Announcements

- **Upcoming deadlines**
  - Project Milestone 1 due on **tonight at 8pm**
  - Quiz 5 due **tomorrow at 8pm**
  - HW 4 due **Wednesday, October 25 at 8pm**

Lecture 14: Neural Networks

CIS 4190/5190

Fall 2023

Agenda

- **Model family**
  - Custom model family rather than a single model family

- **Optimization**
  - Backpropagation algorithm for computing gradient

Historical vs. Modern View

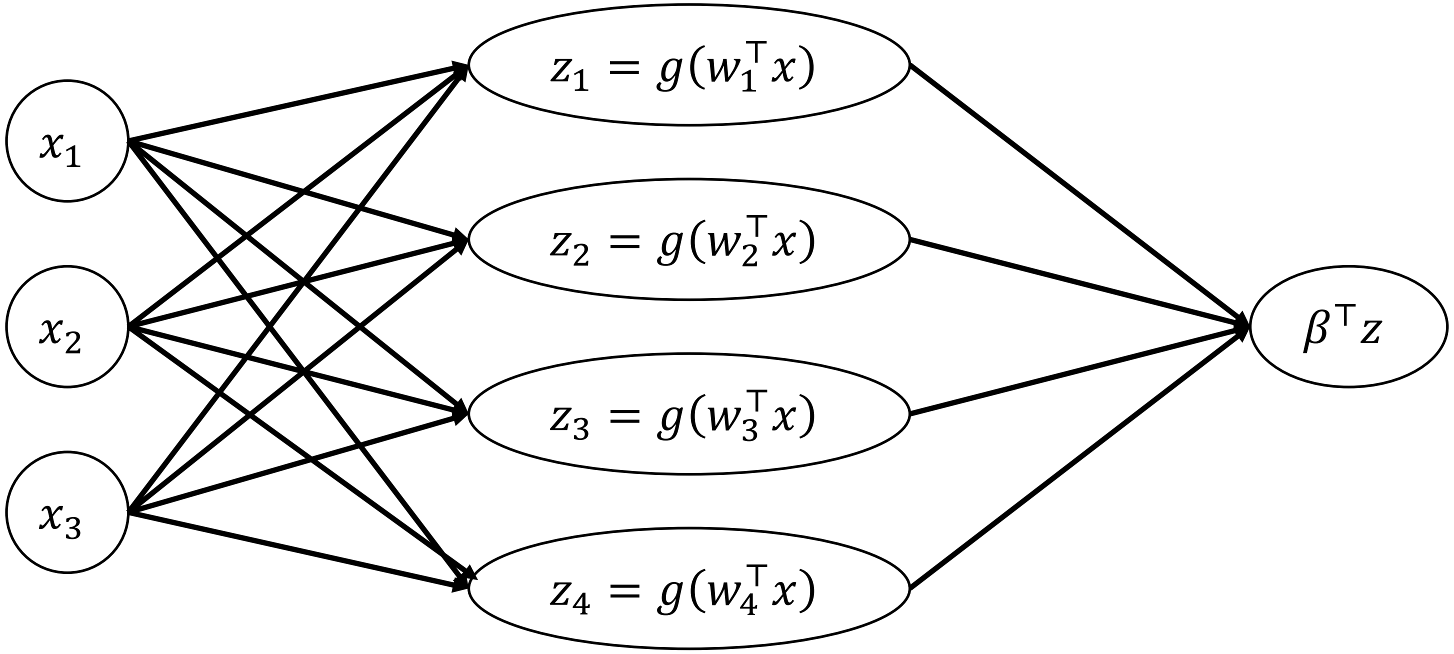
- **Historical view:** Specific model families
  - Feedforward neural networks, convolutional neural networks, etc.
  - Each new model family (“architecture”) requires a custom implementation

- **Modern view:** Design model families by composing building blocks
  - Building blocks are “layers”
  - Layers can be **programmatically** composed together (by composing, concatenating, etc.) to form different model families

Historical View

- **Feedforward neural network model family (for regression):**

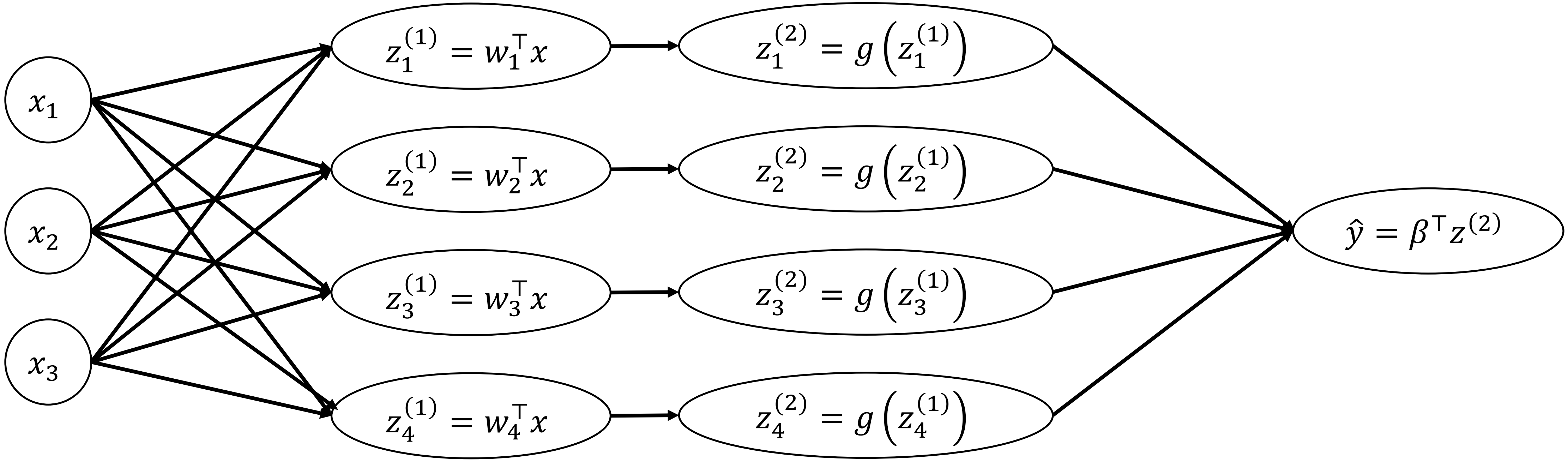
$$f_{W,\beta}(x) = \beta^\top g(Wx)$$



Modern View

- **Feedforward neural network model family (for regression):**

$$f_{W,\beta}(x) = f_\beta\left(g\left(f_W(x)\right)\right) = f_\beta \circ g \circ f_W(x)$$



Modern View

- Each **layer** is a parametric function $f_{W_j}: \mathbb{R}^k \to \mathbb{R}^h$ for some k, h

- Compose sequentially to form model family:

$$f_W(x) = f_{W_m}\left(\dots\left(f_{W_1}(x)\right)\dots\right)$$

- We will use the following notation:

$$f_W = f_{W_m} \circ \cdots \circ f_{W_1}$$

Modern View

- Each **layer** is a parametric function $f_{W_j}: \mathbb{R}^k \to \mathbb{R}^h$ for some k, h

- Can compose layers in other ways, e.g., concatenation:

$$f_W(x) = f_{W_1}(x) \oplus f_{W_2}(x)$$

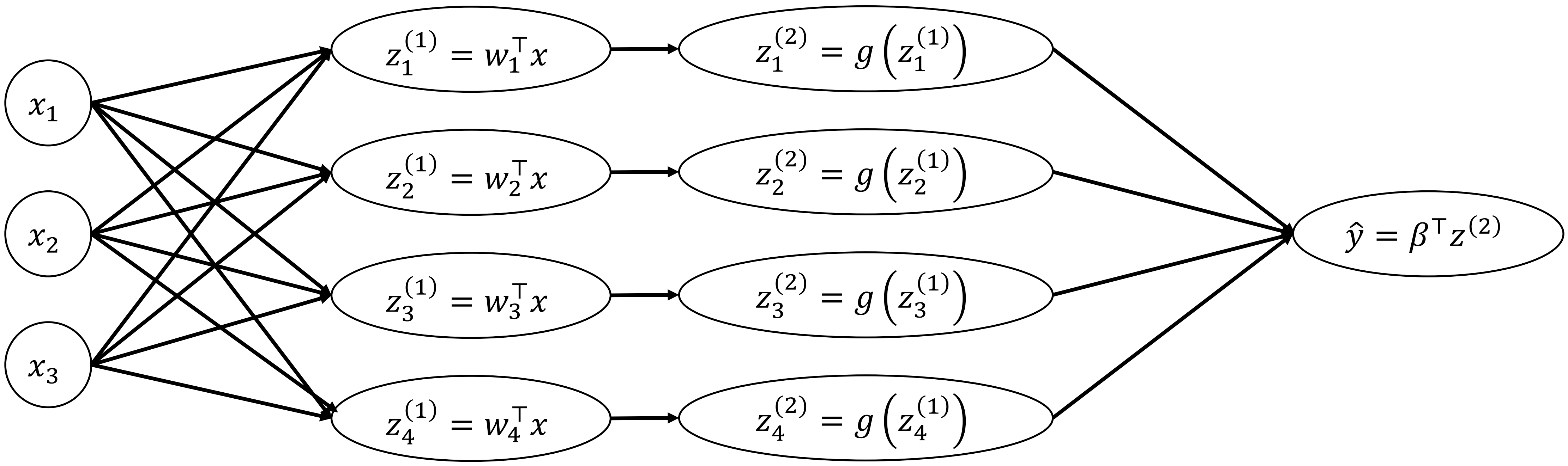
- Here, we have defined

$$\begin{bmatrix} z_1 & \cdots & z_d \end{bmatrix}^\top \oplus \begin{bmatrix} z_1' & \cdots & z_{d'}' \end{bmatrix}^\top = \begin{bmatrix} z_1 & \cdots & z_d & z_1' & \cdots & z_{d'}' \end{bmatrix}^\top$$

Modern View

- **Feedforward neural network model family (for regression):**

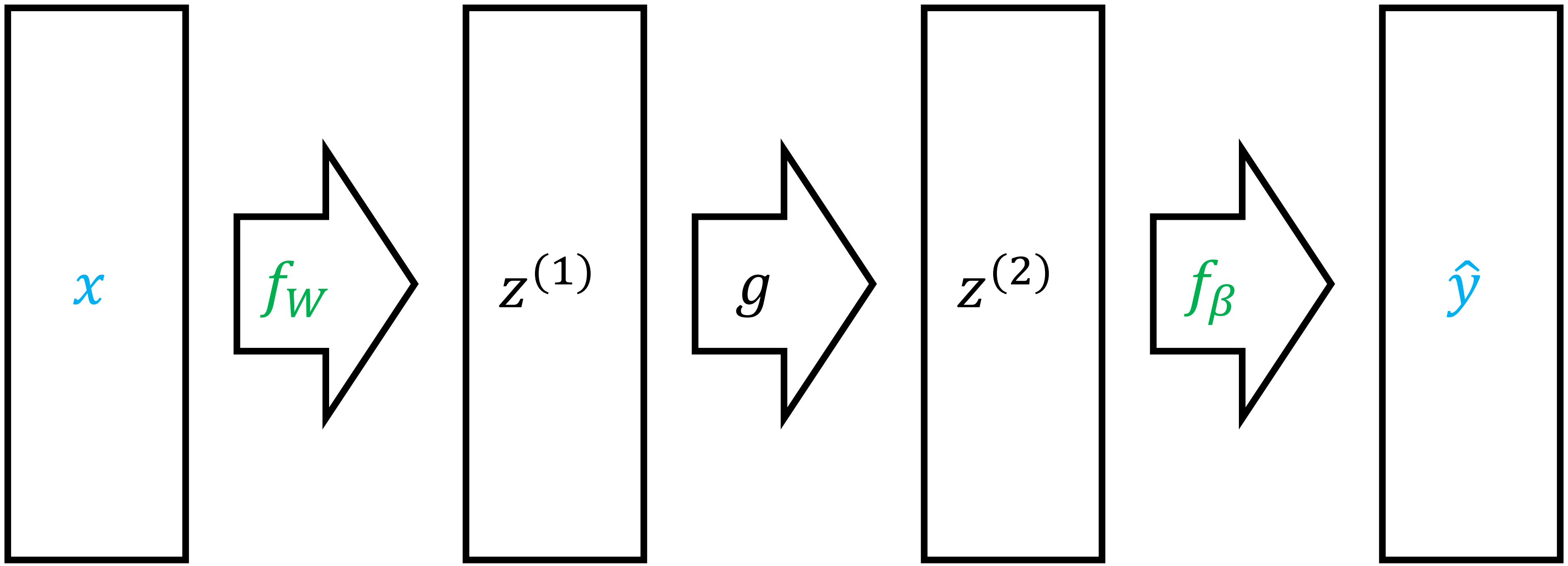
$$f_{W,\beta}(x) = f_\beta \circ g \circ f_W(x)$$



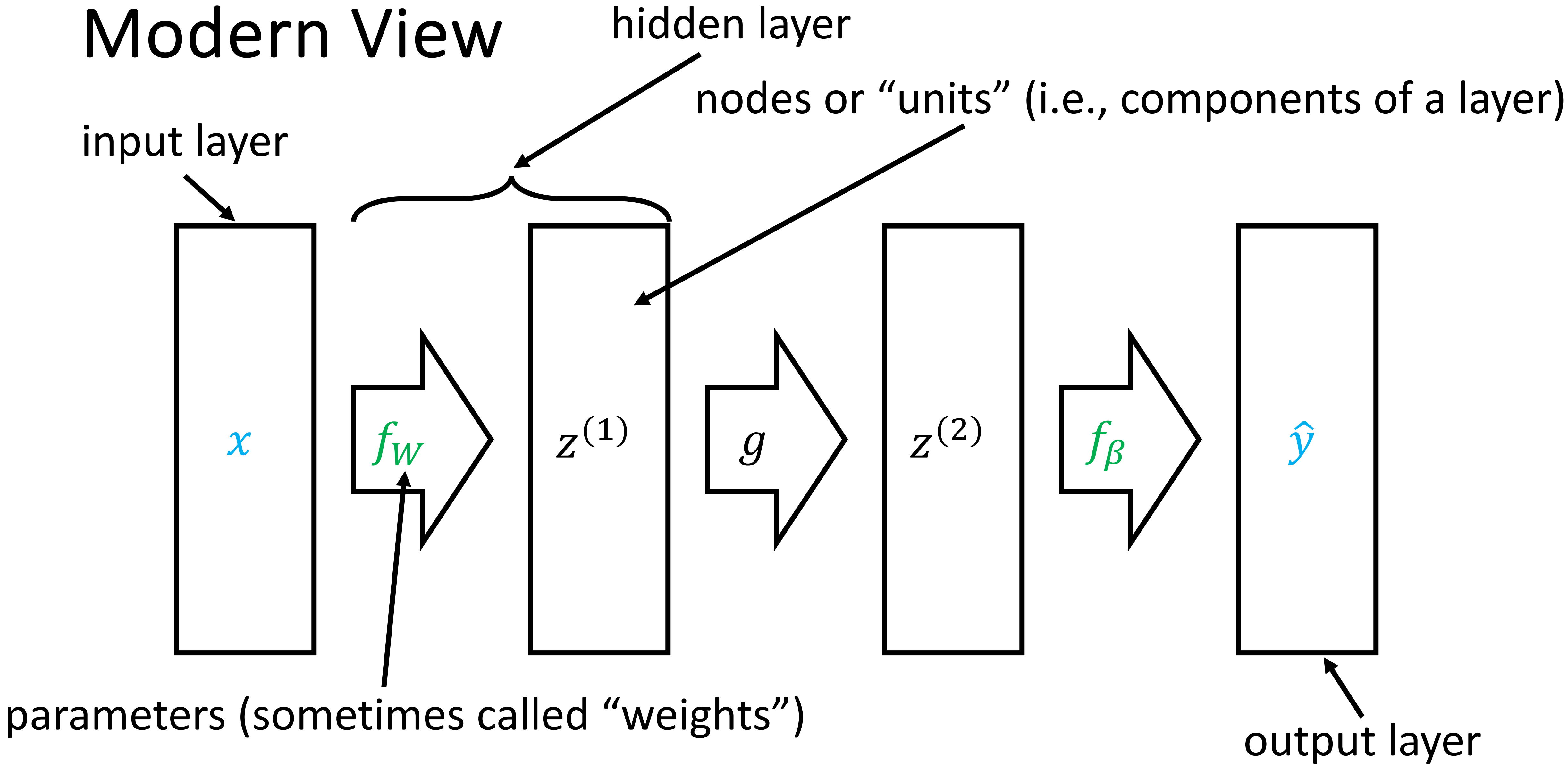
Modern View

- **Feedforward neural network model family (for regression):**

$$f_{W,\beta}(x) = f_\beta \circ g \circ f_W(x)$$



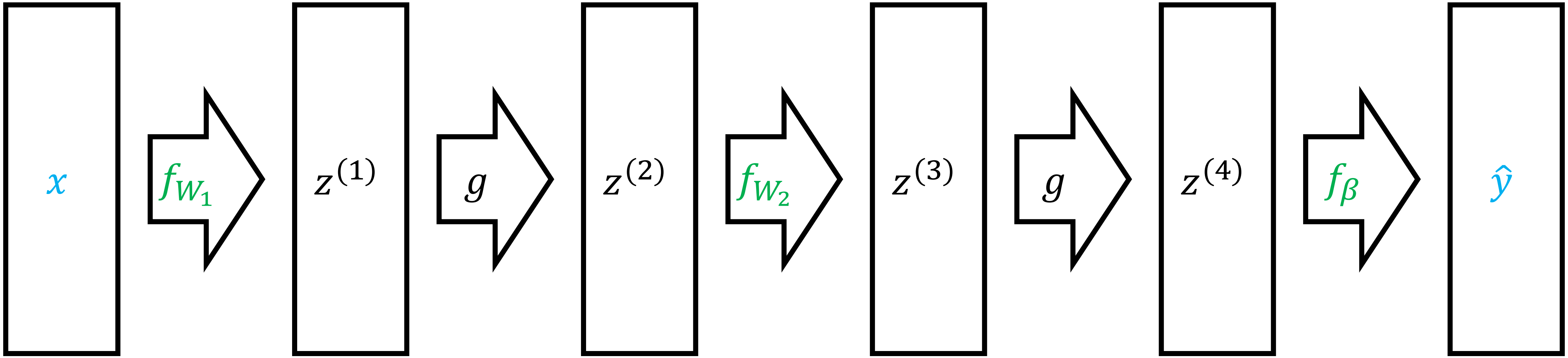
Modern View


hidden layer
nodes or "units" (i.e., components of a layer)
input layer
x
f_W
$z^{(1)}$
g
$z^{(2)}$
f_β
$\hat{y}$
parameters (sometimes called "weights")
output layer

Modern View

- **Neural network with two hidden linear layers:**

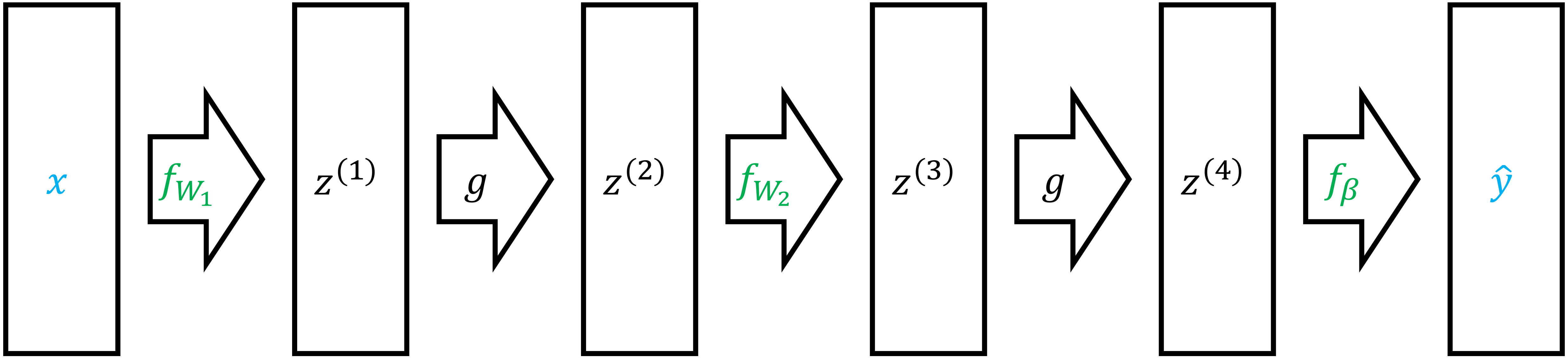
$$f_{W_1,W_2,\beta}(x) = f_\beta \circ g \circ f_{W_2} \circ g \circ f_{W_1}(x)$$



Modern View

- **Neural network with two hidden linear layers:**

$$f_{W_1,W_2,\beta}(x) = f_\beta\left(g\left(f_{W_2}\left(g\left(f_{W_1}(x)\right)\right)\right)\right)$$



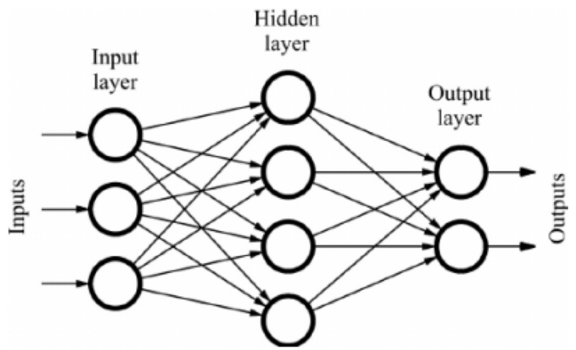
Learn successively more “high-level” representations

Neural Networks

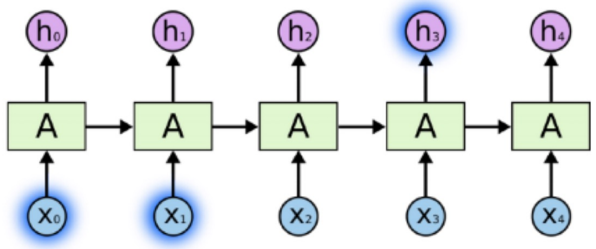
- **Pros**
  - **“Meta” strategy:** Enables users to **design** model family
  - Design model families that capture **symmetries/structure** in the data (e.g., read a sentence forwards, translation invariance for images, etc.)

Common Layers

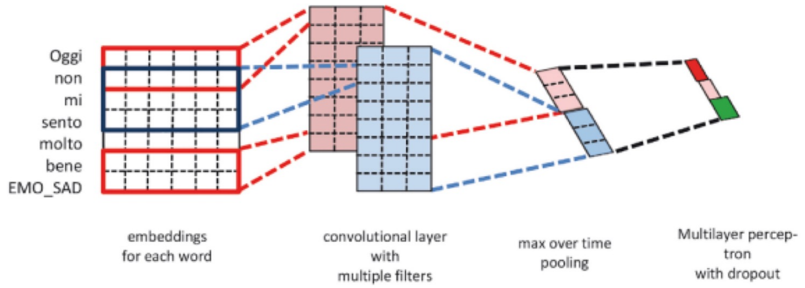
Feed-forward NNs



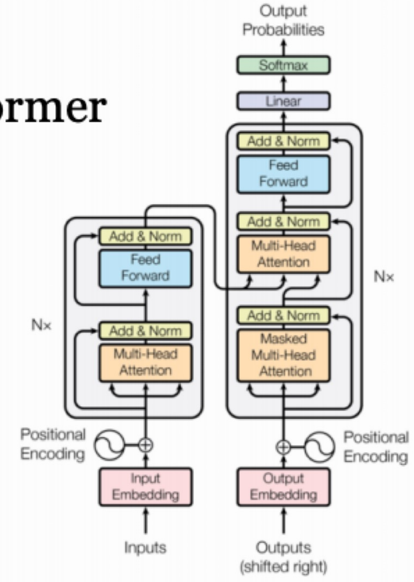
Recurrent NNs



Convolutional NNs



Transformer



Always coupled with word embeddings…

Neural Networks

- **Pros**
  - **“Meta” strategy:** Enables users to **design** model family
  - Design model families that capture **symmetries/structure** in the data (e.g., read a sentence forwards, translation invariance for images, etc.)
  - “Representation learning” (automatically learn features for certain domains)
  - More parameters!

- **Cons**
  - Very hard to train! (Non-convex loss functions)
  - Lots of parameters → need lots of data!
  - Lots of design decisions

Agenda

- **Model family**
  - Custom model family rather than a single model family
- **Optimization**
  - Backpropagation algorithm for computing gradient

Optimization Algorithm

- Based on gradient descent, with a few tweaks
  - **Note:** Loss is nonconvex, but gradient descent works well in practice

- **Key challenge:** How to compute the gradient?
  - **Strategy so far:** Work out gradient for every model family
  - **New strategy:** Algorithm for computing gradient of an arbitrary programmatic composition of layers
  - This algorithm is called **backpropagation**

Gradient Descent

- $W_1 \leftarrow \text{Initialize}(\,)$
- **for** $t \in \{1,2,\dots\}$ **until** convergence:

$$W_{t+1,j} \leftarrow W_{t,j} - \frac{\alpha}{n} \cdot \sum_{i=1}^{n} \nabla_{W_j} L\big(f_{W_t}(x_i), y_i\big) \quad \text{(for each } j\text{)}$$

- **return** f_{W_t}

Backpropagation

- **Input**
  - Example-label pair (x, y)
  - Arbitrary model $f_{W_m} \circ \cdots \circ f_{W_1}$
  - Loss $L(\hat{y}, y)$ for predicted label $\hat{y}$ and true label y
  - Derivative $\nabla_{\hat{y}} L(\hat{y}, y)$ (as a function)
  - Derivatives $D_{W_j} f_{W_j}(z)$ and $D_z f_{W_j}(z)$ (e.g., as a function)

- **Output:** $\nabla_{W_j} L(f_W(x), y)$

Recall: Multi-Dimensional Derivatives

- **Given:**
  - Function $f_W(z)$ mapping parameters $W \in \mathbb{R}^d$ and input vector $z \in \mathbb{R}^k$ to a vector $f_W(z) \in \mathbb{R}^h$
  - Current parameters W and z

- The **derivative** of f_W at W and z with respect to z is a matrix

$$D_z f_W(z) \in \mathbb{R}^{h \times k}$$

Recall: Multi-Dimensional Derivatives

- **Given:**
  - Function $f_W(z)$ mapping parameters $W \in \mathbb{R}^d$ and input vector $z \in \mathbb{R}^k$ to a vector $f_W(z) \in \mathbb{R}^h$
  - Current parameters W and z

- The **derivative** of f_W at W and z with respect to W is a matrix

$$D_W f_W(z) \in \mathbb{R}^{h \times d}$$

Recall: Multi-Dimensional Derivatives

- **Given:**
  - Function $f_W(z)$ mapping parameters $W \in \mathbb{R}^d$ and input vector $z \in \mathbb{R}^k$ to a vector $f_W(z) \in \mathbb{R}^h$
  - Current parameters W and z

- **Intuition:** The linear function that best approximates f_W at W and z:

$$f_{W+dW}(z + dz) \approx f_W(z) + D_z f_W(z) dz + D_W f_W(z) dW$$

Backpropagation Example

- **Gradient of MSE loss (for regression):**

$$\nabla_W L(W, \beta; Z) = \nabla_W \frac{1}{n} \sum_{i=1}^{n} \left(f_{W,\beta}(x_i) - y_i\right)^2$$
$$= \frac{2}{n} \sum_{i=1}^{n} \left(f_{W,\beta}(x_i) - y_i\right) D_W f_{W,\beta}(x_i)$$

$$\nabla_\beta L(W, \beta; Z) = \nabla_\beta \frac{1}{n} \sum_{i=1}^{n} \left(f_{W,\beta}(x_i) - y_i\right)^2$$
$$= \frac{2}{n} \sum_{i=1}^{n} \left(f_{W,\beta}(x_i) - y_i\right) D_\beta f_{W,\beta}(x_i)$$

Backpropagation Example

- **Derivative of neural network:**

$$D_\beta f_{W,\beta}(x) = D_\beta\big(f_\beta \circ g \circ f_W\big)(x)$$
$$= D_\beta f_\beta\big(g \circ f_W(x)\big)$$

$$D_W f_{W,\beta}(x) = D_W\big(f_\beta \circ g \circ f_W\big)(x)$$
$$= D_z f_\beta\big(g \circ f_W(x)\big) D_W(g \circ f_W)(x)$$
$$= D_z f_\beta\big(g \circ f_W(x)\big) D_z g\big(f_W(x)\big) D_W f_W(x)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_m} f_W(x)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_m} f_W(x) = D_{W_m} f_{W_m}\left(z^{(m-1)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_m} f_W(x) = D_{W_m} f_{W_m}\left(z^{(m-1)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-1}} f_W(x)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-1}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_{W_{m-1}} f_{W_{m-1}}\left(z^{(m-2)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-1}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_{W_{m-1}} f_{W_{m-1}}\left(z^{(m-2)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-1}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_{W_{m-1}} f_{W_{m-1}}\left(z^{(m-2)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-2}} f_W(x)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-2}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_z f_{W_{m-1}}\left(z^{(m-2)}\right) D_{W_{m-2}} f_{W_{m-2}}\left(z^{(m-3)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-2}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_z f_{W_{m-1}}\left(z^{(m-2)}\right) D_{W_{m-2}} f_{W_{m-2}}\left(z^{(m-3)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-2}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_z f_{W_{m-1}}\left(z^{(m-2)}\right) D_{W_{m-2}} f_{W_{m-2}}\left(z^{(m-3)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_{m-2}} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) D_z f_{W_{m-1}}\left(z^{(m-2)}\right) D_{W_{m-2}} f_{W_{m-2}}\left(z^{(m-3)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_j} f_W(x)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_j} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) \dots D_z f_{W_{j+1}}\left(z^{(j)}\right) D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_j} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) \ldots D_z f_{W_{j+1}}\left(z^{(j)}\right) D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_j} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) \ldots D_z f_{W_{j+1}}\left(z^{(j)}\right) D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation

- **General case:** Consider a neural network

$$f_W(x) = f_{W_m} \circ f_{W_{m-1}} \circ \cdots \circ f_{W_1}(x)$$

$$z^{(j)} = f_{W_j} \circ \cdots \circ f_{W_1}(x) = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$$

- We have

$$D_{W_j} f_W(x) = D_z f_{W_m}\left(z^{(m-1)}\right) \dots D_z f_{W_{j+1}}\left(z^{(j)}\right) D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation

- We have

$$D_{W_j} f_W(x) = \underbrace{D_z f_{W_m}\left(z^{(m-1)}\right) \dots D_z f_{W_{j+1}}\left(z^{(j)}\right)}_{\substack{\text{Portions shared across terms} \\ \text{Denote it by } D^{(j)}}} D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation Algorithm

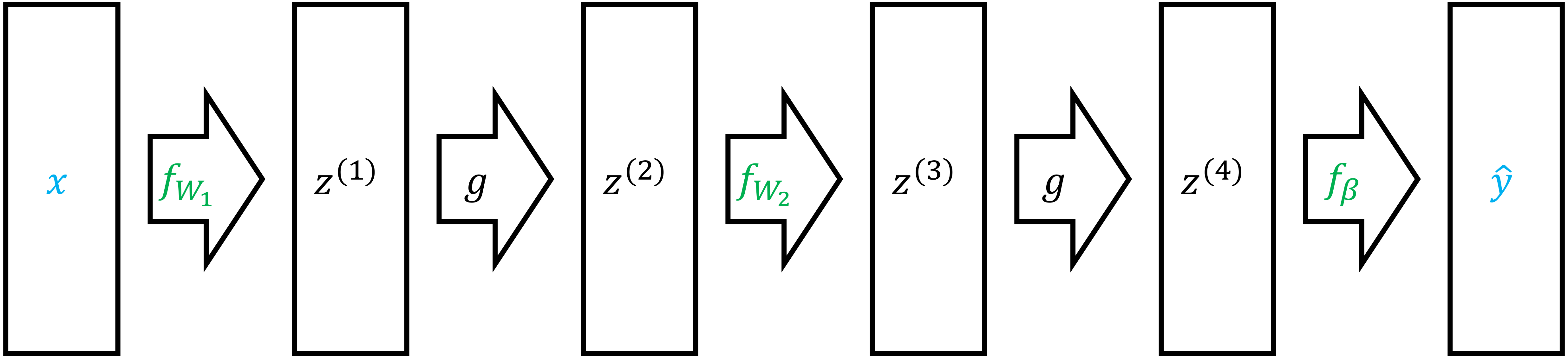
- Compute recursively starting from $j = m$ to $j = 1$:

$$D^{(j)} = D_z f_{W_m}\left(z^{(m-1)}\right) \dots D_z f_{W_{j+1}}\left(z^{(j)}\right)$$

$$= \begin{cases} 1 & \text{if } j = m \\ D^{(j+1)} D_z f_{W_{j+1}}\left(z^{(j)}\right) & \text{if } j < m \end{cases}$$

$$D_{W_j} f_W(x) = D^{(j)} D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$$

Backpropagation

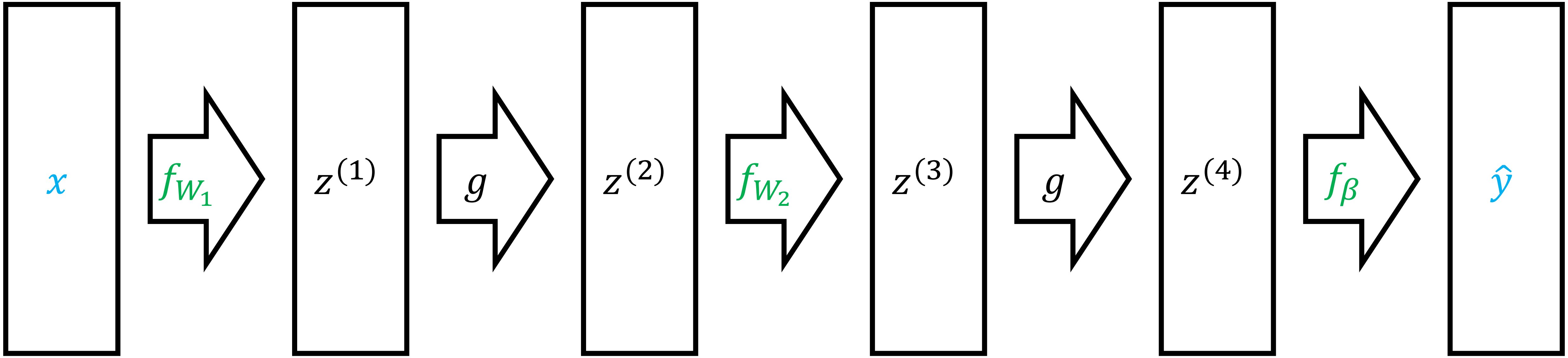


Forward pass: Compute $z^{(j)} = f_{W_j}\left(z^{(j-1)}\right)$

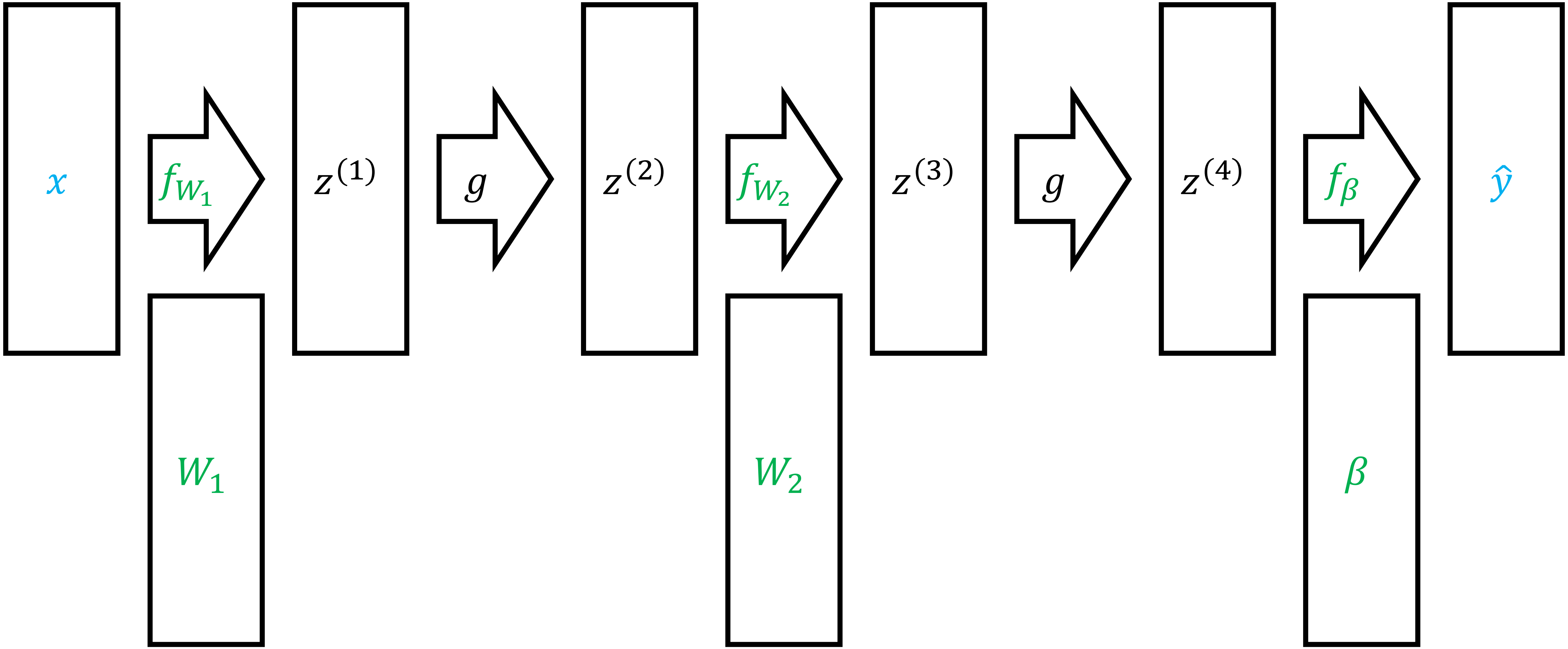
Backward pass: Compute $D^{(j)} = D^{(j+1)} D_z f_{W_{j+1}}\left(z^{(j)}\right)$ and $D_{W_j} f_W(x) = D^{(j)} D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$

Final output: $\nabla_{\hat{y}} L\left(z^{(m)}, y\right)^\top D_{W_j} f_W(x)$

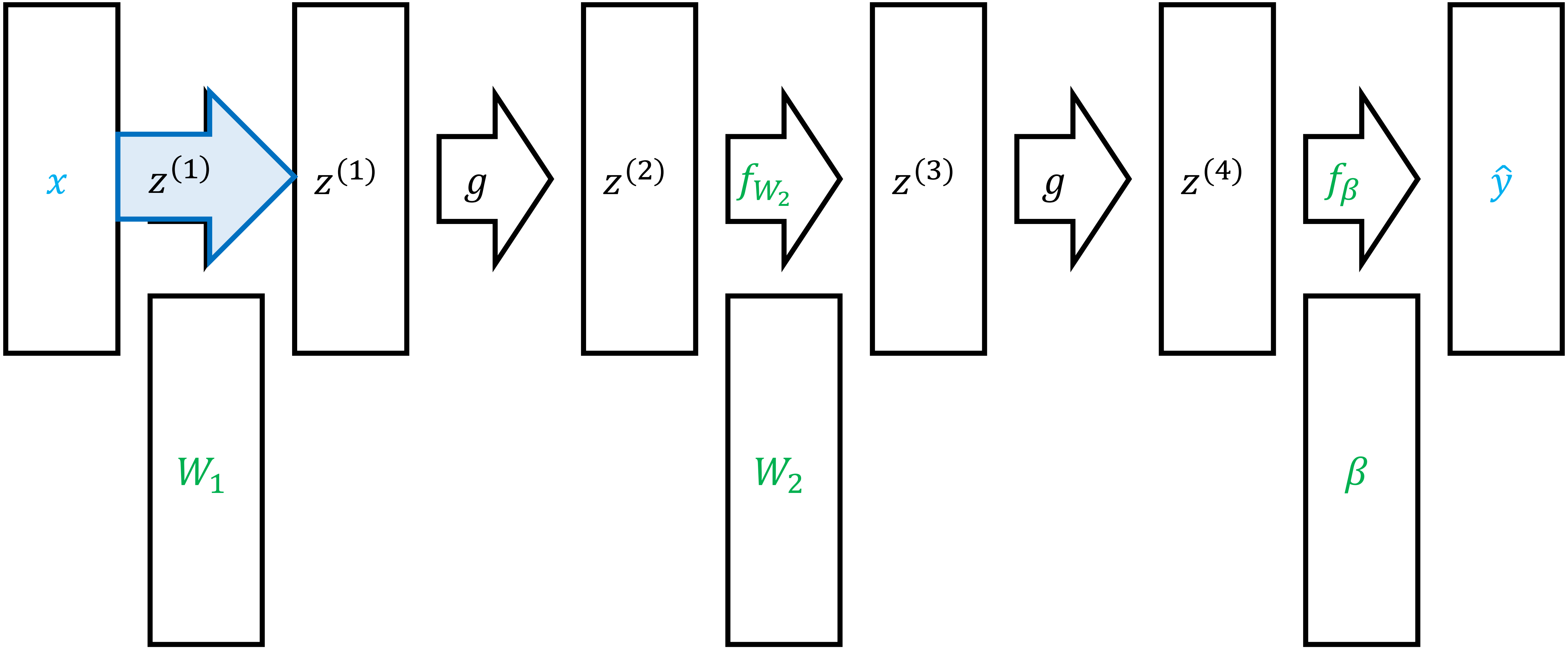
Backpropagation



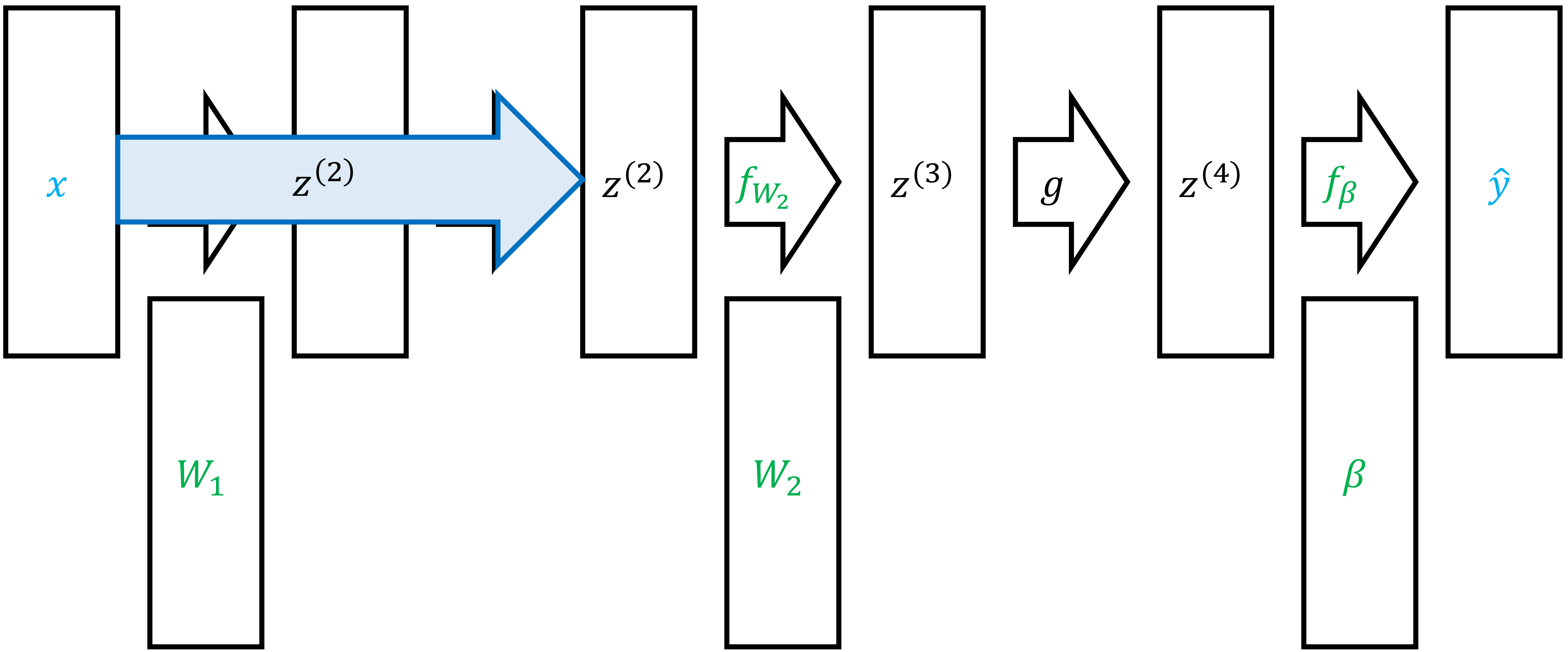
Backpropagation



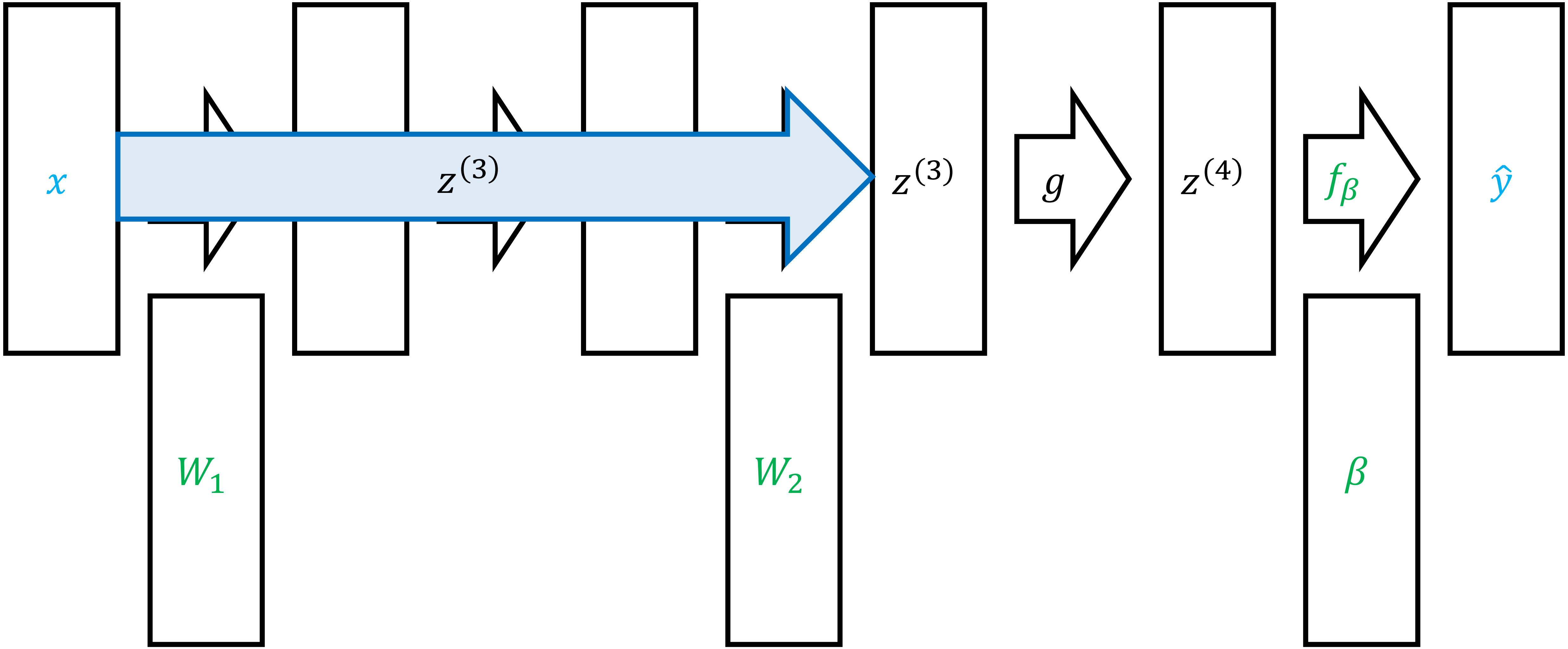
Backpropagation



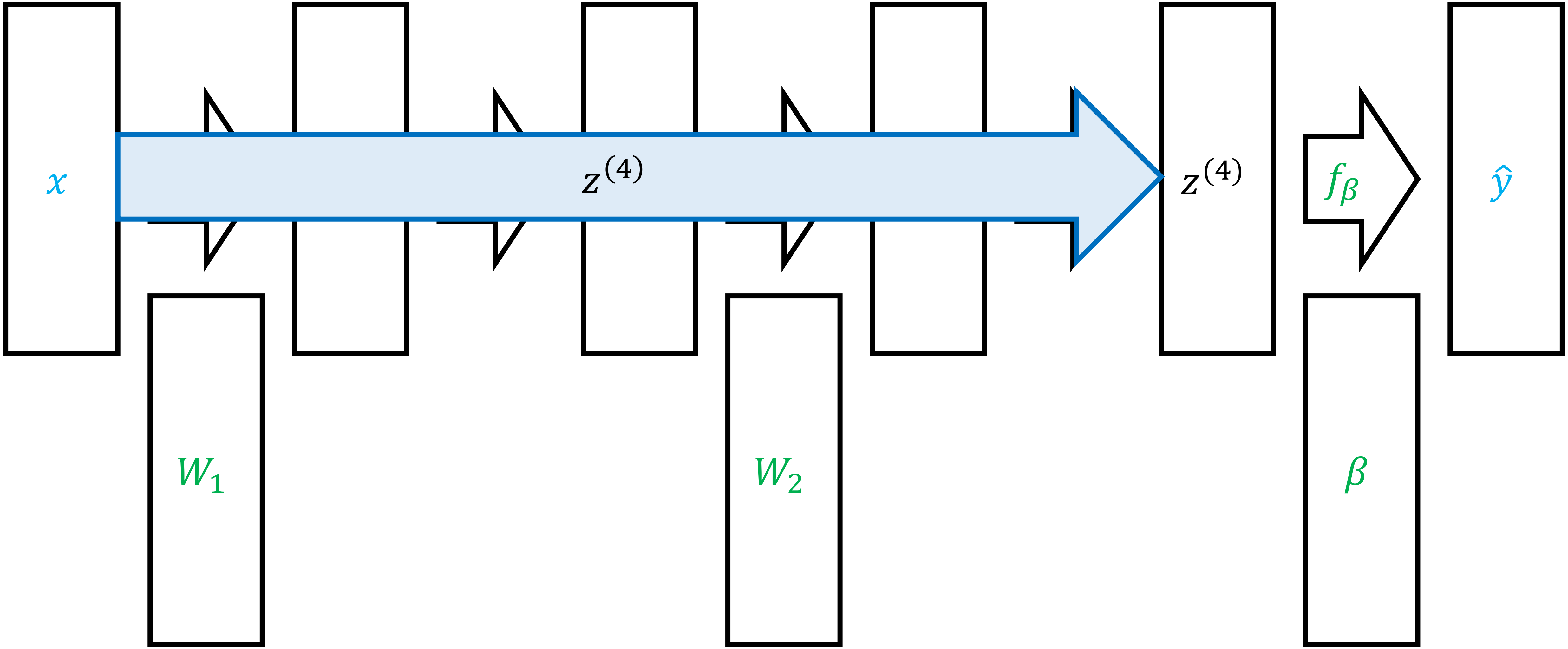
Backpropagation



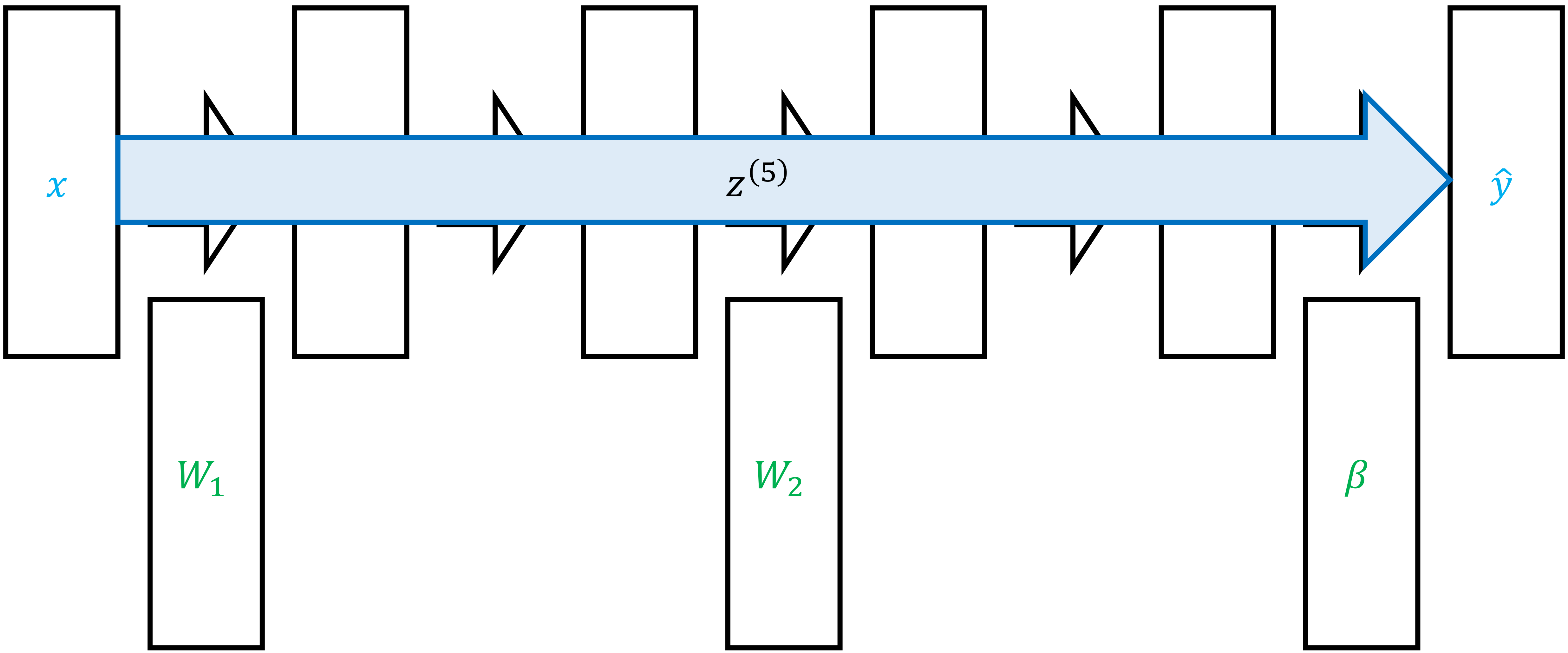
Backpropagation



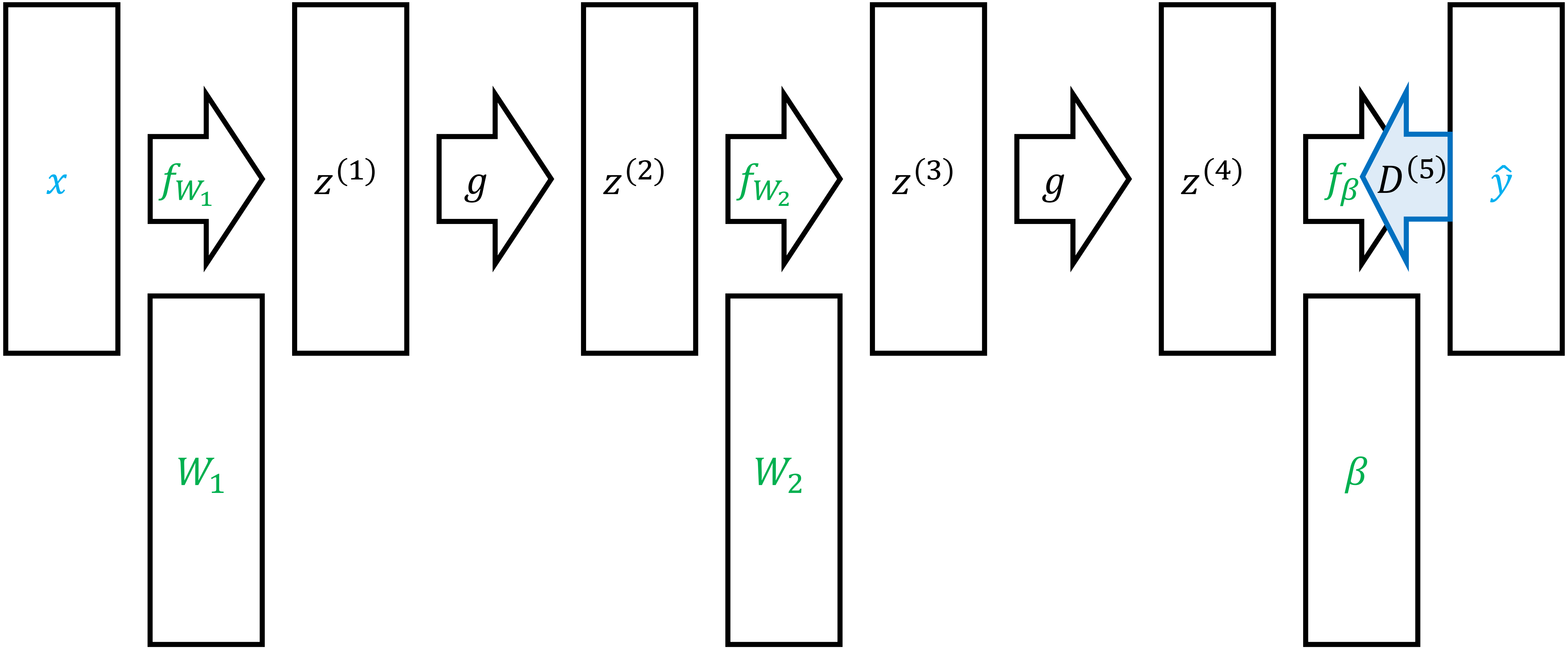
Backpropagation



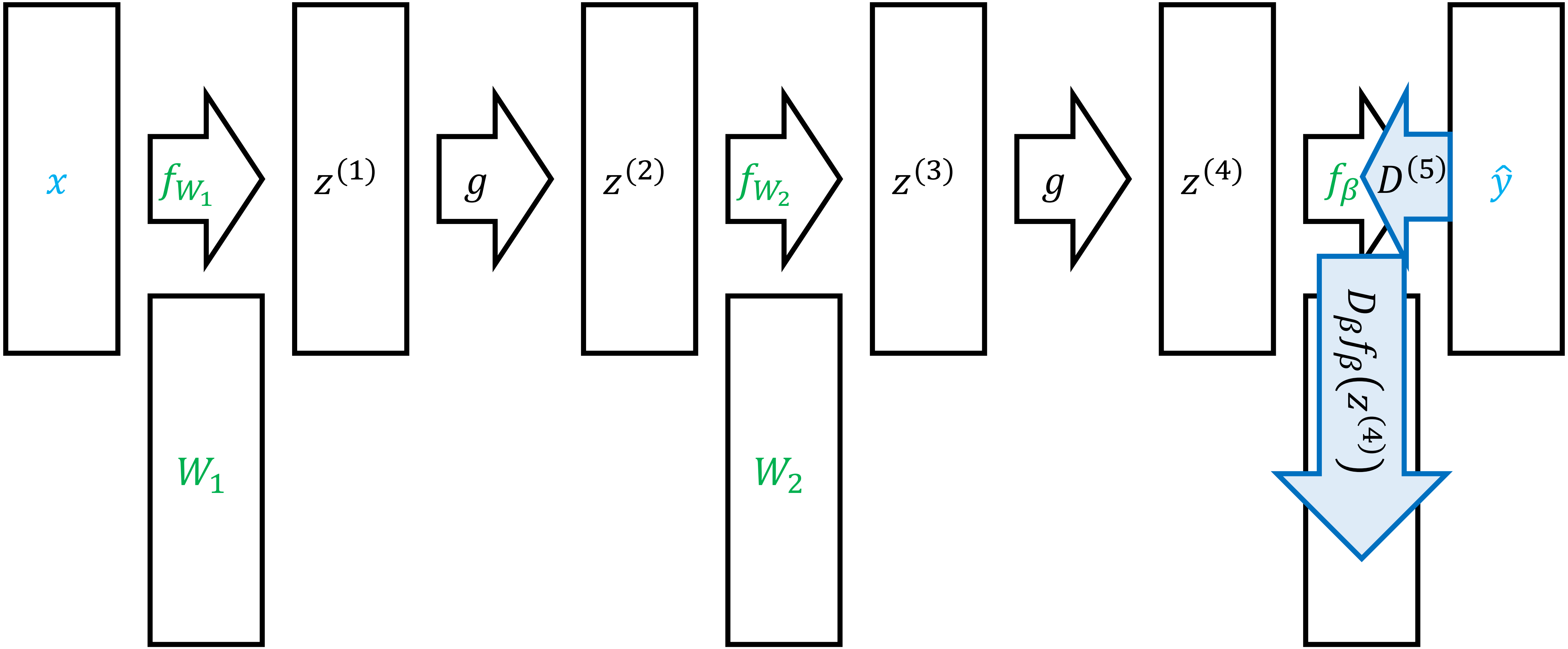
Backpropagation



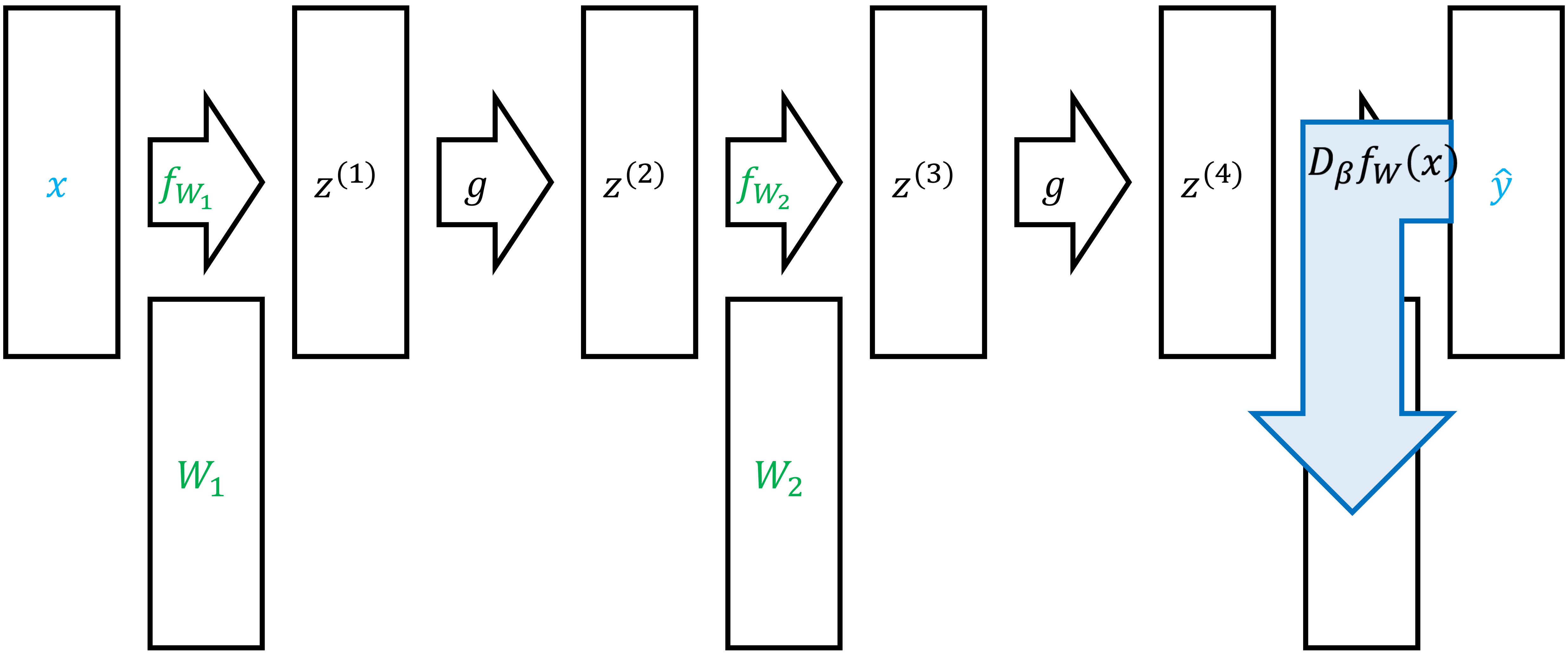
Backpropagation



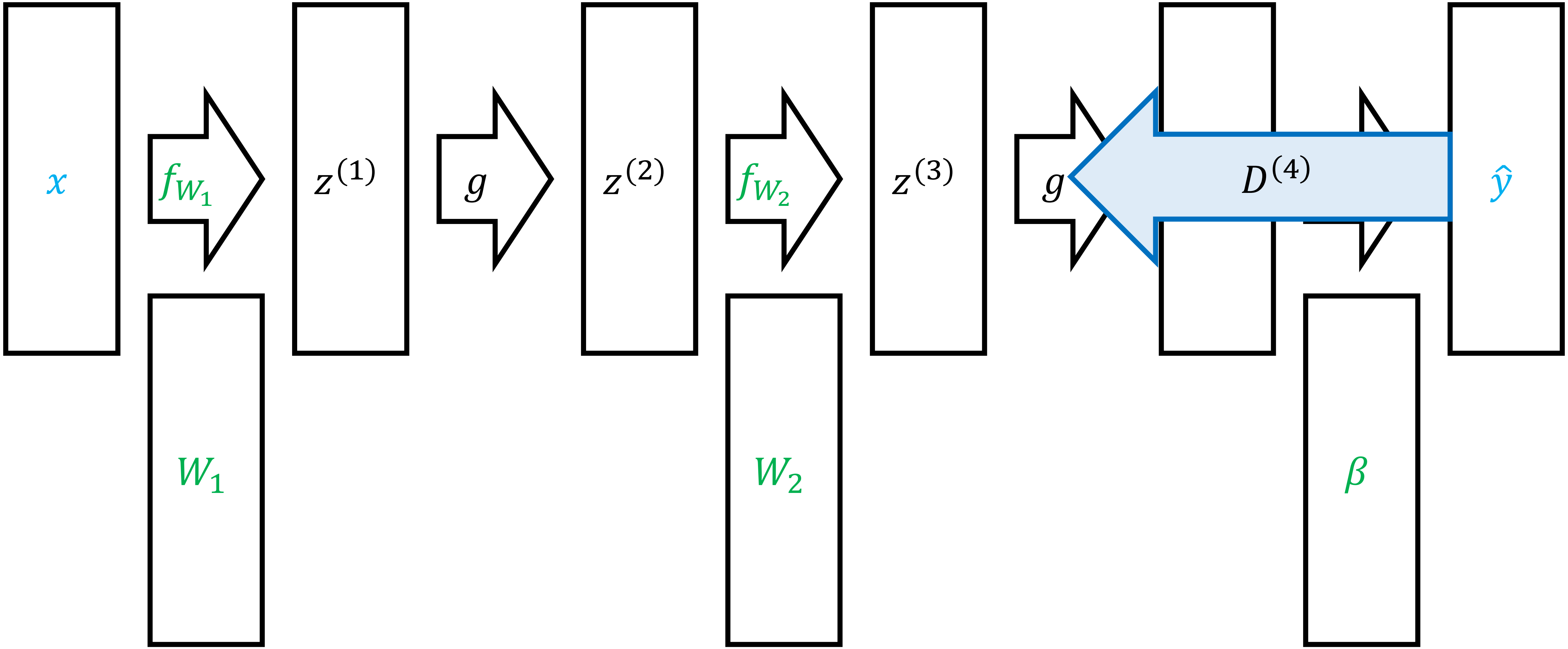
Backpropagation



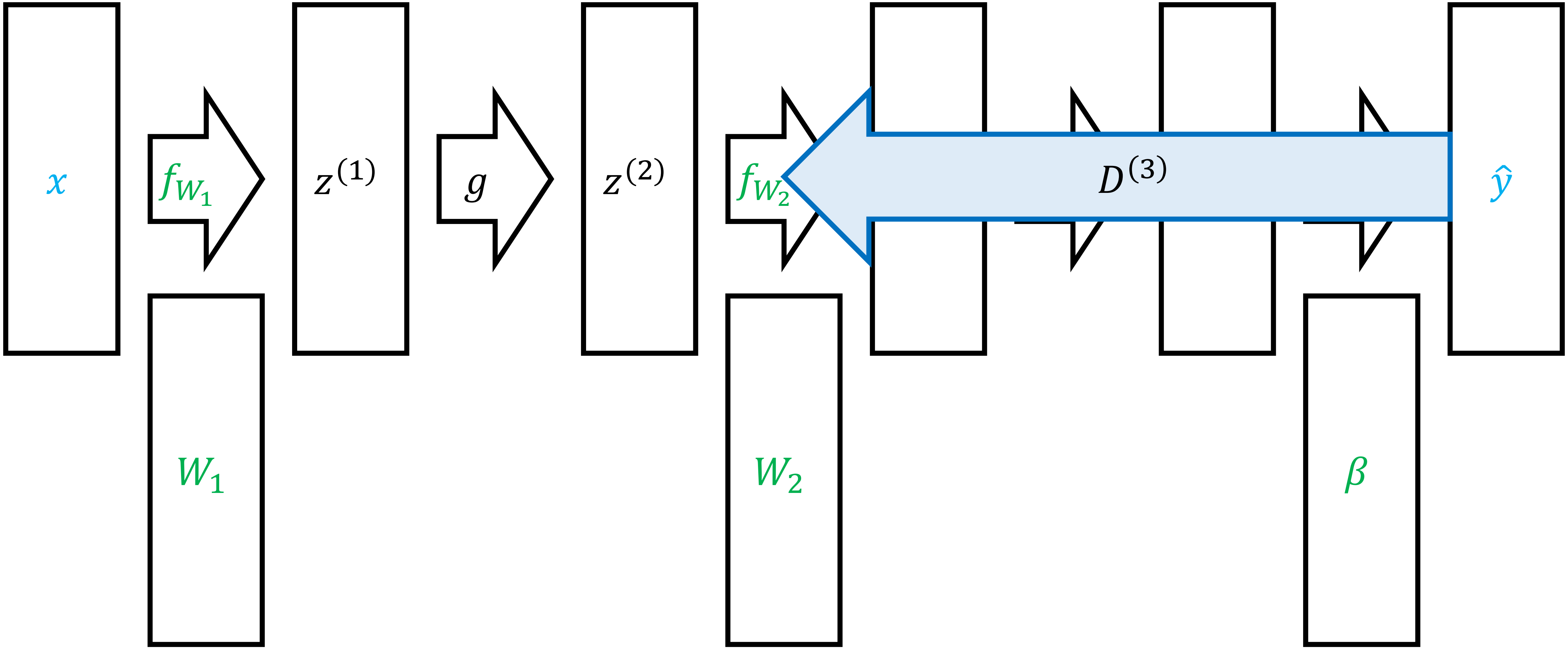
Backpropagation



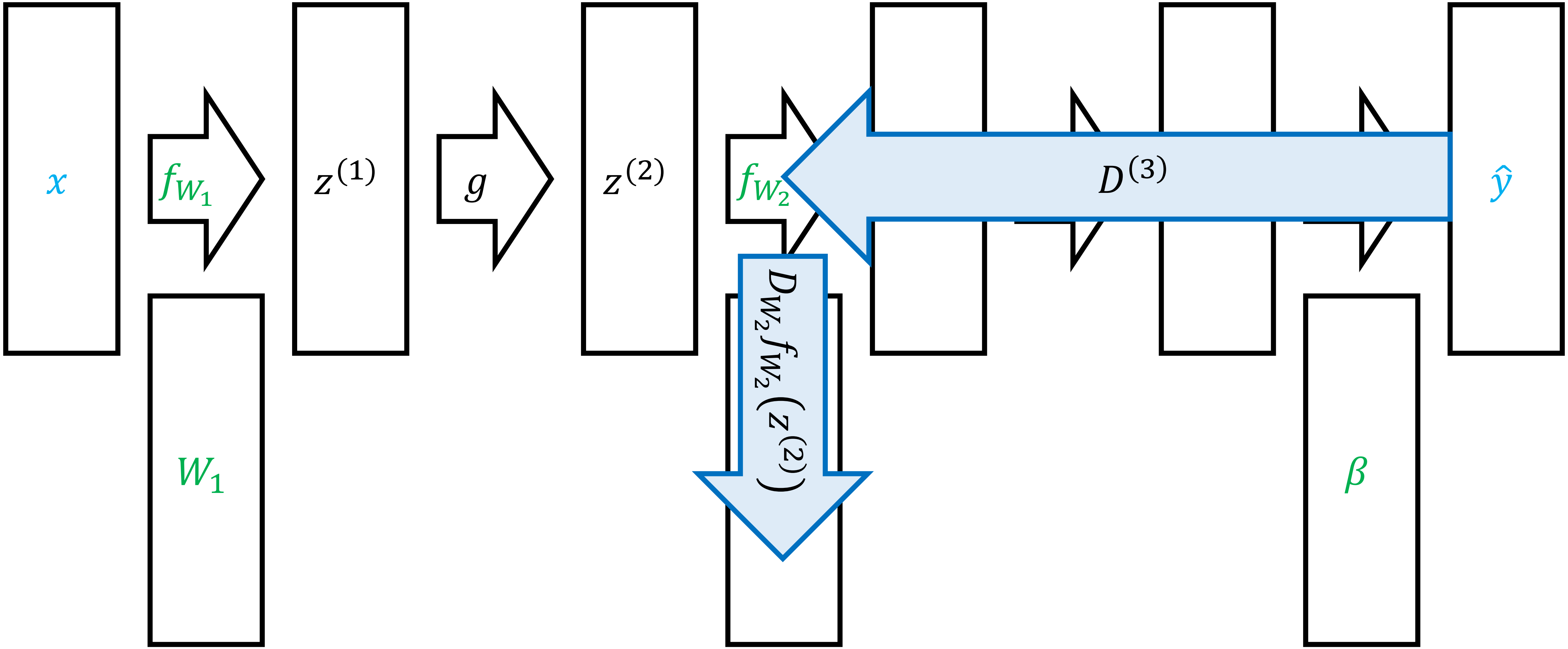
Backpropagation



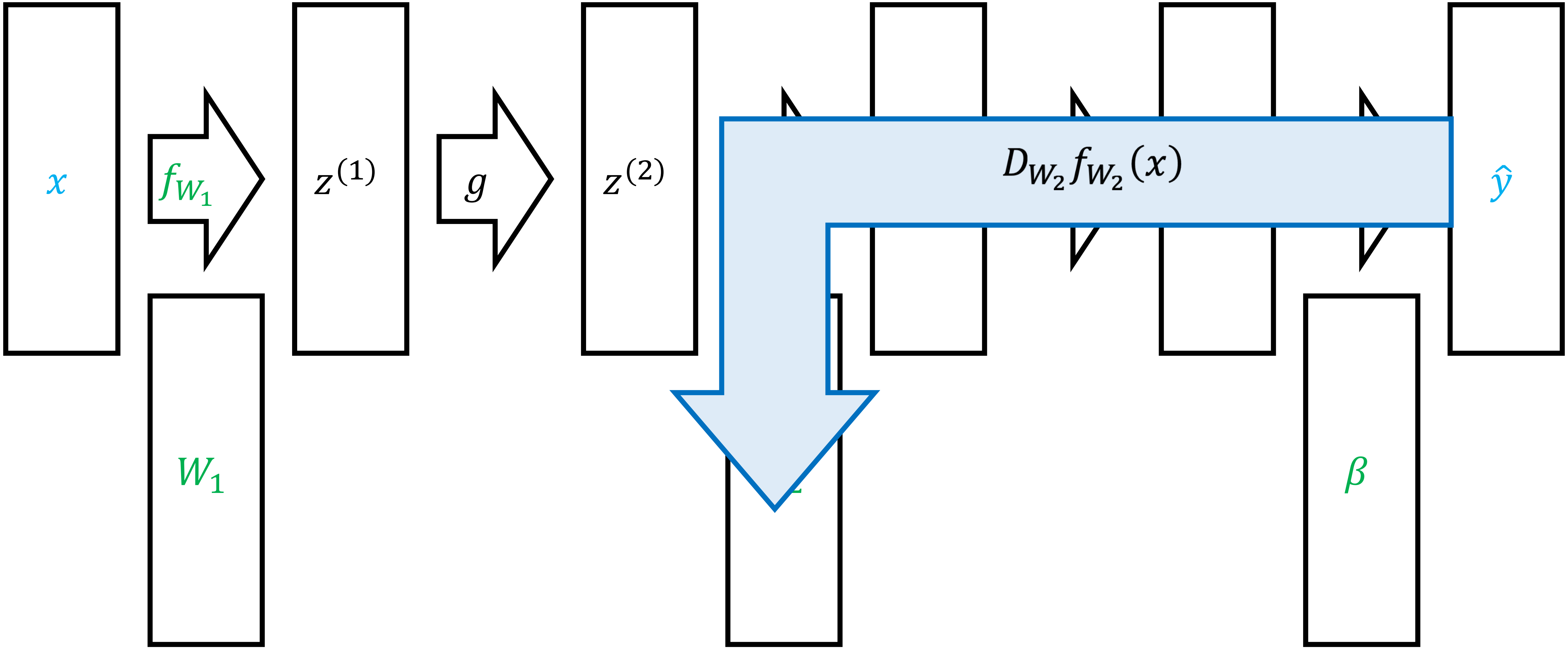
Backpropagation



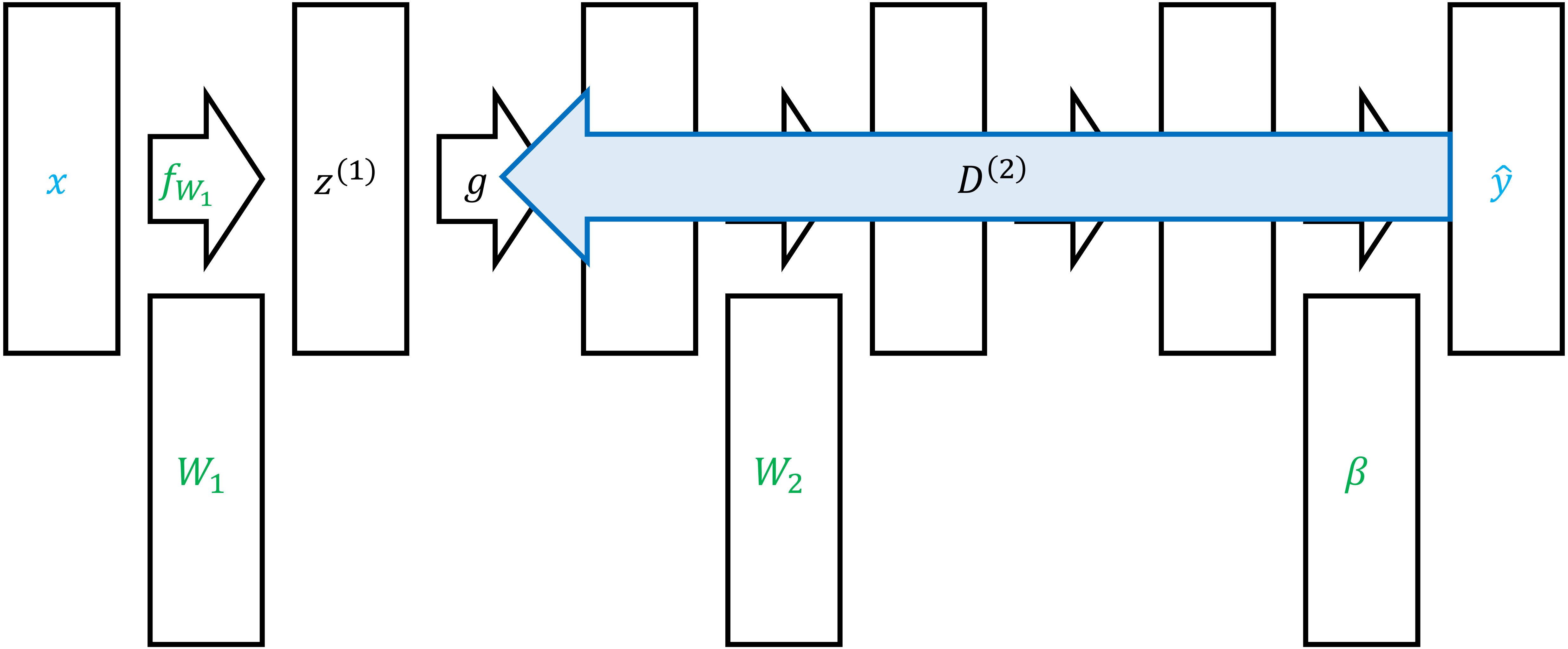
Backpropagation



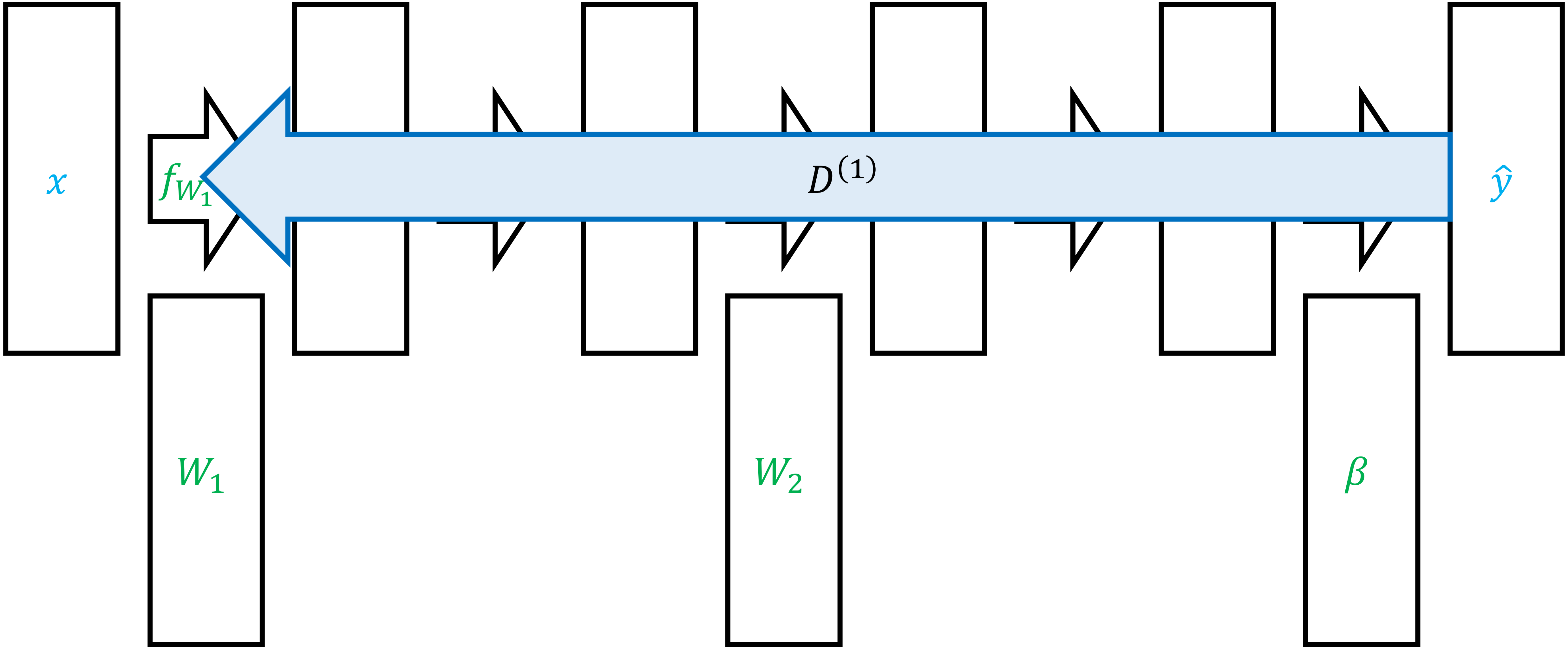
Backpropagation



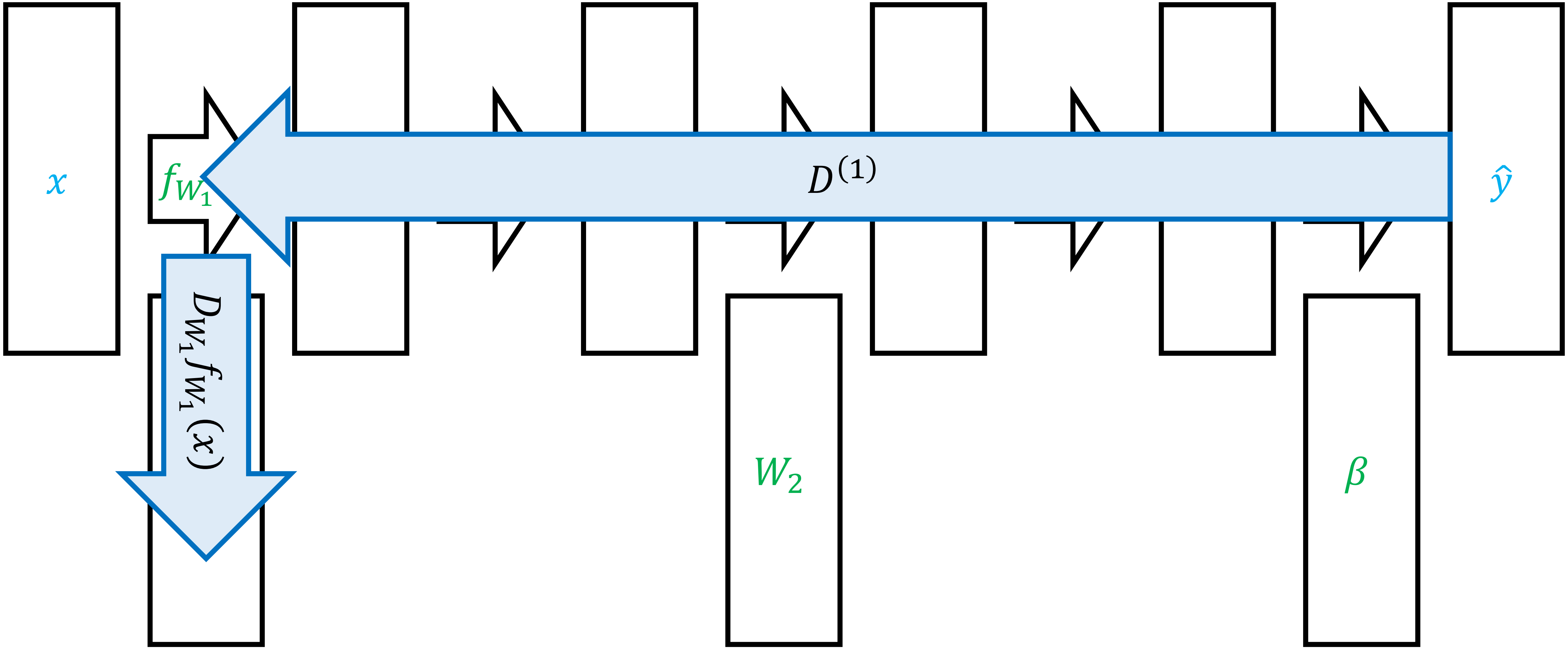
Backpropagation



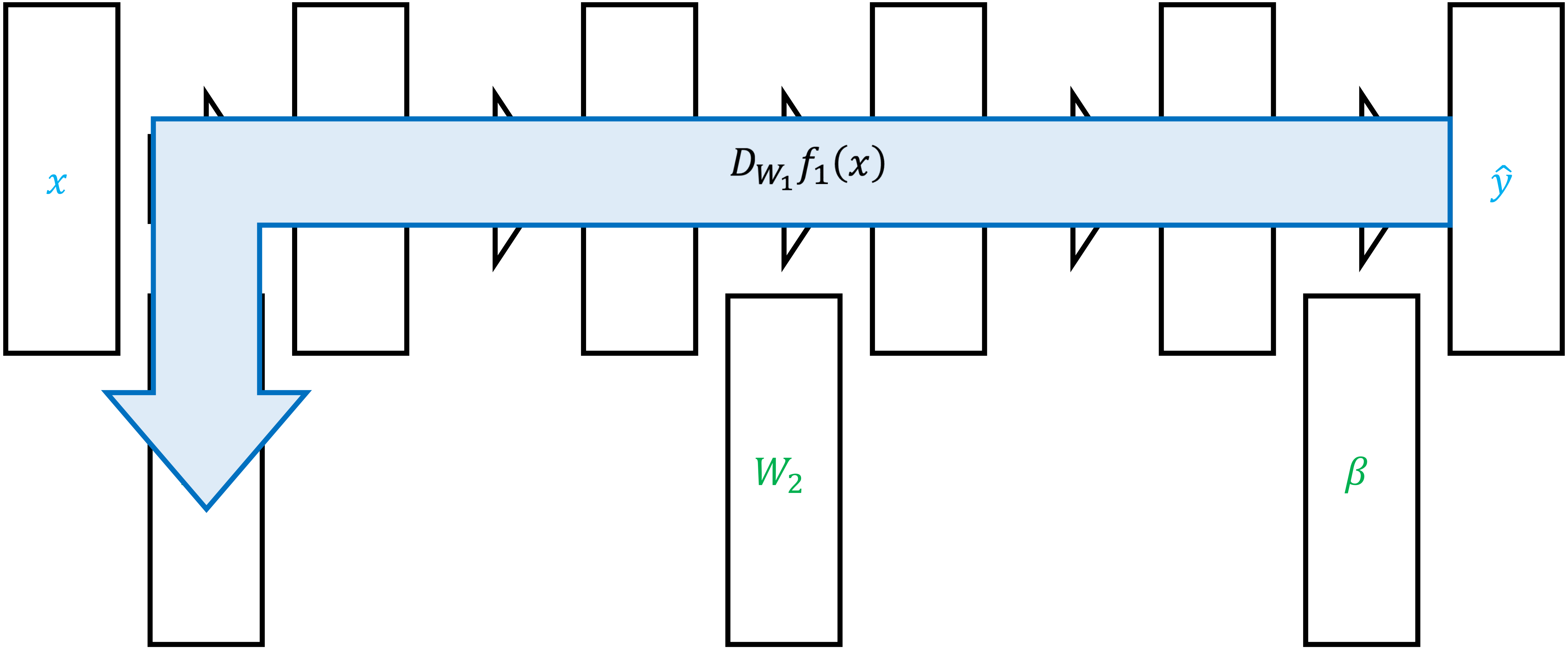
Backpropagation



Backpropagation



Backpropagation



Backpropagation Algorithm

- **Forward pass:** Compute forwards from $j = 0$ to $j = m$
  - $z^{(j)} = \begin{cases} x & \text{if } j = 0 \\ f_{W_j}\left(z^{(j-1)}\right) & \text{if } j > 0 \end{cases}$
- **Backward pass:** Compute backwards from $j = m$ to $j = 1$
  - $D^{(j)} = \begin{cases} 1 & \text{if } j = m \\ D^{(j+1)} D_z f_{W_{j+1}}\left(z^{(j)}\right) & \text{if } j < m \end{cases}$
  - $D_{W_j} f_W(x) = D^{(j)} D_{W_j} f_{W_j}\left(z^{(j-1)}\right)$
- **Final output:** $\nabla_{W_j} L(f_W(x), y)^\top = \nabla_{\hat{y}} L\left(z^{(m)}, y\right)^\top D_{W_j} f_W(x)$ for each j

Gradient Descent

- $W_1 \leftarrow \text{Initialize}(\,)$
- **for** $t \in \{1,2,\dots\}$ **until** convergence:

$$W_{t+1,j} \leftarrow W_{t,j} - \frac{\alpha}{n} \cdot \sum_{i=1}^{n} \nabla_{W_j} L\big(f_{W_t}(x_i), y_i\big) \quad \text{(for each } j\text{)}$$

- **return** f_{W_t}

Gradient Descent

- $W_1 \leftarrow \text{Initialize}(\,)$
- **for** $t \in \{1, 2, \ldots\}$ **until** convergence:
  - Compute gradients $\nabla_{W_j} L\big(f_{W_t}(x_i), y_i\big)$ using backpropagation
  - Update parameters:

$$W_{t+1,j} \leftarrow W_{t,j} - \frac{\alpha}{n} \cdot \sum_{i=1}^{n} \nabla_{W_j} L\big(f_{W_t}(x_i), y_i\big) \quad \text{(for each } j\text{)}$$

- **return** f_{W_t}