PAT 498/598 (Winter 2025)

Music & AI

Lecture 8: Deep Learning Fundamentals II

Instructor: Hao-Wen Dong

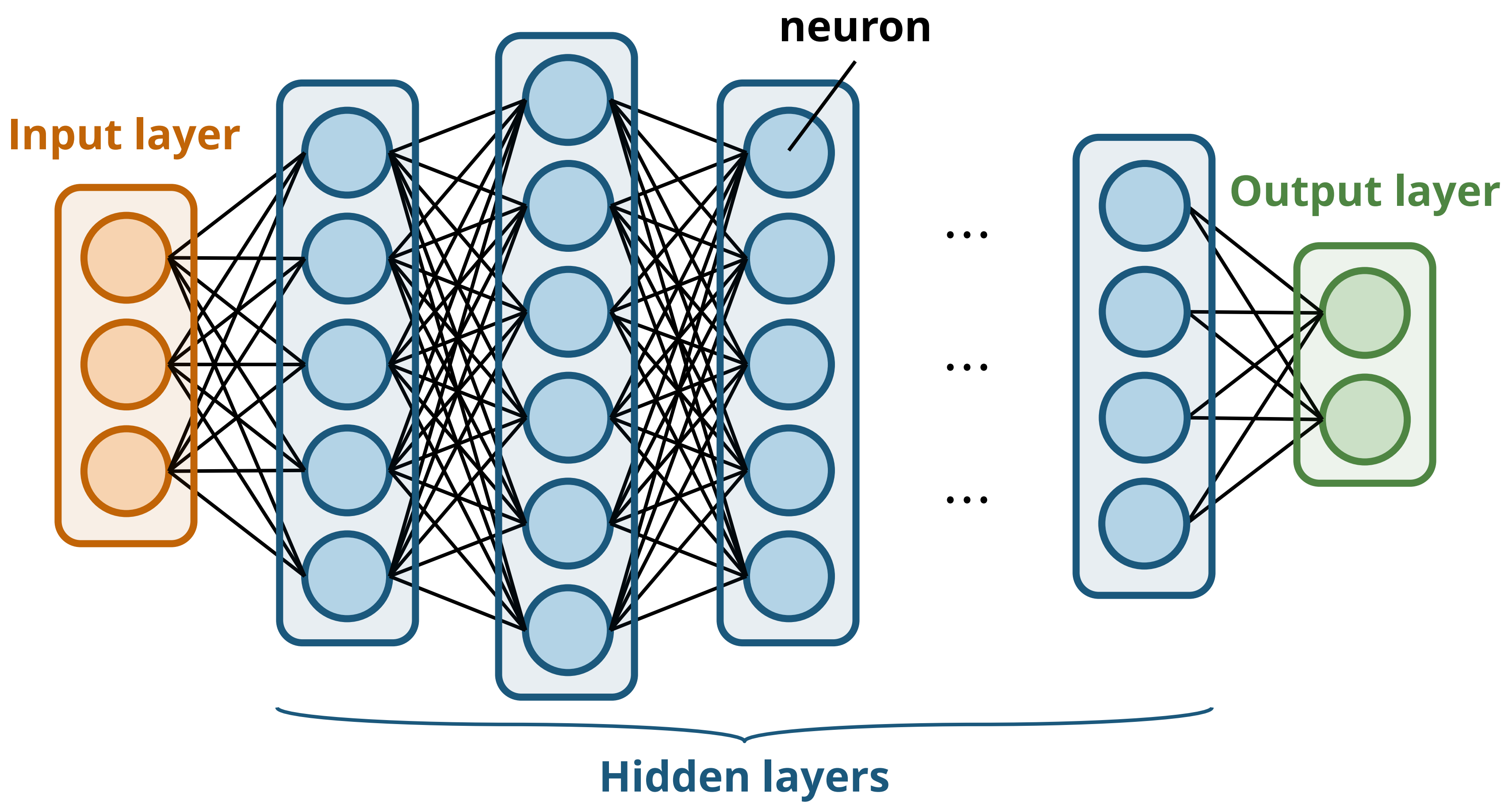
SCHOOL OF MUSIC, THEATRE & DANCE
PERFORMING ARTS TECHNOLOGY
UNIVERSITY OF MICHIGAN

Homework 2: Music & Audio Processing

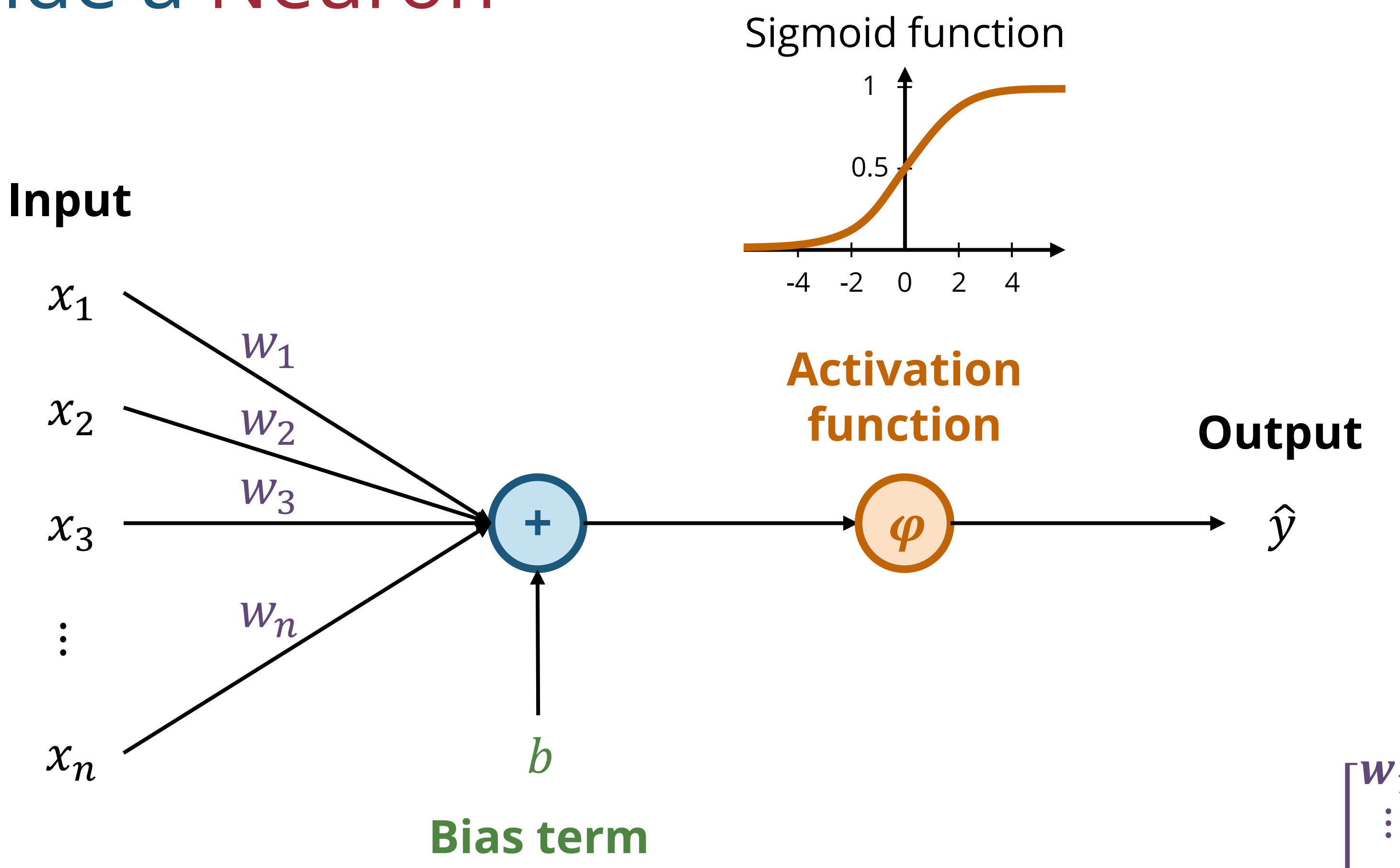
- Instructions will be sent by **emails** and released on the **course website**
- Please submit you work to **Gradescope**
- Due at **11:59pm ET** on **February 7**
- Late submissions: **1 point deducted per day**

(Recap) What is Deep Learning?

- A type of machine learning that uses **deep neural networks**



(Recap) Inside a Neuron

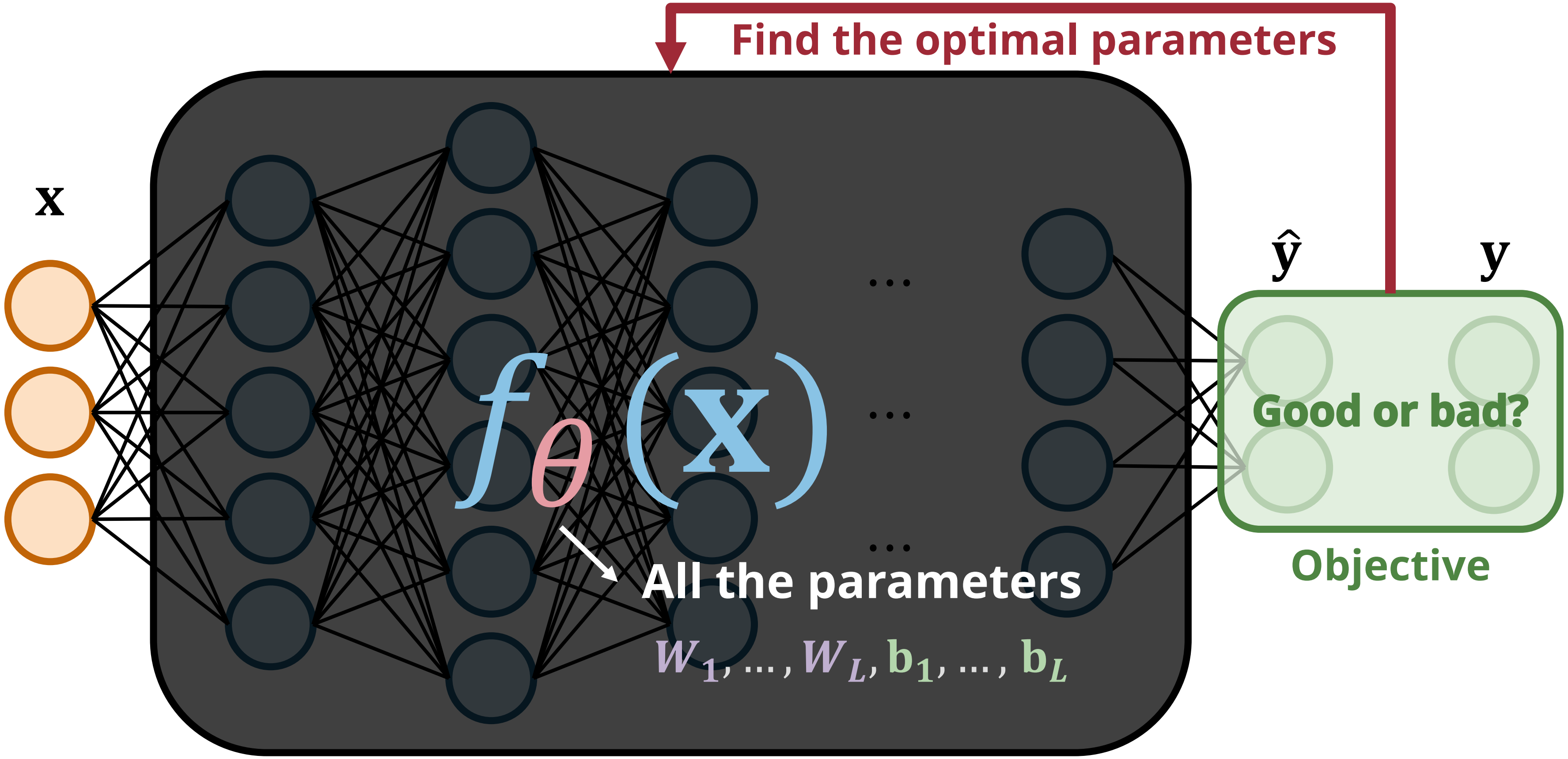


$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\hat{\mathbf{y}} = \varphi(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) = \varphi\left(\sum_{i=1}^{n} w_ix_i + b\right) = \varphi(\mathbf{w} \cdot \mathbf{x} + b)$$

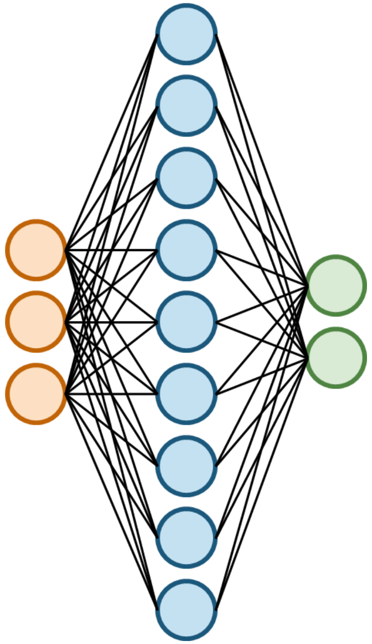
(Recap) Neural Networks are Parameterized Functions

- A neural network represents **a set of functions**



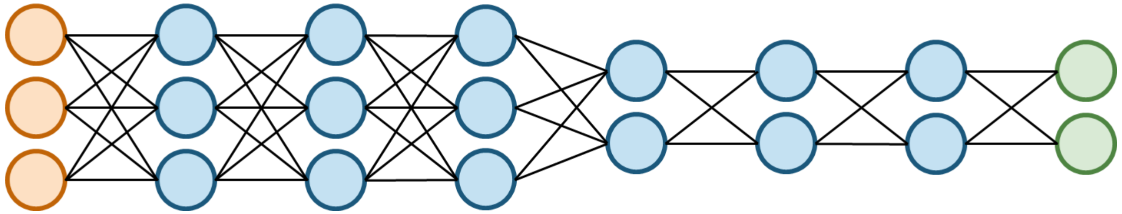
(Recap) Shallow vs Deep Neural Networks – In Practice

Shallow neural nets



Less expressive
(less parameter efficient)

Deep neural nets



More expressive
(more parameter efficient)

How to Train a Neural Network?

Training a Neural Network

Build a neural network
(which defines a set of functions)

↓

Define the objective
(i.e., what is good for a function)

↓

Find the optimal parameters
(which leads to the best function)

Training a Neural Network

Build a neural network
(which defines a set of functions)

↓

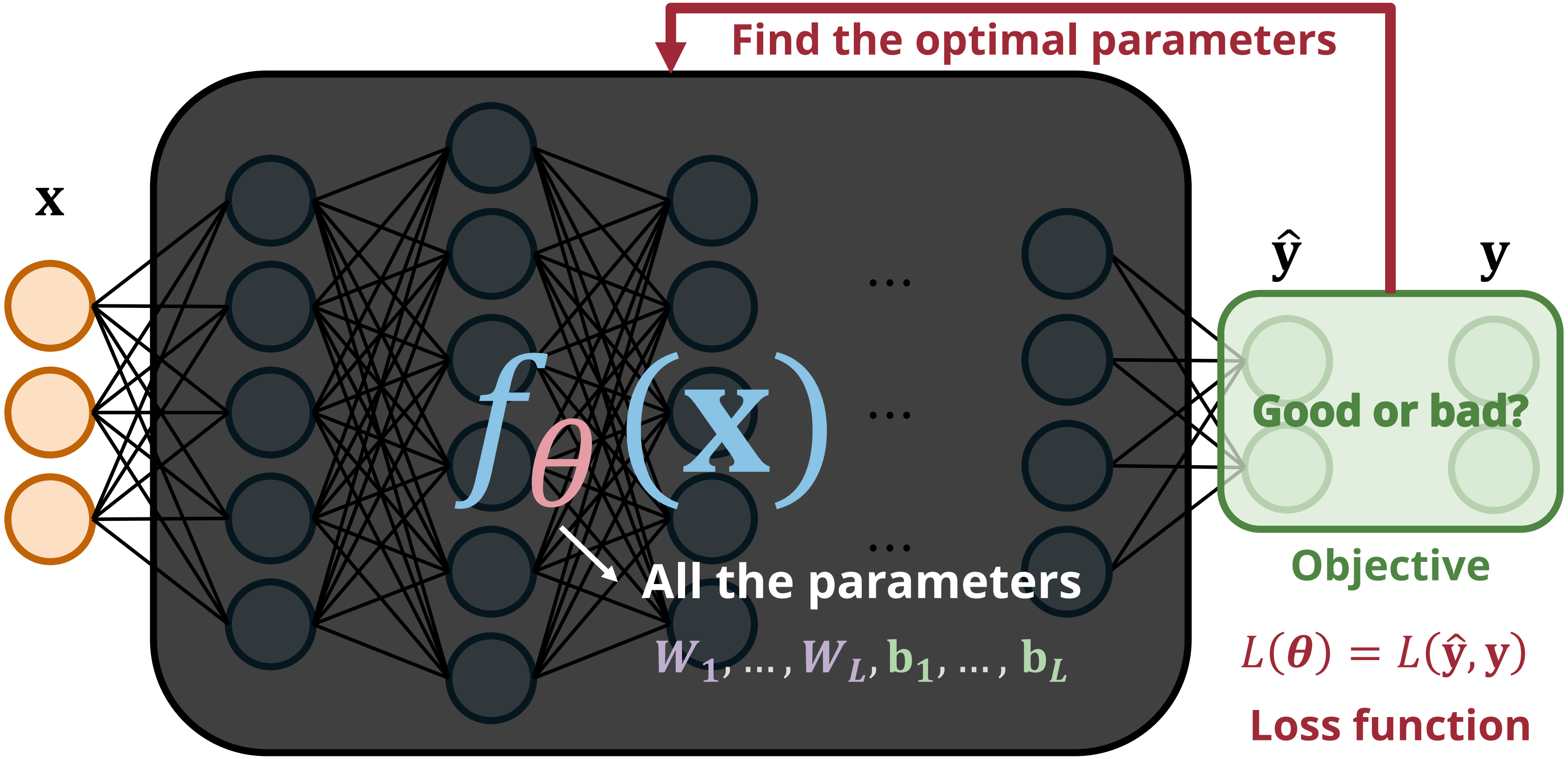
Define the objective
(i.e., what is good for a function)

↓

Find the optimal parameters
(which leads to the best function)

(Recap) Neural Networks are Parameterized Functions

- A neural network represents **a set of functions**



Loss Function

- Measure **how well the model perform** (in the opposite way)
- The choice of loss function depends on the task and the goals

$$L(\boldsymbol{\theta}) = L(\hat{\mathbf{y}}, \mathbf{y})$$

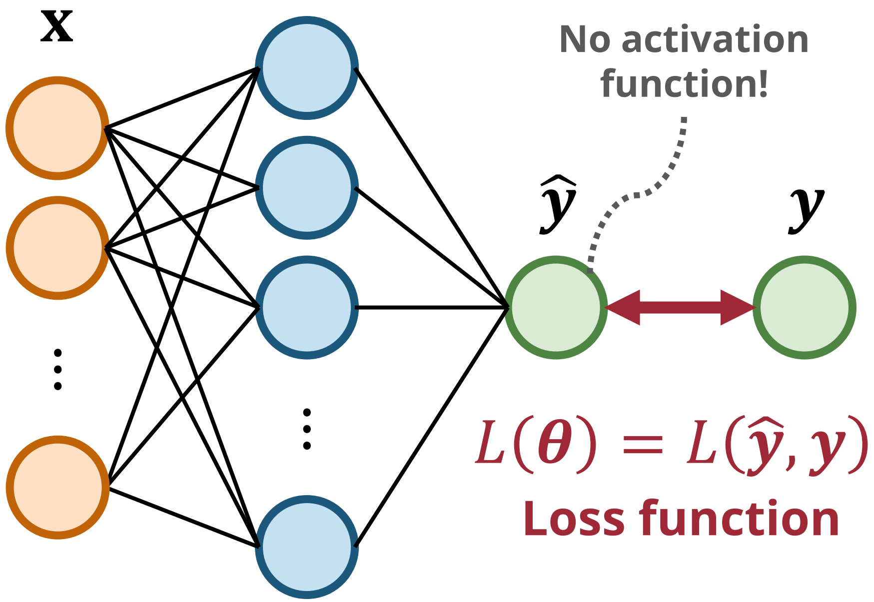
Loss Function – The Many Names

- Sometimes called
  - **Cost** function
  - **Error** function
- The opposite is known as
  - **Objective** function
  - **Reward** function (reinforcement learning)
  - **Fitness** function (evolutionary algorithms & genetic algorithms)
  - **Utility** function (economics)
  - **Profit** function (economics)

Example: Audio Codec

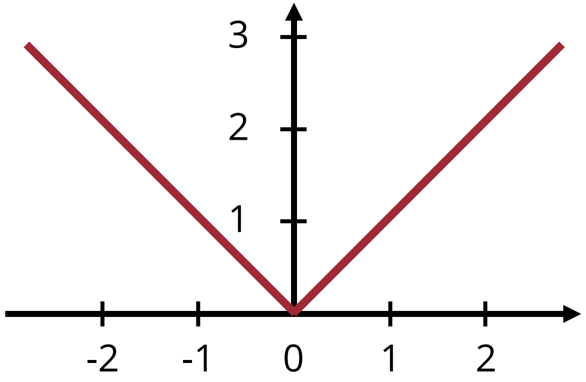
- What would be **a good objective to train a neural audio codec**?
- What do we care about for a codec?
  - Reconstruction quality **Trainable**
  - Bit rate (compression rate) **Likely not trainable but searchable**
  - Encoding/decoding speed **Likely not trainable but searchable**
- How do we measure reconstruction quality?
  - Difference in raw waveforms?
  - Difference in spectrograms?
  - Perceptual quality (psychoacoustics)?

Common Loss Functions for Regression



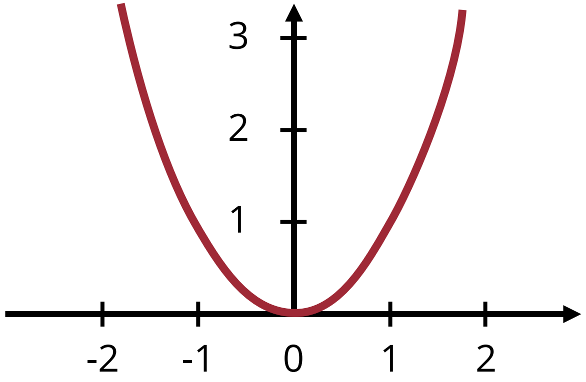
L1 loss

$$L(\hat{y}, y) = |\hat{y} - y|$$



L2 loss

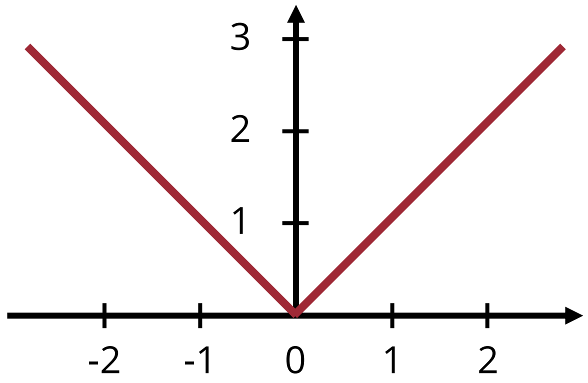
$$L(\hat{y}, y) = (\hat{y} - y)^2$$



Why not $L(\hat{y}, y) = \hat{y} - y$?

L1 vs L2 Losses

L1 loss

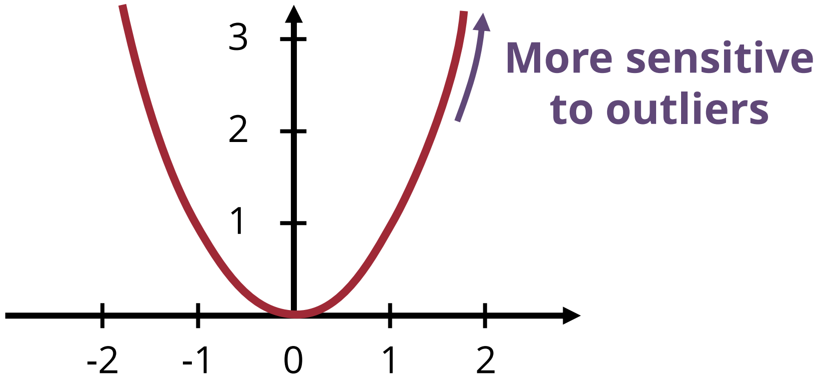


$$L(\hat{y}, y) = |\hat{y} - y|$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \mathbf{MAE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|$$

Mean Absolute Error (MAE)

L2 loss



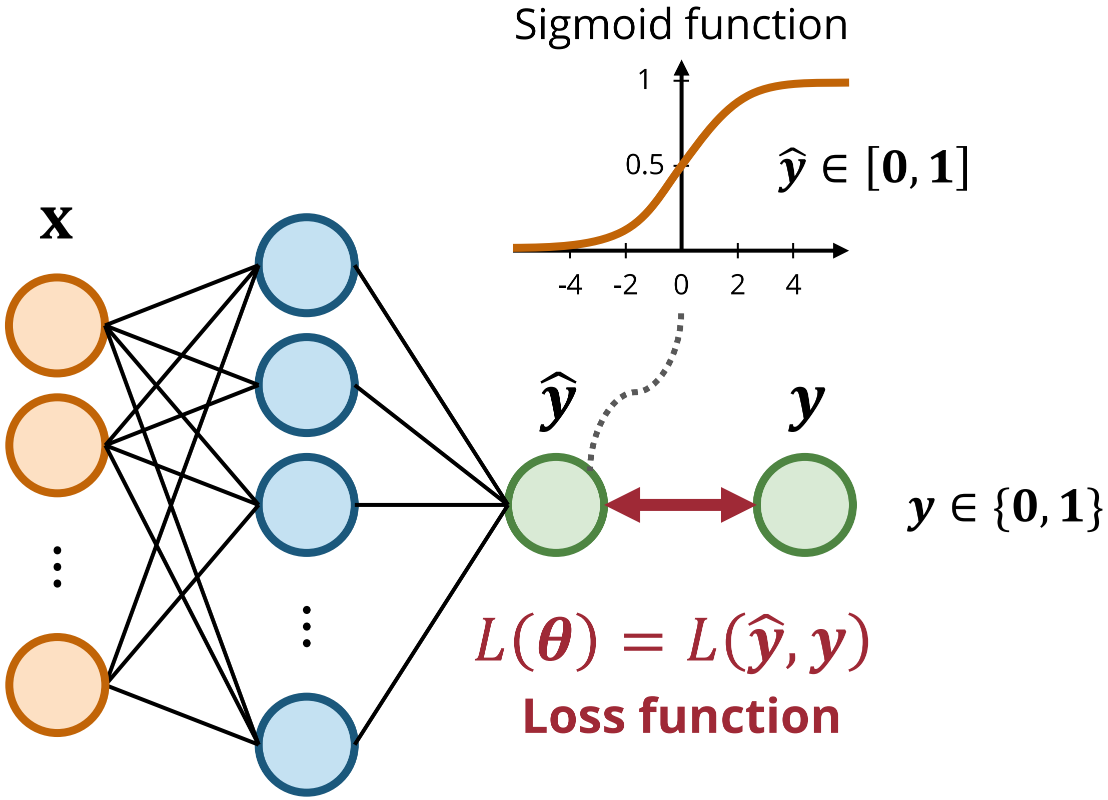
$$L(\hat{y}, y) = (\hat{y} - y)^2$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \mathbf{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Mean Squared Error (MSE)

Binary Cross Entropy for Binary Classification

- **Logistic regression** approaches classification like regression

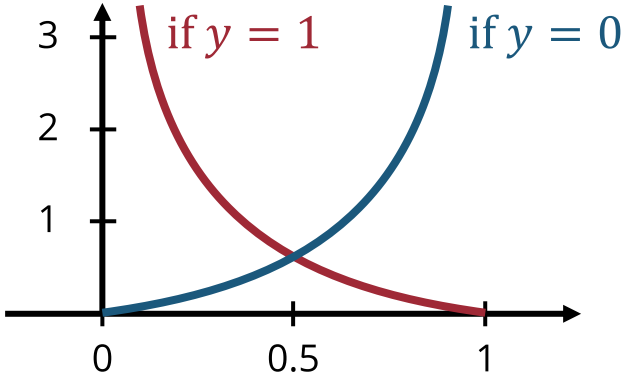


Binary cross entropy

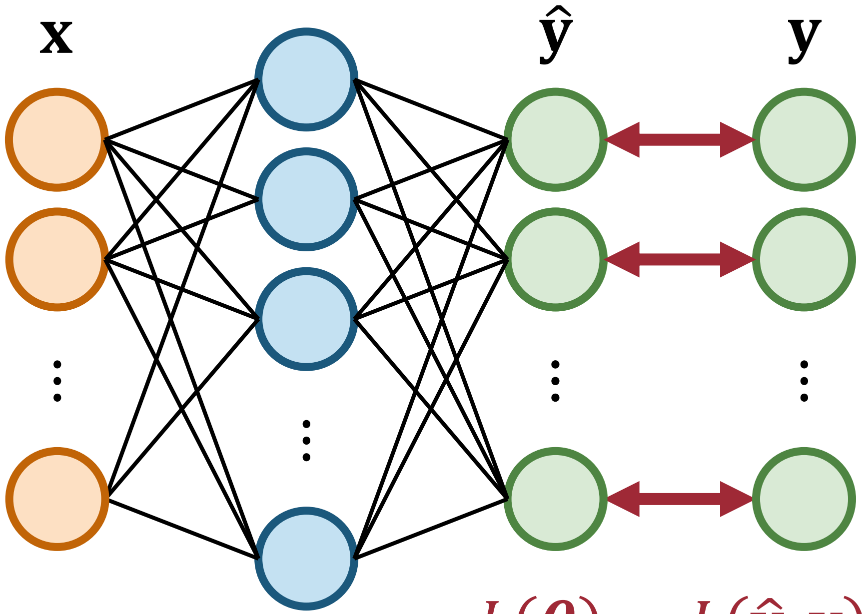
(Also called log loss)

$$L(\hat{y}, y) = \begin{cases} -\log \hat{y}\,, & \text{if } y = 1 \\ -\log(1-\hat{y})\,, & \text{if } y = 0 \end{cases}$$

$$= -y \log \hat{y} - (1-y)\log(1-\hat{y})$$



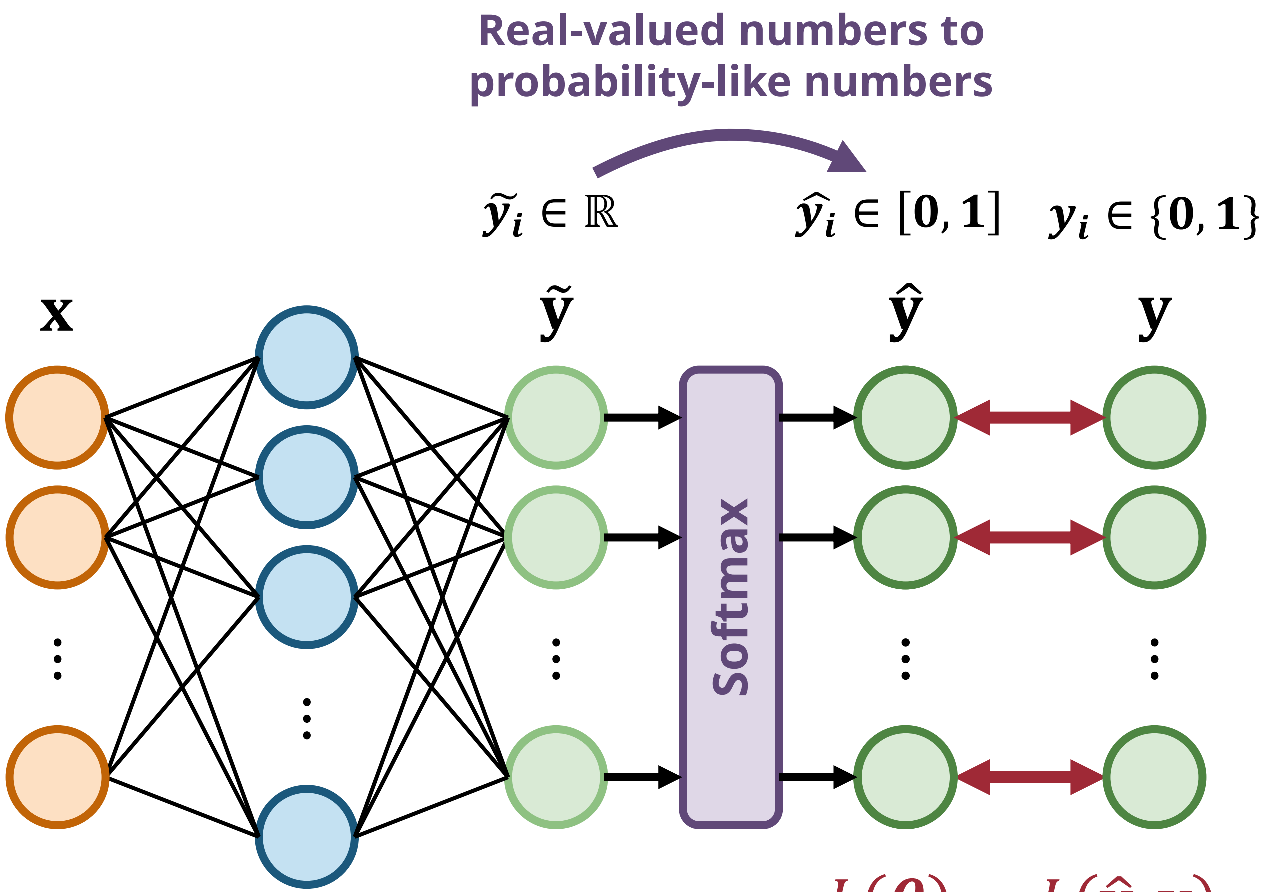
Cross Entropy for Multiclass Classification



$L(\boldsymbol{\theta}) = L(\hat{\mathbf{y}}, \mathbf{y})$

Loss function

Cross Entropy for Multiclass Classification



$$L(\theta) = L(\hat{\mathbf{y}}, \mathbf{y})$$

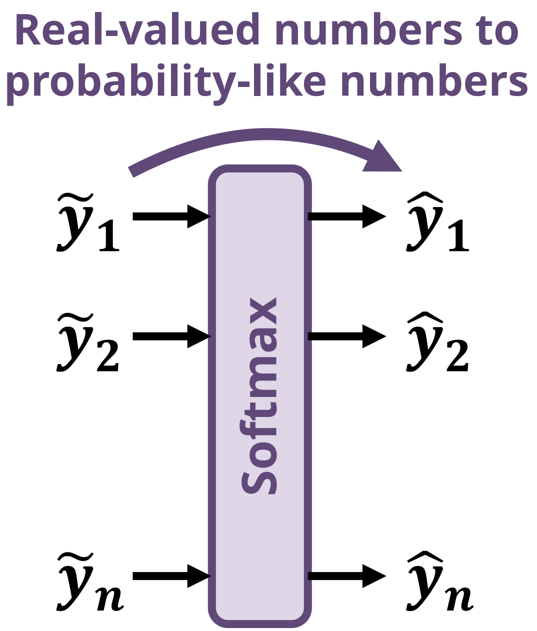
Loss function

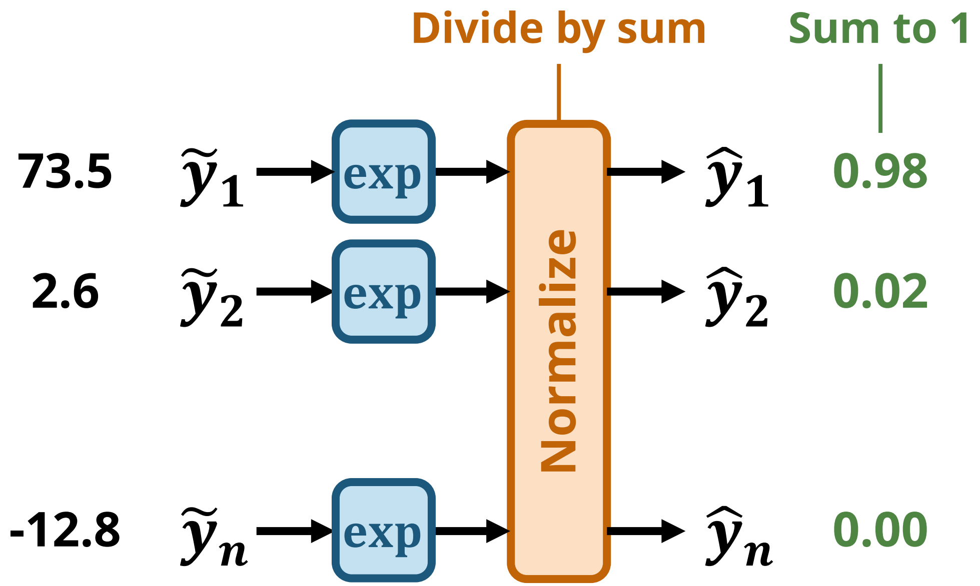
Softmax

$$\hat{y}_i = \frac{e^{\tilde{y}_i}}{\sum_{j=1}^{n} e^{\tilde{y}_j}}$$

Softmax

- **Intuition**: Map several numbers to $[0, 1]$ while **keeping their relative magnitude**
  - Softmax is like the **multivariate version of sigmoid**





Cross Entropy for Multiclass Classification

Binary Cross Entropy

Only one of them will be one!

$$L(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

Cross Entropy

Only one of them will be one!

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -y_1 \log \hat{y}_1 - y_2 \log \hat{y}_2 - \cdots - y_i \log \hat{y}_n$$

$$= -\sum_{i}^{n} y_i \log \hat{y}_i$$

Log likelihood

Optimization

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

$$L(\boldsymbol{\theta})$$

↓

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

Define the objective
(i.e., what is good for a function)

$$L(\boldsymbol{\theta})$$

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Optimizing the Parameters of a Neural Network

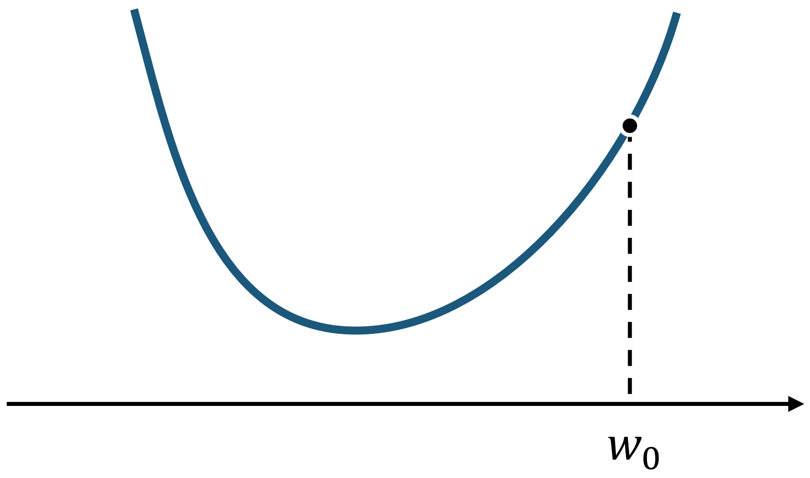
- Many, many ways…
- Most commonly through **gradient descent** in deep learning
- Alternatively, we can use search or genetic algorithm

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Gradient Descent

- **Intuition**: Gradient can suggest a good direction to tune the parameters

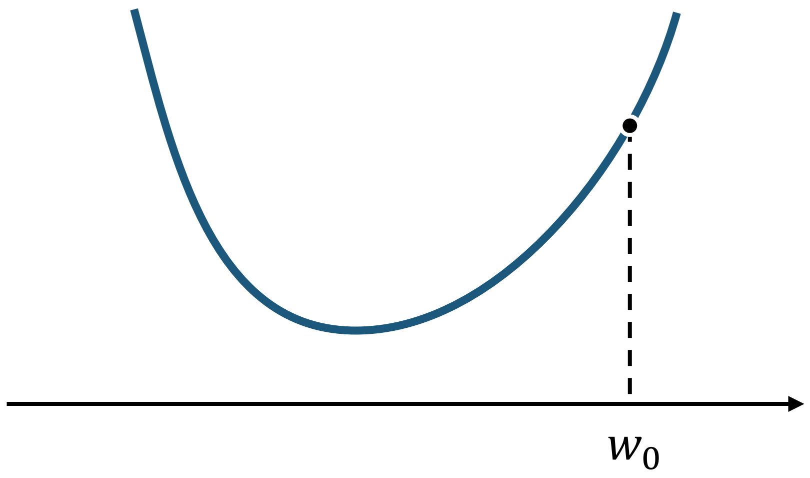
Derivative for a vector, matrix or tensor



Gradient Descent – Pseudocode

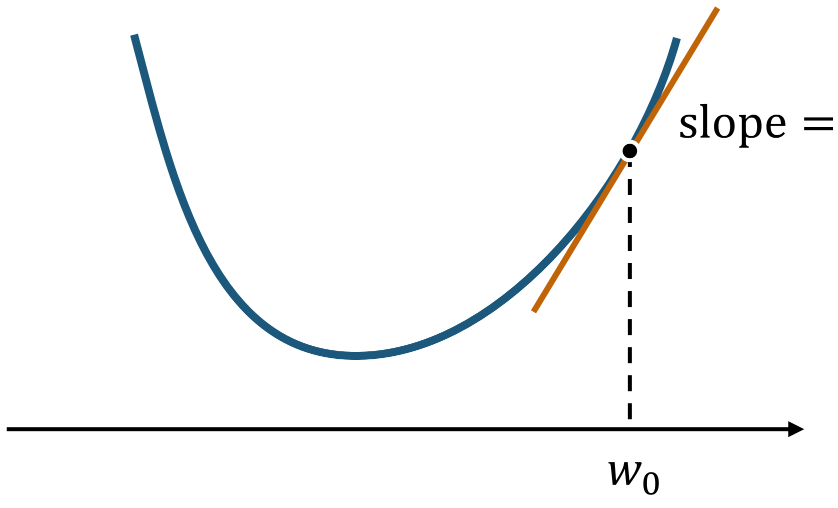
- Pick an **initial weight vector** w_0 and **learning rate** η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$

Gradient of function f with respect to weight w



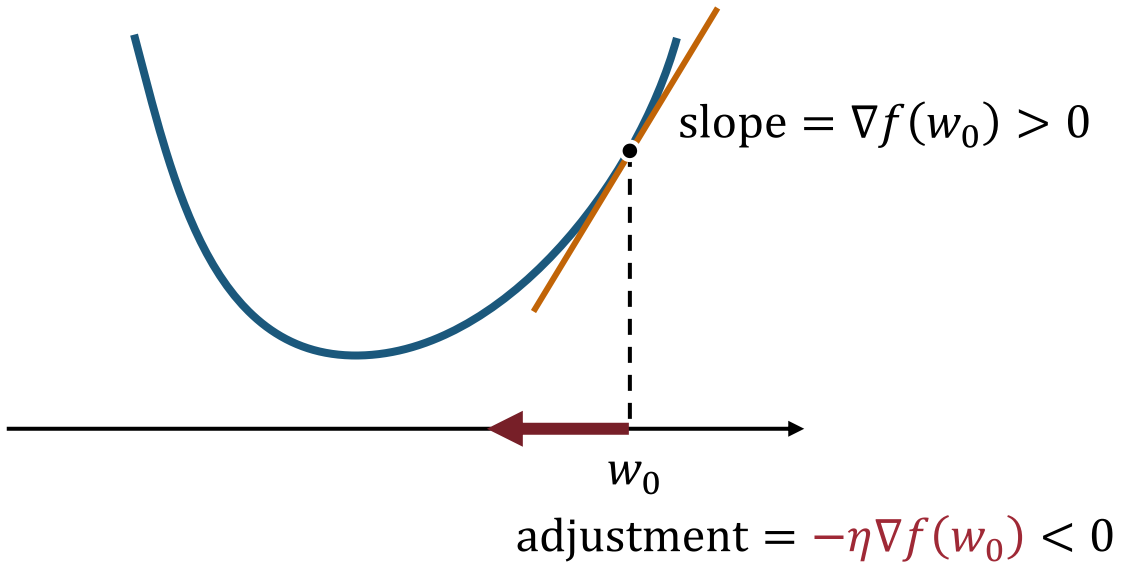
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



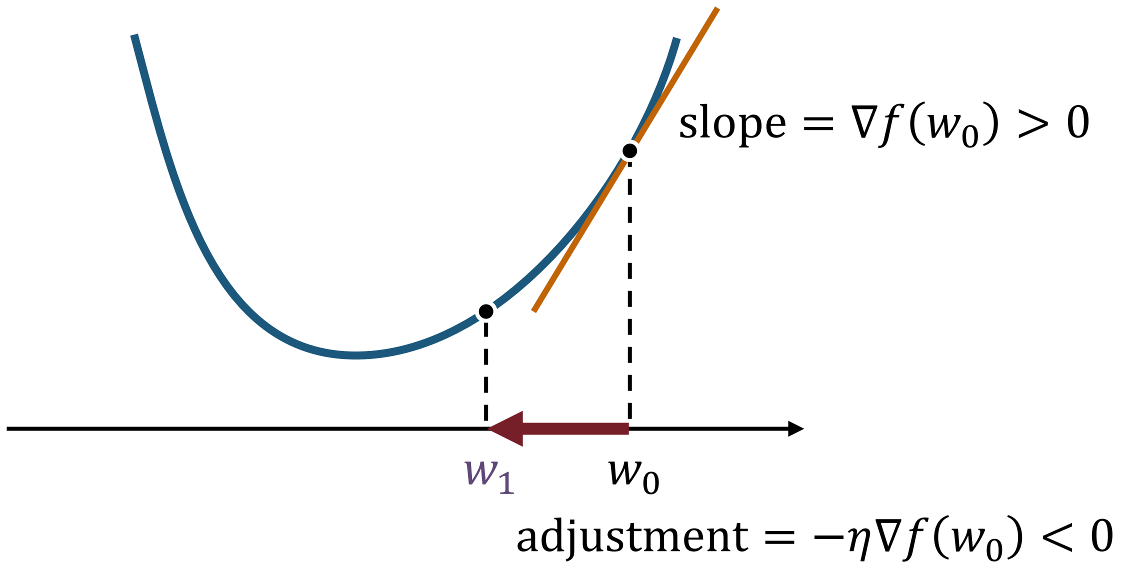
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



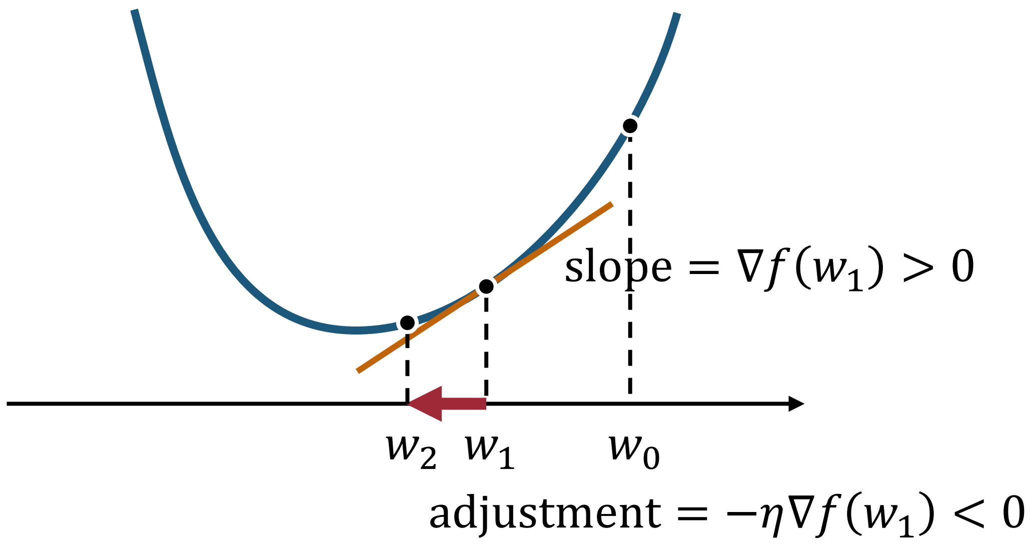
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



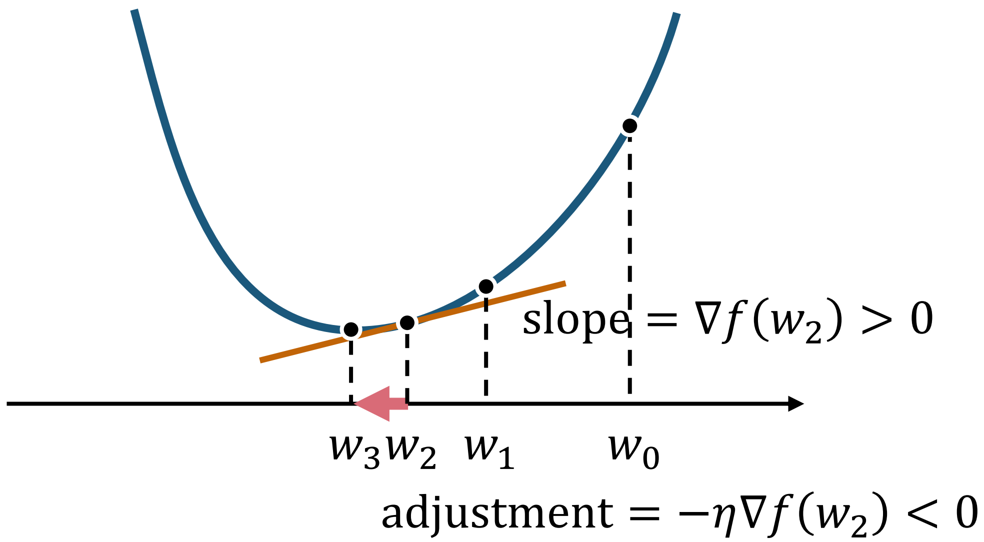
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



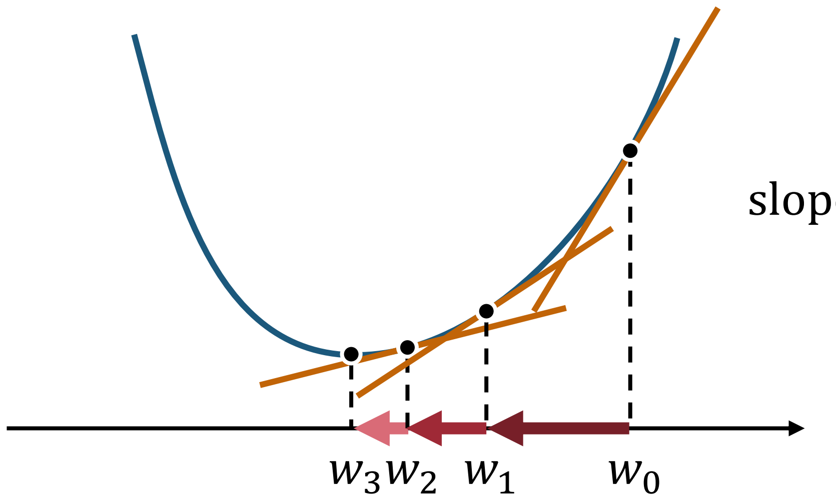
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



Gradient Descent – Pseudocode

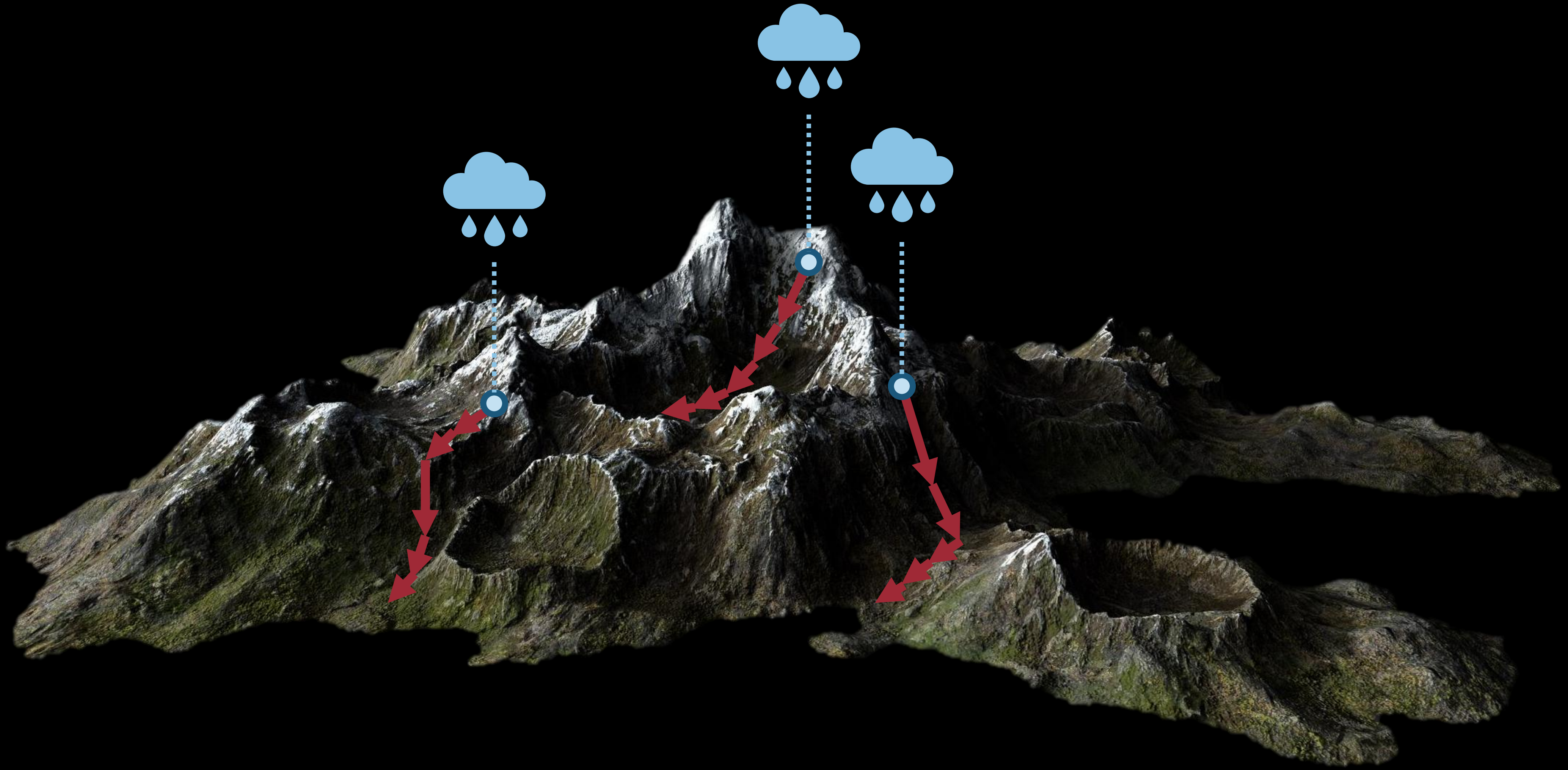
- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



slope $= \nabla f(w_t) > 0$

adjustment $= -\eta \nabla f(w_t) < 0$

Gradient Descent – 3D Case



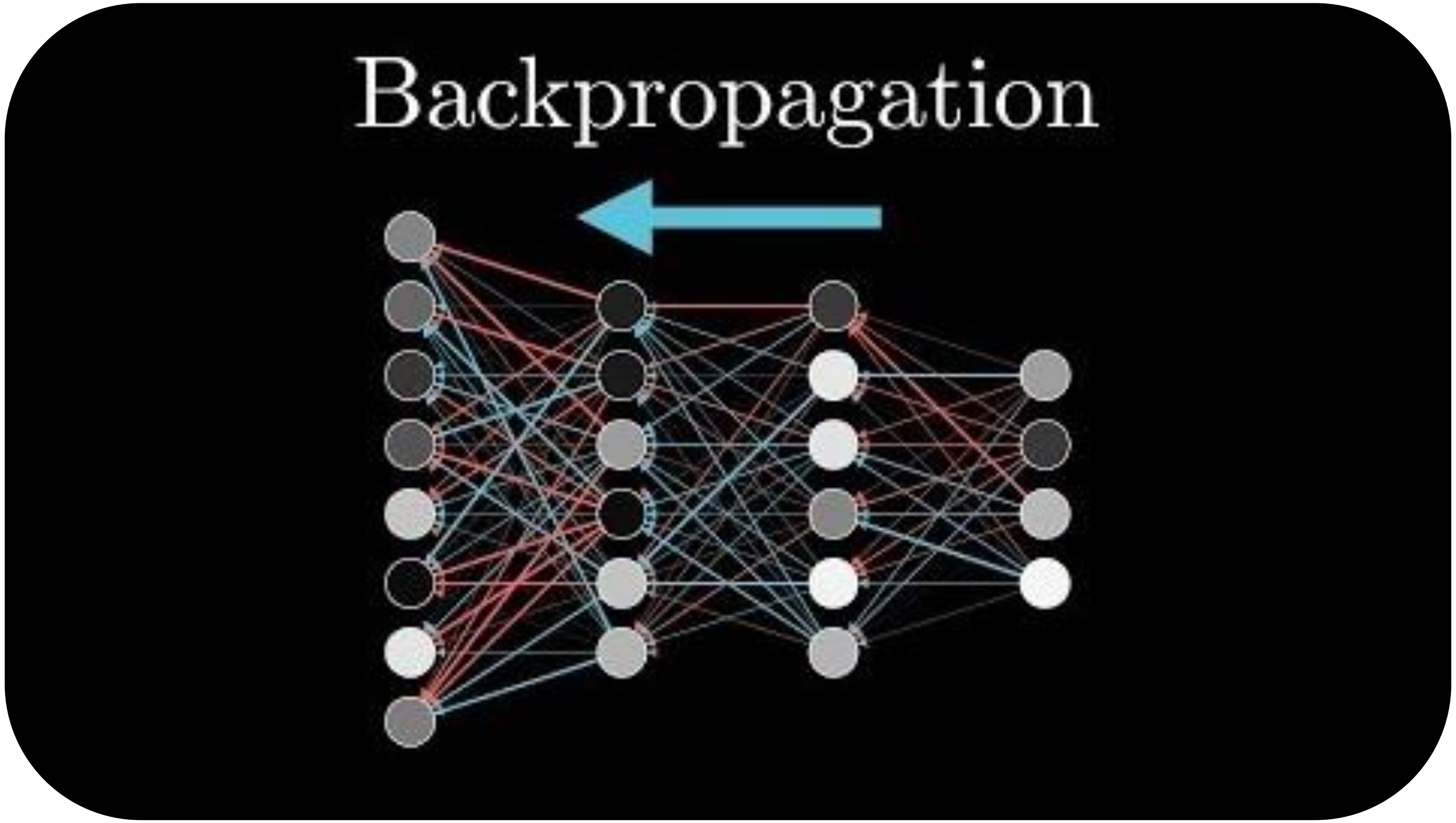
substance3d.adobe.com/community-assets/assets/9fe77d4279e12fe2fbd90b8f7ca18146a565f7ee

Backpropagation: Efficiently Computing the Gradients

- An efficient way of **computing gradients** using chain rule
- The reason why we want **everything to be differentiable** in deep learning

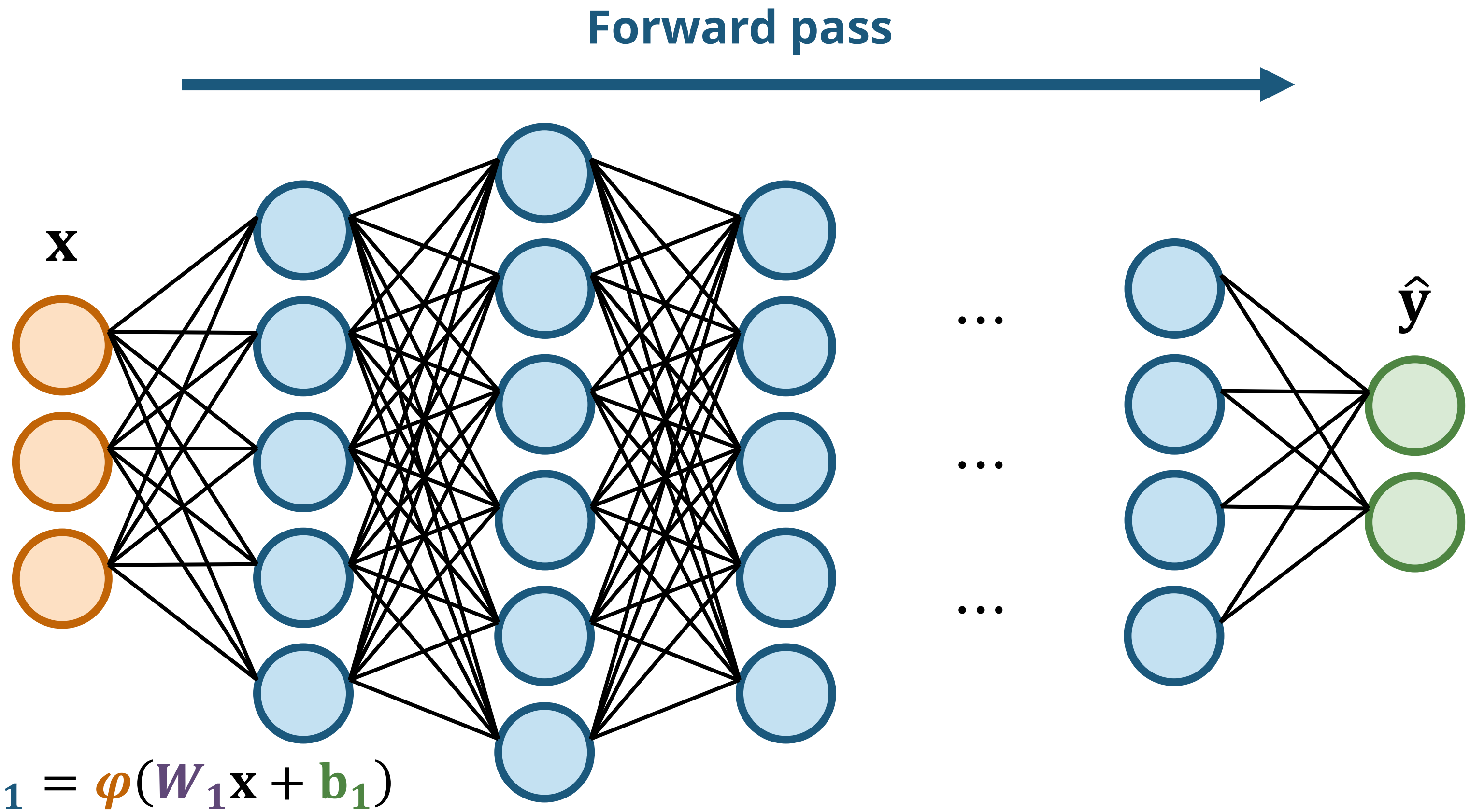
$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Backpropagation: Efficiently Computing the Gradients



youtu.be/Ilg3gGewQ5U?t=196

Forward Pass & Backward Pass



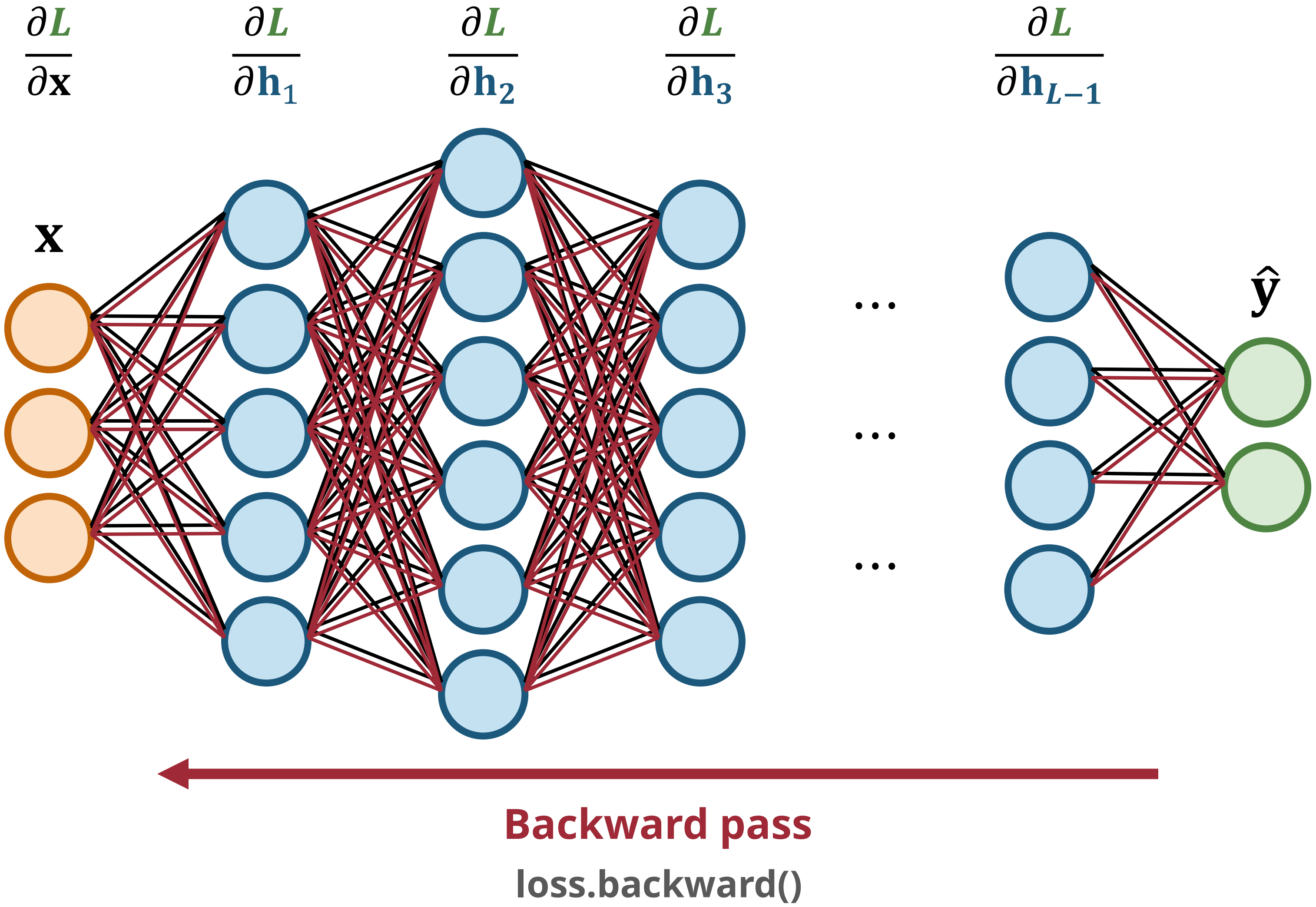
$\mathbf{h}_1 = \varphi(W_1\mathbf{x} + \mathbf{b}_1)$

$\mathbf{h}_2 = \varphi(W_2\mathbf{h}_1 + \mathbf{b}_2)$

$\mathbf{h}_3 = \varphi(W_2\mathbf{h}_2 + \mathbf{b}_2)$

$\hat{\mathbf{y}} = \varphi(W_L\mathbf{h}_{L-1} + \mathbf{b}_L)$

Forward Pass & Backward Pass



Advanced Optimization

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

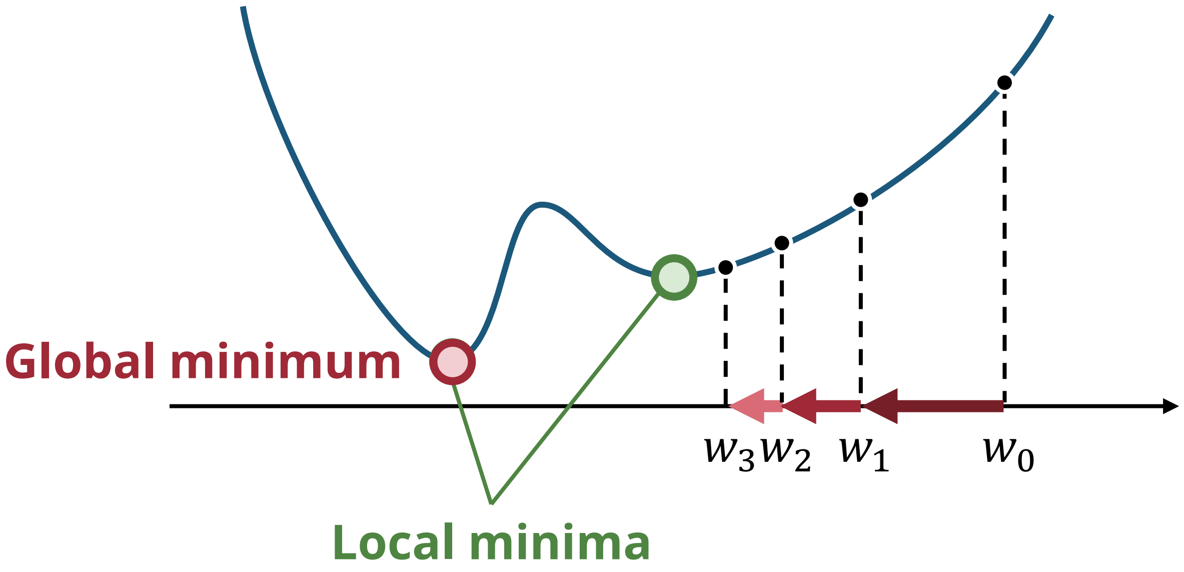
Define the objective
(i.e., what is good for a function)

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

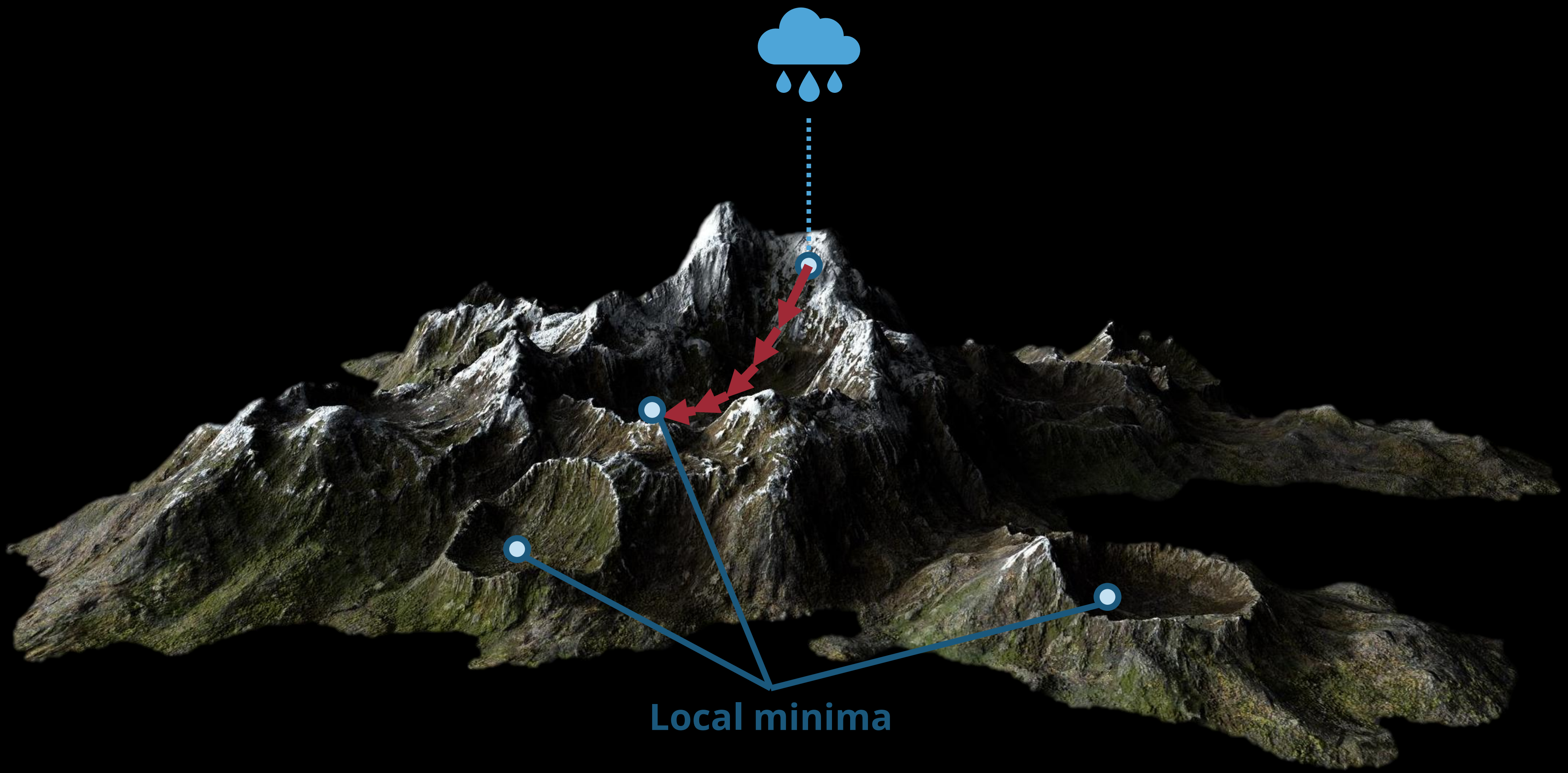
Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Gradient Descent Finds a Local Minimum

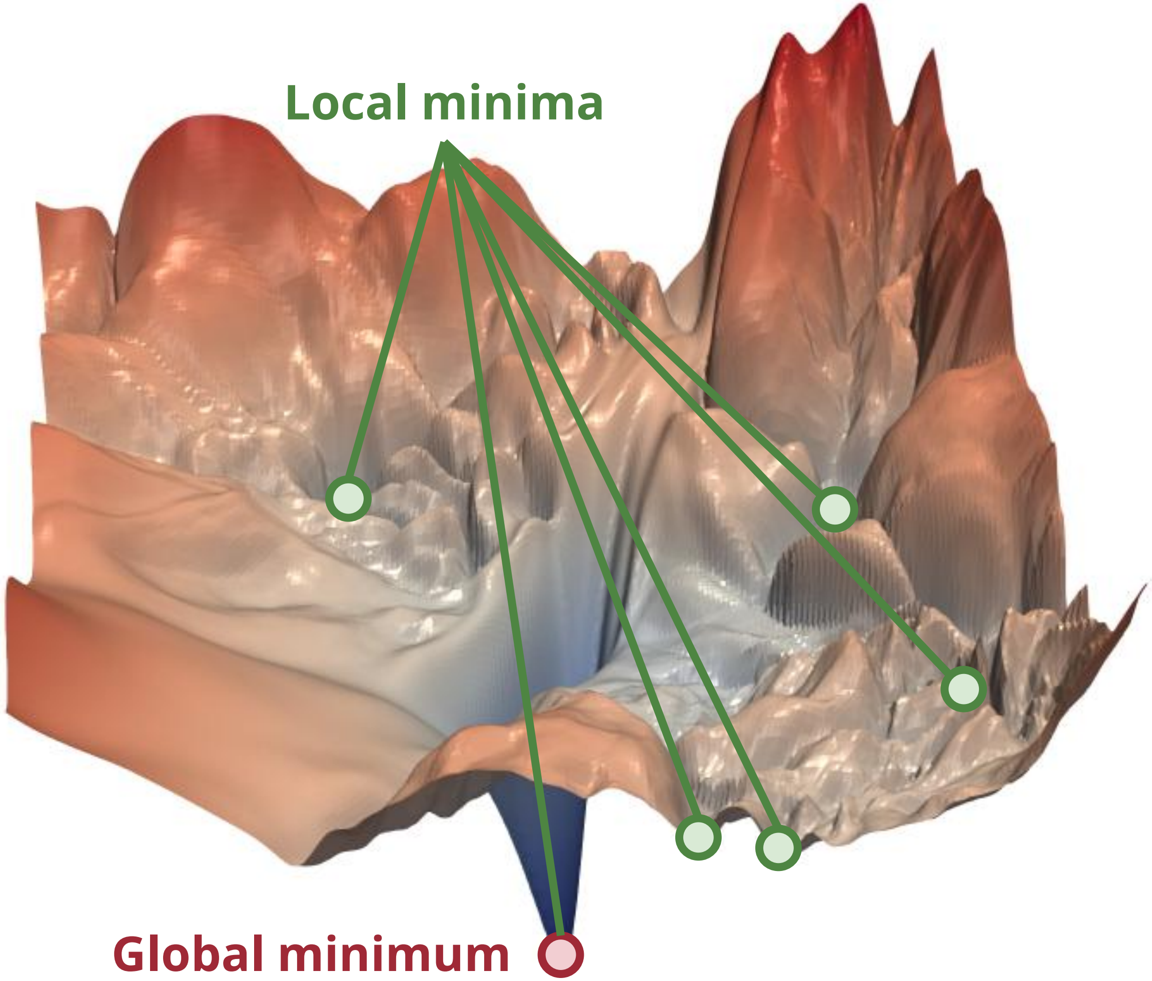


Gradient Descent Finds a Local Minimum



substance3d.adobe.com/community-assets/assets/9fe77d4279e12fe2fbd90b8f7ca18146a565f7ee

Local Minima in Complex Loss Landscape



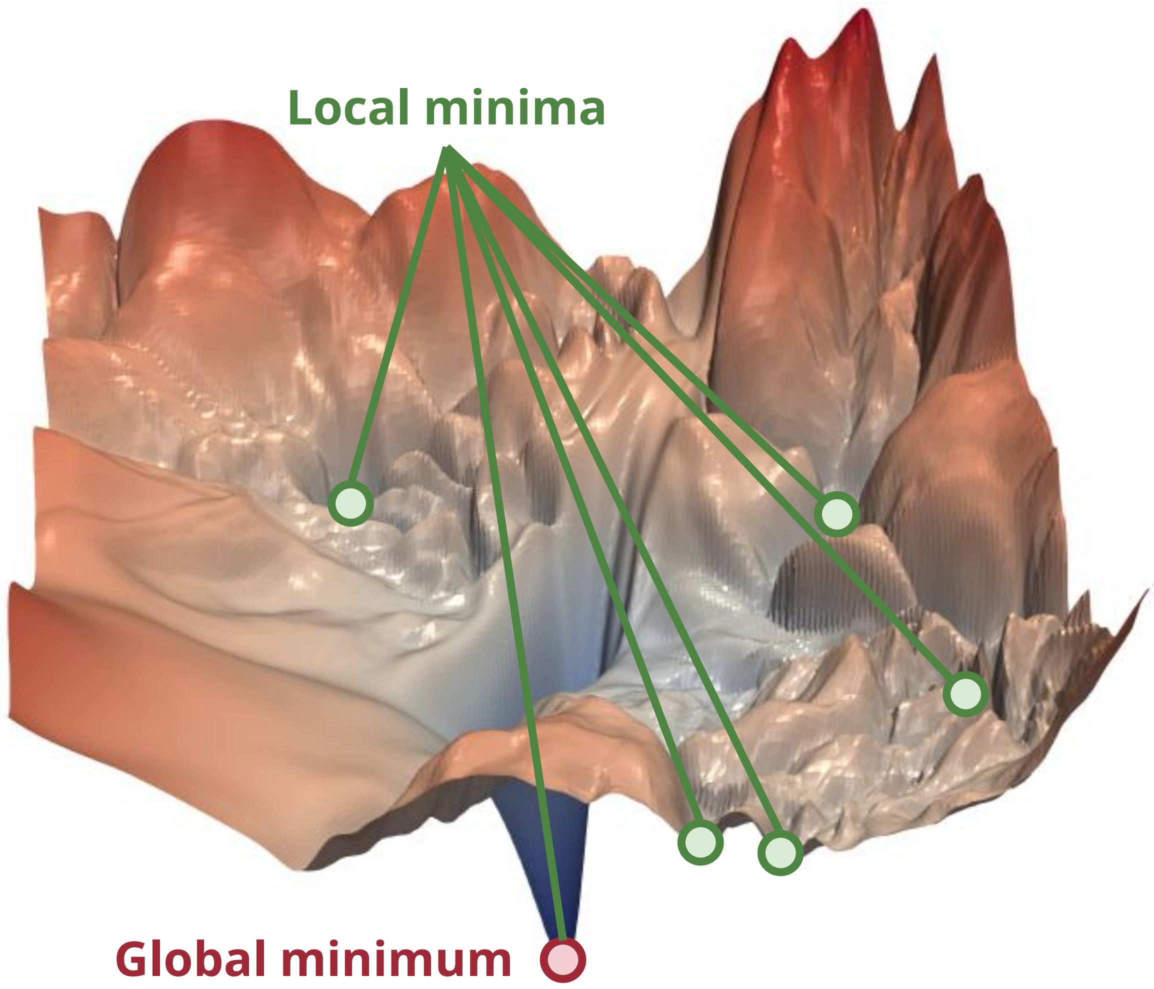
Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Local Minima in Complex Loss Landscape



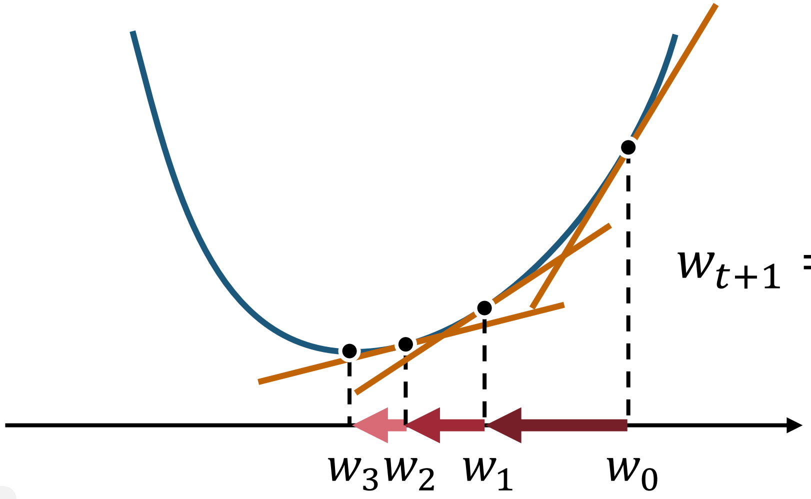
Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

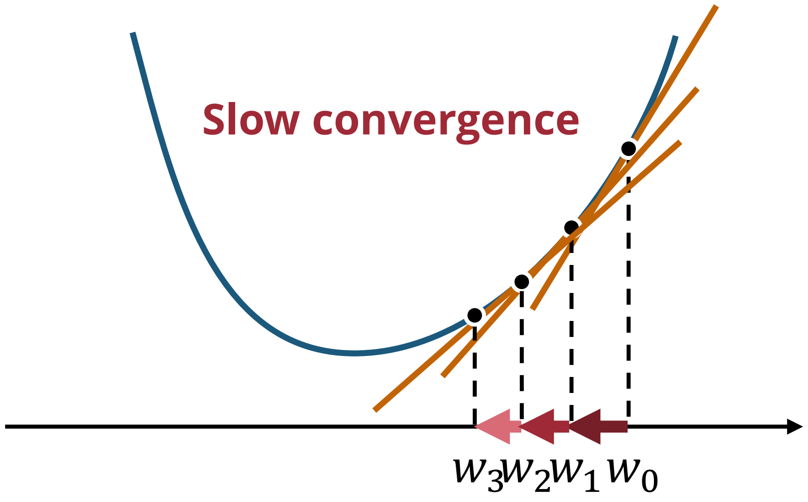
Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Learning Rate in Gradient Descent

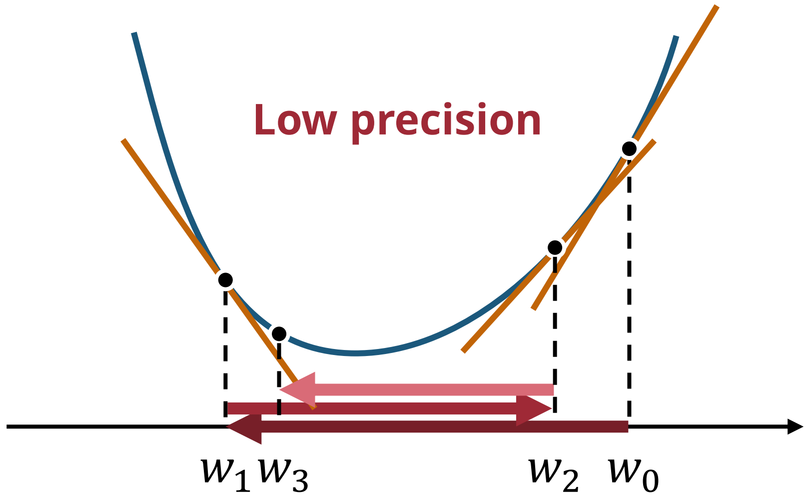


$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Smaller learning rate

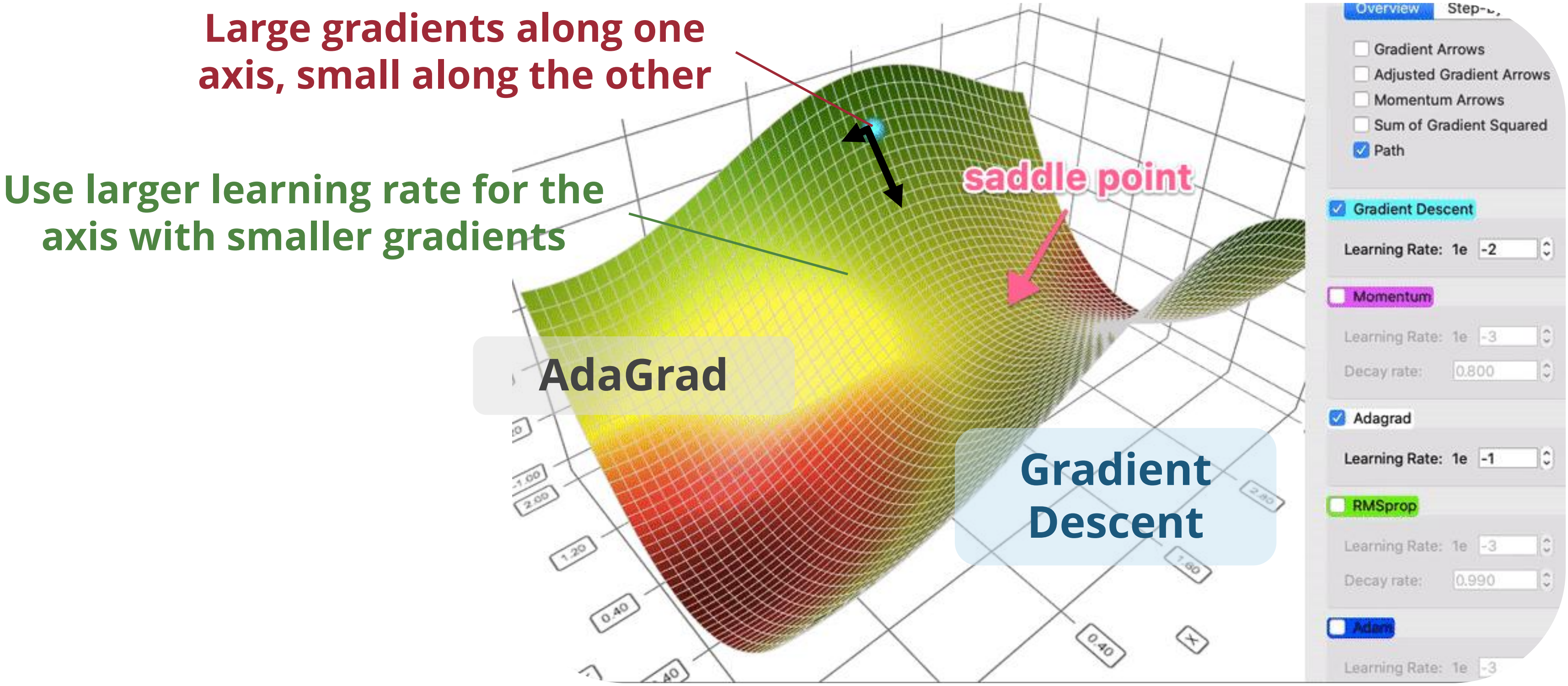


Larger learning rate



Gradient-based Adaptive Learning Rate

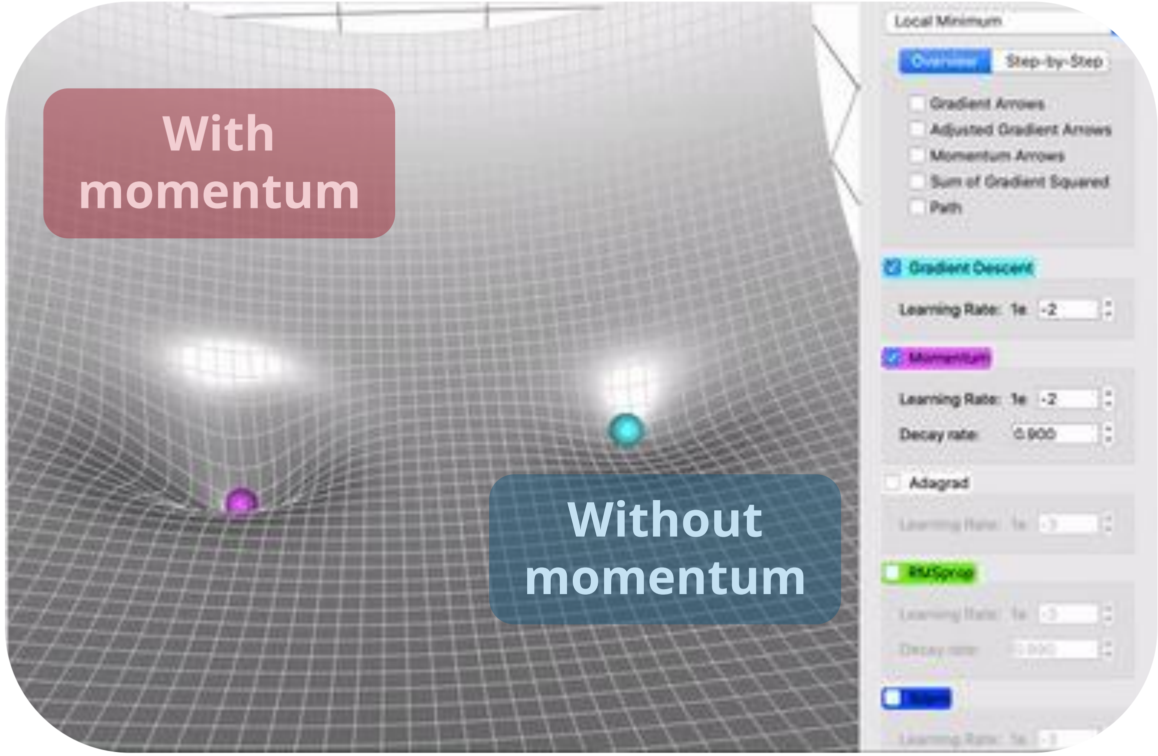
- **Intuition**: **Compensate axis that has little progress** by comparing the current gradients to the previous gradients



towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

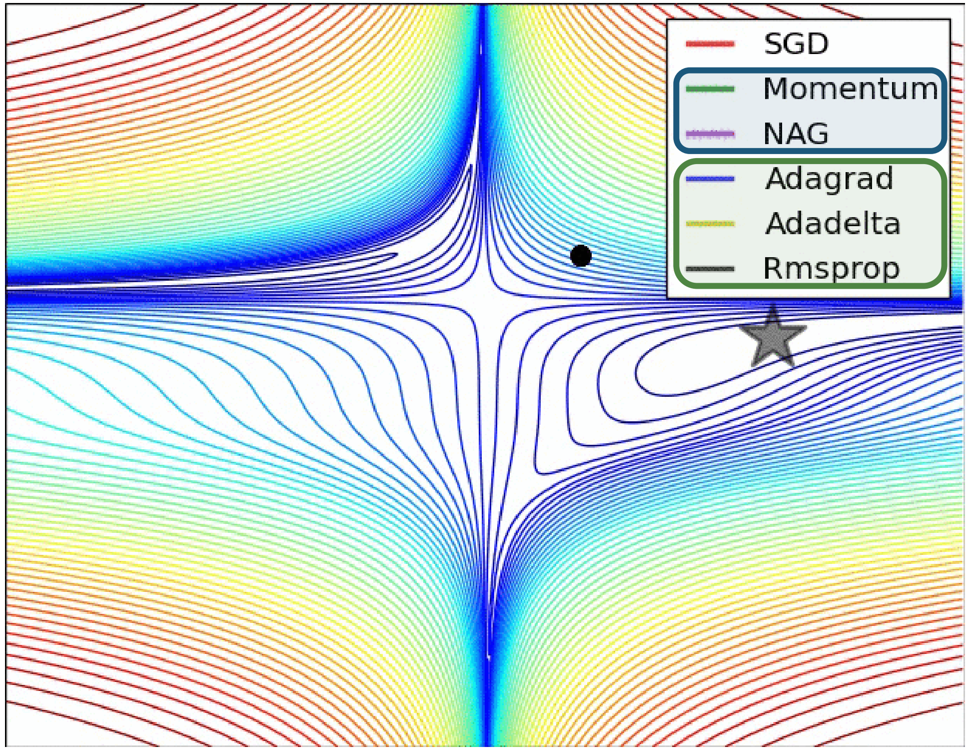
Momentum

- **Intuition**: Maintain the momentum to **escape from local minima**



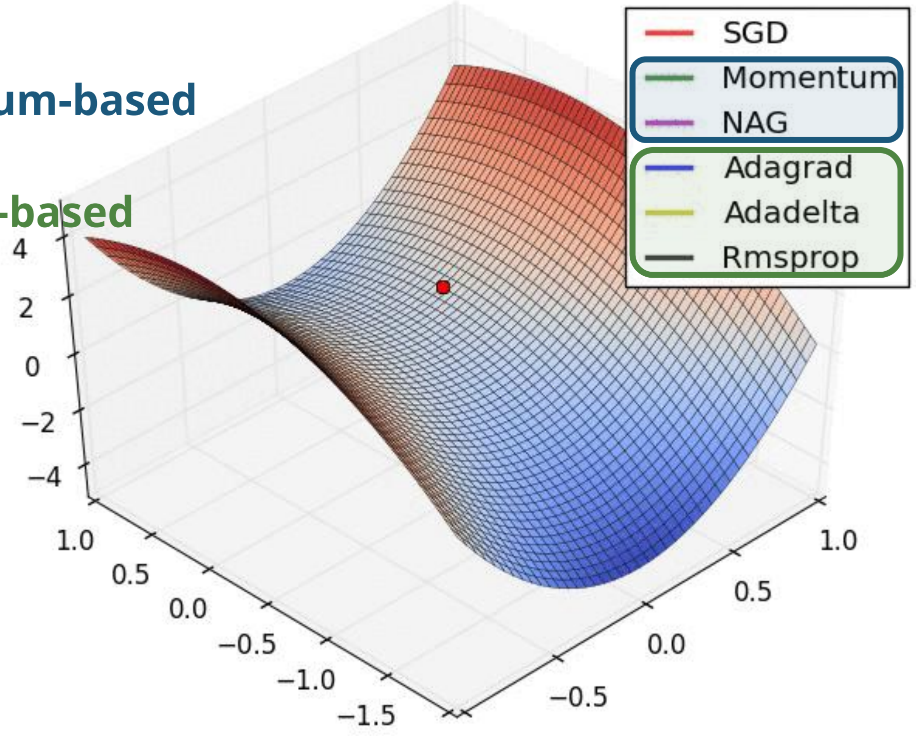
towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Comparison of Optimizers



Momentum-based

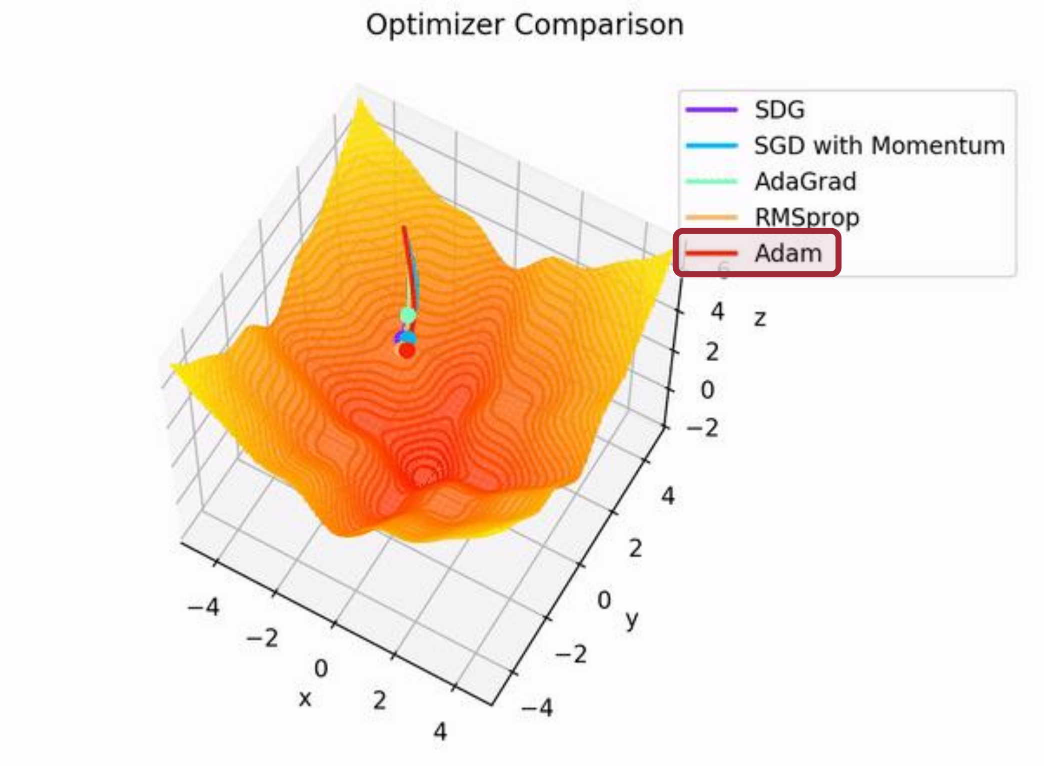
Gradient-based



Can we combine them?

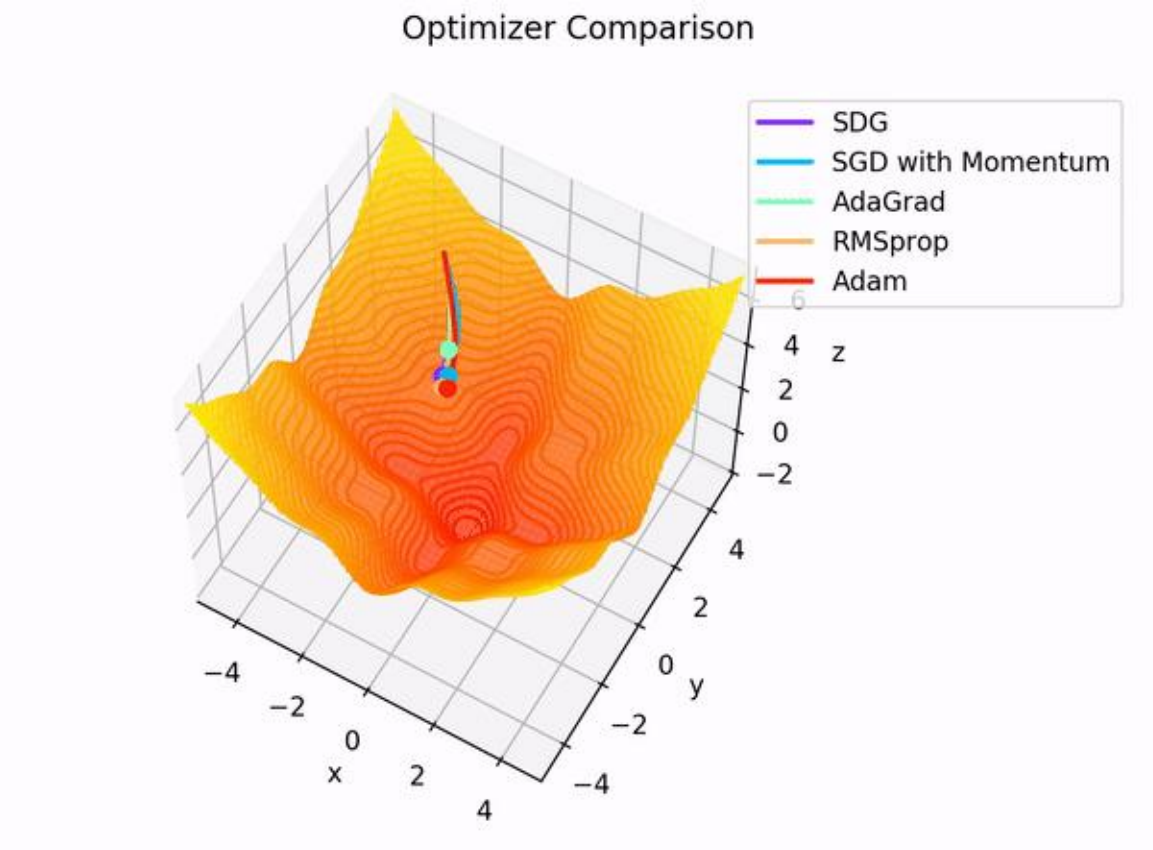
Adam Optimizer

- Combine the idea of **adaptive learning rate** and **momentum**
- Work **empirically well** in complex neural network
- The **go-to choice** for most cases



Comparison of Optimizers

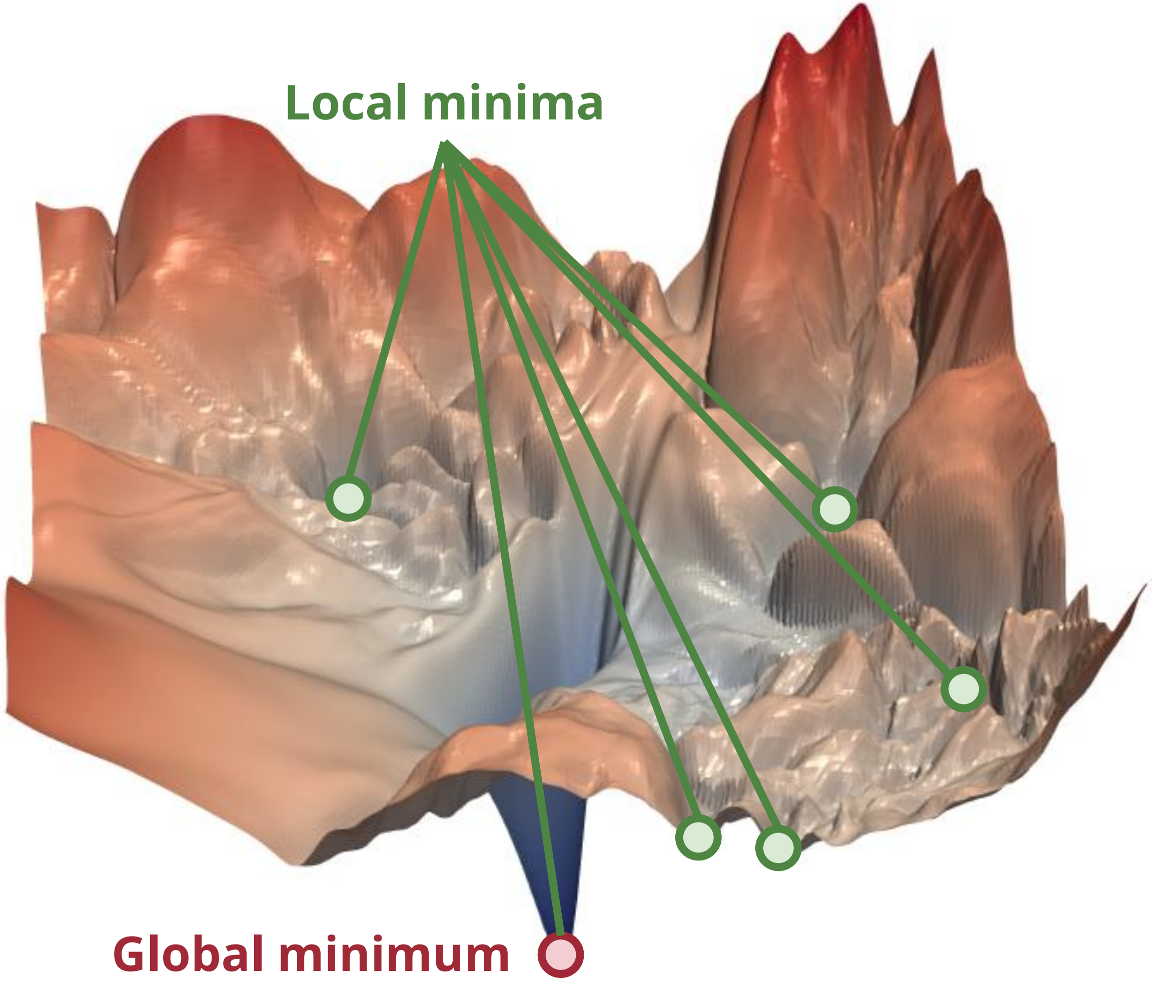
- **Momentum**
  - Gets you out of spurious local minima
  - Allows the model to explore around



- **Gradient-based adaption**
  - Maintains steady improvement
  - Allows faster convergence

medium.com/@LayanSA/complete-guide-to-adam-optimization-1e5f29532c3d

Local Minima in Complex Loss Landscape



Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Batch Gradient Descent

- How to aggregate the gradients obtained from different training samples?
- Batch gradient descent computes the mean gradients **over the whole training set**

MSE loss

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{k}^{N} \sum_{i}^{n} \left(\hat{y}_i^{(k)} - y_i^{(k)}\right)^2$$

Binary cross entropy

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}, y) = \sum_{k}^{N} -y \log \hat{y} - (1 - y) \ \log(1 - \hat{y})$$

Cross entropy

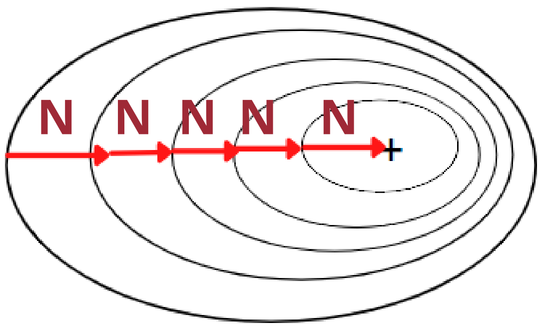
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k}^{N} \sum_{i}^{n} y_i \log \hat{y}_i$$

Stochastic Gradient Descent (SGD)

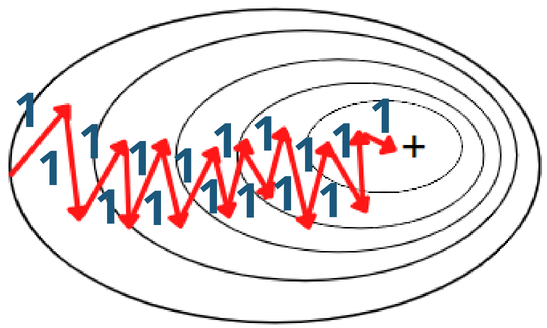
- **Intuition:** **Estimate** the gradient using **one random training sample**
- Benefits
  - **Speed up** the computation of the gradient **N computations → 1 computation**
  - Add some **randomness** to the gradient descent algorithm **Help escape spurious local minima**

Gradient descent

5N gradient computations



Stochastic gradient descent

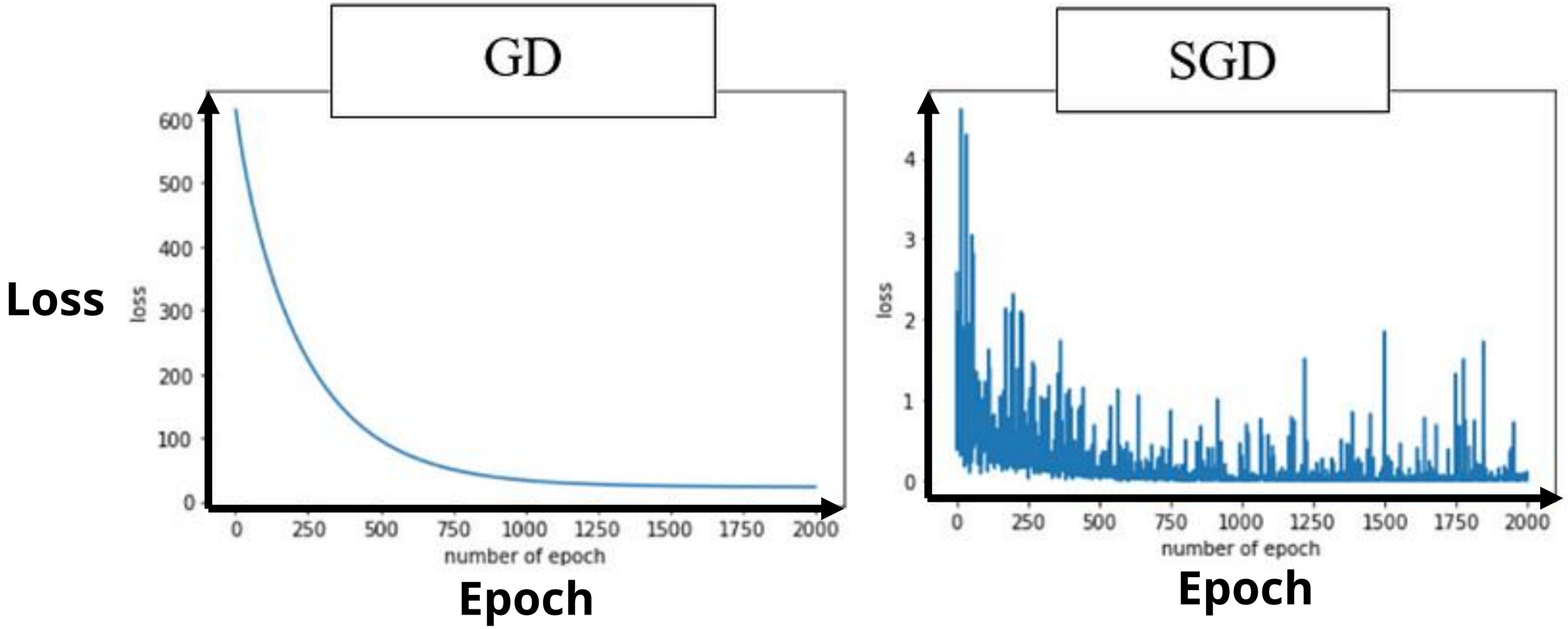


16 gradient computations

analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Stochastic Gradient Descent is Noisy and Unstable

- Gradient estimate using one single sample can be unreliable



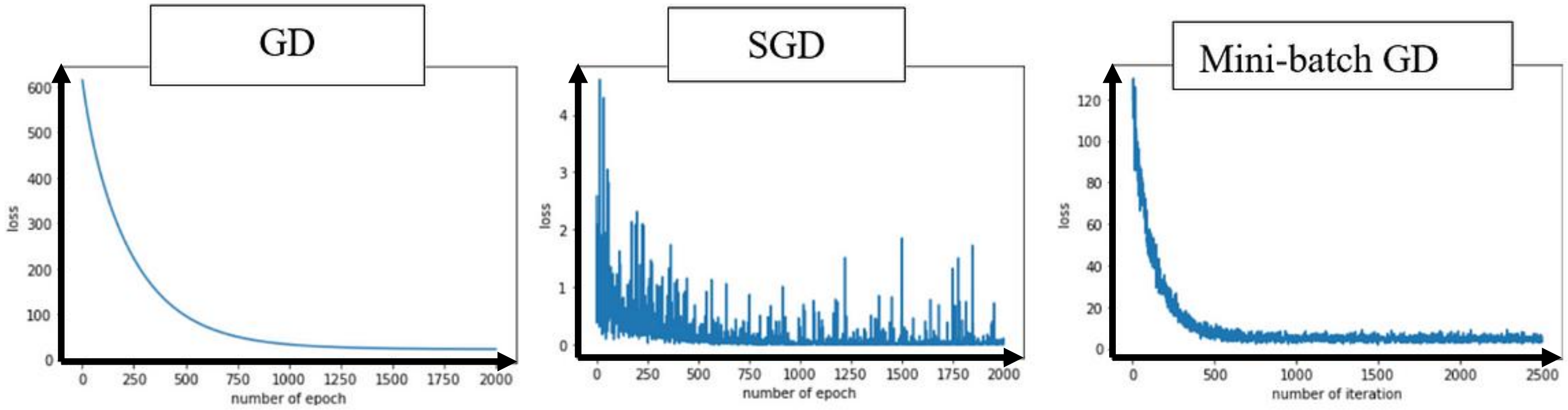
How about we use more samples to estimate the gradient?

towardsdatascience.com/deep-learning-optimizers-436171c9e23f

Mini-batch Gradient Descent

- **Intuition:** **Estimate** the gradient using **several random training samples**

Loss



Epoch | Epoch | Epoch

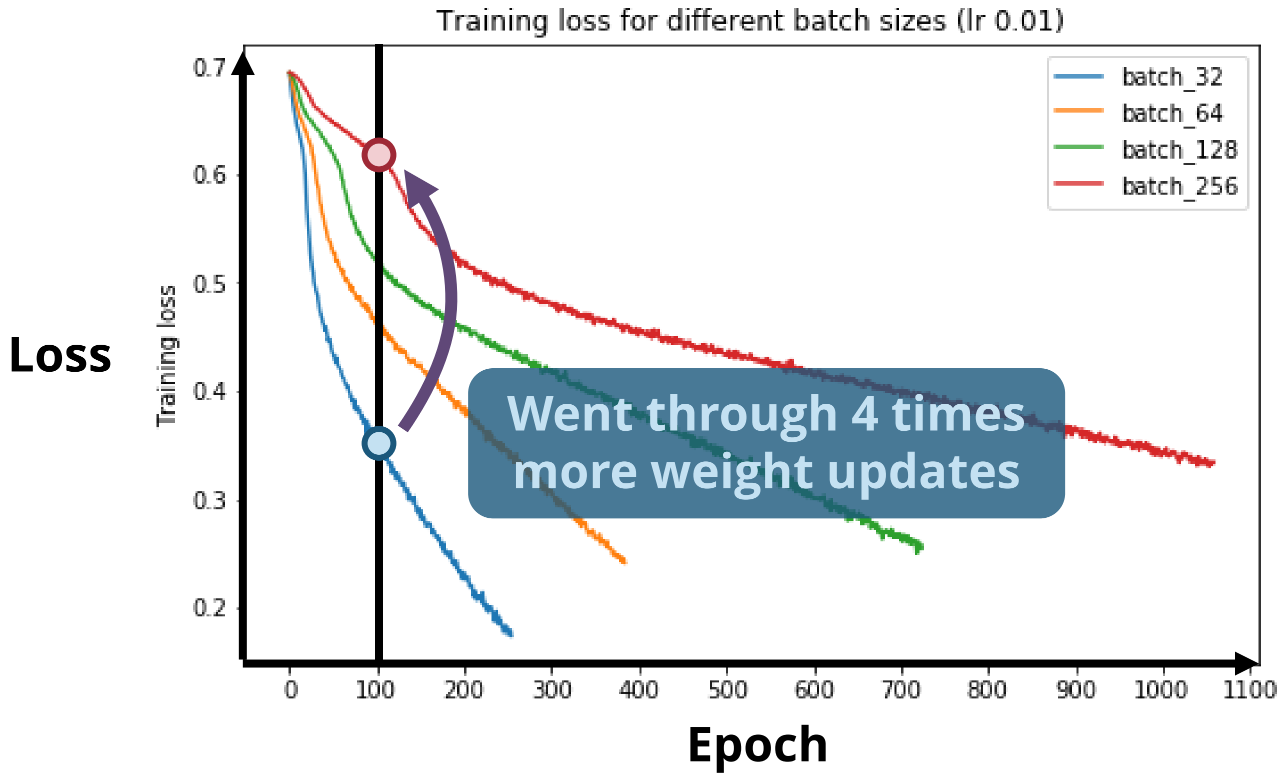
$\text{batch size} = N$ | $\text{batch size} = 1$ | $1 < \text{batch size} < N$

analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Effects of Batch Size

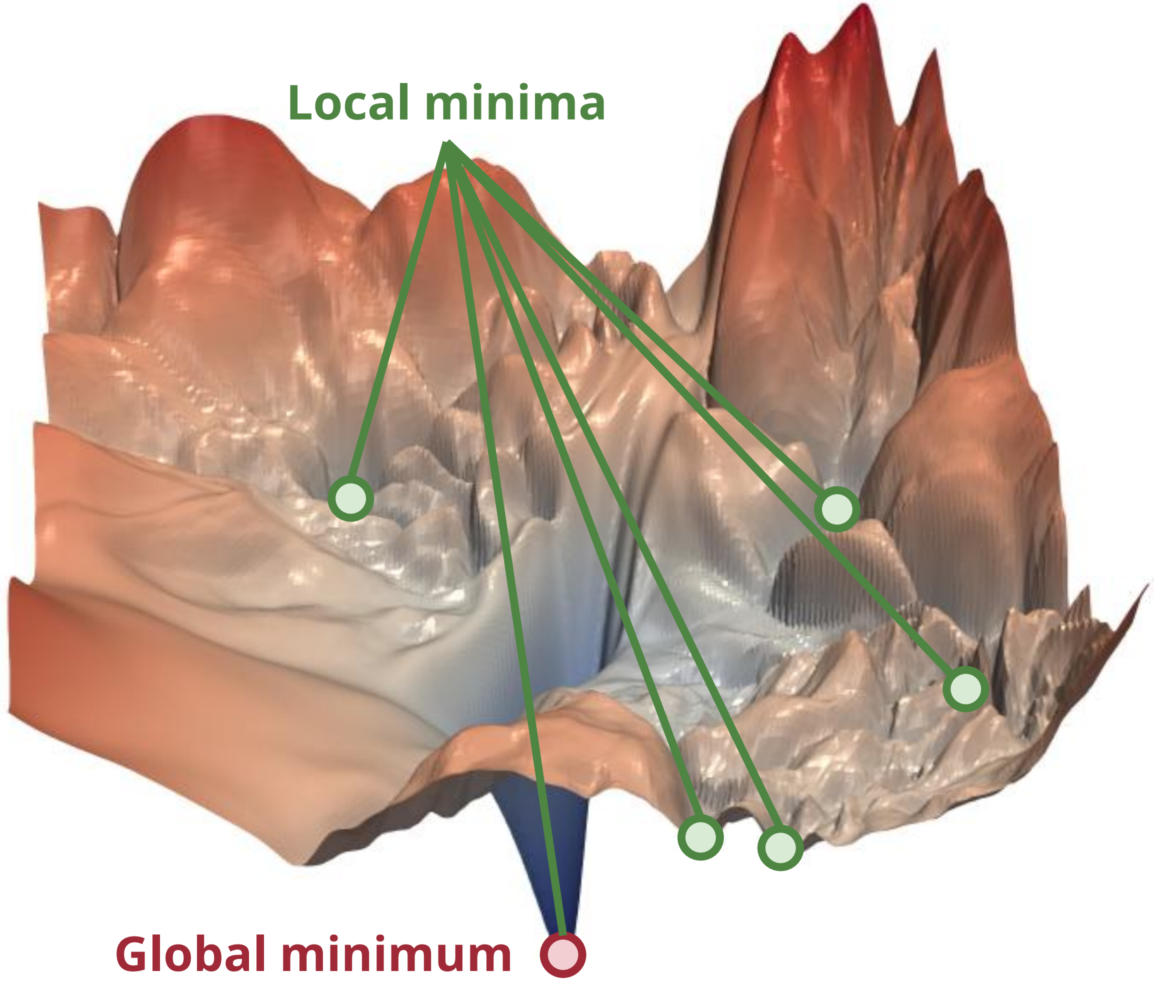
- An **epoch** is a full run of the whole dataset
- Steps per epoch depends on the batch size

$$\#(\mathbf{steps}) = \frac{\#(\mathbf{training\ samples})}{\mathbf{batch\ size}}$$



medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57

Local Minima in Complex Loss Landscape



Solution 1
Use an optimizer with adaptive learning rate

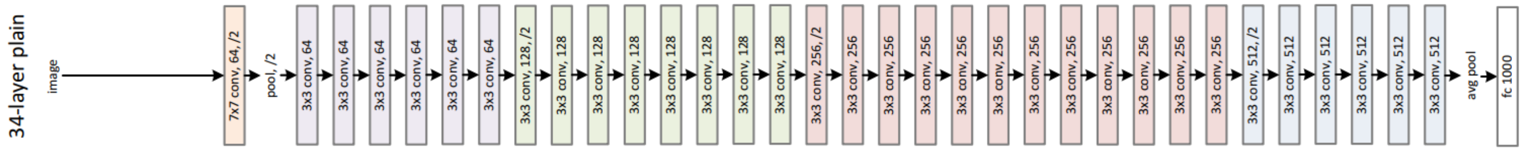
Solution 2
Use a stochastic optimizer

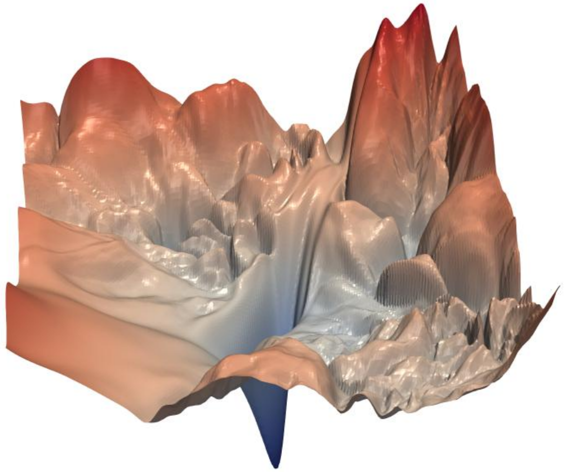
Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

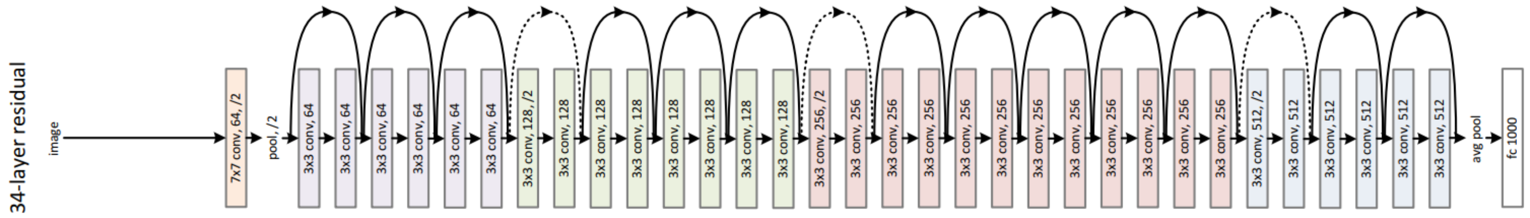
Skip Connections

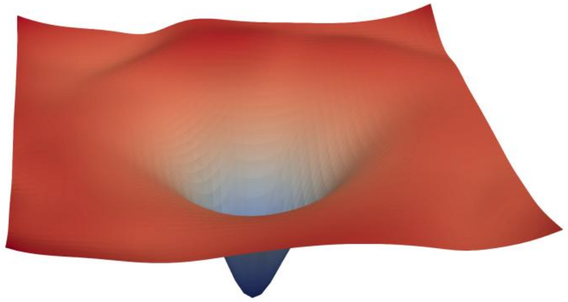
Without skip connections





With skip connections





Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.