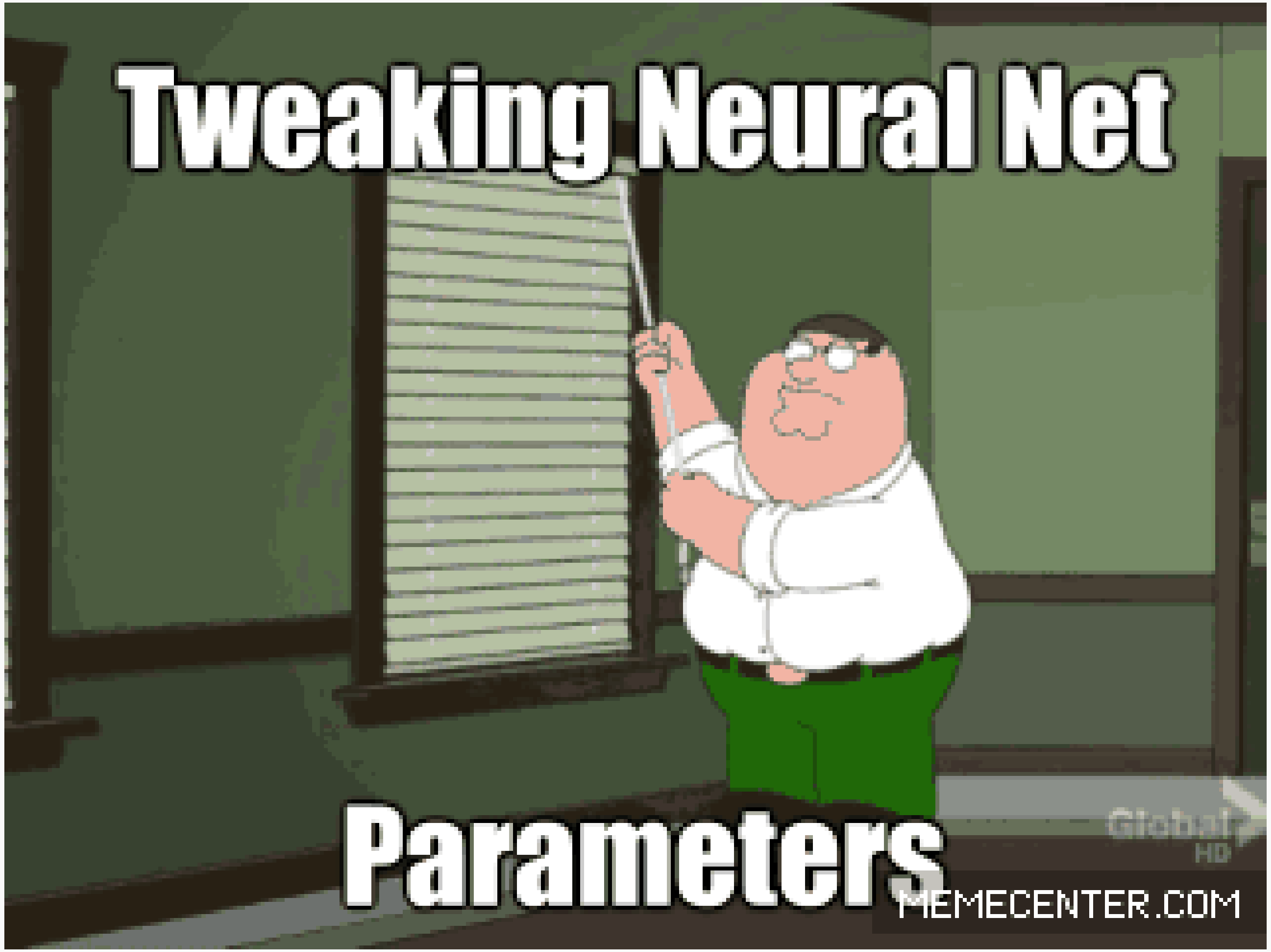
Fitting Neural Networks and Stochastic Gradient Descent

CS1090B Data Science II – Spring 2025

Pavlos Protopapas, Natesh Pillai, and Chris Gumb

Delphine Veronese-Milin
Monument Valley
Arizona, USA


Tweaking Neural Net
Parameters
MEMECENTER.COM

Gradient Descent Considerations

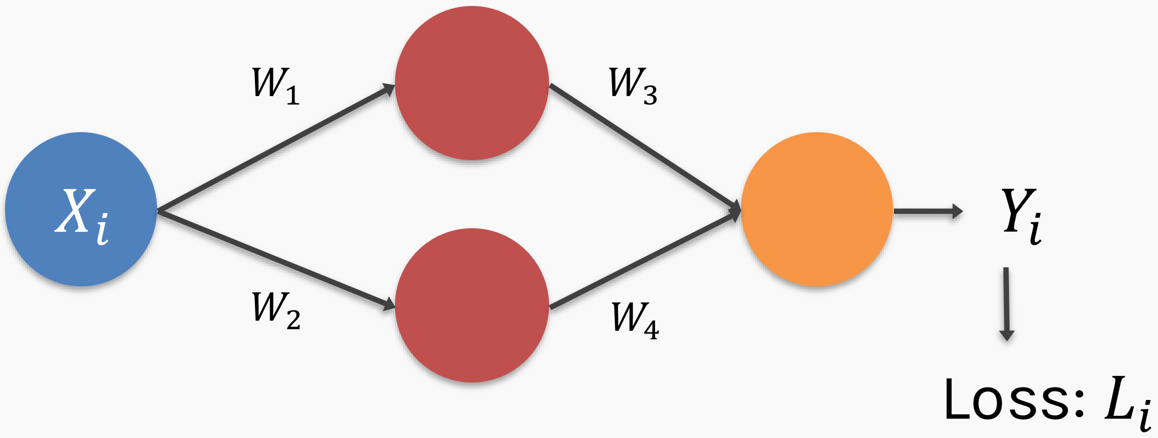
- We still need to calculate the derivatives.
- We need to set the learning rate.
- Local vs global minima.
- **The full likelihood function includes summing up all individual '*errors*'. Sometimes this includes hundreds of thousands of examples.**

Stochastic Gradient Descent

Deep learning models perform best with large amounts of data. As dataset size increases, models generally improve in performance.

In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in **one epoch** for SGD:

1. Take one example
2. Feed it to Neural Network (forward mode) and calculate the loss function

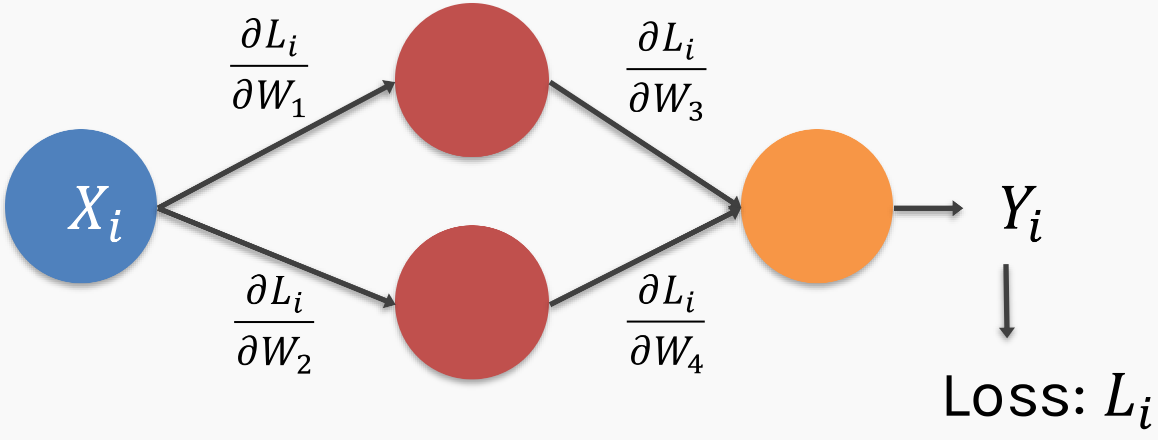


Stochastic Gradient Descent

Deep learning models perform best with large amounts of data. As dataset size increases, models generally improve in performance.

In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in **one epoch** for SGD:

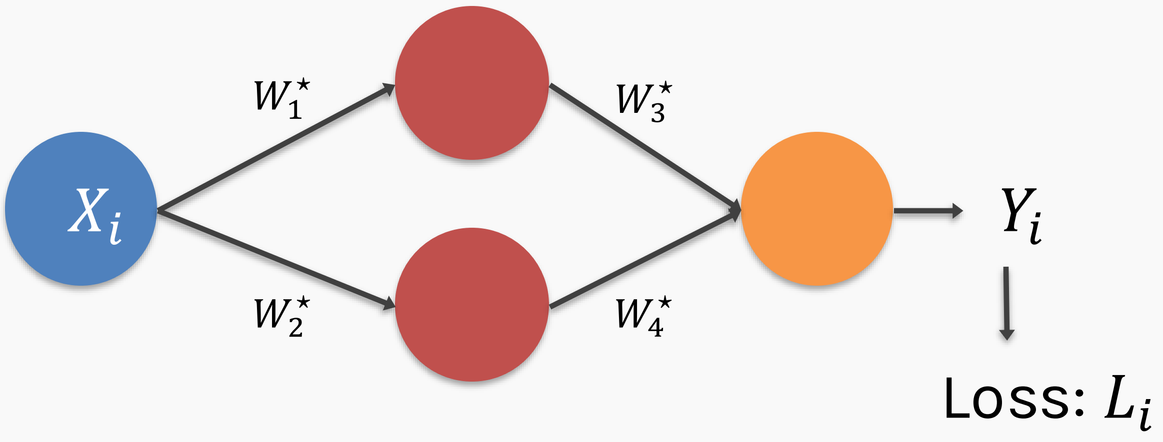
1. Take one example
2. Feed it to Neural Network (forward mode) and calculate the loss function
3. Calculate its gradient wrt to W's



Stochastic Gradient Descent

Deep learning models perform best with large amounts of data. As dataset size increases, models generally improve in performance.

In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in **one epoch** for SGD:



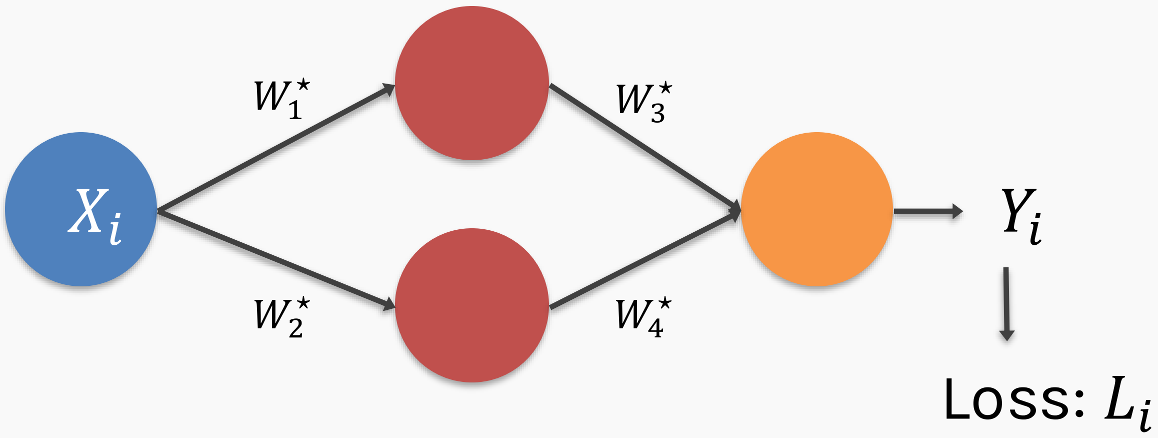
1. Take one example
2. Feed it to Neural Network (forward mode)
   and calculate the loss function
3. Calculate its gradient wrt to W's
4. Use the gradient we calculated in step 3 to update the weights using

   gradient descent update $W_j^\star = W_j - \eta \frac{\partial L_i}{dW_j}$

Stochastic Gradient Descent

Deep learning models perform best with large amounts of data. As dataset size increases, models generally improve in performance.

In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in **one epoch** for SGD:



1. Take one example
2. Feed it to Neural Network (forward mode) and calculate the loss function
3. Calculate its gradient wrt to W's
4. Use the gradient we calculated in step 3 to update the weights using gradient descent update $W_j^\star = W_j - \eta \frac{\partial L_i}{dW_j}$

5. Repeat

Stochastic Gradient Descent

Epoch:

One forward pass and one backward pass of ***all*** the training examples is called an **epoch**.

Why is this stochastic?

Order the algorithm sees data is random.

Mini-Batch Stochastic Gradient Descent

Let $\mathcal{L}_i$ be the loss of an individual sample

Instead of using one example for every step, use a subset of them which we call mini batch.

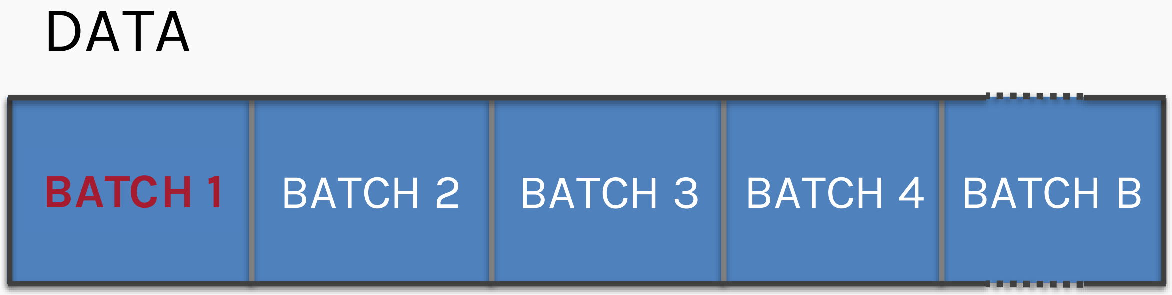
We use only the points in the mini batch to calculate the loss function:

$$\boxed{\mathcal{L}^k = \sum_{i \in b^k} \mathcal{L}_i}$$

which **approximates** the full loss function.

Note: The process will be same for any loss function, be it, regression or classification

1. Divide data into mini-batches
2. Pick a mini-batch
3. Feed it to Neural Network
4. Calculate the mean gradient of the mini-batch
5. Use the mean gradient we calculated in step 4 to update the weights
6. Repeat steps 1–5 for the mini-batches we created



Calculate $L_1 \Rightarrow \frac{\partial L_1}{\partial w} \quad \Rightarrow \quad w^\star = w - \eta \frac{\partial L_1}{\partial w}$

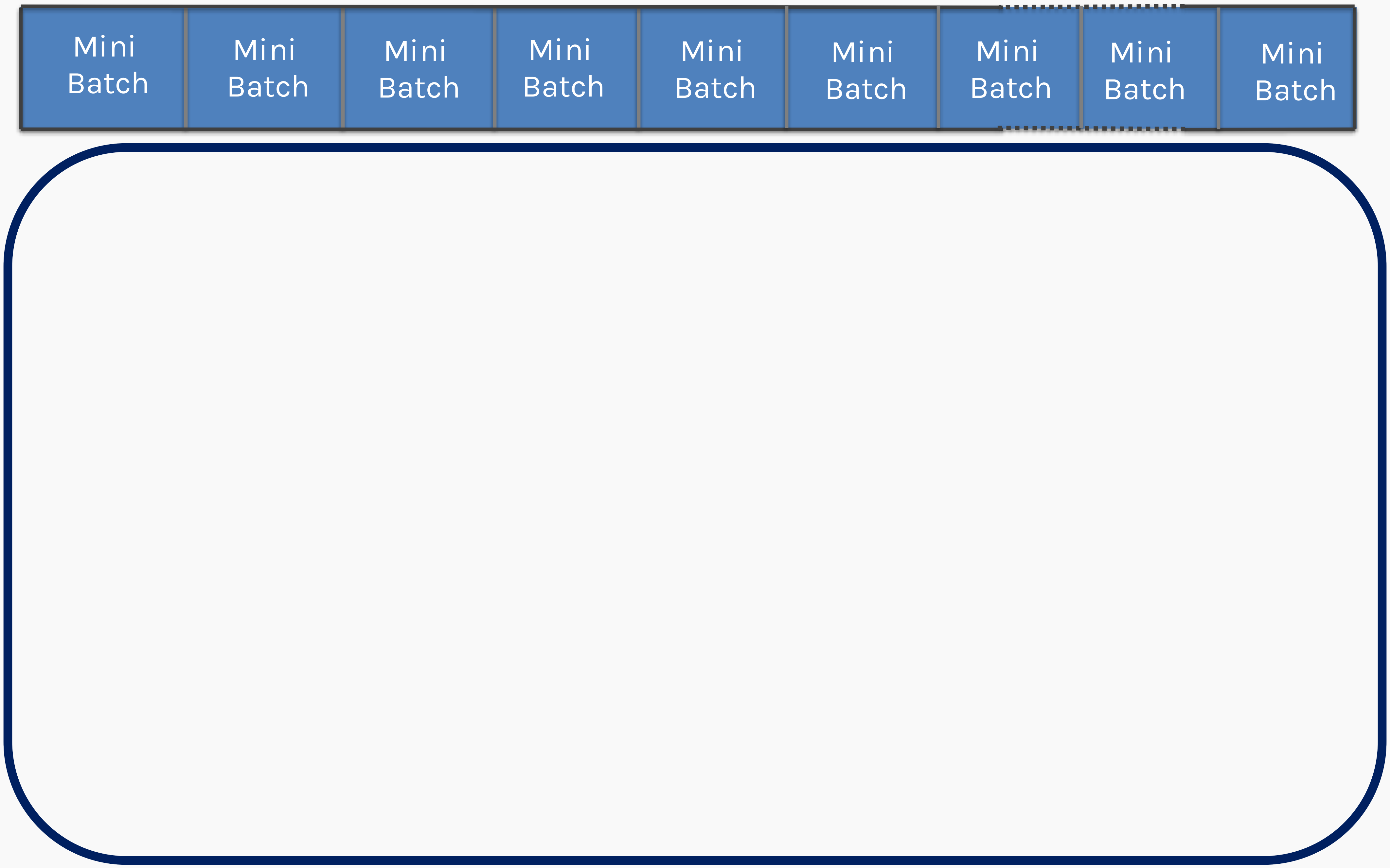
Calculate $L_2 \Rightarrow \frac{\partial L_2}{\partial w} \quad \Rightarrow \quad w^\star = w - \eta \frac{\partial L_2}{\partial w}$

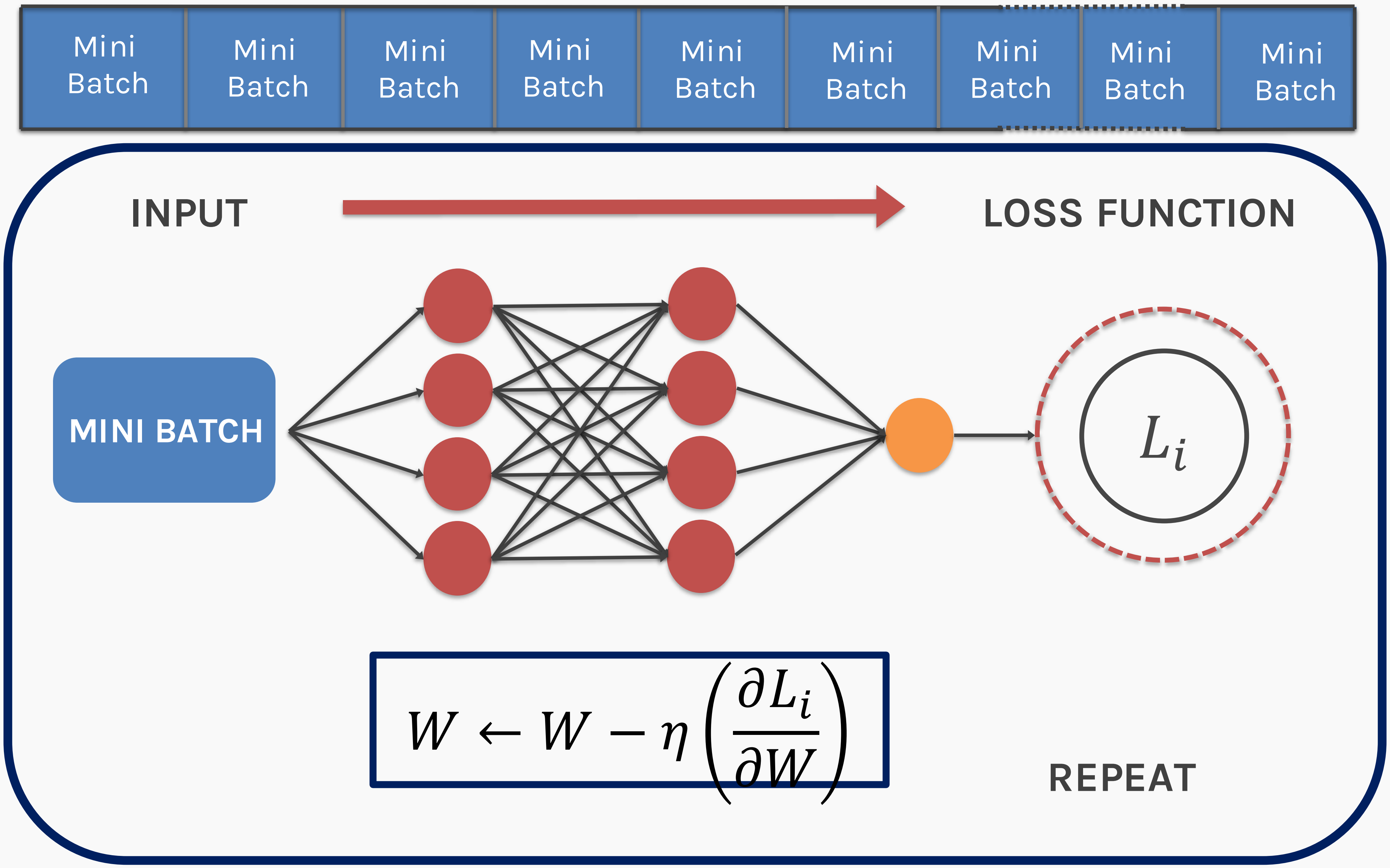
$L_B \Rightarrow \frac{\partial L_B}{\partial w} \quad \Rightarrow \quad w^\star = w - \eta \frac{\partial L_B}{\partial w}$

1. Divide data into mini-batches
2. Pick a mini-batch
3. Feed it to Neural Network
4. Calculate the mean gradient of the mini-batch
5. Use the mean gradient we calculated in step 4 to update the weights
6. Repeat steps 1–5 for the mini-batches we created

One epoch

Reshuffle data and repeat

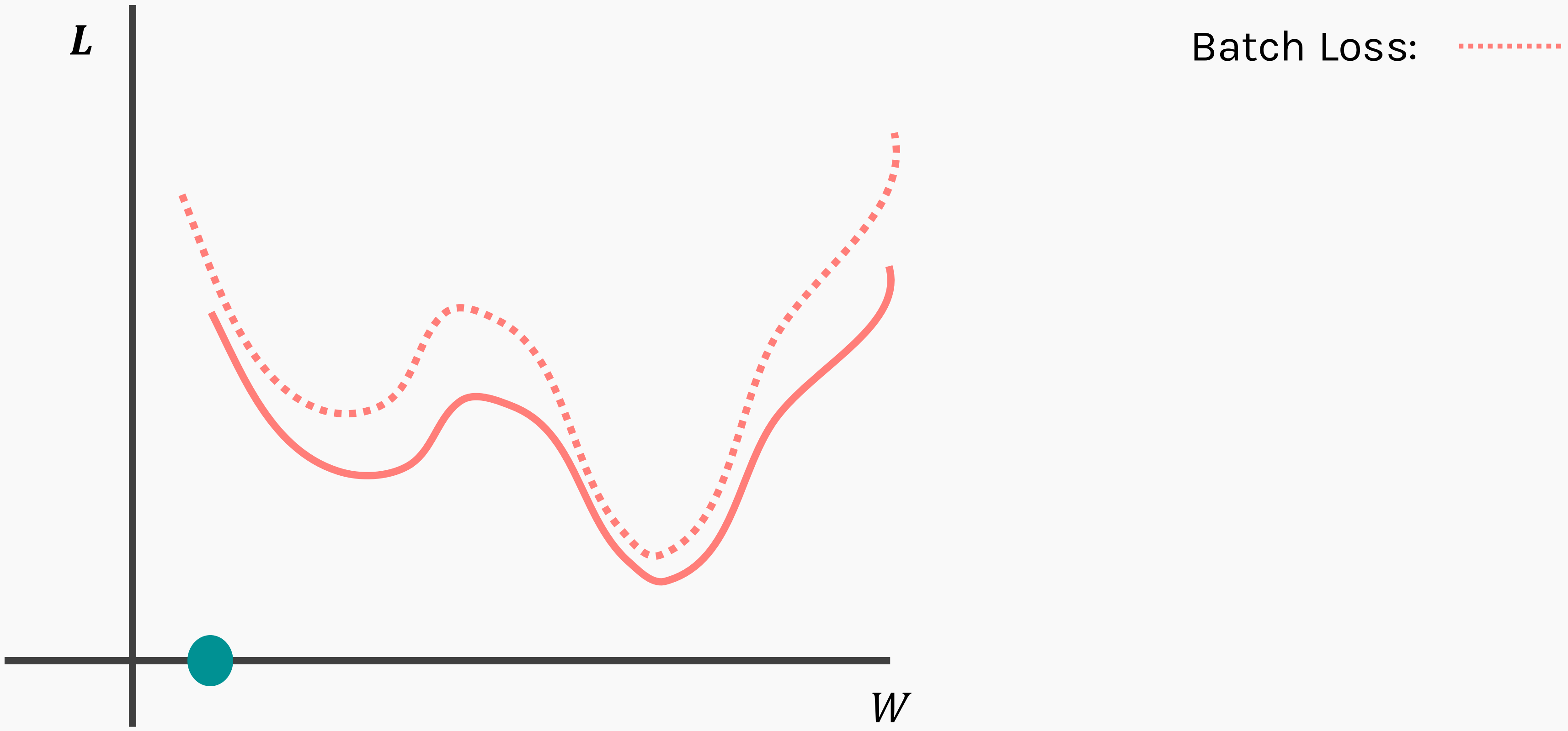

Mini Batch
Mini Batch
Mini Batch
Mini Batch
Mini Batch
Mini Batch
Mini Batch
Mini Batch
Mini Batch



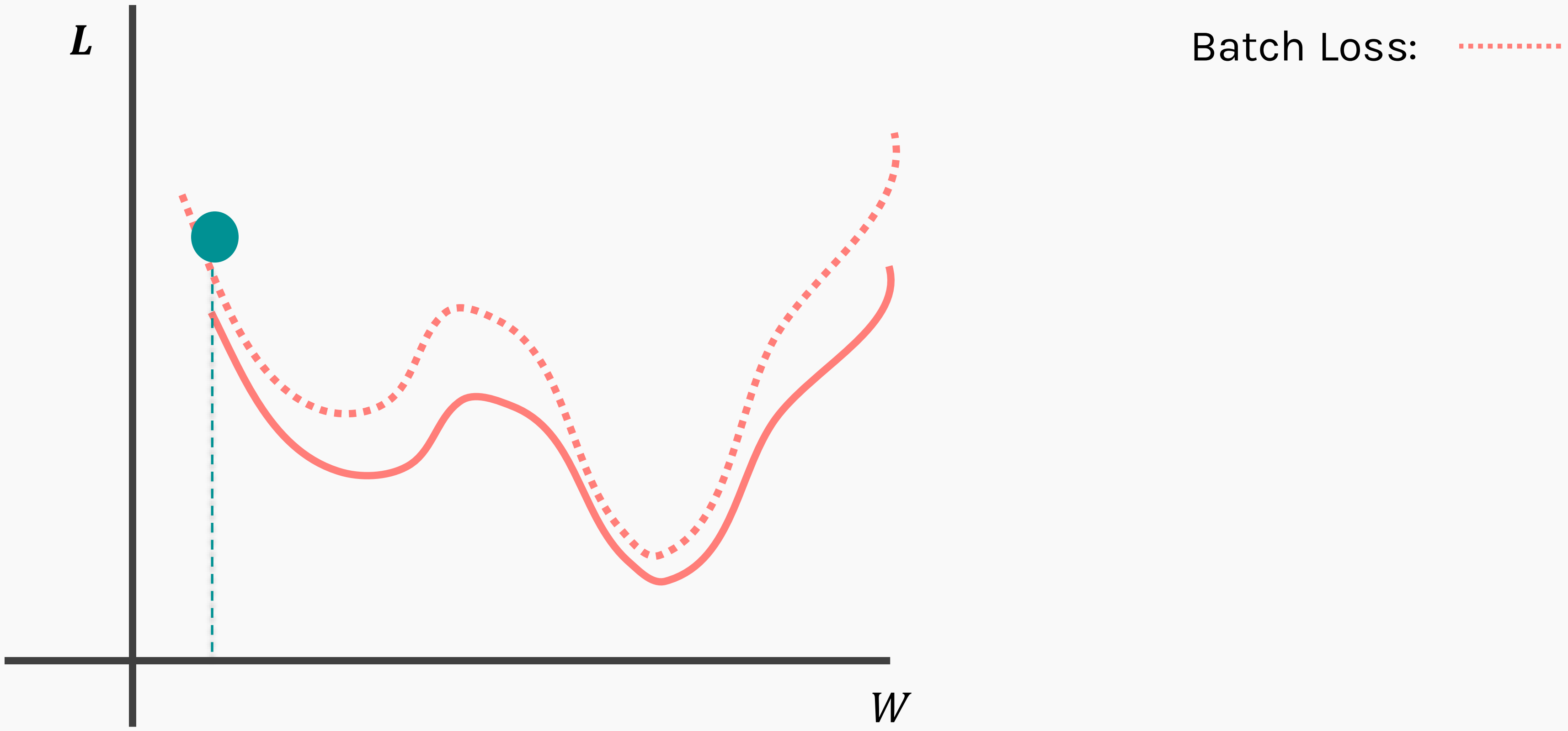
$$W \leftarrow W - \eta \left(\frac{\partial L_i}{\partial W} \right)$$

Ready for some magic?

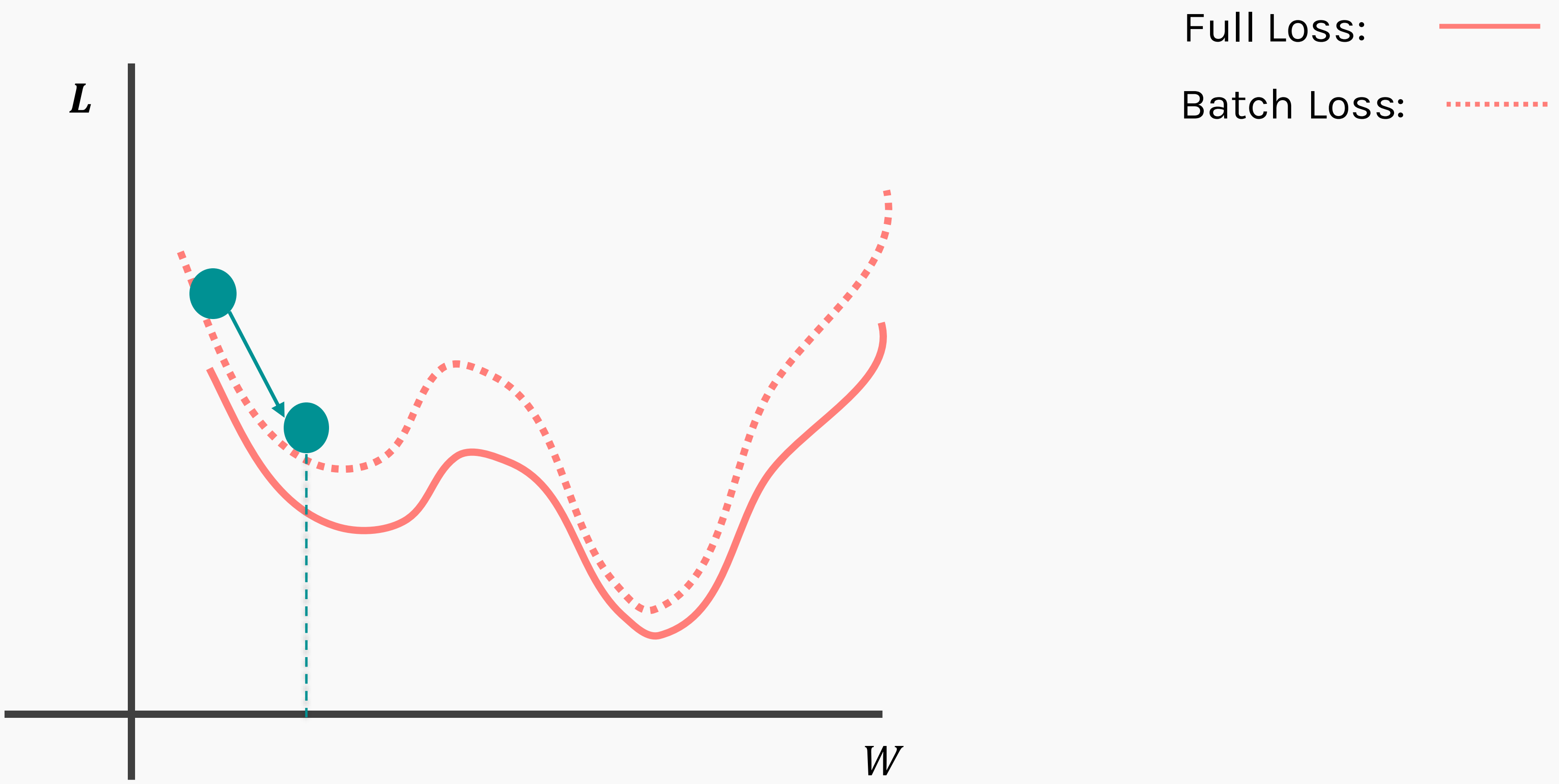
Batch and Stochastic Gradient Descent



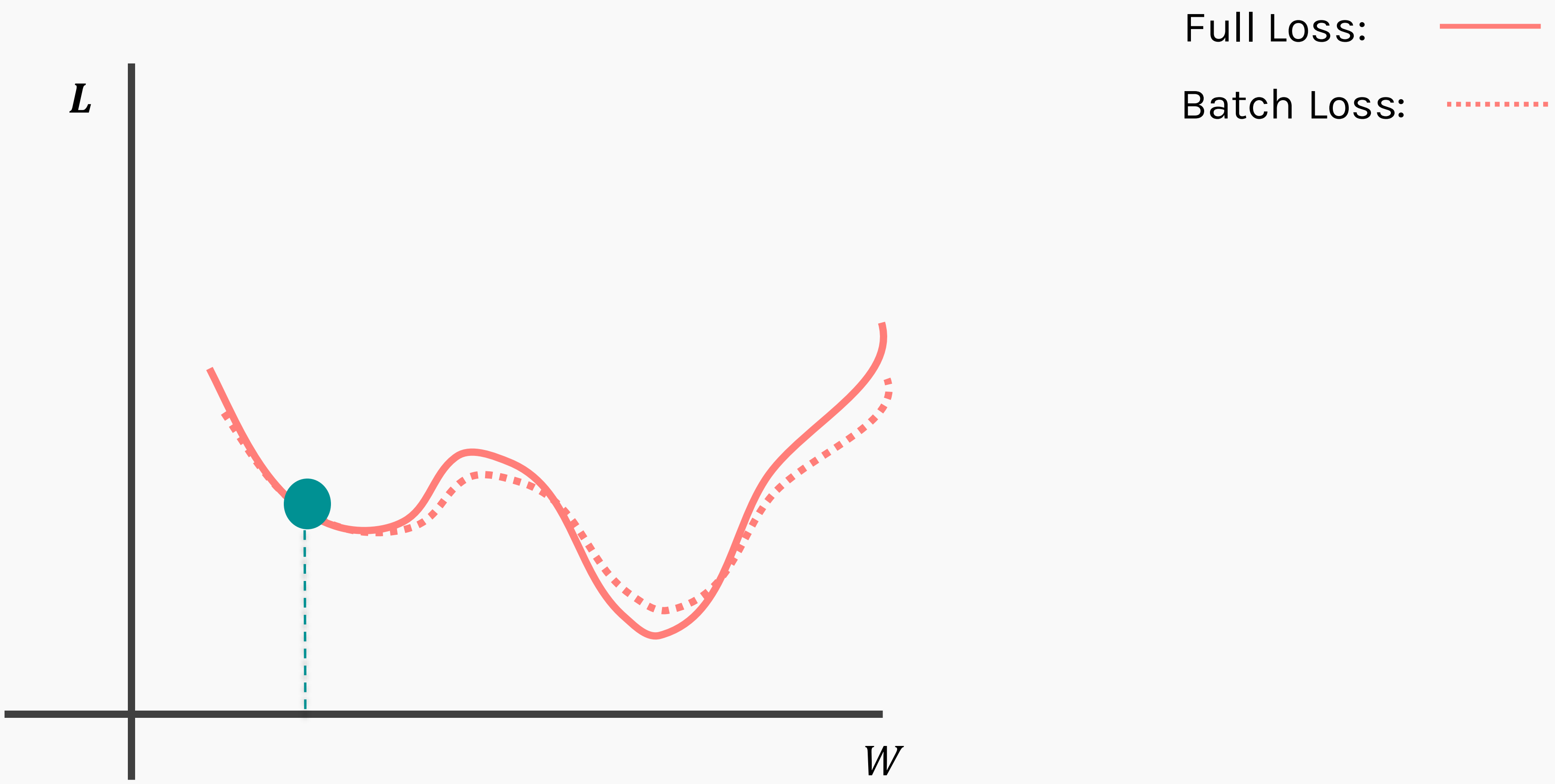
Batch and Stochastic Gradient Descent



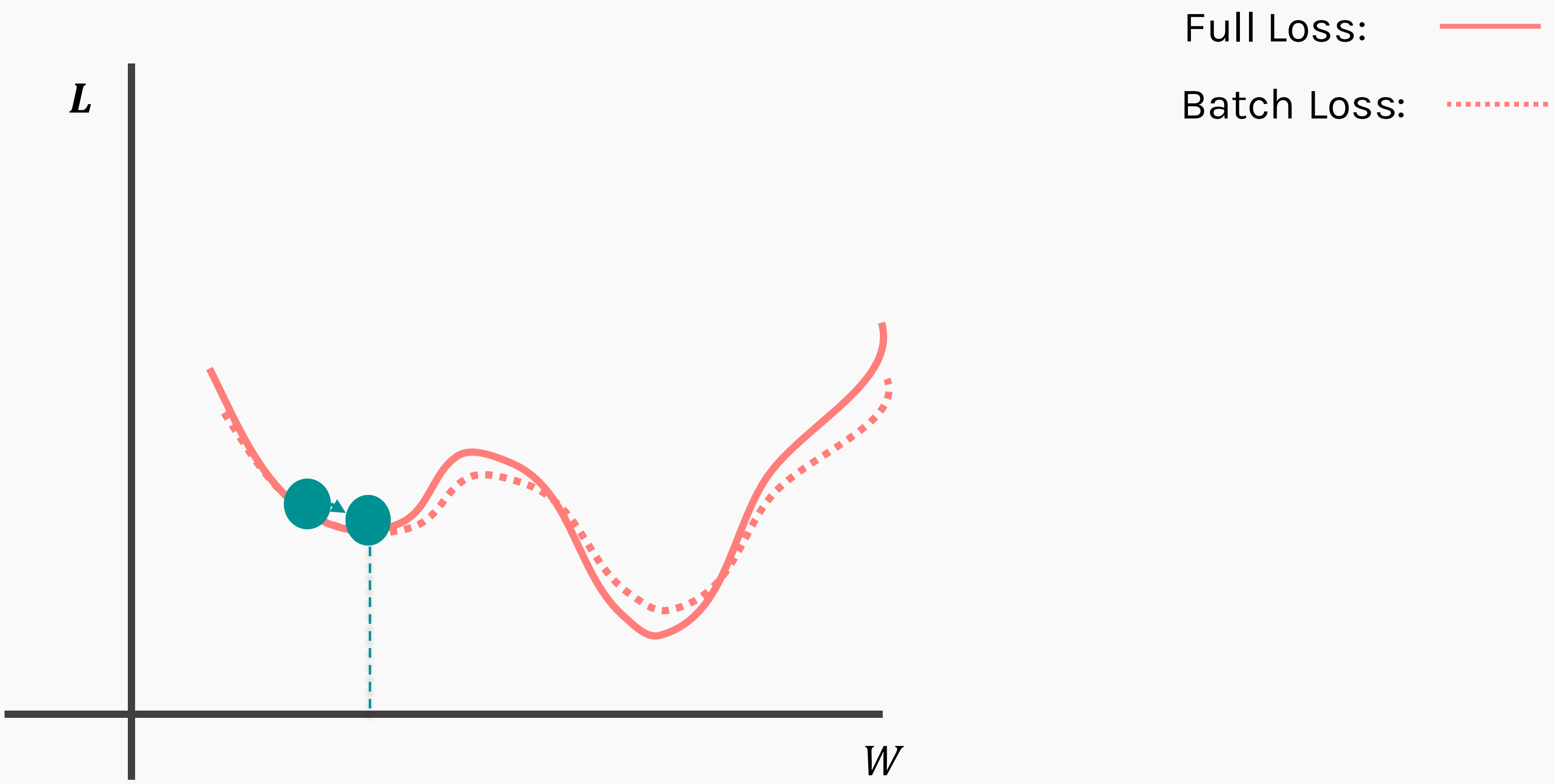
Batch and Stochastic Gradient Descent



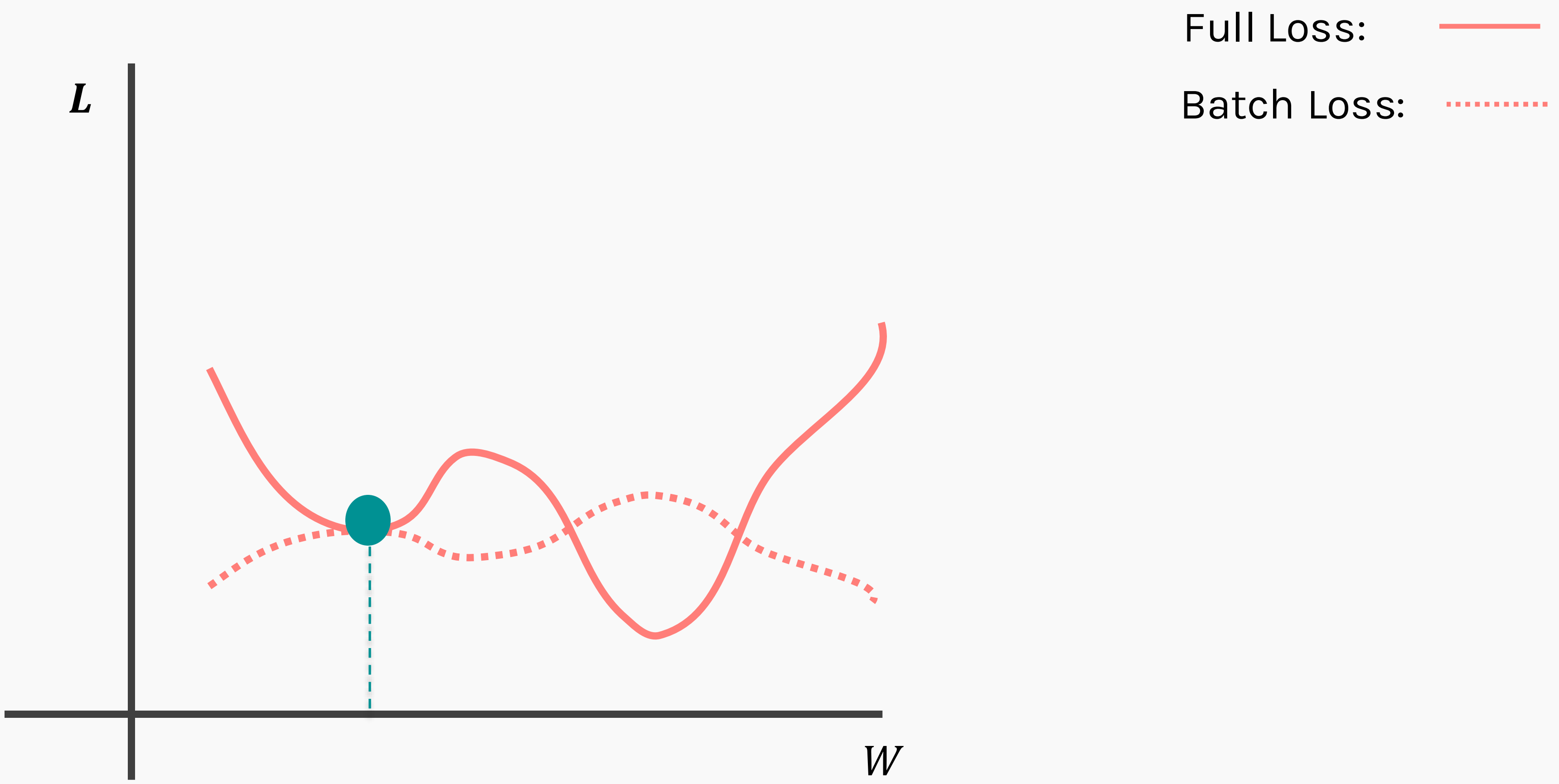
Batch and Stochastic Gradient Descent



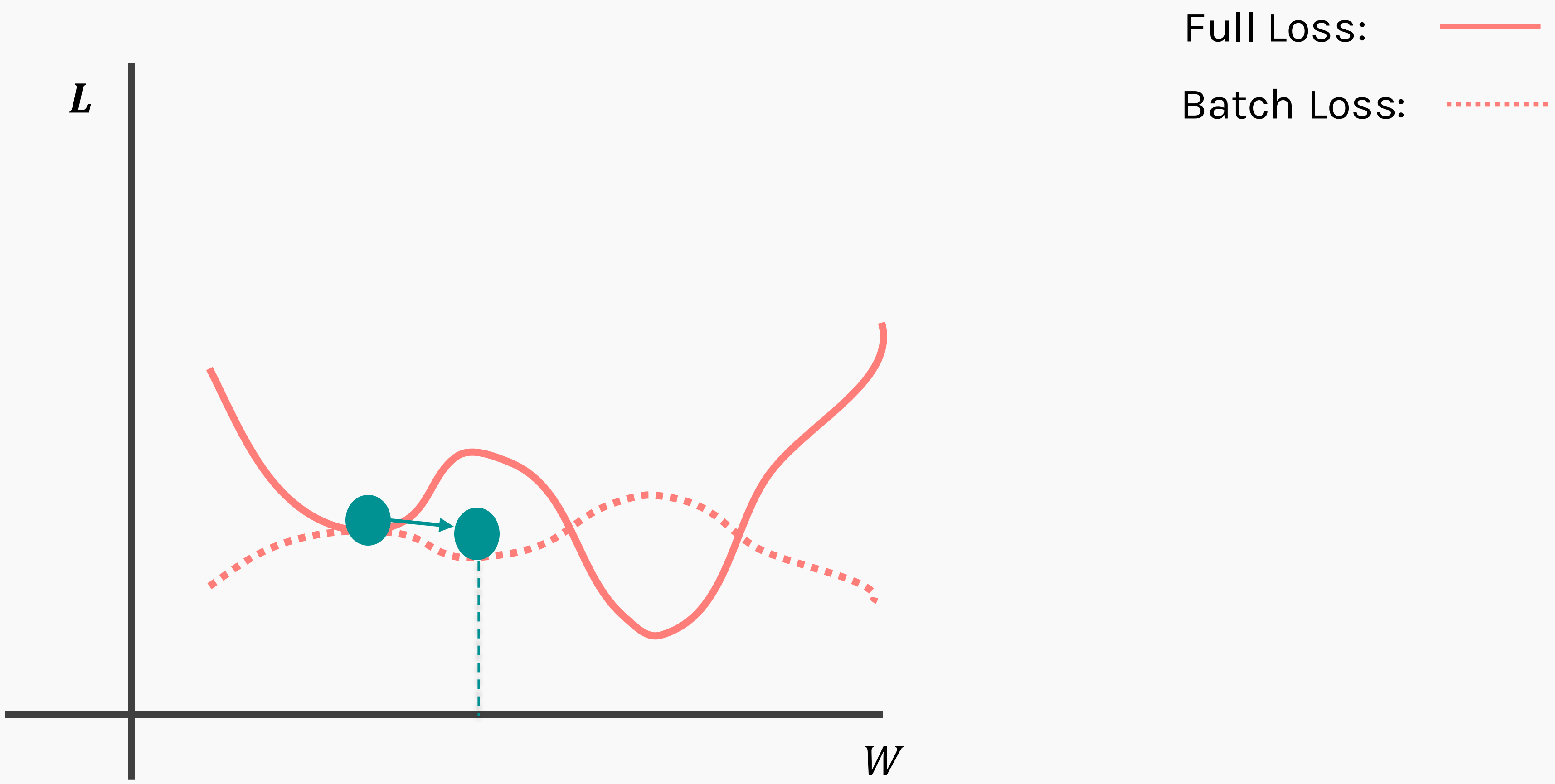
Batch and Stochastic Gradient Descent



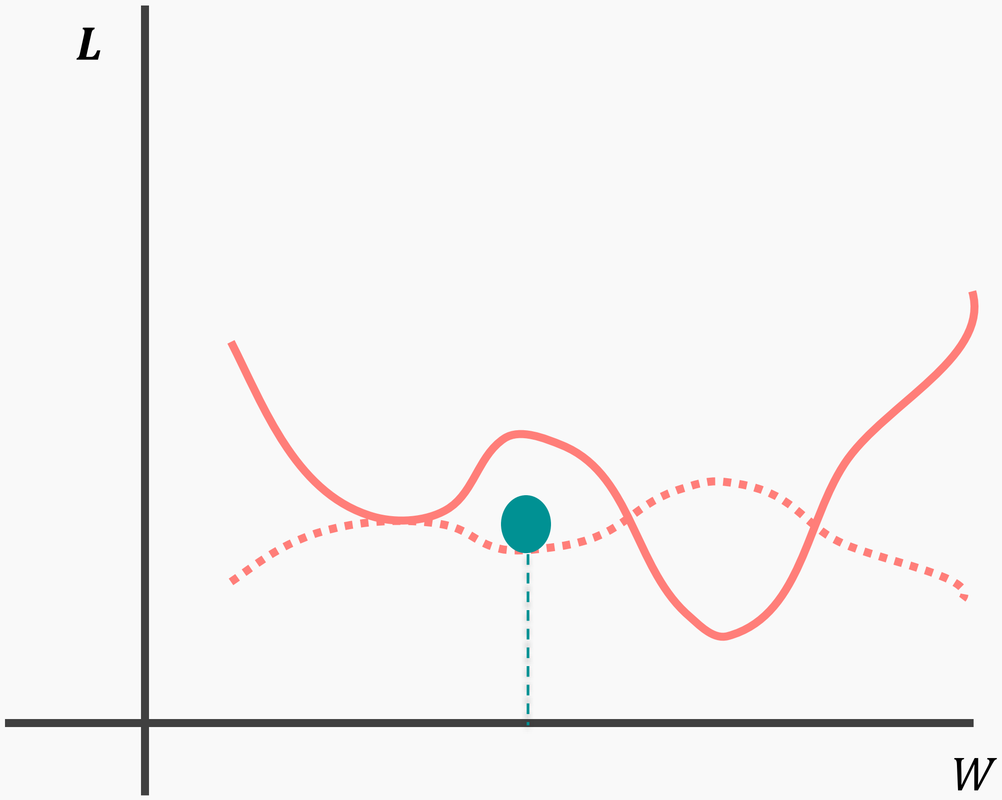
Batch and Stochastic Gradient Descent



Batch and Stochastic Gradient Descent



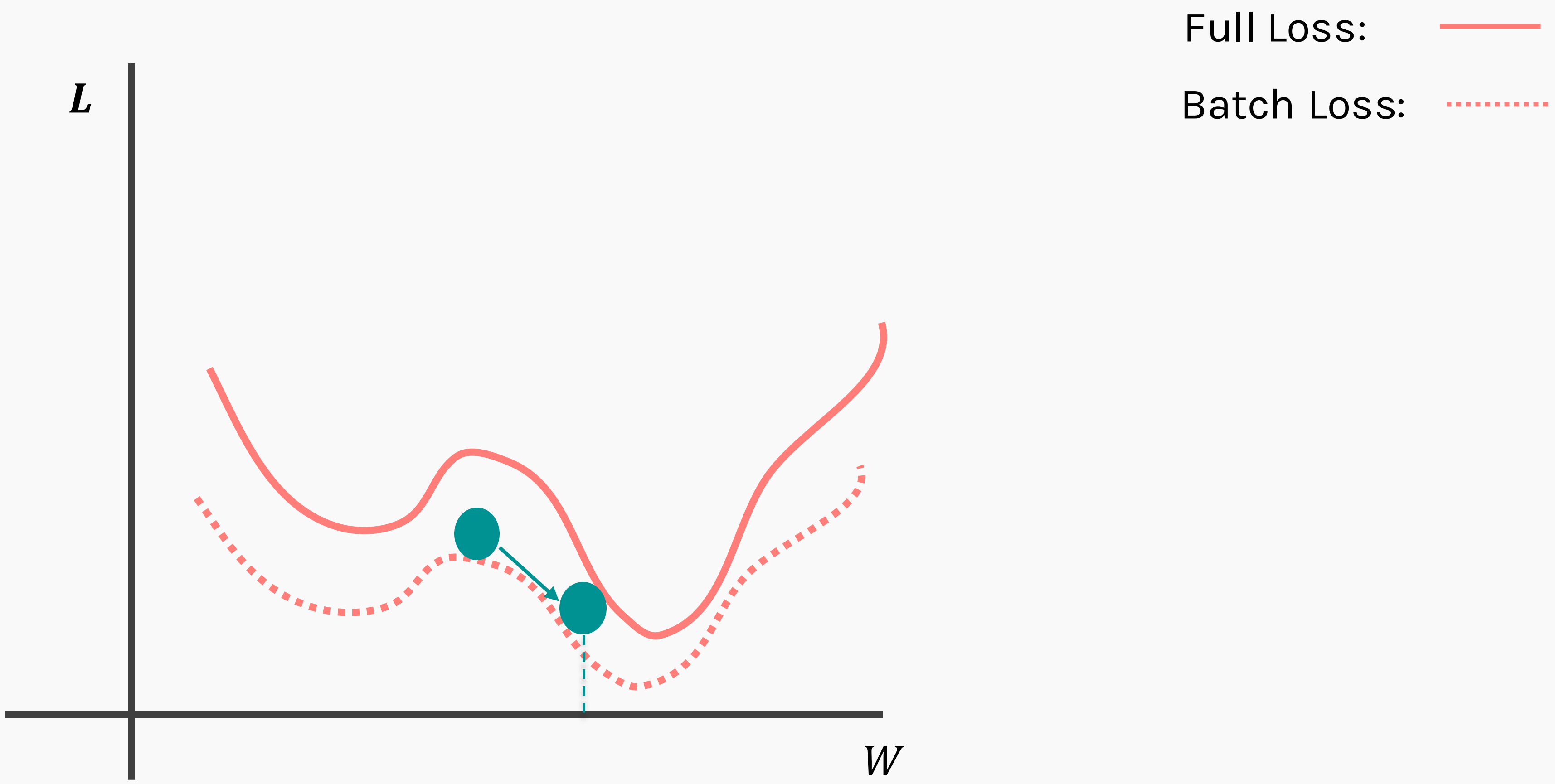
Batch and Stochastic Gradient Descent



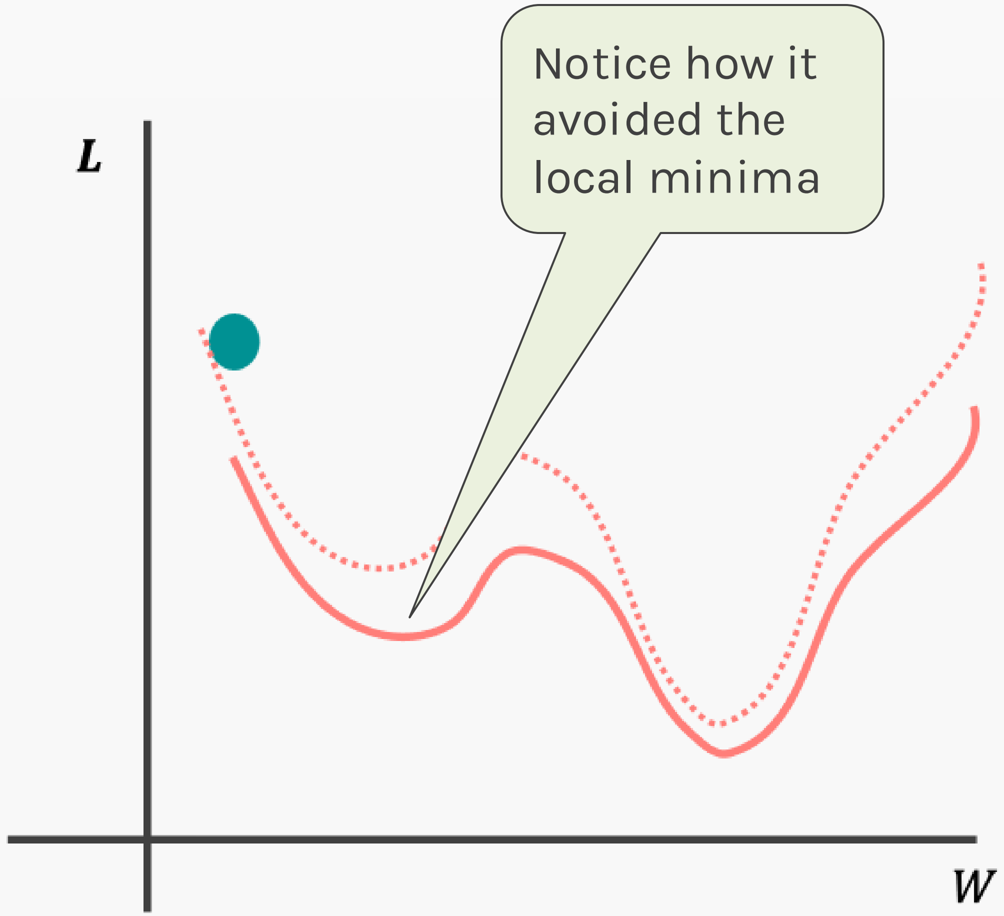
Full Loss:

Batch Loss:

Batch and Stochastic Gradient Descent



Batch and Stochastic Gradient Descent



Full Loss:

Batch Loss:

Stochastic Gradient Descent Summary

SGD is faster than the full gradient descent.

We can see a change in the loss landscape for every epoch because of the limited batch size.

The changes in loss landscape helps the model to get past the local minima and converge faster towards the global minima.

At the same time, this keeps the model from converging to the exact global minima (it keeps oscillating near the minima).

Mini-Batch Stochastic Gradient Descent

Mini-batch size:

Often referred to as batch size, it's common practice to select powers of 2 due to memory considerations, particularly when utilizing GPUs.

A large batch size is equivalent to gradient descent, also known as batch gradient descent.

Conversely, a very small batch size resembles vanilla SGD, although it can be erratic but effective in avoiding local minima.