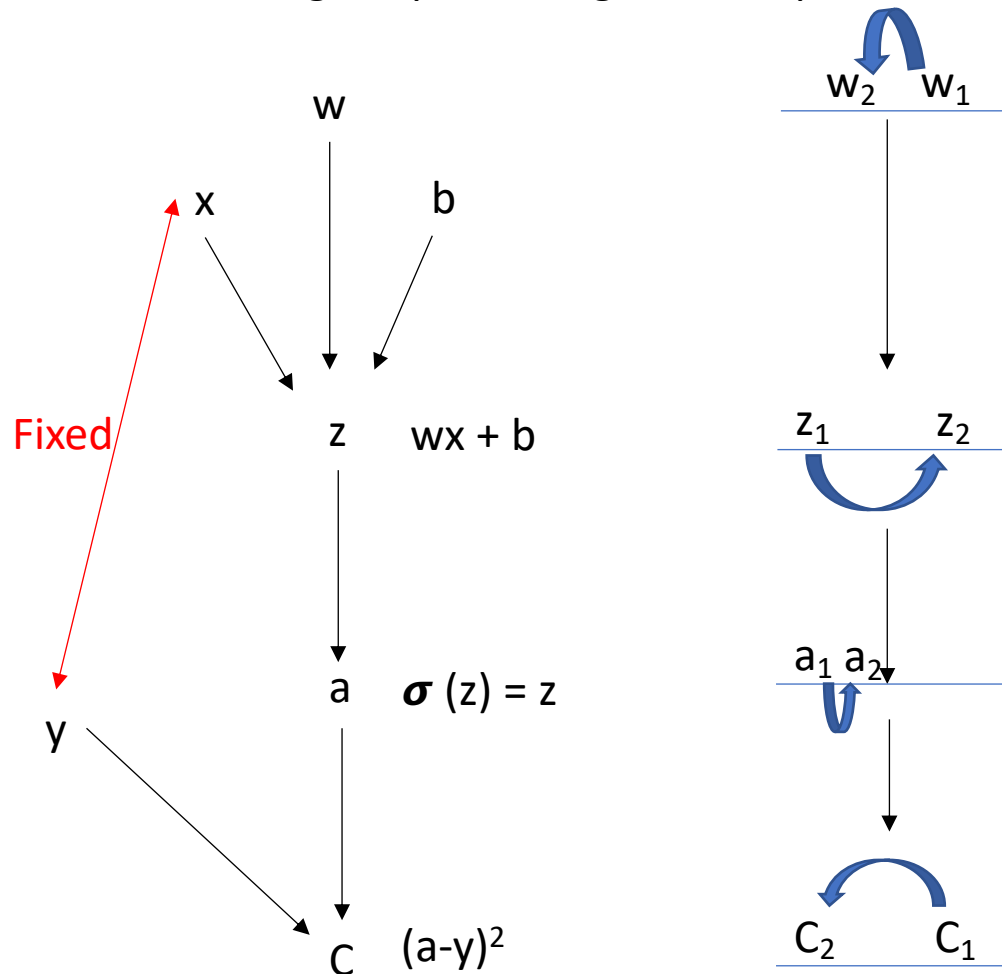
- Linear regression $y = w * x + b$ (i.e., a NN of a single neuron, and identity, $f(x) = x$, as activation function)



- Loss function defined as $C = (a - y)^2$
- How does C change with w and b variations?
 - compute the ratio at which C changes with changes in w and b
 - use this ratio to modify then w and b in order to move C towards a minimum

Computing the Gradient

Gradient with a single input, that generates prediction a



$$\frac{\partial C}{\partial w} = \frac{\partial z}{\partial w} \frac{\partial a}{\partial z} \frac{\partial C}{\partial a} = 2x(a - y)$$

$$\frac{\partial z}{\partial w} = x$$

$$\frac{\partial a}{\partial z} = 1$$

$$\frac{\partial C}{\partial a} = 2(a - y)$$

$$\frac{\partial C}{\partial b} = \frac{\partial z}{\partial b} \frac{\partial a}{\partial z} \frac{\partial C}{\partial a} = 2(a - y)$$

$$\frac{\partial z}{\partial b} = 1$$

$$\frac{\partial a}{\partial z} = 1$$

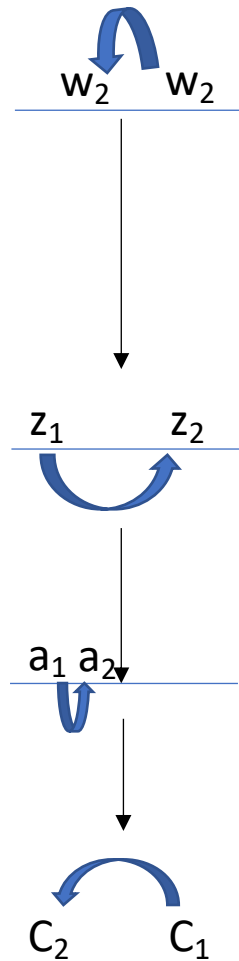
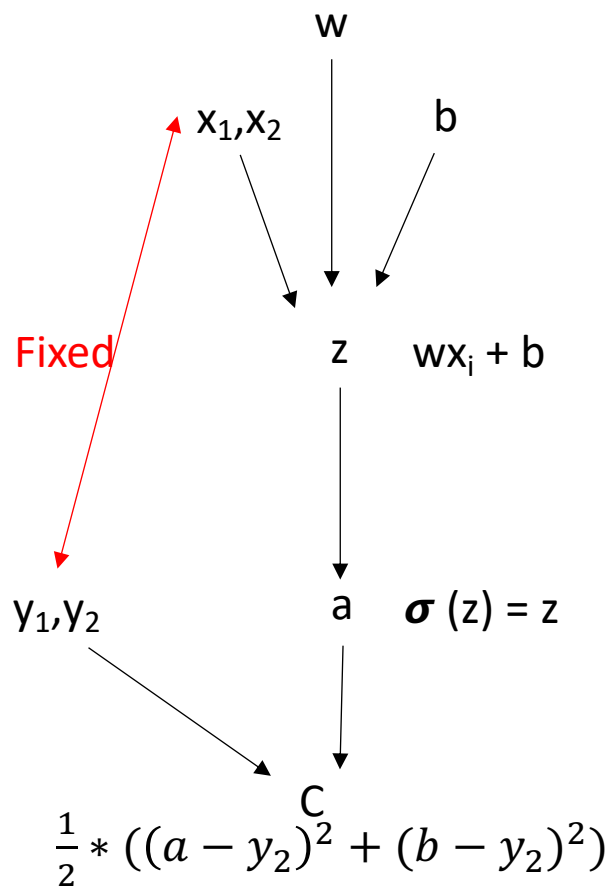
$$\frac{\partial C}{\partial a} = 2(a - y)$$

Gradient Vector

$$\begin{pmatrix} \frac{\partial C}{\partial w} \\ \frac{\partial C}{\partial b} \end{pmatrix} = \begin{pmatrix} 2x(a - y) \\ 2(a - y) \end{pmatrix}$$

Computing the Gradient

Gradient with two inputs that generates predictions:
 a and b



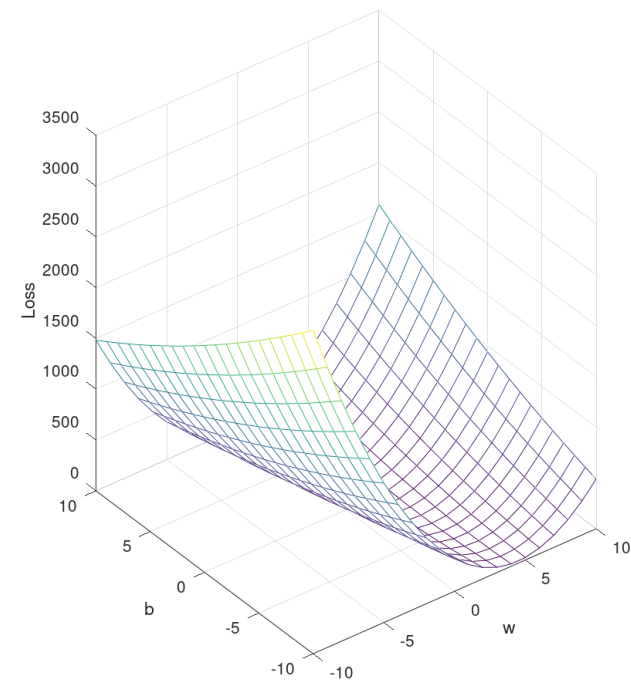
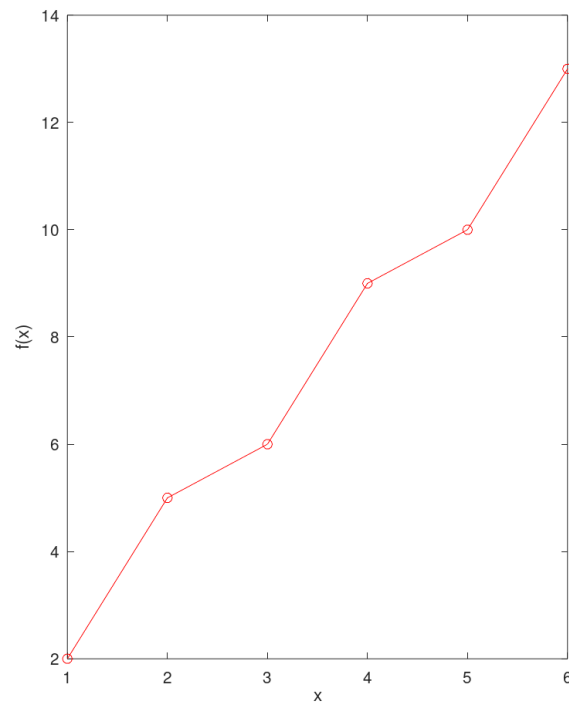
$$\frac{\partial C}{\partial w} = \frac{1}{2} * (2x(a - y_1) + 2x(b - y_2))$$

$$\frac{\partial C}{\partial b} = \frac{1}{2} * (2(a - y_1) + 2(b - y_2))$$

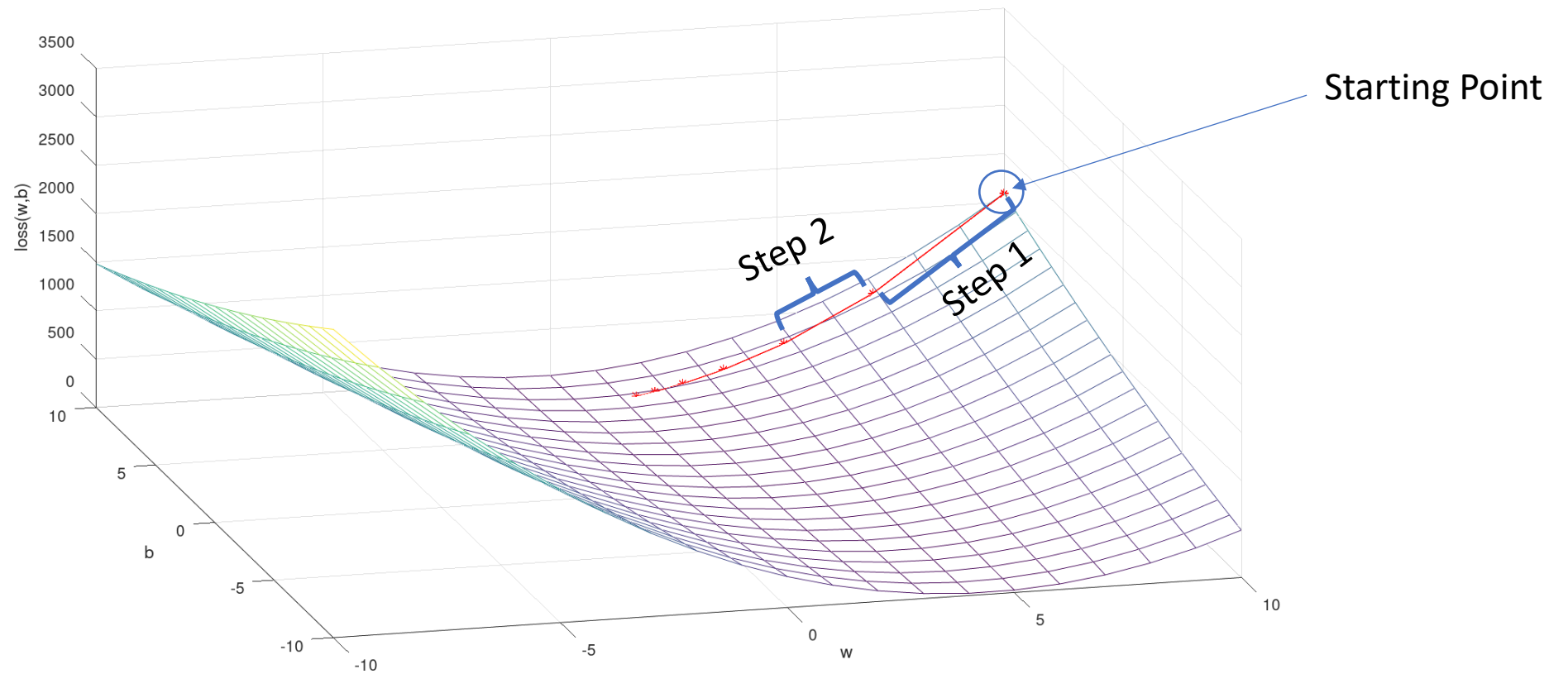
Gradient Vector

$$\begin{pmatrix} \frac{\partial C}{\partial w} \\ \frac{\partial C}{\partial b} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} * (2x(a - y_1) + 2x(b - y_2)) \\ \frac{1}{2} * (2(a - y_1) + 2(b - y_2)) \end{pmatrix}$$

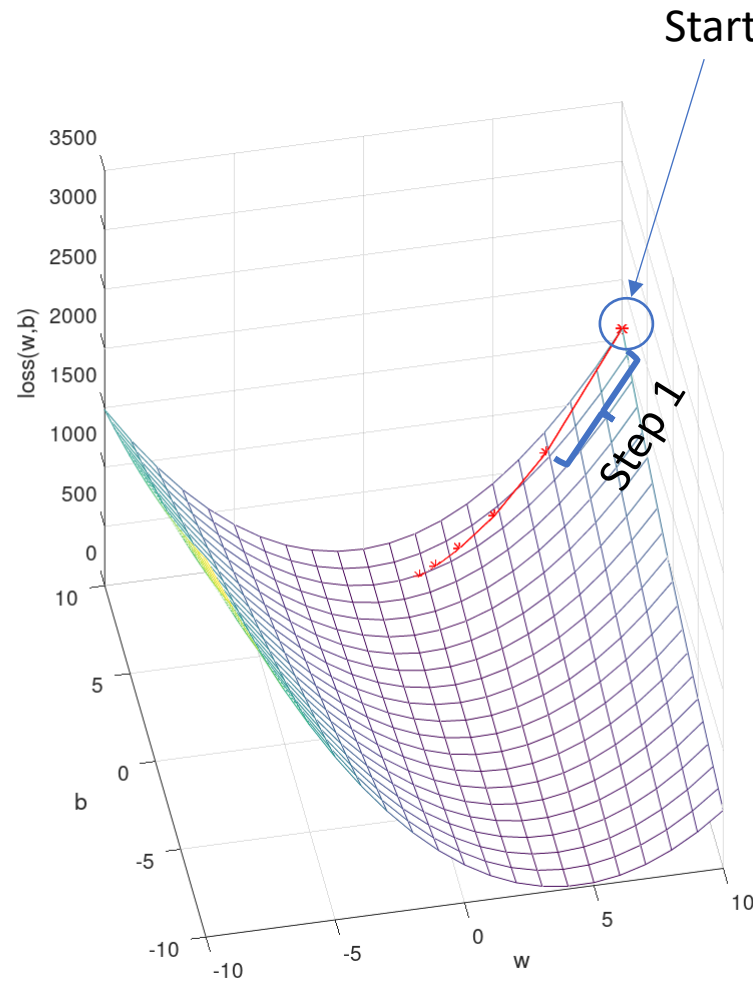
Back Propagation and Gradient Descent



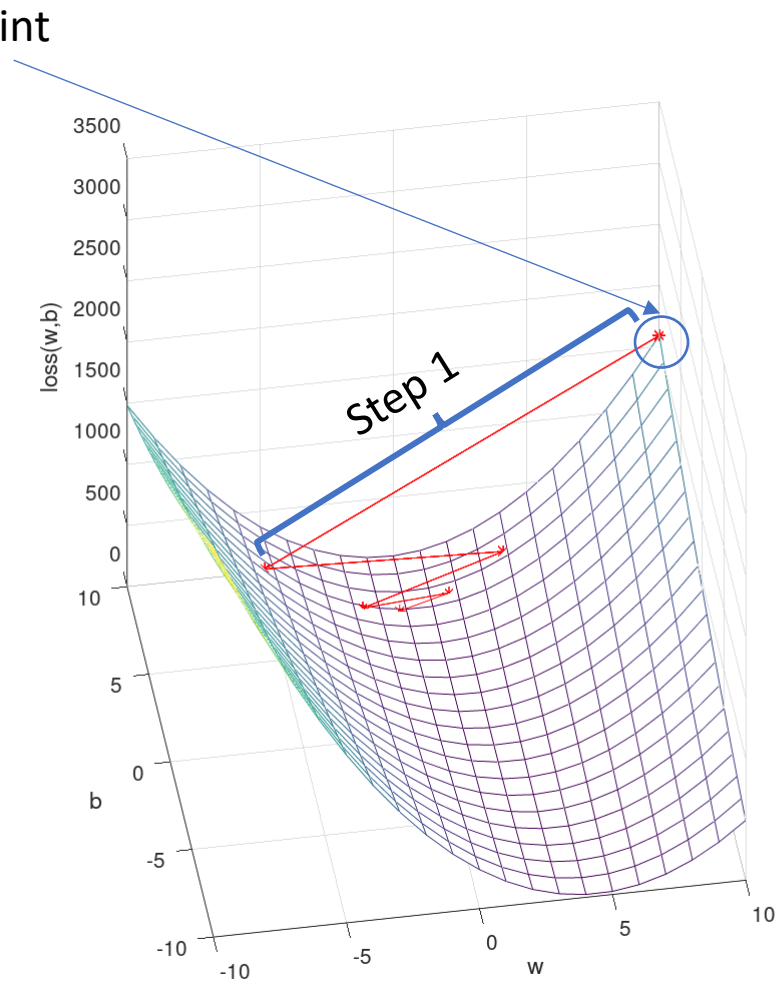
Back Propagation and Gradient Descent



Back Propagation and Gradient Descent

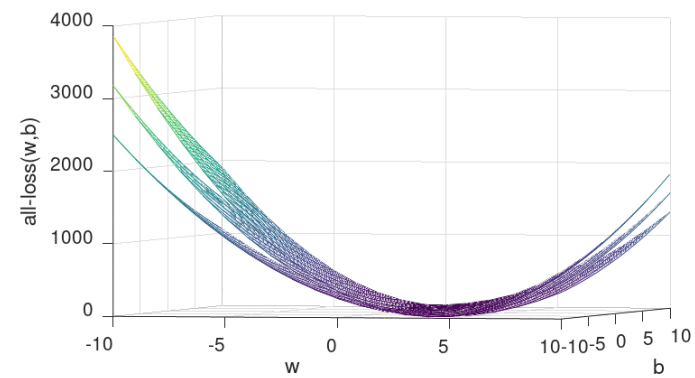
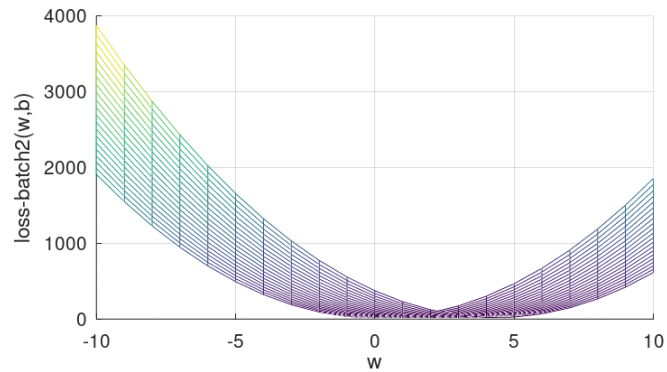
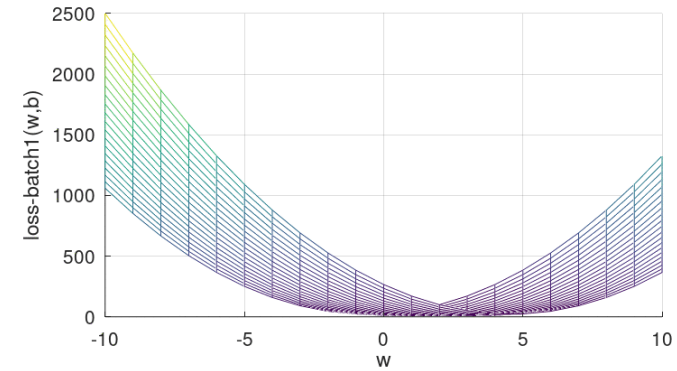
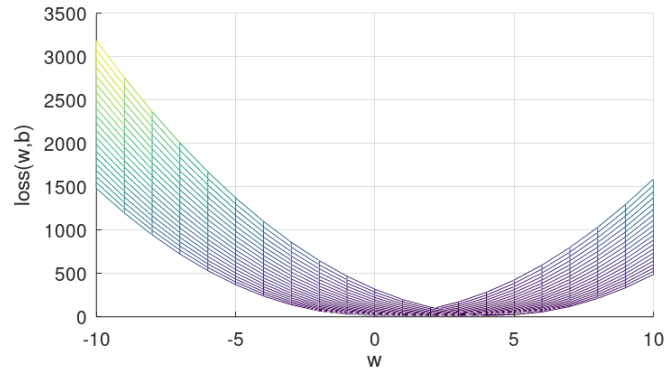


smaller learning rate

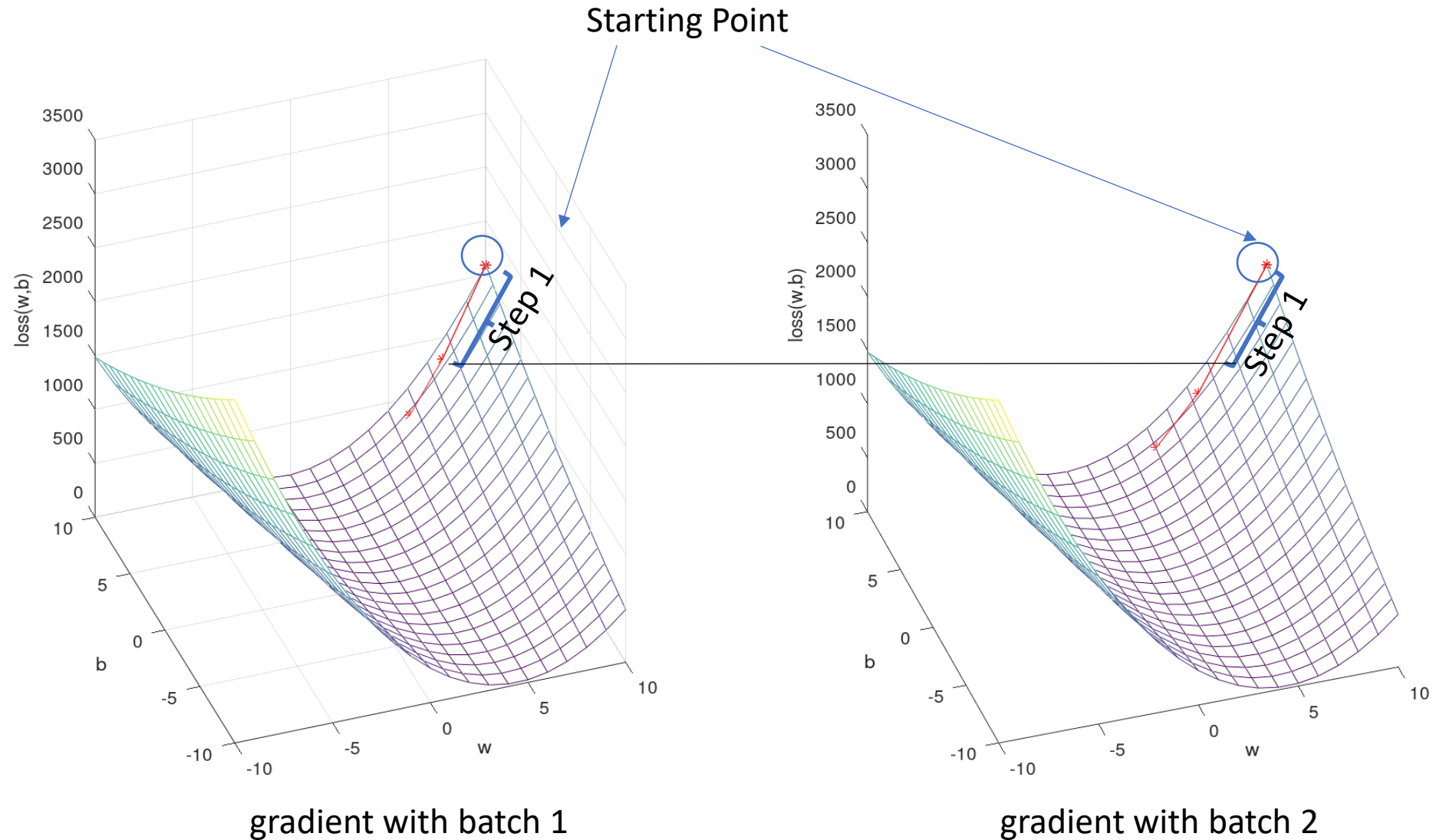


larger learning rate

Back Propagation and Gradient Descent



Back Propagation and Gradient Descent

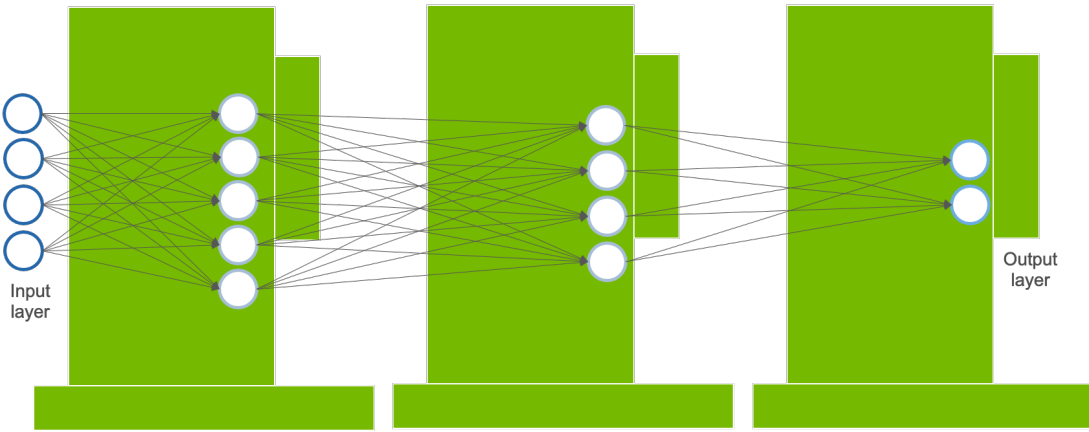


Back Propagation and Gradient Descent

- Batch size implications
 - Smaller batches imply more steps per epoch:
 - More updates to weights --> More updates to the net
 - Smaller batches do not imply larger/smaller gradients



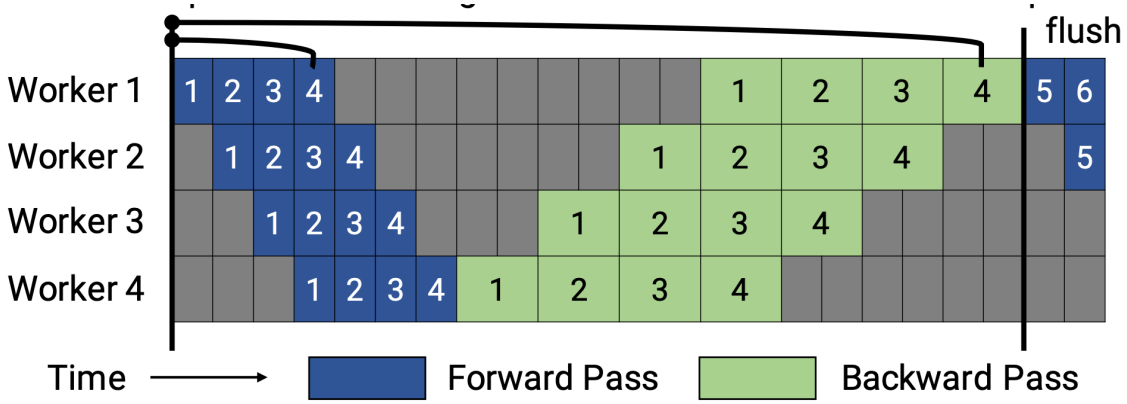
Parallel/Distributed ML Training



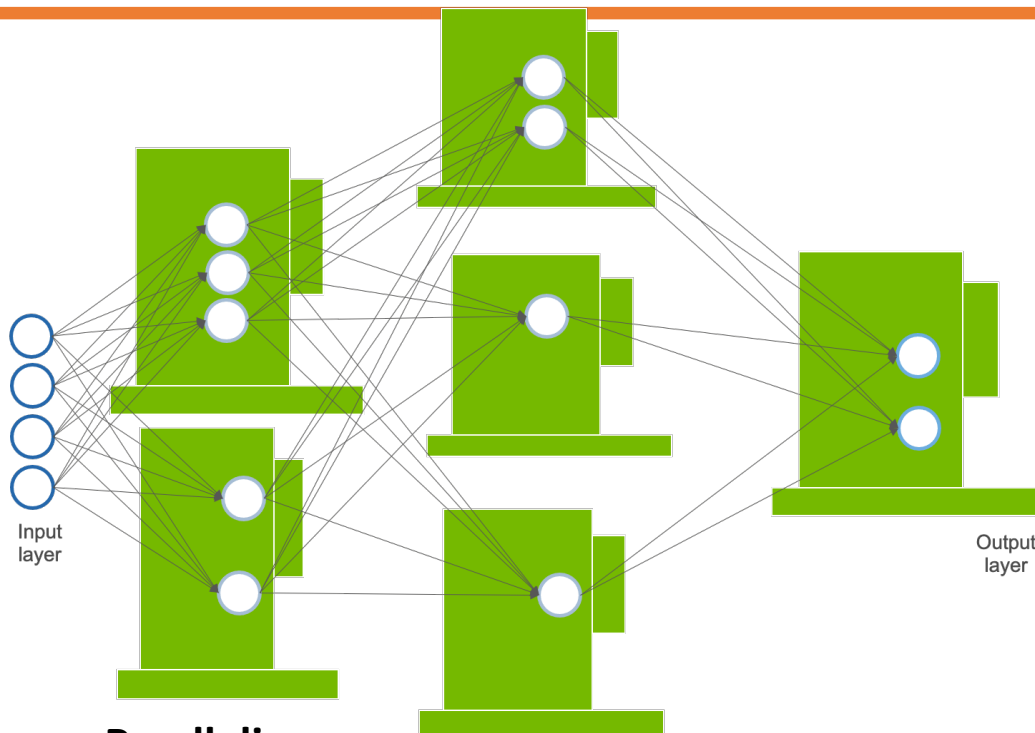
- 1. Model Parallelism: Memory usage and computation of a model distributed across devices
- Two main variants:
 - a) Pipeline parallelism
 - b) Tensor parallelism

Pipeline Model

- Complete layer per device
 - Weights stay within device
- Activations are communicated between GPUs
- Non efficient implementations may lead to inefficient usage of resources
 - Research area



Parallel/Distributed ML Training



1. Model Parallelism: Memory usage and computation of a model distributed across devices

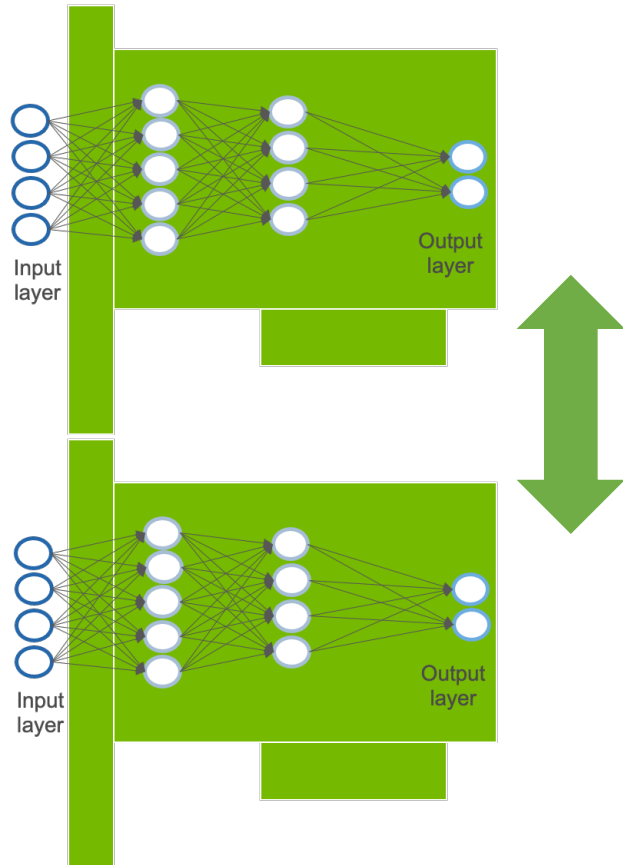
Two main variants:

- Pipeline parallelism
- Tensor parallelism

Tensor Parallelism

- Tensor operations (e.g., computing a layer output) distributed across device
 - Allows larger, more computationally expensive models
- Activations are communicated between GPUs
- Further points for inefficiencies
 - A device might depend on the activations computed by more than one device

Parallel/Distributed ML Training



2. Data Parallelism: Training mini-batch is split across devices

- Model must fit into the memory of a single device
- Weights are the same in each device
 - Gradients are communicated across all devices (all-to-all)