Deep Learning Principles & Applications

Chapter 4 – Deep Neural Networks

Sudarsan N.S. Acharya (sudarsan.acharya@manipal.edu)

Topics

1. Deep L-layer neural network: architecture
2. Deep L-layer neural network: notation
3. Deep L-layer neural network forward propagation
4. Accounting for bias in a deep L-layer neural network
5. Minibatch forward propagation in a deep L-layer neural network
6. Deep L-layer neural network gradient calculation using backward propagation
7. Minibatch backward propagation in a deep L-layer neural network

Topics

1. Deep L-layer neural network: architecture
2. Deep L-layer neural network: notation
3. Deep L-layer neural network forward propagation
4. Accounting for bias in a deep L-layer neural network
5. Minibatch forward propagation in a deep L-layer neural network
6. Deep L-layer neural network gradient calculation using backward propagation
7. Minibatch backward propagation in a deep L-layer neural network

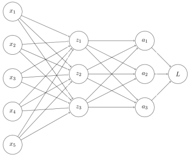
Deep L-layer neural network: architecture

Deep L-layer neural network: architecture

Zero hidden layer neural network

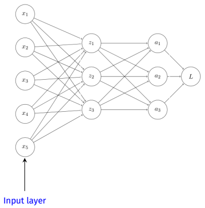
Deep L-layer neural network: architecture

Zero hidden layer neural network



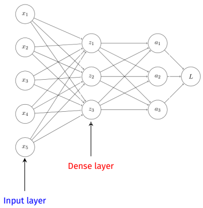
Deep L-layer neural network: architecture

Zero hidden layer neural network



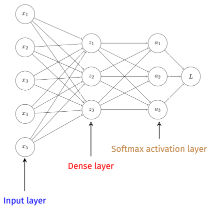
Deep L-layer neural network: architecture

Zero hidden layer neural network



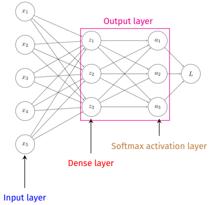
Deep L-layer neural network: architecture

Zero hidden layer neural network



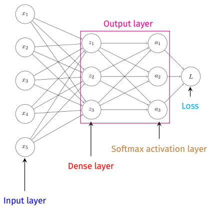
Deep L-layer neural network: architecture

Zero hidden layer neural network



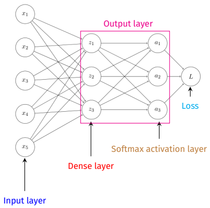
Deep L-layer neural network: architecture

Zero hidden layer neural network



Deep L-layer neural network: architecture

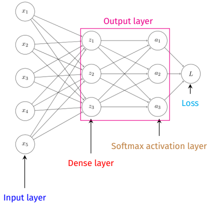
Zero hidden layer neural network



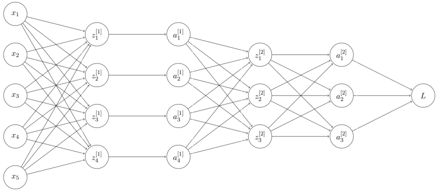
1 hidden layer neural network

Deep L-layer neural network: architecture

Zero hidden layer neural network

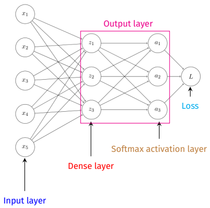


1 hidden layer neural network

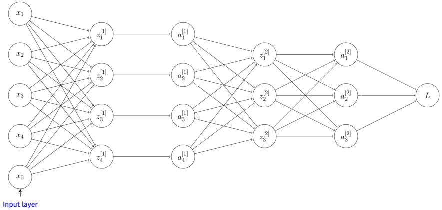


Deep L-layer neural network: architecture

Zero hidden layer neural network

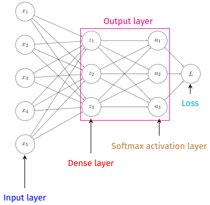


1 hidden layer neural network

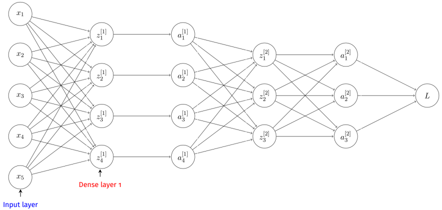


Deep L-layer neural network: architecture

Zero hidden layer neural network

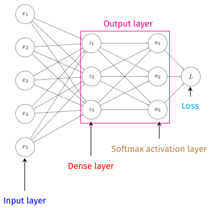


1 hidden layer neural network

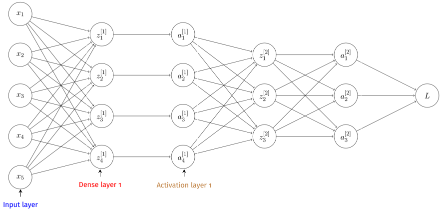


Deep L-layer neural network: architecture

Zero hidden layer neural network

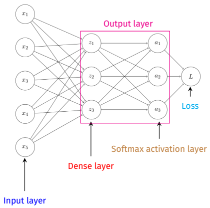


1 hidden layer neural network

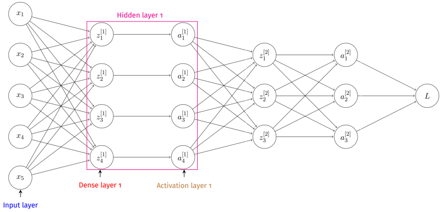


Deep L-layer neural network: architecture

Zero hidden layer neural network

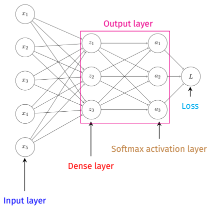


1 hidden layer neural network

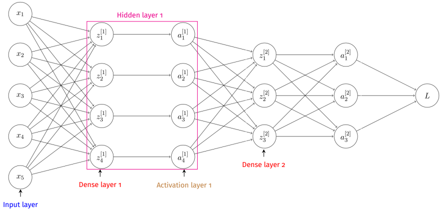


Deep L-layer neural network: architecture

Zero hidden layer neural network

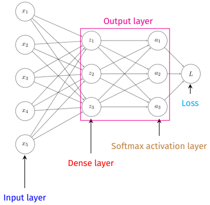


1 hidden layer neural network

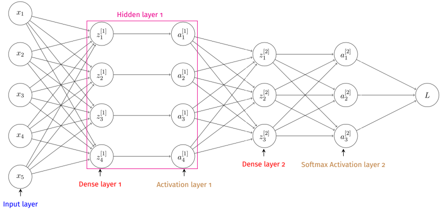


Deep L-layer neural network: architecture

Zero hidden layer neural network

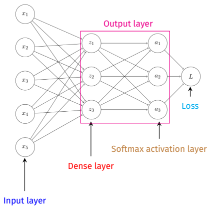


1 hidden layer neural network

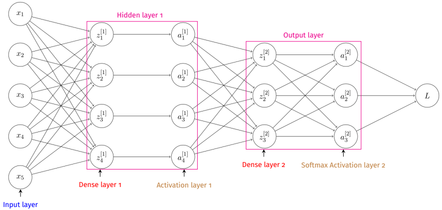


Deep L-layer neural network: architecture

Zero hidden layer neural network

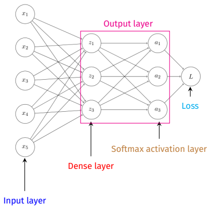


1 hidden layer neural network

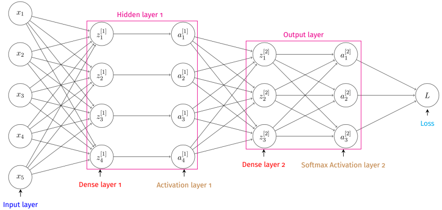


Deep L-layer neural network: architecture

Zero hidden layer neural network

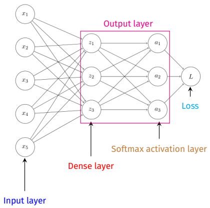


1 hidden layer neural network

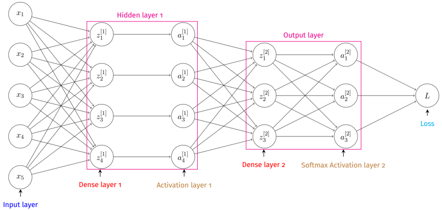


Deep L-layer neural network: architecture

Zero hidden layer neural network



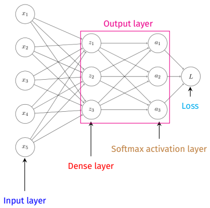
1 hidden layer neural network



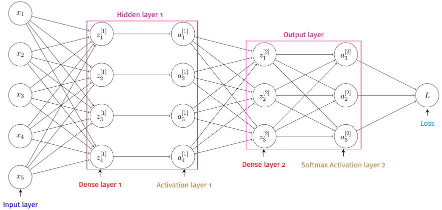
2 hidden layer neural network

Deep L-layer neural network: architecture

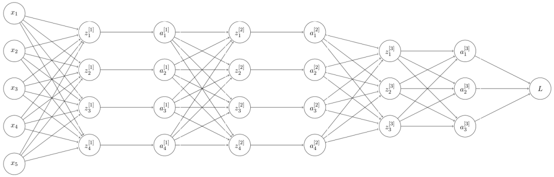
Zero hidden layer neural network



1 hidden layer neural network

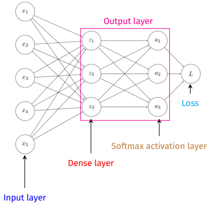


2 hidden layer neural network

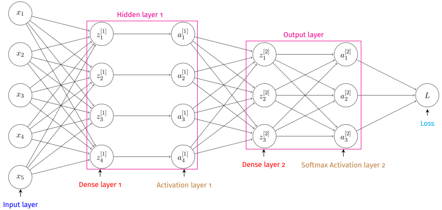


Deep L-layer neural network: architecture

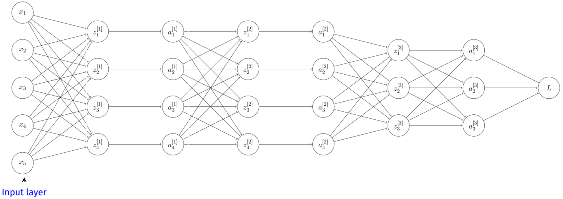
Zero hidden layer neural network



1 hidden layer neural network

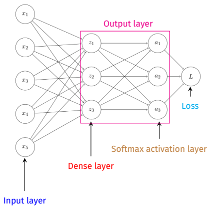


2 hidden layer neural network

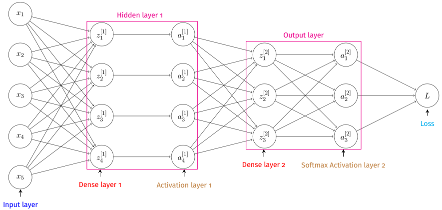


Deep L-layer neural network: architecture

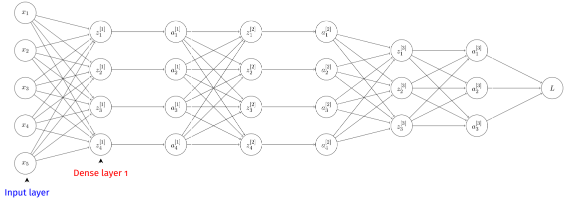
Zero hidden layer neural network



1 hidden layer neural network

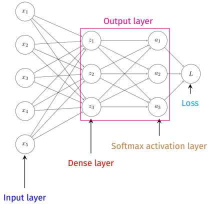


2 hidden layer neural network

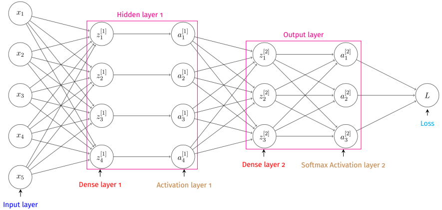


Deep L-layer neural network: architecture

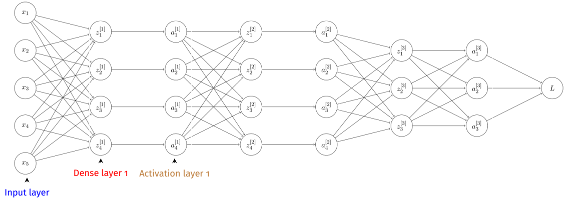
Zero hidden layer neural network



1 hidden layer neural network

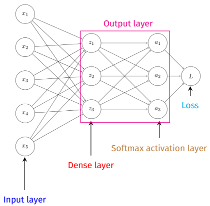


2 hidden layer neural network

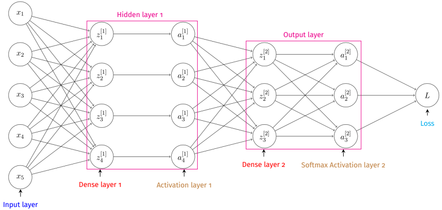


Deep L-layer neural network: architecture

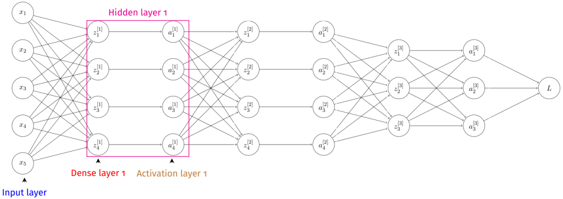
Zero hidden layer neural network



1 hidden layer neural network

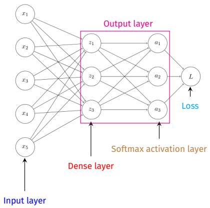


2 hidden layer neural network

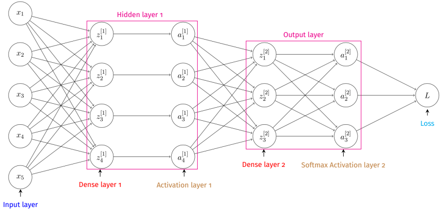


Deep L-layer neural network: architecture

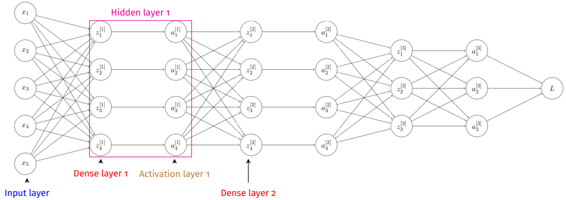
Zero hidden layer neural network



1 hidden layer neural network

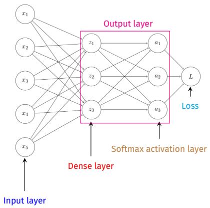


2 hidden layer neural network

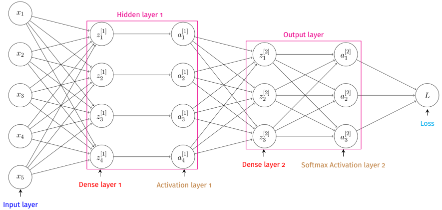


Deep L-layer neural network: architecture

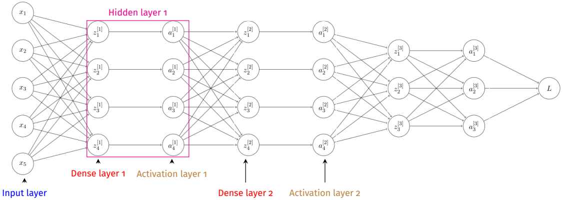
Zero hidden layer neural network



1 hidden layer neural network

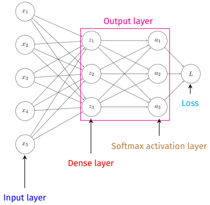


2 hidden layer neural network

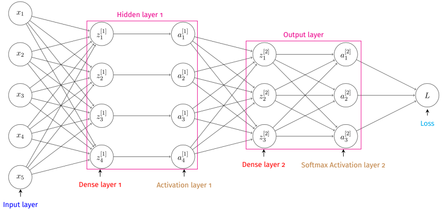


Deep L-layer neural network: architecture

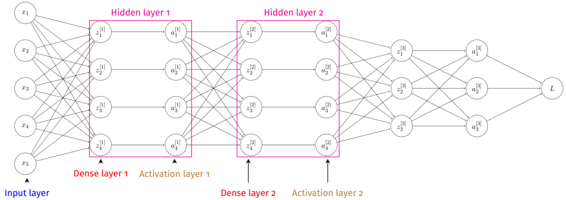
Zero hidden layer neural network



1 hidden layer neural network

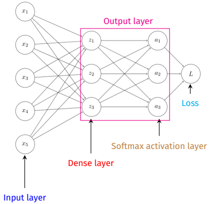


2 hidden layer neural network

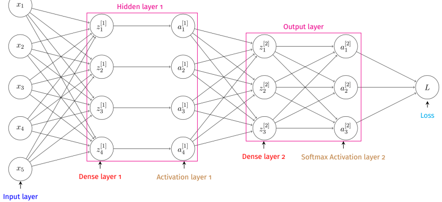


Deep L-layer neural network: architecture

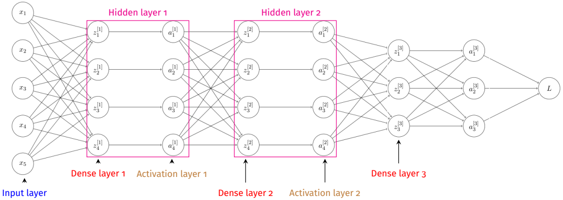
Zero hidden layer neural network



1 hidden layer neural network

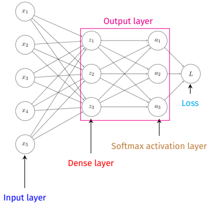


2 hidden layer neural network

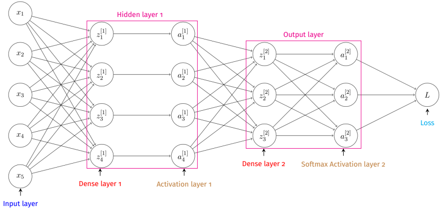


Deep L-layer neural network: architecture

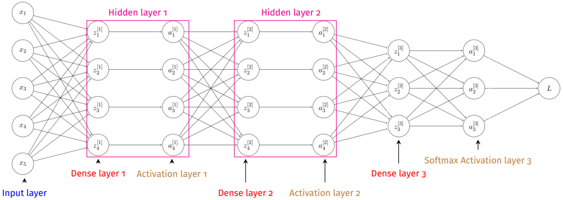
Zero hidden layer neural network



1 hidden layer neural network

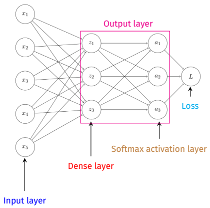


2 hidden layer neural network

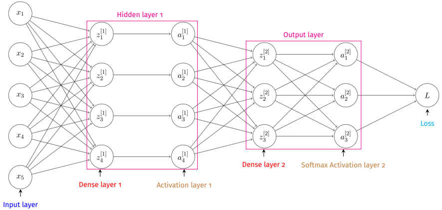


Deep L-layer neural network: architecture

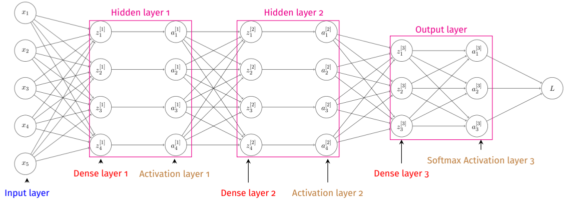
Zero hidden layer neural network



1 hidden layer neural network



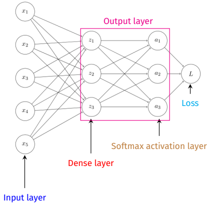
2 hidden layer neural network



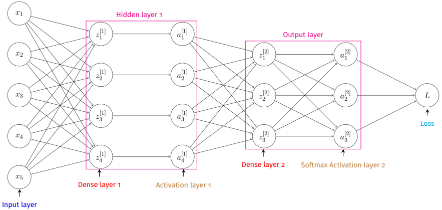
Deep L-layer neural network: architecture

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

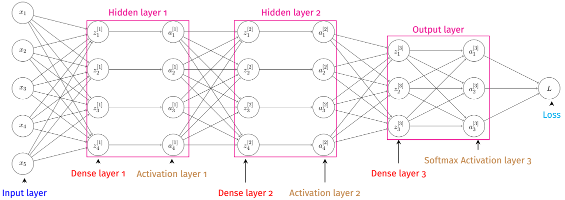
Zero hidden layer neural network



1 hidden layer neural network

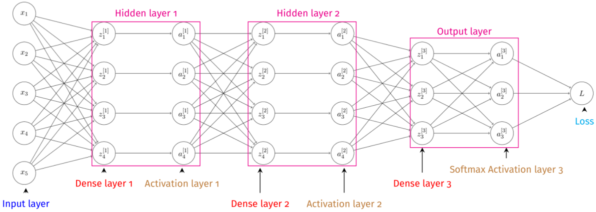


2 hidden layer neural network

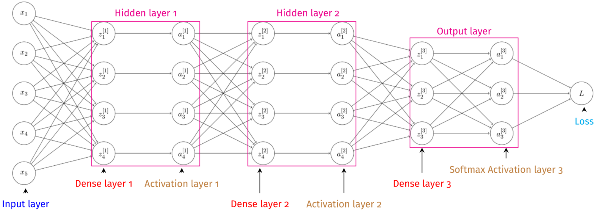


Deep L-layer neural network: notation and building blocks

Deep L-layer neural network: notation and building blocks

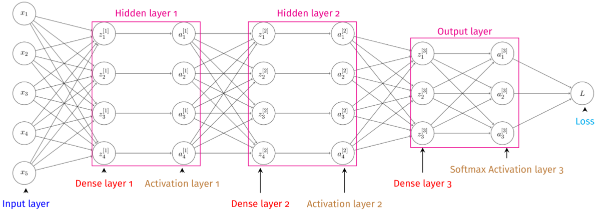


Deep L-layer neural network: notation and building blocks



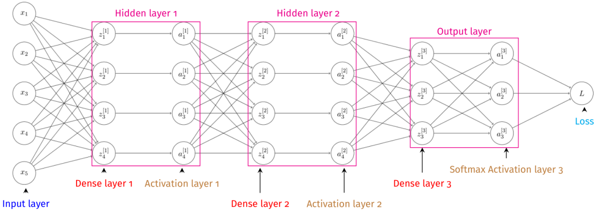
- Total number of layers $= L + 1$.

Deep L-layer neural network: notation and building blocks



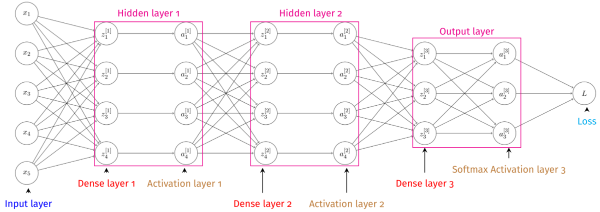
- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}.$

Deep L-layer neural network: notation and building blocks



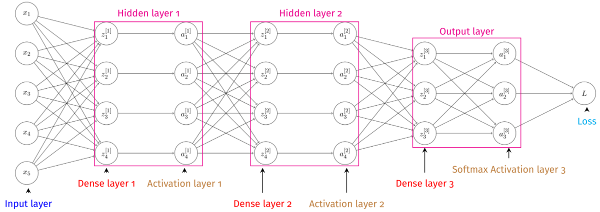
- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}.$
- $n^{[l]}$: number of nodes in layer l.

Deep L-layer neural network: notation and building blocks



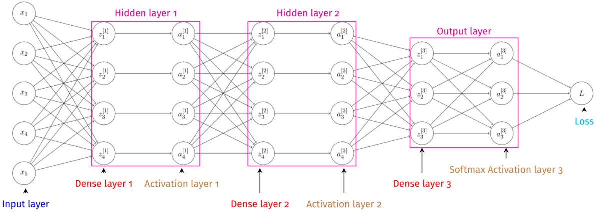
- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}.$
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.

Deep L-layer neural network: notation and building blocks



- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}$.
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.

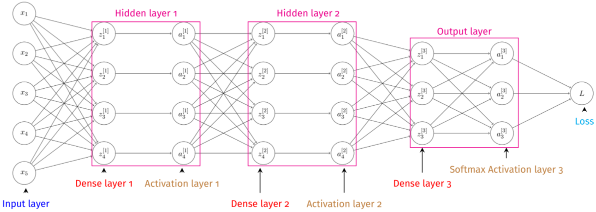
Deep L-layer neural network: notation and building blocks



- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{(\text{input layer})}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{(\text{output layer})}.$
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{W}^{[l]}$: weights matrix associated with dense layer l of shape $n^{[l]} \times n^{[l-1]}$ for $l = 1, 2, \ldots, L$.

Deep L-layer neural network: notation and building blocks

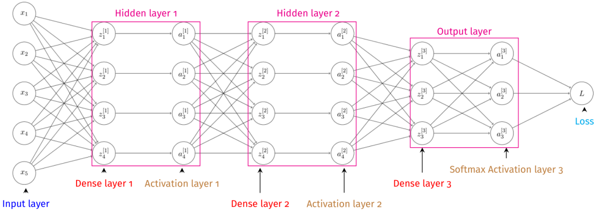
MANIPAL ACADEMY *of* HIGHER EDUCATION
(Institution of Eminence Deemed to be University)



- Total number of layers $= L+1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}.$
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{W}^{[l]}$: weights matrix associated with dense layer l of shape $n^{[l]} \times n^{[l-1]}$ for $l = 1, 2, \ldots, L$.
- $g^{[l]}$: activation function associated with activation layer l.

Deep L-layer neural network: notation and building blocks

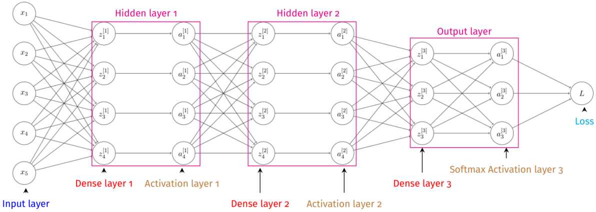
MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)



- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}$.
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{W}^{[l]}$: weights matrix associated with dense layer l of shape $n^{[l]} \times n^{[l-1]}$ for $l = 1, 2, \ldots, L$.
- $g^{[l]}$: activation function associated with activation layer l.
- For a batch of samples of size b, for layer l with $l = 1, 2, \ldots, L$

Deep L-layer neural network: notation and building blocks

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

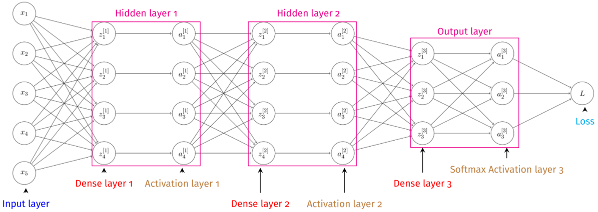


- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}$.
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{W}^{[l]}$: weights matrix associated with dense layer l of shape $n^{[l]} \times n^{[l-1]}$ for $l = 1, 2, \ldots, L$.
- $g^{[l]}$: activation function associated with activation layer l.
- For a batch of samples of size b, for layer l with $l = 1, 2, \ldots, L$

$$\text{raw scores matrix } \mathbf{Z}^{[l]} = \begin{bmatrix} \mathbf{z}^{[l](0)} & \mathbf{z}^{[l](1)} & \ldots & \mathbf{z}^{[l](b-1)} \end{bmatrix},$$

Deep L-layer neural network: notation and building blocks

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)



- Total number of layers $= L + 1$.
- Layer index: $l = \underbrace{0}_{\text{(input layer)}}, \underbrace{1, 2, \ldots L-1,}_{\text{hidden layers}} \underbrace{L}_{\text{(output layer)}}.$
- $n^{[l]}$: number of nodes in layer l.
- $\mathbf{z}^{[l]}$: raw scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{a}^{[l]}$: activated scores vector for hidden layer l of shape $n^{[l]}$.
- $\mathbf{W}^{[l]}$: weights matrix associated with dense layer l of shape $n^{[l]} \times n^{[l-1]}$ for $l = 1, 2, \ldots, L$.
- $g^{[l]}$: activation function associated with activation layer l.
- For a batch of samples of size b, for layer l with $l = 1, 2, \ldots, L$

$$\text{raw scores matrix } \mathbf{Z}^{[l]} = \begin{bmatrix} \mathbf{z}^{[l](0)} & \mathbf{z}^{[l](1)} & \ldots & \mathbf{z}^{[l](b-1)} \end{bmatrix},$$

$$\text{activated scores matrix } \mathbf{A}^{[l]} = \begin{bmatrix} \mathbf{a}^{[l](0)} & \mathbf{a}^{[l](1)} & \ldots & \mathbf{a}^{[l](b-1)} \end{bmatrix}.$$

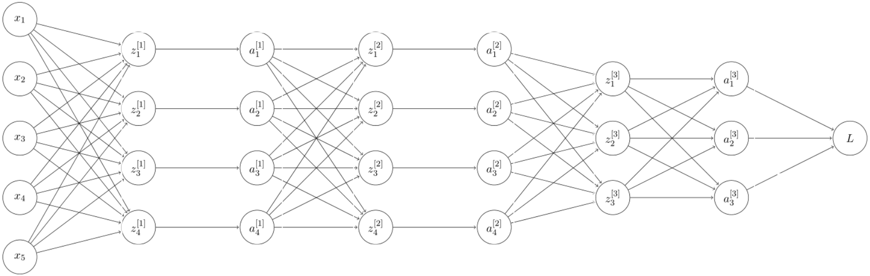
Deep L-layer neural network forward propagation

Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):

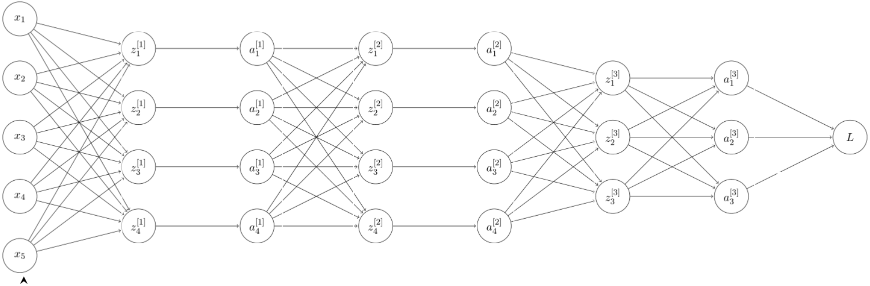
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Deep L-layer neural network forward propagation

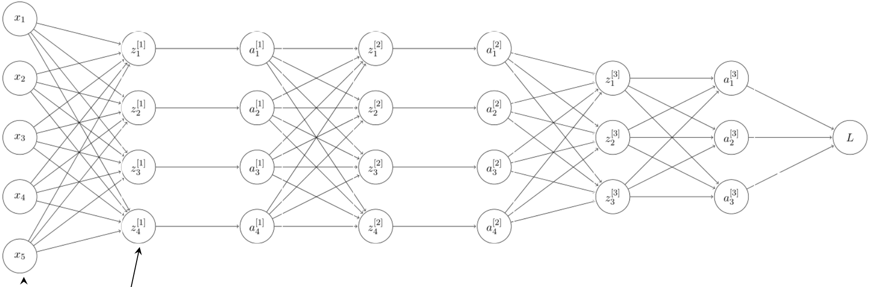
Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Input layer: $\mathbf{a}^{[0]}$

Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):

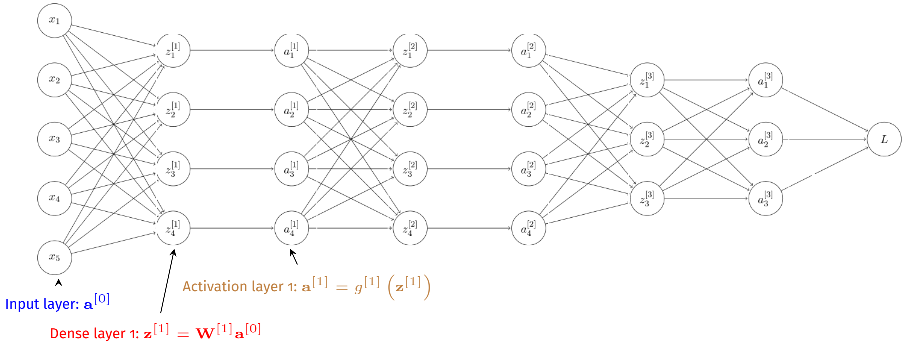


Input layer: $\mathbf{a}^{[0]}$

Dense layer 1: $\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]}$

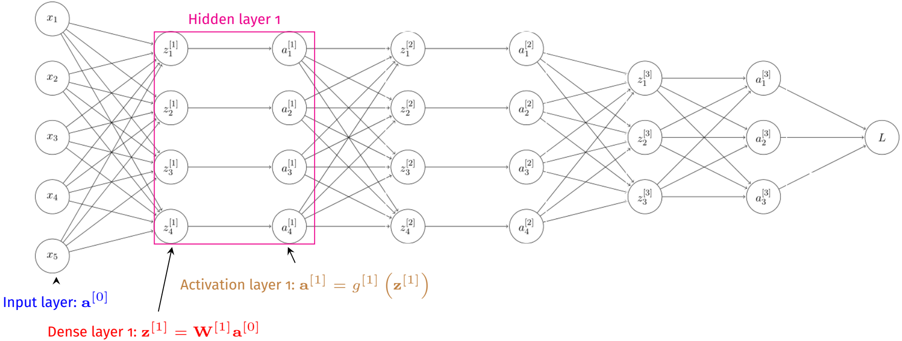
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Deep L-layer neural network forward propagation

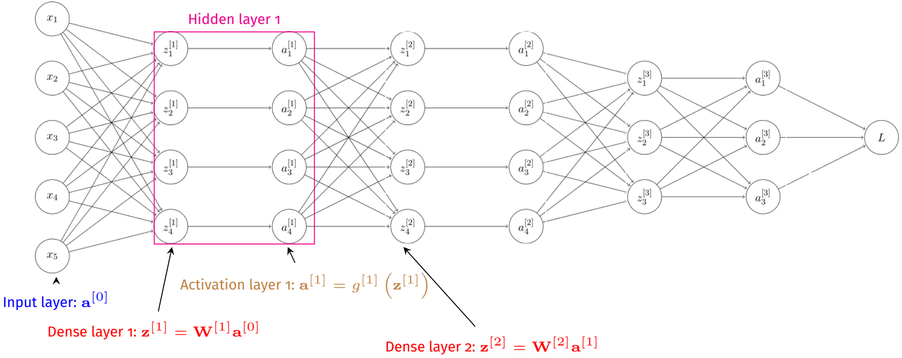
Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Deep L-layer neural network forward propagation

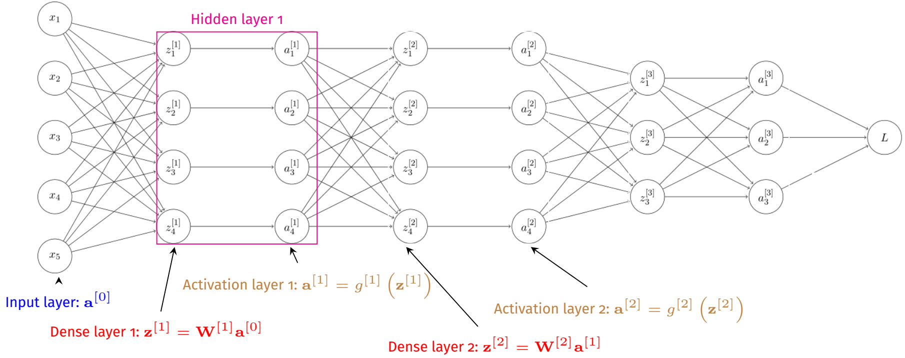
MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



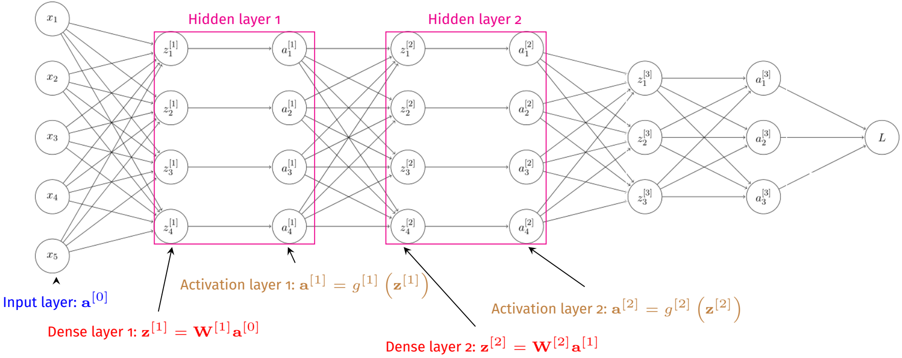
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



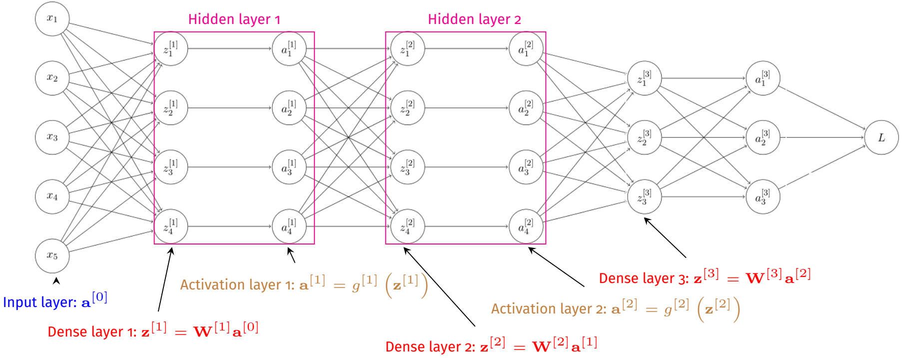
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



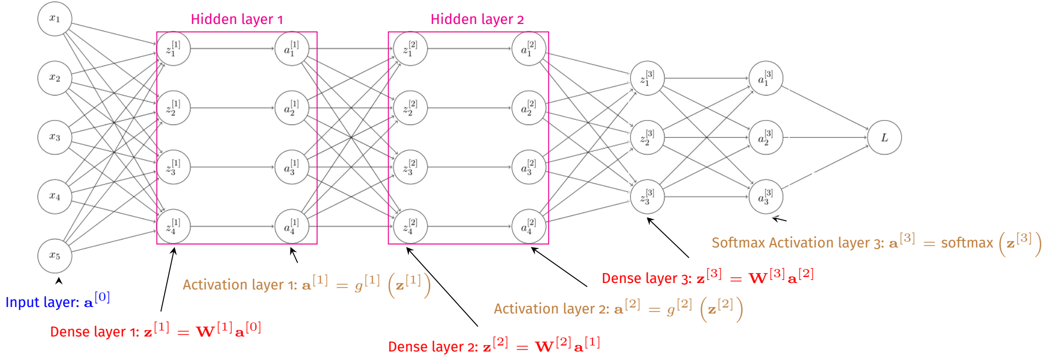
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



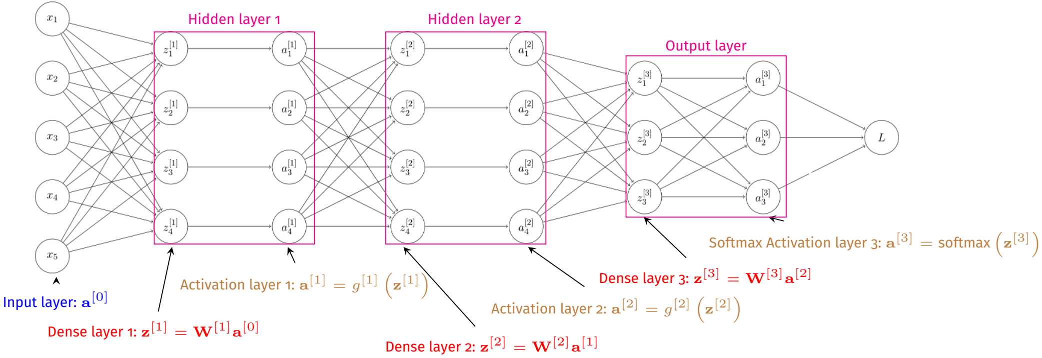
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



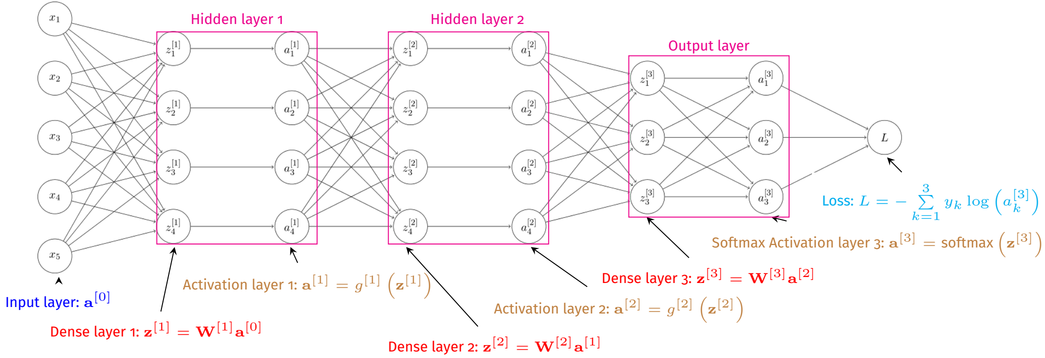
Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Deep L-layer neural network forward propagation

Consider applying a 3-layer neural network to a sample $\mathbf{x}$ with 5 features and correct output label y from 3 possible output labels (bias feature 1 ignored for clarity):



Accounting for bias in a deep L-layer neural network

Accounting for bias in a deep L-layer neural network

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.

Accounting for bias in a deep L-layer neural network

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

Accounting for bias in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector}$$

Accounting for bias in a deep L-layer neural network

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector} \Rightarrow \mathbf{z}^{[1]} = \mathbf{W}_B^{[1]}\mathbf{a}_B^{[0]}, \text{ where } \mathbf{W}_B^{[1]} \text{ is a } \underbrace{4 \times 6}_{n^{[1]} \times \left(n^{[0]}+1\right)} \text{-matrix.}$$

Accounting for bias in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector} \Rightarrow \mathbf{z}^{[1]} = \mathbf{W}_B^{[1]}\mathbf{a}_B^{[0]}, \text{ where } \mathbf{W}_B^{[1]} \text{ is a } \underbrace{4 \times 6}_{n^{[1]} \times \left(n^{[0]}+1\right)} \text{-matrix.}$$

- For simplicity, we will refer to $\mathbf{W}_B^{[1]}$ as just $\mathbf{W}^{[1]}$.

Accounting for bias in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)
INSPIRED BY LIFE

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector} \Rightarrow \mathbf{z}^{[1]} = \mathbf{W}_B^{[1]}\mathbf{a}_B^{[0]}, \text{ where } \mathbf{W}_B^{[1]} \text{ is a } \underbrace{4 \times 6}_{n^{[1]}\times\left(n^{[0]}+1\right)} \text{-matrix.}$$

- For simplicity, we will refer to $\mathbf{W}_B^{[1]}$ as just $\mathbf{W}^{[1]}$.
- Similarly, we have

$$\Bigg\{$$

Accounting for bias in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector} \Rightarrow \mathbf{z}^{[1]} = \mathbf{W}_B^{[1]}\mathbf{a}_B^{[0]}, \text{ where } \mathbf{W}_B^{[1]} \text{ is a } \underbrace{4 \times 6}_{n^{[1]} \times (n^{[0]}+1)} \text{-matrix.}$$

- For simplicity, we will refer to $\mathbf{W}_B^{[1]}$ as just $\mathbf{W}^{[1]}$.
- Similarly, we have

$$\begin{cases} \mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}_B^{[1]}, \text{ where } \mathbf{W}^{[2]} \text{ is a } n^{[2]} \times \left(n^{[1]} + 1\right) = 4 \times 5\text{-matrix,} \\ \end{cases}$$

Accounting for bias in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

- In the 3-layer neural network, we have $n^{[0]} = 5, n^{[1]} = 4, n^{[2]} = 4$, and $n^{[3]} = 3$.
- The bias feature 1 is appended to the activated scores prior to feeding as input to the next dense layer. For example:

$$\mathbf{a}_B^{[0]} = \begin{bmatrix} \mathbf{a}^{[0]} \\ 1 \end{bmatrix} \text{ is a } 6\text{-vector} \Rightarrow \mathbf{z}^{[1]} = \mathbf{W}_B^{[1]}\mathbf{a}_B^{[0]}, \text{where } \mathbf{W}_B^{[1]} \text{ is a } \underbrace{4 \times 6}_{n^{[1]} \times \left(n^{[0]}+1\right)} \text{-matrix.}$$

- For simplicity, we will refer to $\mathbf{W}_B^{[1]}$ as just $\mathbf{W}^{[1]}$.
- Similarly, we have

$$\begin{cases} \mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\mathbf{a}_B^{[1]}, \text{ where } \mathbf{W}^{[2]} \text{ is a } n^{[2]} \times \left(n^{[1]} + 1\right) = 4 \times 5\text{-matrix,} \\ \mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\mathbf{a}_B^{[2]}, \text{ where } \mathbf{W}^{[3]} \text{ is a } n^{[3]} \times \left(n^{[2]} + 1\right) = 3 \times 5\text{-matrix.} \end{cases}$$

Minibatch forward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$,

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \ldots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \ldots & \mathbf{y}^{(b)} \end{bmatrix}$.

Minibatch forward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\left\{ \begin{array}{l} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \\ \\ \\ \end{array} \right.$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \ldots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \ldots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\left\{ \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \ldots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \ldots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \right.$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\left\{ \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \right.$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\left\{ \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \right.$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \end{cases}$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = \end{cases}$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = \end{cases}$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}}\right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$,

Minibatch forward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \ldots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \ldots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \ldots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \ldots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \ldots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \ldots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \ldots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \ldots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$, and for hidden layer 3, $\mathbf{Z}^{[3]} = \mathbf{W}^{[3]}\mathbf{A}_B^{[2]}$ and $\mathbf{A}^{[3]} = \text{softmax}\left(\mathbf{Z}^{[3]}\right)$.

Minibatch forward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$, and for hidden layer 3, $\mathbf{Z}^{[3]} = \mathbf{W}^{[3]}\mathbf{A}_B^{[2]}$ and $\mathbf{A}^{[3]} = \text{softmax}\left(\mathbf{Z}^{[3]}\right)$.
- Setting $\hat{\mathbf{Y}} = \mathbf{A}^{[3]}$, the predicted probabilities matrix for the minibatch,

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$, and for hidden layer 3, $\mathbf{Z}^{[3]} = \mathbf{W}^{[3]}\mathbf{A}_B^{[2]}$ and $\mathbf{A}^{[3]} = \text{softmax}\left(\mathbf{Z}^{[3]}\right)$.
- Setting $\hat{\mathbf{Y}} = \mathbf{A}^{[3]}$, the predicted probabilities matrix for the minibatch, the categorical crossentropy (CCE) loss for the ith sample is $L_i = \sum_{k=1}^{3} -y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right)$,

Minibatch forward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$, and for hidden layer 3, $\mathbf{Z}^{[3]} = \mathbf{W}^{[3]}\mathbf{A}_B^{[2]}$ and $\mathbf{A}^{[3]} = \text{softmax}\left(\mathbf{Z}^{[3]}\right)$.
- Setting $\hat{\mathbf{Y}} = \mathbf{A}^{[3]}$, the predicted probabilities matrix for the minibatch, the categorical crossentropy (CCE) loss for the ith sample is $L_i = \sum_{k=1}^{3} -y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right)$, which leads to the average categorical crossentropy (CCE) batch loss for the minibatch as

Minibatch forward propagation in a deep L-layer neural network

- Suppose we have minibatch comprising b samples represented as the $5 \times b$-matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(b)} \end{bmatrix}$, with one-hot encoded true labels represented as the $3 \times b$-matrix (3 possible output categories) $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(b)} \end{bmatrix}$.
- For hidden layer 1, we calculate

$$\begin{cases} \underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{\mathbf{Z}^{[1]}} = \begin{bmatrix} \mathbf{W}^{[1]}\mathbf{a}_B^{[0](1)} & \dots & \mathbf{W}^{[1]}\mathbf{a}_B^{[0](b)} \end{bmatrix} = \mathbf{W}^{[1]} \underbrace{\begin{bmatrix} \mathbf{a}_B^{[0](1)} & \dots & \mathbf{a}_B^{[0](b)} \end{bmatrix}}_{\mathbf{A}_B^{[0]}} = \mathbf{W}^{[1]}\mathbf{A}_B^{[0]}, \\ \underbrace{\begin{bmatrix} \mathbf{a}^{[1](1)} & \dots & \mathbf{a}^{[1](b)} \end{bmatrix}}_{\mathbf{A}^{[1]}} = \begin{bmatrix} g^{[1]}\left(\mathbf{z}^{[1](1)}\right) & \dots & g^{[1]}\left(\mathbf{z}^{[1](b)}\right) \end{bmatrix} = g^{[1]}\left(\underbrace{\begin{bmatrix} \mathbf{z}^{[1](1)} & \dots & \mathbf{z}^{[1](b)} \end{bmatrix}}_{=\mathbf{Z}^{[1]}} \right) = g^{[1]}\left(\mathbf{Z}^{[1]}\right). \end{cases}$$

- Similarly, we calculate for hidden layer 2, $\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}_B^{[1]}$ and $\mathbf{A}^{[2]} = g^{[2]}\left(\mathbf{Z}^{[2]}\right)$, and for hidden layer 3, $\mathbf{Z}^{[3]} = \mathbf{W}^{[3]}\mathbf{A}_B^{[2]}$ and $\mathbf{A}^{[3]} = \text{softmax}\left(\mathbf{Z}^{[3]}\right)$.
- Setting $\hat{\mathbf{Y}} = \mathbf{A}^{[3]}$, the predicted probabilities matrix for the minibatch, the categorical crossentropy (CCE) loss for the ith sample is $L_i = \sum_{k=1}^{3} -y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right)$, which leads to the average categorical crossentropy (CCE) batch loss for the minibatch as $L = \frac{1}{b}\left[L_1 + \dots + L_b\right] = \frac{1}{b}\left[\sum_{k=1}^{3} -y_k^{(1)} \log\left(\hat{y}_k^{(1)}\right) + \dots + \sum_{k=1}^{3} -y_k^{(b)} \log\left(\hat{y}_k^{(b)}\right)\right]$.

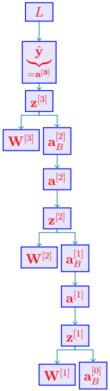
Deep L-layer neural network gradient calculation using backward propagation

Deep L-layer neural network gradient calculation using backward propagation

Computation graph:

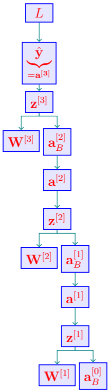
Deep L-layer neural network gradient calculation using backward propagation

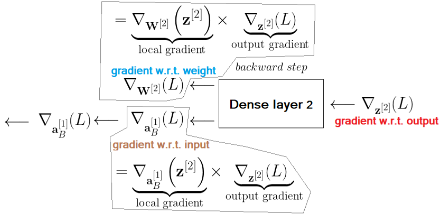
Computation graph:



Deep L-layer neural network gradient calculation using backward propagation

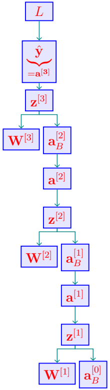
Computation graph:

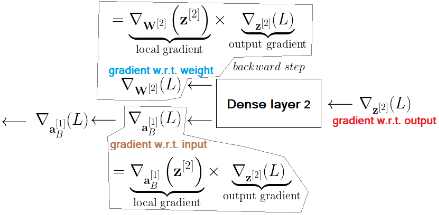




Deep L-layer neural network gradient calculation using backward propagation

Computation graph:

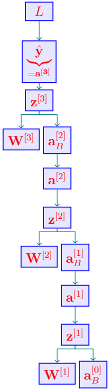


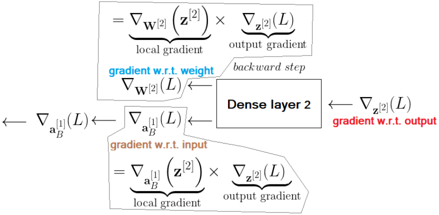


During backward propagation, a dense layer looks at the gradient flowing backward from its output side and does two things:

Deep L-layer neural network gradient calculation using backward propagation

Computation graph:



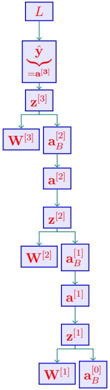


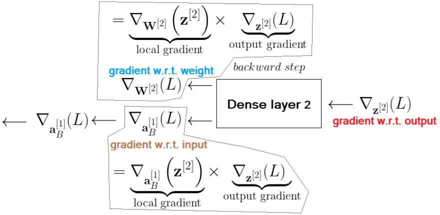
During backward propagation, a dense layer looks at the gradient flowing backward from its output side and does two things: (1) calculates the local gradient w.r.t. its weights and multiplies that with the output gradient so that the weights can be updated

Deep L-layer neural network gradient calculation using backward propagation

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Computation graph:

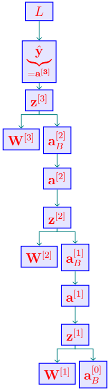


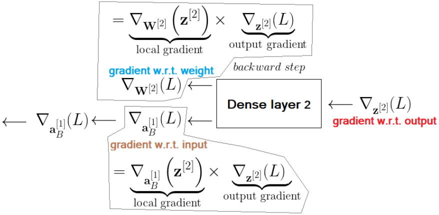


During backward propagation, a dense layer looks at the gradient flowing backward from its output side and does two things: (1) calculates the local gradient w.r.t. its weights and multiplies that with the output gradient so that the weights can be updated (2) calculates the local gradient w.r.t. its input and multiplies that with the output gradient so that the resulting gradient flowing backward from its input side is returned;

Deep L-layer neural network gradient calculation using backward propagation

Computation graph:





During backward propagation, a dense layer looks at the gradient flowing backward from its output side and does two things: (1) calculates the local gradient w.r.t. its weights and multiplies that with the output gradient so that the weights can be updated (2) calculates the local gradient w.r.t. its input and multiplies that with the output gradient so that the resulting gradient flowing backward from its input side is returned; this then becomes the gradient flowing backward from the output side of the previous activation layer.

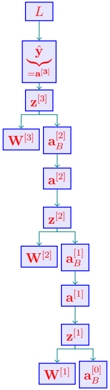
Deep L-layer neural network gradient calculation using backward propagation – continued

Deep L-layer neural network gradient calculation using backward propagation – continued

Computation graph:

Deep L-layer neural network gradient calculation using backward propagation – continued

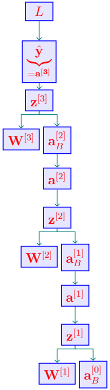
Computation graph:



Deep L-layer neural network gradient calculation using backward propagation – continued

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Computation graph:



gradient w.r.t input

$$\nabla_{\mathbf{z}^{[1]}}(L) = \underbrace{\nabla_{\mathbf{z}^{[1]}}\left(\mathbf{a}^{[1]}\right)}_{\text{local gradient}} \times \underbrace{\nabla_{\mathbf{a}^{[1]}}(L)}_{\text{output gradient}}$$

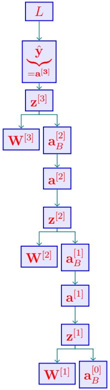
$\longleftarrow$ **Activation layer 1** $\longleftarrow$

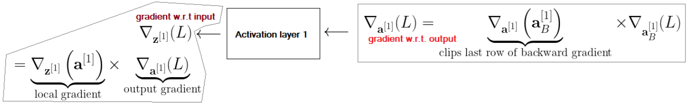
gradient w.r.t. output

$$\nabla_{\mathbf{a}^{[1]}}(L) = \underbrace{\nabla_{\mathbf{a}^{[1]}}\left(\mathbf{a}_B^{[1]}\right)}_{\text{clips last row of backward gradient}} \times \nabla_{\mathbf{a}_B^{[1]}}(L)$$

Deep L-layer neural network gradient calculation using backward propagation – continued

Computation graph:

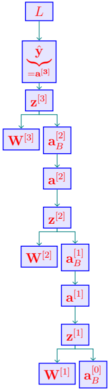




During backward propagation, an activation layer does the following:

Deep L-layer neural network gradient calculation using backward propagation – continued

Computation graph:



gradient w.r.t input

$$\nabla_{\mathbf{z}^{[1]}}(L) \longleftarrow \boxed{\text{Activation layer 1}} \longleftarrow \nabla_{\mathbf{a}^{[1]}}(L) = \underbrace{\nabla_{\mathbf{a}^{[1]}}\left(\mathbf{a}_B^{[1]}\right)}_{\text{clips last row of backward gradient}} \times \nabla_{\mathbf{a}_B^{[1]}}(L)$$

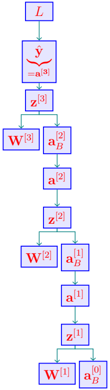
gradient w.r.t. output

$$= \underbrace{\nabla_{\mathbf{z}^{[1]}}\left(\mathbf{a}^{[1]}\right)}_{\text{local gradient}} \times \underbrace{\nabla_{\mathbf{a}^{[1]}}(L)}_{\text{output gradient}}$$

During backward propagation, an activation layer does the following: it clips the gradient flowing backward sent by the subsequent dense layer such that the last row of the gradient is removed;

Deep L-layer neural network gradient calculation using backward propagation – continued

Computation graph:



gradient w.r.t input

$$\nabla_{\mathbf{z}^{[1]}}(L) = \underbrace{\nabla_{\mathbf{z}^{[1]}}\left(\mathbf{a}^{[1]}\right)}_{\text{local gradient}} \times \underbrace{\nabla_{\mathbf{a}^{[1]}}(L)}_{\text{output gradient}} \longleftarrow \boxed{\text{Activation layer 1}} \longleftarrow$$

$$\nabla_{\mathbf{a}^{[1]}}(L) = \underbrace{\nabla_{\mathbf{a}^{[1]}}\left(\mathbf{a}_B^{[1]}\right)}_{\text{clips last row of backward gradient}} \times \nabla_{\mathbf{a}_B^{[1]}}(L)$$

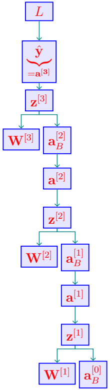
gradient w.r.t. output

During backward propagation, an activation layer does the following: it clips the gradient flowing backward sent by the subsequent dense layer such that the last row of the gradient is removed; the resulting gradient is then sent flowing backward on the output side for the activation layer;

Deep L-layer neural network gradient calculation using backward propagation – continued

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Computation graph:



$$\underbrace{\nabla_{z^{[1]}}(L)}_{\text{gradient w.r.t input}} \longleftarrow \boxed{\text{Activation layer 1}} \longleftarrow \underbrace{\nabla_{a^{[1]}}(L)}_{\text{gradient w.r.t. output}} = \underbrace{\nabla_{a^{[1]}}\left(a_B^{[1]}\right)}_{\text{clips last row of backward gradient}} \times \nabla_{a_B^{[1]}}(L)$$

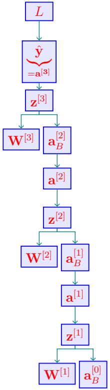
$$= \underbrace{\nabla_{z^{[1]}}\left(a^{[1]}\right)}_{\text{local gradient}} \times \underbrace{\nabla_{a^{[1]}}(L)}_{\text{output gradient}}$$

During backward propagation, an activation layer does the following: it clips the gradient flowing backward sent by the subsequent dense layer such that the last row of the gradient is removed; the resulting gradient is then sent flowing backward on the output side for the activation layer; after that, it calculates its local gradient which is multiplied by the clipped output gradient so that the resulting gradient flowing backward from its input side is returned;

Deep L-layer neural network gradient calculation using backward propagation – continued

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Computation graph:



gradient w.r.t input

$$\nabla_{\mathbf{z}^{[1]}}(L) \longleftarrow \boxed{\text{Activation layer 1}} \longleftarrow \underbrace{\nabla_{\mathbf{a}^{[1]}}(L)}_{\text{gradient w.r.t. output}} = \underbrace{\nabla_{\mathbf{a}^{[1]}}\left(\mathbf{a}_B^{[1]}\right)}_{\text{clips last row of backward gradient}} \times \nabla_{\mathbf{a}_B^{[1]}}(L)$$

$$= \underbrace{\nabla_{\mathbf{z}^{[1]}}\left(\mathbf{a}^{[1]}\right)}_{\text{local gradient}} \times \underbrace{\nabla_{\mathbf{a}^{[1]}}(L)}_{\text{output gradient}}$$

During backward propagation, an activation layer does the following: it clips the gradient flowing backward sent by the subsequent dense layer such that the last row of the gradient is removed; the resulting gradient is then sent flowing backward on the output side for the activation layer; after that, it calculates its local gradient which is multiplied by the clipped output gradient so that the resulting gradient flowing backward from its input side is returned; this then becomes the gradient flowing backward from the output side of the previous dense layer.

Minibatch backward propagation in a deep L-layer neural network

Minibatch backward propagation in a deep L-layer neural network

Suppose we have minibatch comprising b samples;

Minibatch backward propagation in a deep L-layer neural network

Suppose we have minibatch comprising b samples; the gradient of the average loss w.r.t. a particular set of weights is simply the average of the gradient of each sample's loss w.r.t. those weights;

Minibatch backward propagation in a deep L-layer neural network

Suppose we have minibatch comprising b samples; the gradient of the average loss w.r.t. a particular set of weights is simply the average of the gradient of each sample's loss w.r.t. those weights; for example, consider the gradient of the average loss w.r.t. $\mathbf{W}^{[2]}$ focusing on the gradient flowing through hidden layer 2:

Minibatch backward propagation in a deep L-layer neural network

Suppose we have minibatch comprising b samples; the gradient of the average loss w.r.t. a particular set of weights is simply the average of the gradient of each sample's loss w.r.t. those weights; for example, consider the gradient of the average loss w.r.t. $\mathbf{W}^{[2]}$ focusing on the gradient flowing through hidden layer 2:

$$L = \frac{1}{b}\left[L_1 + \cdots + L_b\right]$$

Minibatch backward propagation in a deep L-layer neural network

Suppose we have minibatch comprising b samples; the gradient of the average loss w.r.t. a particular set of weights is simply the average of the gradient of each sample's loss w.r.t. those weights; for example, consider the gradient of the average loss w.r.t. $\mathbf{W}^{[2]}$ focusing on the gradient flowing through hidden layer 2:

$$L = \frac{1}{b}\left[L_1 + \cdots + L_b\right]$$

$$\Rightarrow \nabla_{\mathbf{W}^{[2]}}(L) = \frac{1}{b}\left[\nabla_{\mathbf{W}^{[2]}}(L_1) + \cdots + \nabla_{\mathbf{W}^{[2]}}(L_b)\right]$$

Minibatch backward propagation in a deep L-layer neural network

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Suppose we have minibatch comprising b samples; the gradient of the average loss w.r.t. a particular set of weights is simply the average of the gradient of each sample's loss w.r.t. those weights; for example, consider the gradient of the average loss w.r.t. $\mathbf{W}^{[2]}$ focusing on the gradient flowing through hidden layer 2:

$$
\begin{aligned}
L &= \frac{1}{b}\left[L_1 + \cdots + L_b\right] \\
\Rightarrow \nabla_{\mathbf{W}^{[2]}}(L) &= \frac{1}{b}\left[\nabla_{\mathbf{W}^{[2]}}(L_1) + \cdots + \nabla_{\mathbf{W}^{[2]}}(L_b)\right] \\
&= \frac{1}{b}\left(\underbrace{\left[\nabla_{\mathbf{W}^{[2]}}\left(\mathbf{z}^{[2](1)}\right) \times \nabla_{\mathbf{z}^{[2](1)}}\left(\mathbf{a}^{[2](1)}\right) \times \nabla_{\mathbf{a}^{[2](1)}}(L_1)\right]}_{\text{sample 1}} + \cdots\right. \\
&\quad \left. + \underbrace{\left[\nabla_{\mathbf{W}^{[2]}}\left(\mathbf{z}^{[2](b)}\right) \times \nabla_{\mathbf{z}^{[2](b)}}\left(\mathbf{a}^{[2](b)}\right) \times \nabla_{\mathbf{a}^{[2](b)}}(L_b)\right]}_{\text{sample } b}\right).
\end{aligned}
$$