Books

» http://www.deeplearningbook.org/

Deep Learning

An MIT Press book

Ian Goodfellow and Yoshua Bengio and Aaron Courville

Exercises Lectures External Links

The Deep Learning textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular. The online version of the book is now complete and will remain available online for free.

The deep learning textbook can now be pre-ordered on Amazon. Pre-orders should ship on December 16, 2016.

For up to date announcements, join our mailing list.

Citing the book

To cite this book, please use this bibtex entry:

```
@book{Goodfellow-et-al-2016,
  title={Deep Learning},
  author={Ian Goodfellow and Yoshua Bengio and Aaron Courville},
  publisher={MIT Press},
  note={\url{http://www.deeplearningbook.org}},
  year={2016}
}
```

Books

http://neuralnetworksanddeeplearning.com/.org/

Neural Networks and Deep Learning

Neural Networks and Deep Learning is a free online book. The book will teach you about:

- Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data
- Deep learning, a powerful set of techniques for learning in neural networks

Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. This book will teach you many of the core concepts behind neural networks and deep learning.

For more details about the approach taken in the book, see here. Or you can jump directly to Chapter 1 and get started.

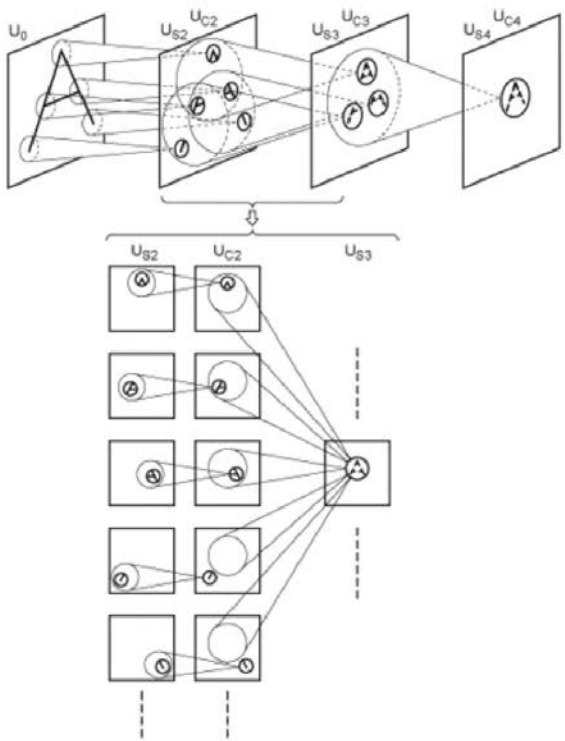
reviews

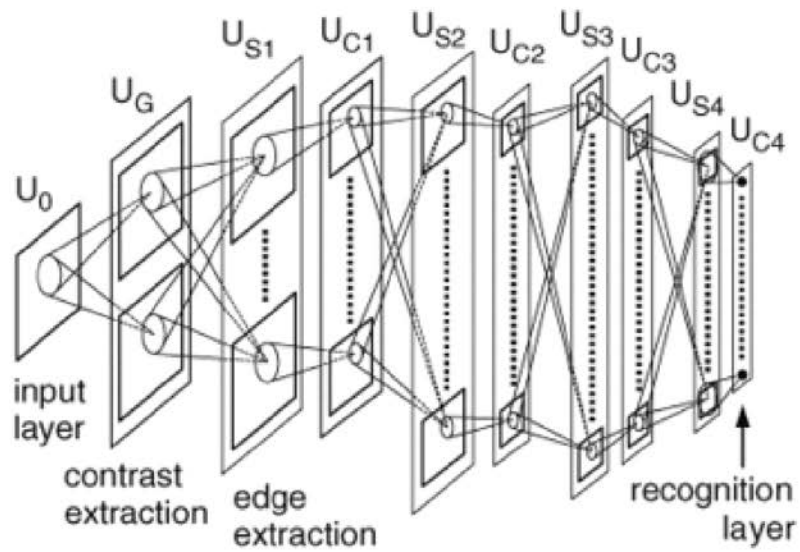
- http://www.deeplearningbook.org/contents/linear_algebra.html
- http://www.deeplearningbook.org/contents/prob.html
- http://www.deeplearningbook.org/contents/numerical.html

Convolutional Neural Networks

Earliest “deep” architecture

Neocognitron





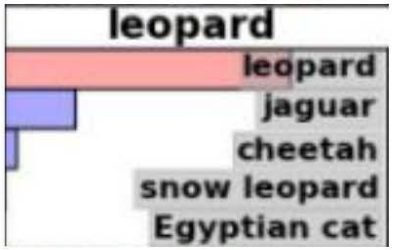
(Fukushima 1974-1982)

Goal: Given an image, we want to identify what class that image belongs to.

Input:

Classification

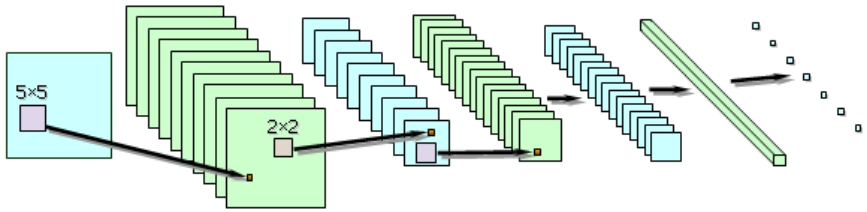
Output:



Pipeline:

Input

Convolutional Neural Network (CNN)

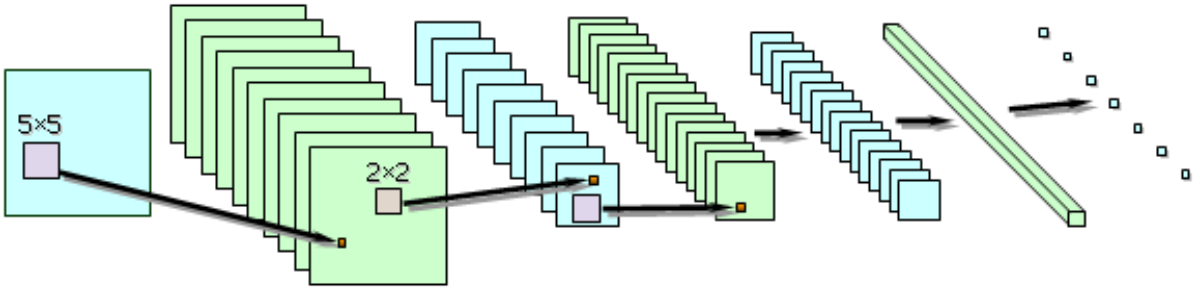


Output

A Monitor

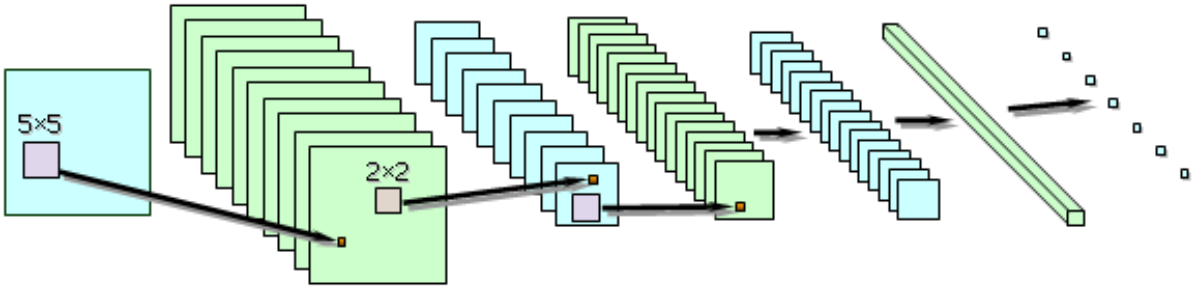
Convolutional Neural Nets (CNNs) in a nutshell:

- A typical CNN takes a raw RGB image as an input.
- It then applies a series of non-linear operations on top of each other.
- These include convolution, sigmoid, matrix multiplication, and pooling (subsampling) operations.
- The output of a CNN is a highly non-linear function of the raw RGB image pixels.



How the key operations are encoded in standard CNNs:

- Convolutional Layers: 2D Convolution
- Fully Connected Layers: Matrix Multiplication
- Sigmoid Layers: Sigmoid function
- Pooling Layers: Subsampling



2D convolution:

$$h = f \otimes g$$

f - the values in a 2D grid that we want to convolve

g - convolutional weights of size MxN

$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} f(i-m, j-n) g(m,n)$$

A sliding window operation across the entire grid f .

$f =$ 

$g_1 =$

0	0	0
0	1	0
0	0	0

$g_2 =$

0.107	0.113	0.107
0.113	0.119	0.113
0.107	0.113	0.107

$g_3 =$

-1	0	1
-2	0	2
-1	0	1

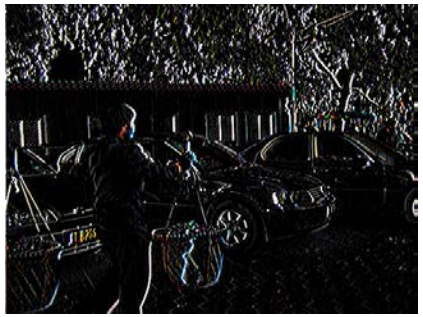
$f \otimes g_1$

Unchanged Image

$f \otimes g_2$

Blurred Image

$f \otimes g_3$



Vertical Edges

$f =$ 

$g_1 =$

0	0	0
0	1	0
0	0	0

$g_2 =$

0.107	0.113	0.107
0.113	0.119	0.113
0.107	0.113	0.107

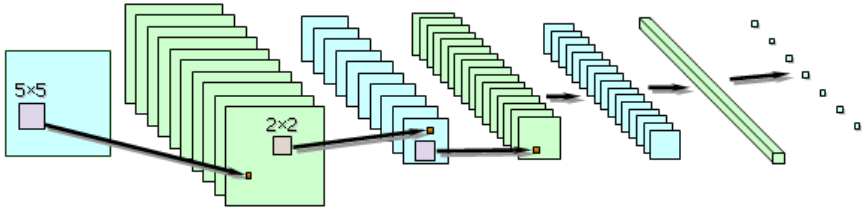
$g_3 =$

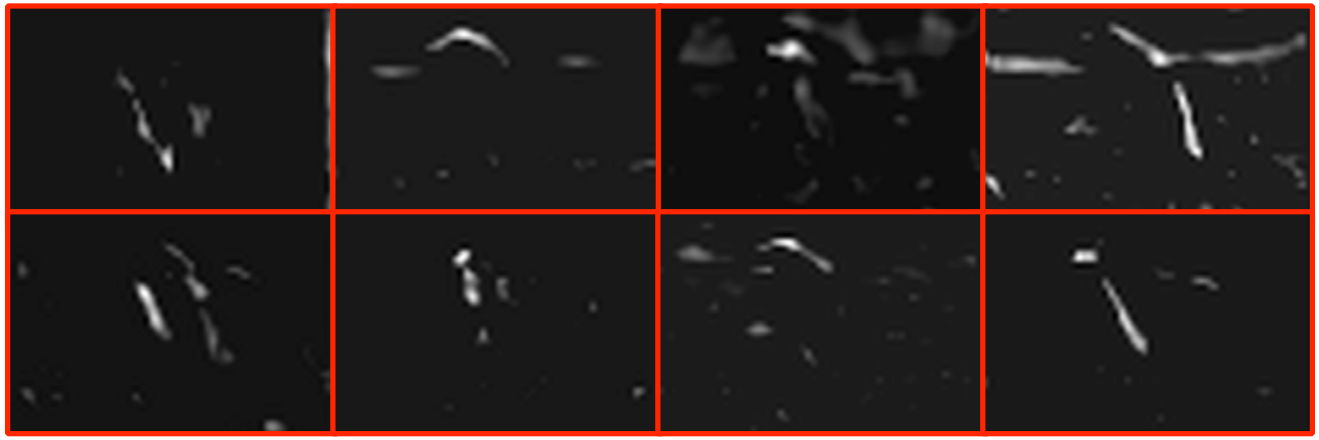
-1	0	1
-2	0	2
-1	0	1

CNNs aim to learn convolutional weights directly from the data

Input:

Convolutional Neural Network (CNN)

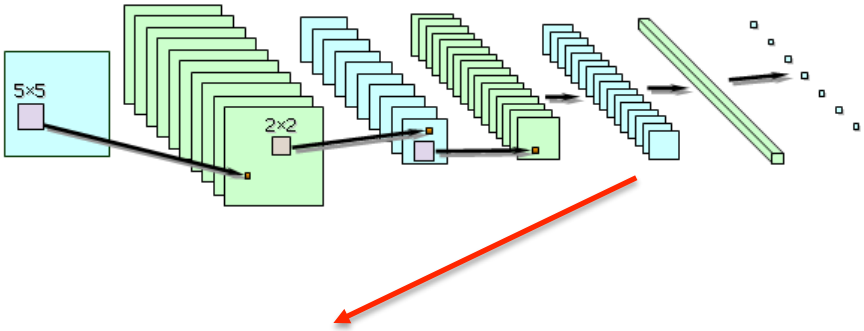


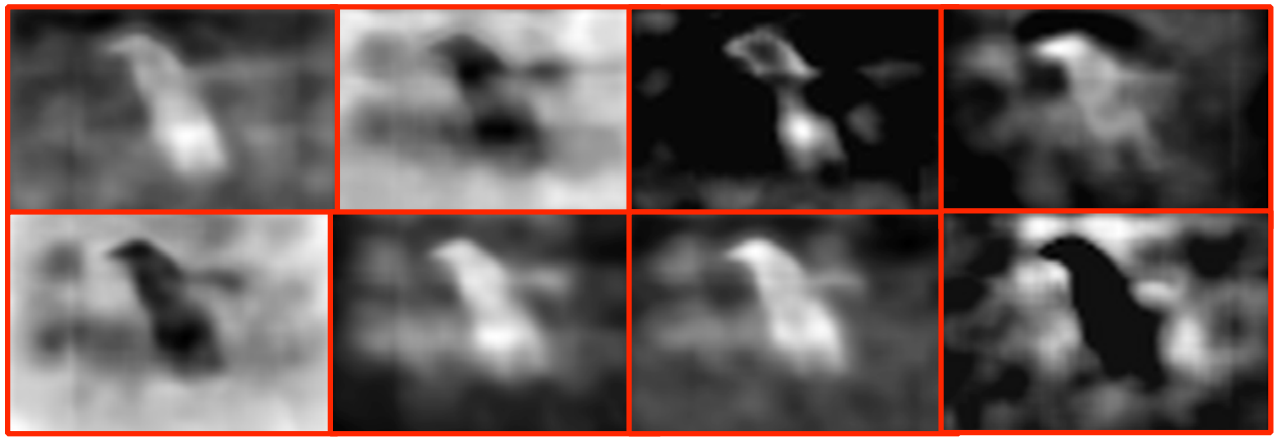


Early layers learn to detect low level structures such as oriented edges, colors and corners

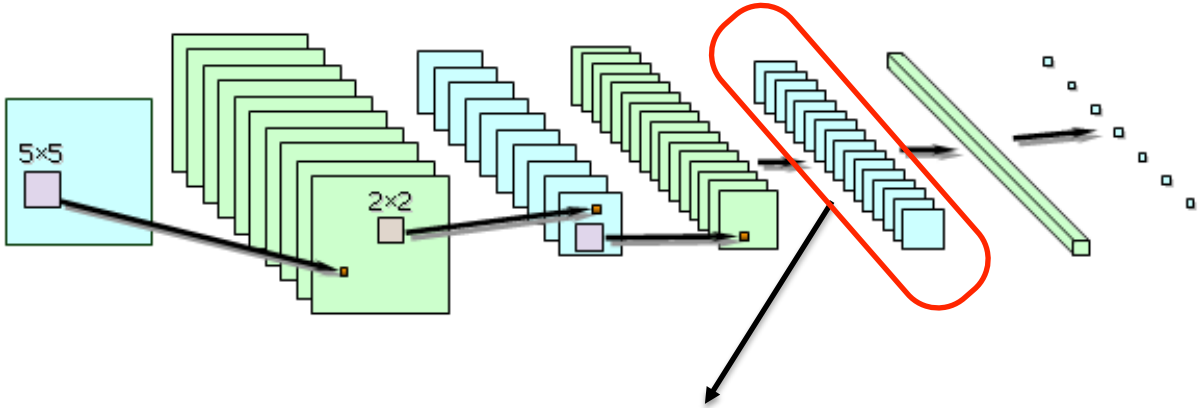
Input:

Convolutional Neural Network (CNN)

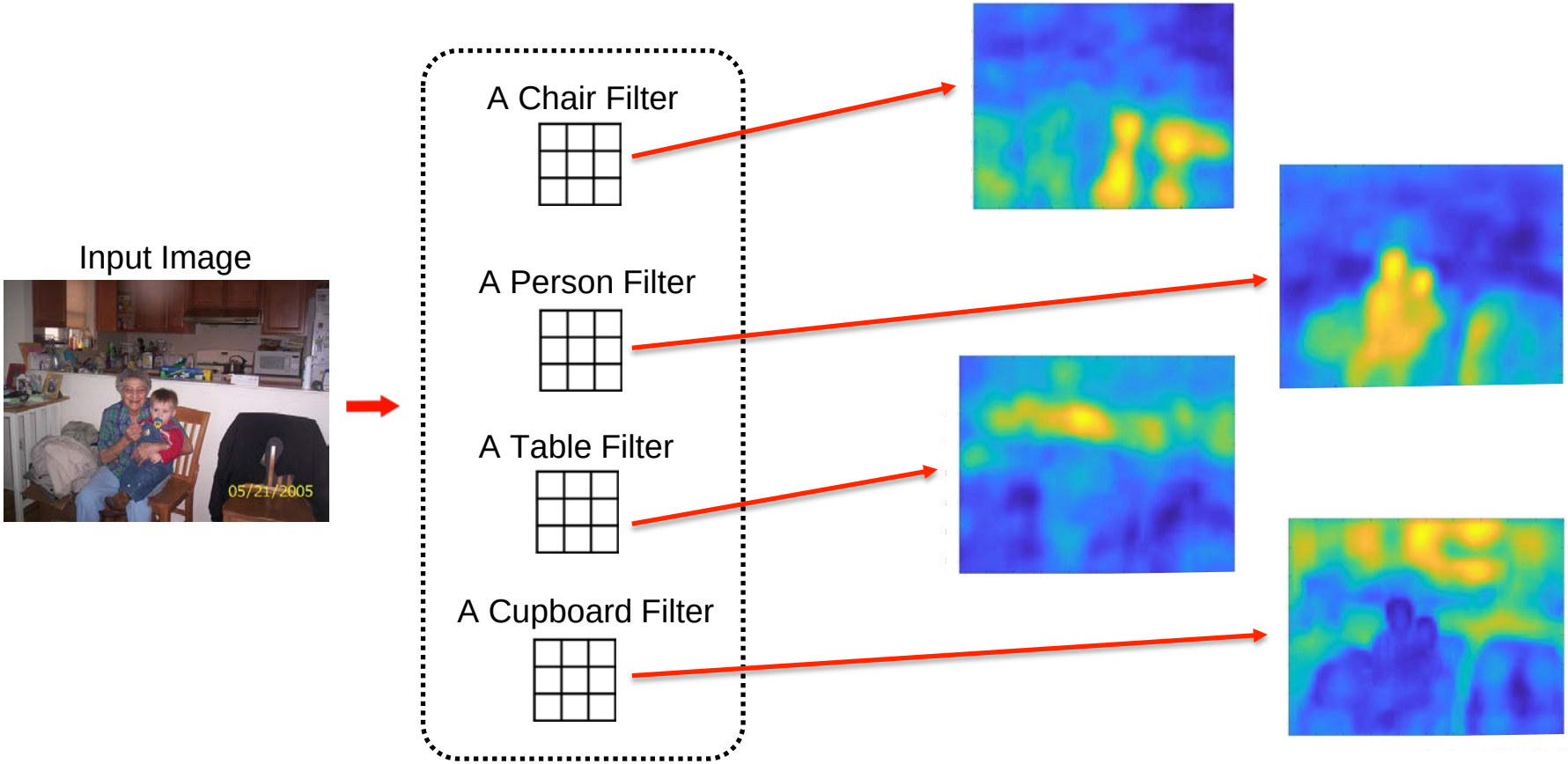




Deep layers learn to detect high-level object structures and their parts.



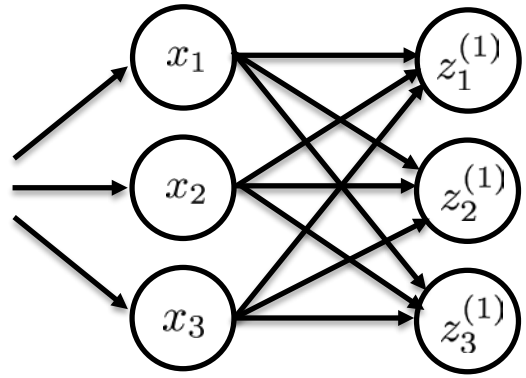
A Closer Look inside the Convolutional Layer



Fully Connected Layers:

Hidden Layer
Connections

Input Data



Layer 0 Layer 1

$z_i^{(l)}$ - the output unit i in layer l

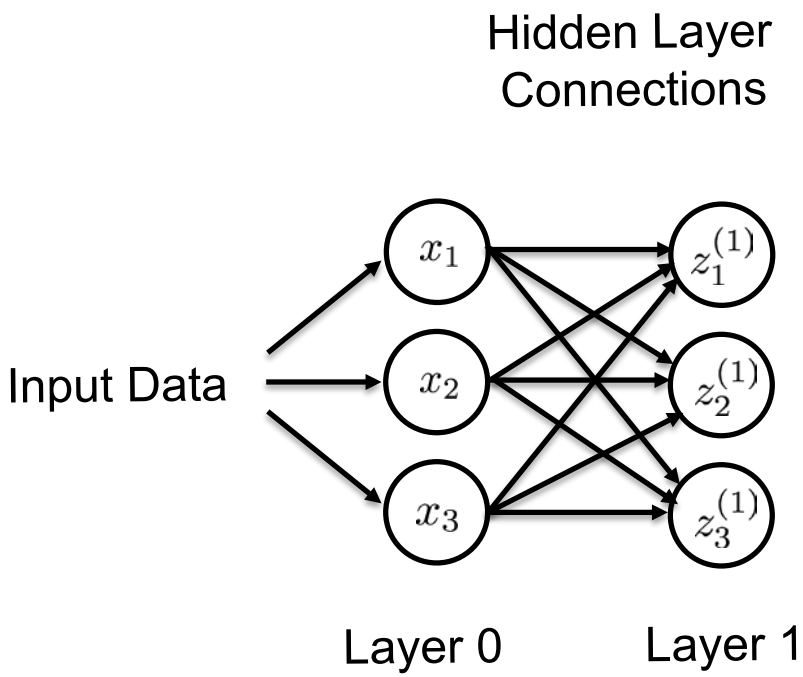
$W_{ij}^{(l)}$ - the weight connection between unit j in layer l and unit i in layer $l+1$

$$z_1^{(1)} = W_{11}^{(0)} x_1 + W_{12}^{(0)} x_2 + W_{13}^{(0)} x_3$$

$$z_2^{(1)} = W_{21}^{(0)} x_1 + W_{22}^{(0)} x_2 + W_{23}^{(0)} x_3$$

$$z_3^{(1)} = W_{31}^{(0)} x_1 + W_{32}^{(0)} x_2 + W_{33}^{(0)} x_3$$

Fully Connected Layers:

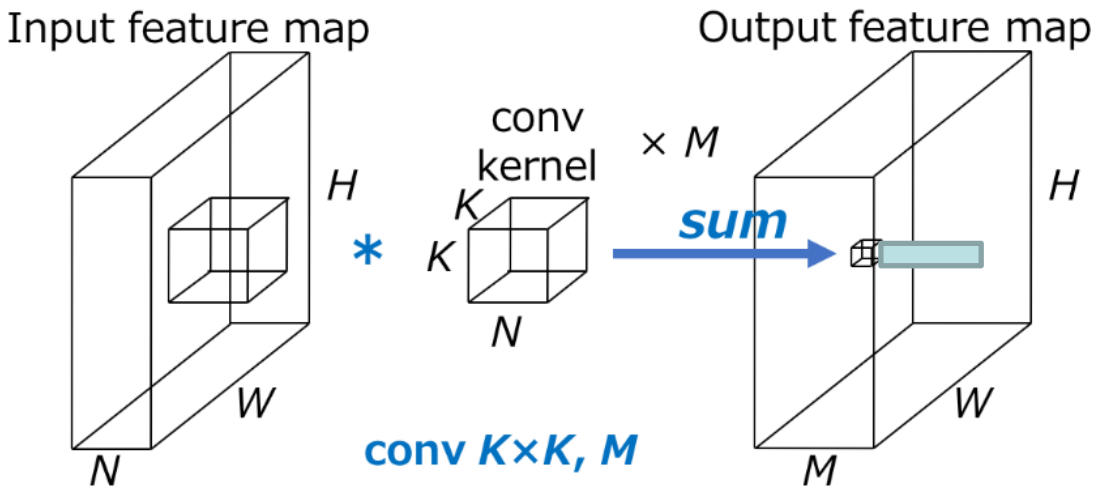


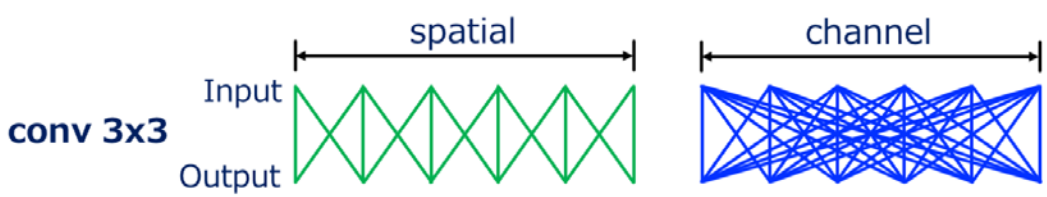
$z_i^{(l)}$ - the output unit i in layer l

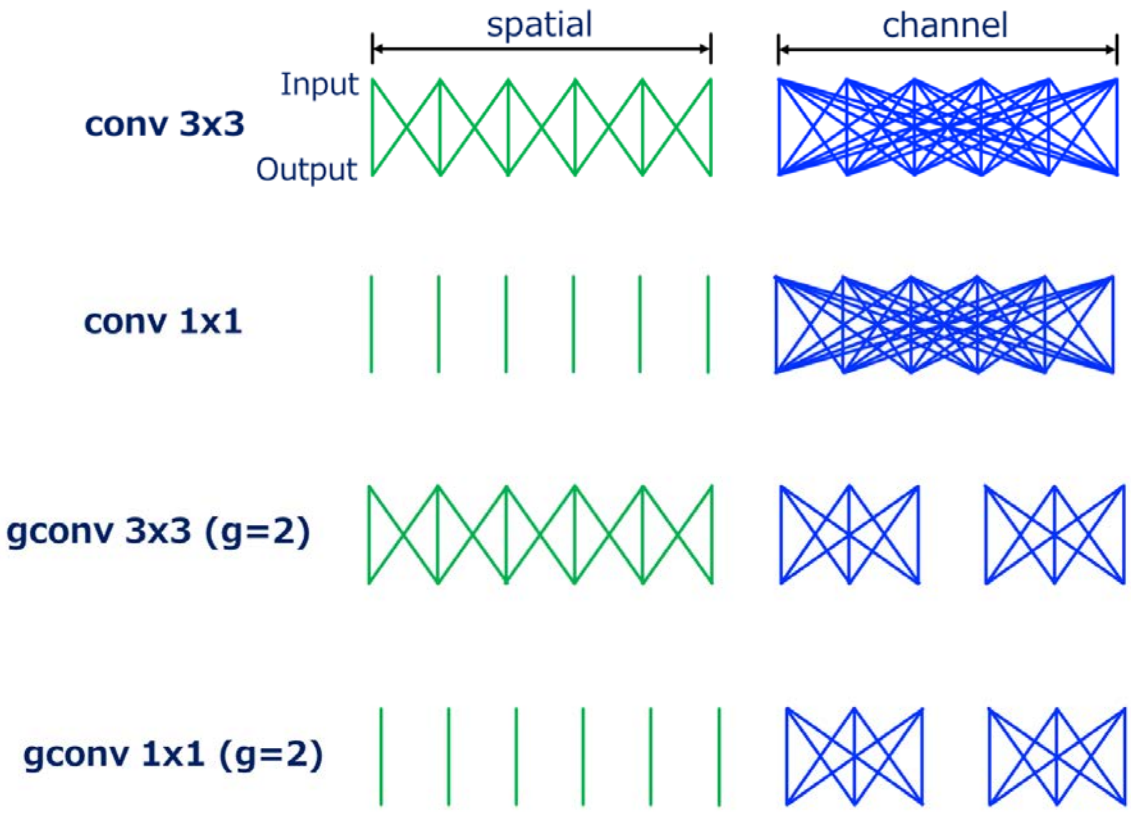
$W_{ij}^{(l)}$ - the weight connection between unit j in layer l and unit i in layer $l+1$

$$z^{(1)} = W^{(0)} x$$

matrix multiplication


Input feature map
Output feature map
conv
kernel
× M
H
K
K
N
W
*
sum
H
W
M
N
conv K×K, M

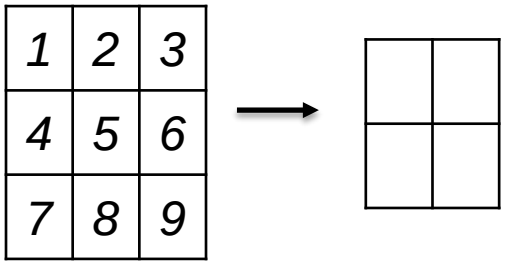

spatial
channel
Input
conv 3x3
Output



https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

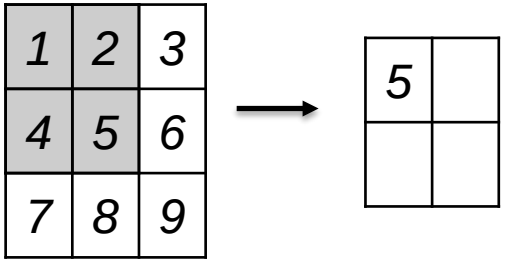
Max Pooling Layer:

- Sliding window is applied on a grid of values.
- The maximum is computed using the values in the current window.



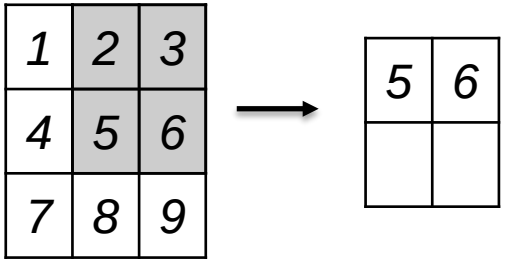
Max Pooling Layer:

- Sliding window is applied on a grid of values.
- The maximum is computed using the values in the current window.



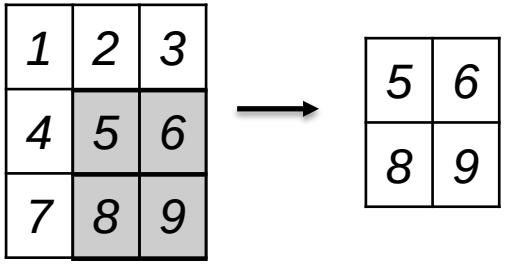
Max Pooling Layer:

- Sliding window is applied on a grid of values.
- The maximum is computed using the values in the current window.



Max Pooling Layer:

- Sliding window is applied on a grid of values.
- The maximum is computed using the values in the current window.



Sigmoid Layer:

- Applies a sigmoid function on an input

$$a^{(l)} = f(z^{(l)}) = \frac{1}{1 + \exp\left(-z^{(l)}\right)}$$

Convolutional Networks

Let us now consider a CNN with a specific architecture:

- 2 convolutional layers.
- 2 pooling layers.
- 2 fully connected layers.
- 3 sigmoid layers.

Notation:

- convolutional layer output

- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:

x

Notation:

- convolutional layer output

- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:

x

$z^{(1)}$

Notation:

- convolutional layer output

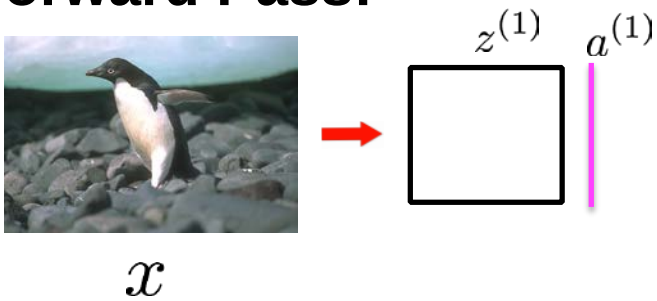
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

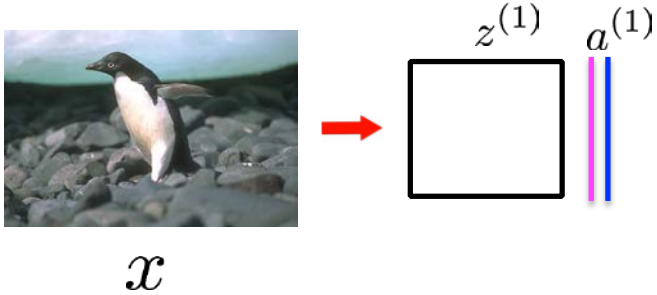
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

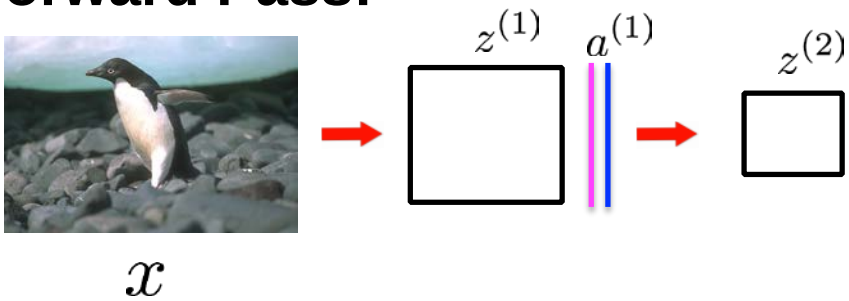
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

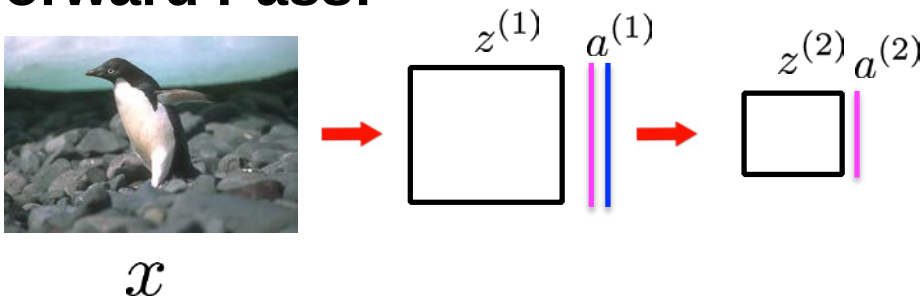
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

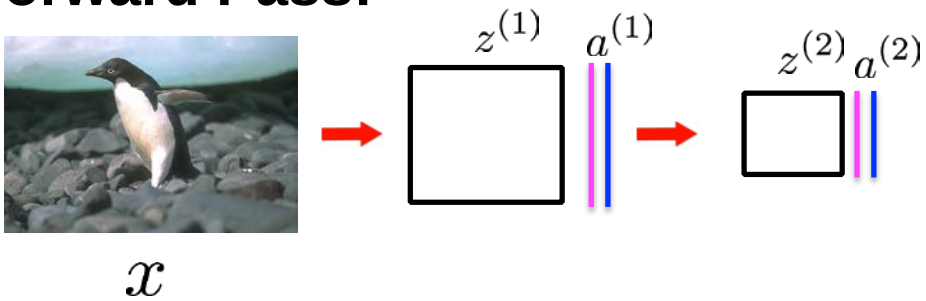
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

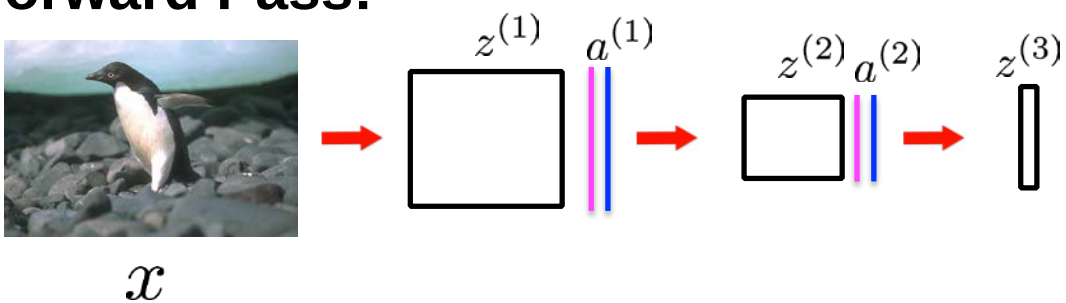
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

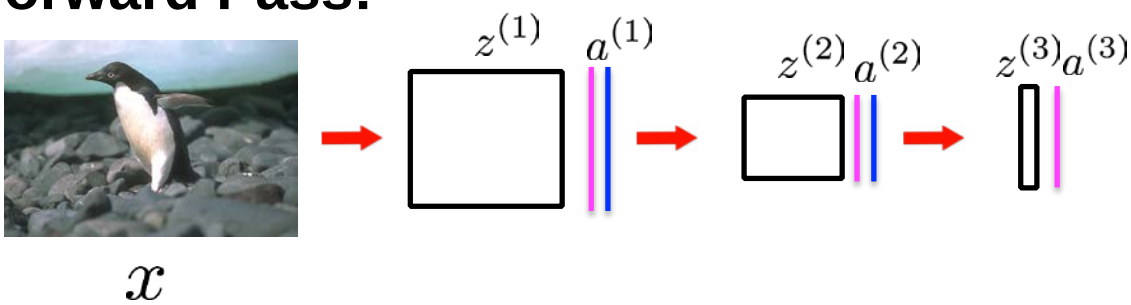
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Convolutional Networks

Notation:

- convolutional layer output

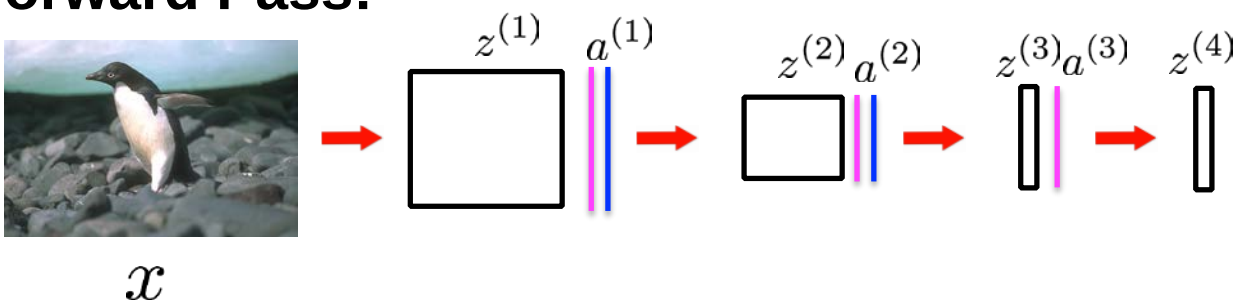
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

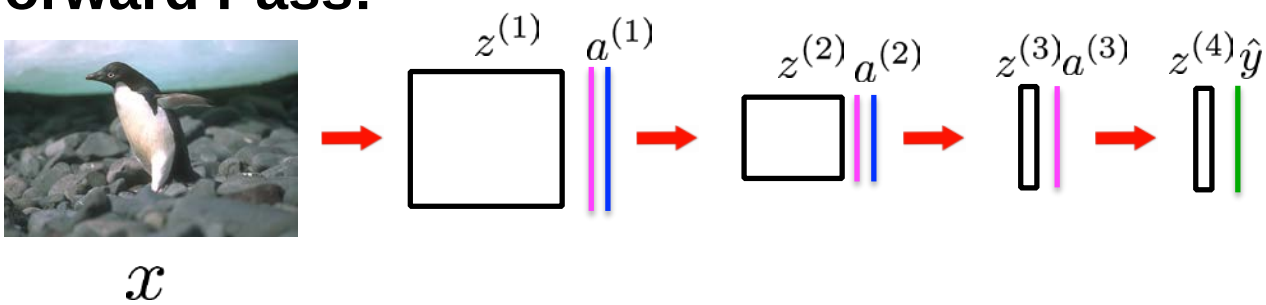
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

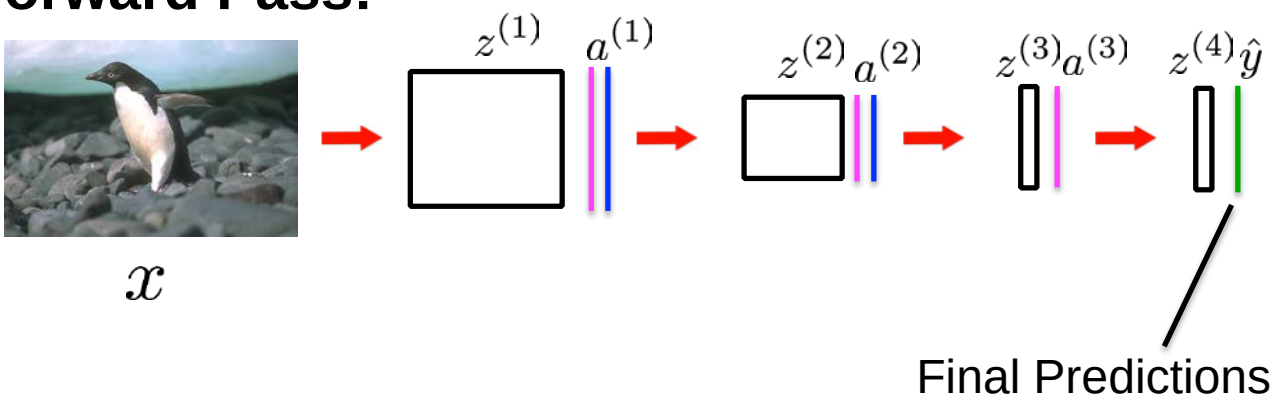
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

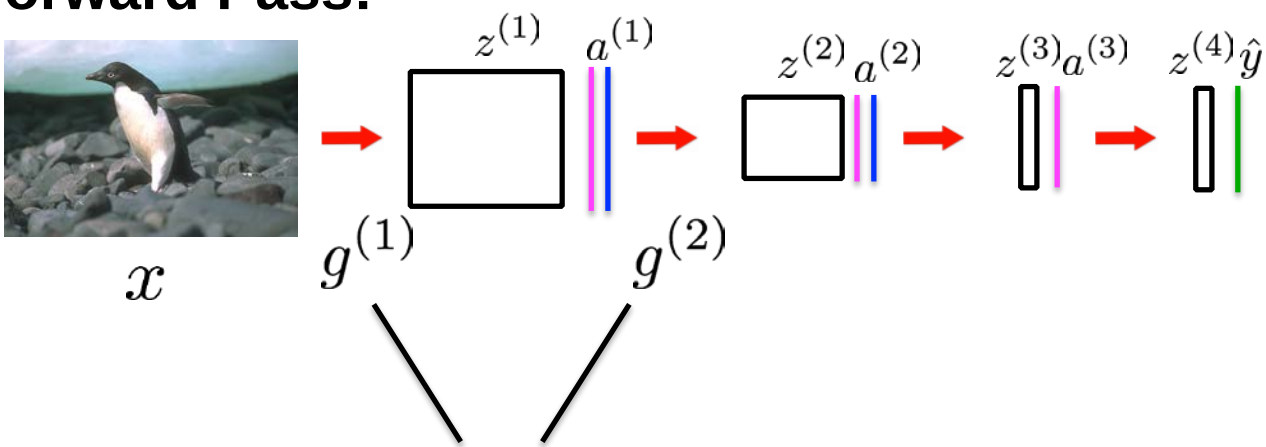
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Convolutional layer parameters in layers 1 and 2

Notation:

- convolutional layer output

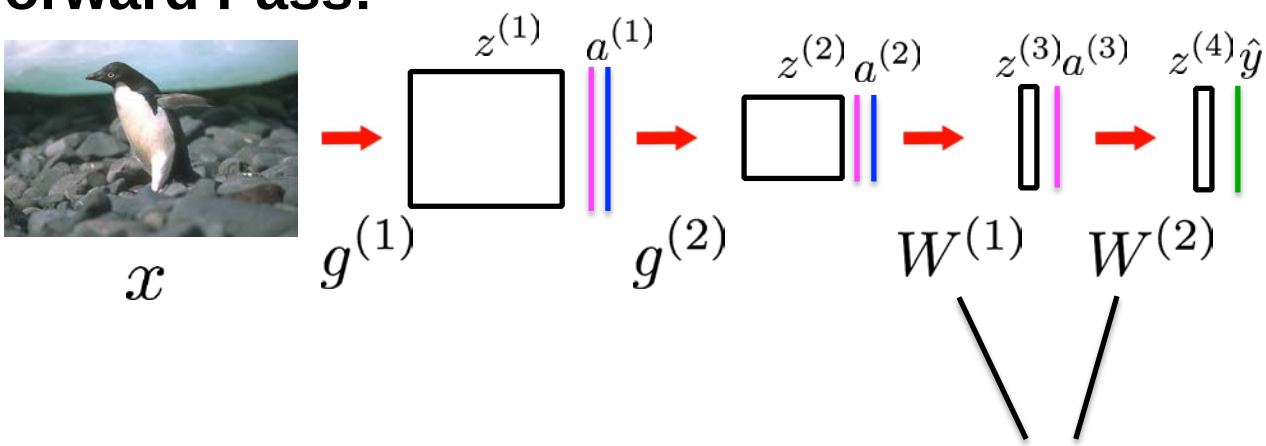
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Fully connected layer parameters in the fully connected layers 1 and 2

Notation:

- convolutional layer output

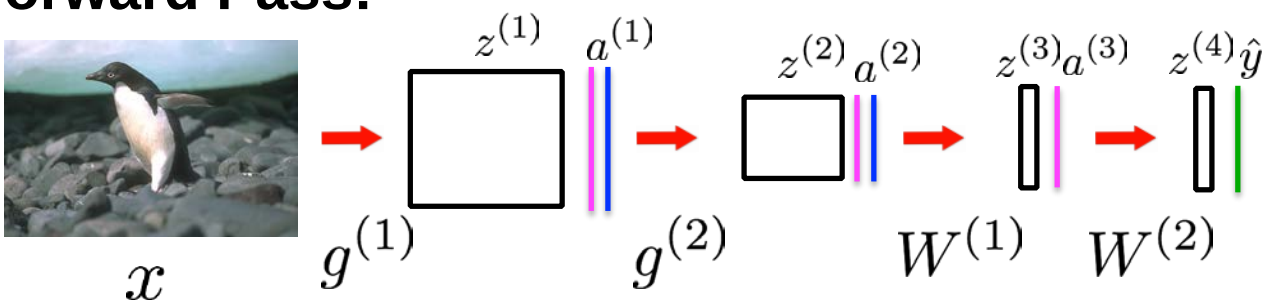
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:



Notation:

- convolutional layer output

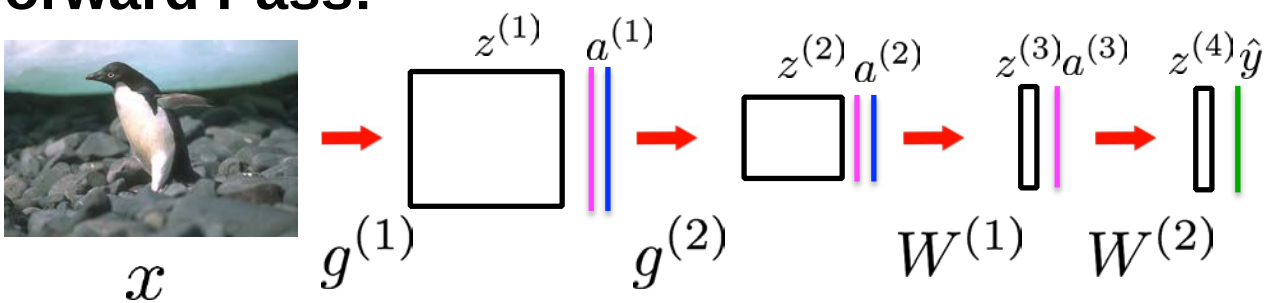
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:

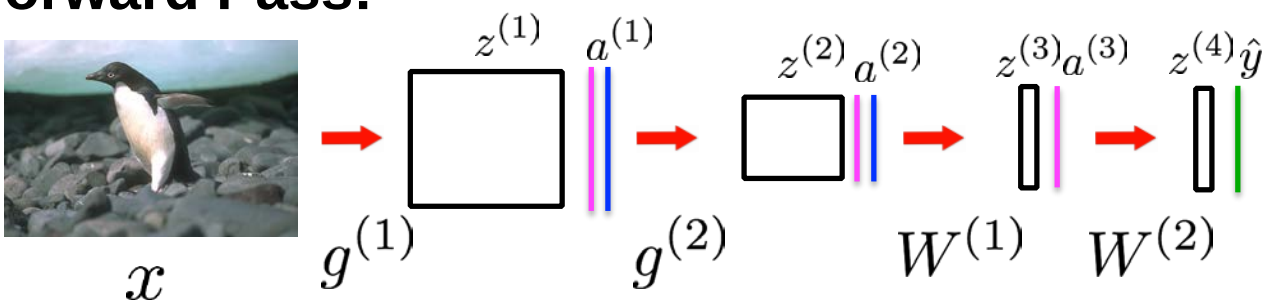


1. $$a^{(1)} = pool(f(g^{(1)} * x))$$

Notation:

- convolutional layer output
- fully connected layer output
- pooling layer
- sigmoid function f
- softmax function

Forward Pass:

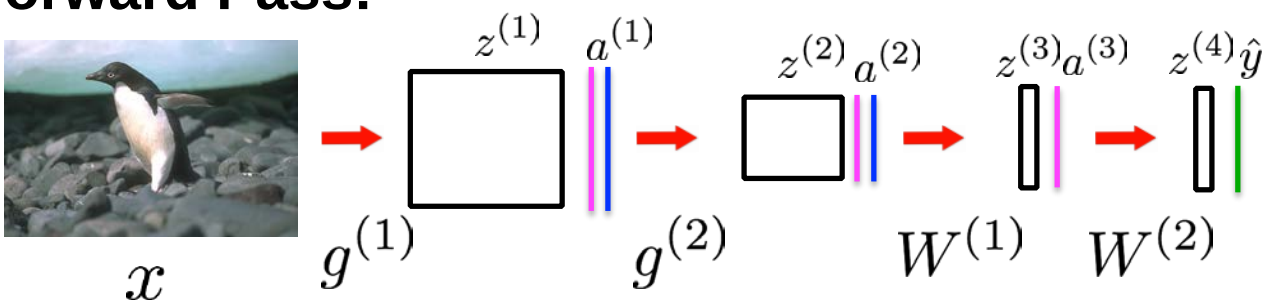


1. $a^{(1)} = pool(f(g^{(1)} * x))$
2. $a^{(2)} = pool(f(g^{(2)} * a^{(1)}))$

Notation:

- convolutional layer output
- fully connected layer output
- pooling layer
- sigmoid function f
- softmax function

Forward Pass:

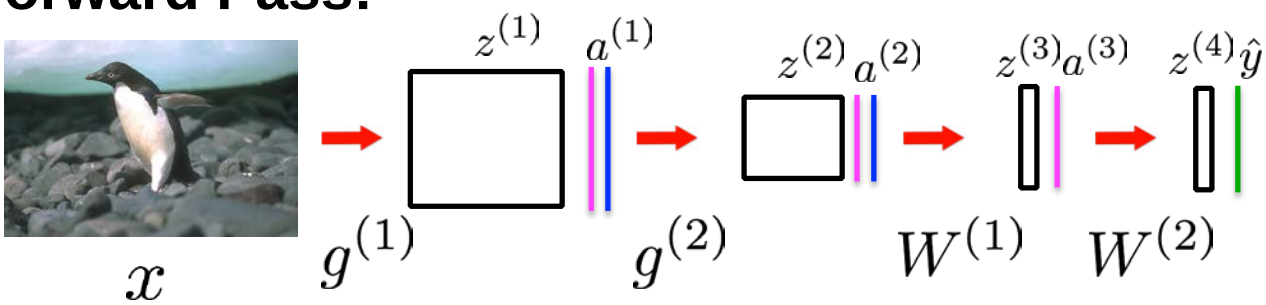


1. $a^{(1)} = pool(f(g^{(1)} * x))$
2. $a^{(2)} = pool(f(g^{(2)} * a^{(1)}))$
3. $a^{(3)} = f(W^{(1)} a^{(2)})$

Notation:

- convolutional layer output
- fully connected layer output
- pooling layer
- sigmoid function f
- softmax function

Forward Pass:



1. $a^{(1)} = pool(f(g^{(1)} * x))$
2. $a^{(2)} = pool(f(g^{(2)} * a^{(1)}))$
3. $a^{(3)} = f(W^{(1)} a^{(2)})$
4. $\hat{y} = softmax(W^{(2)} a^{(3)})$

Notation:

- convolutional layer output

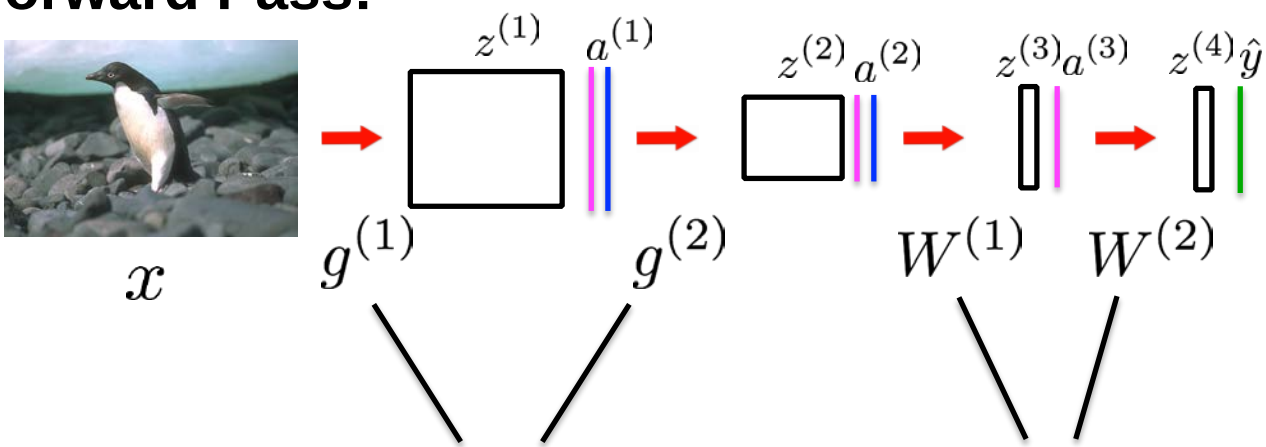
- fully connected layer output

- pooling layer

- sigmoid function f

- softmax function

Forward Pass:

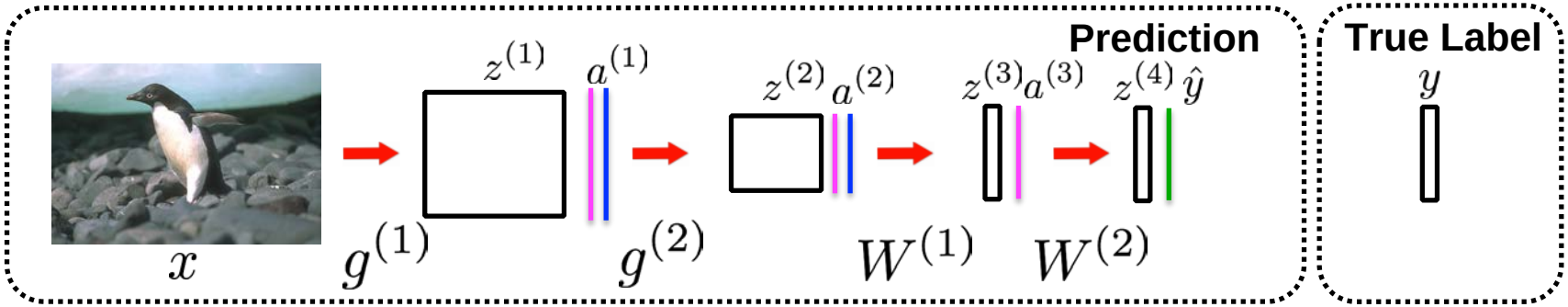


Key Question: How to learn the parameters from the data?

Backpropagation

for

Convolutional Neural Networks



How to learn the parameters of a CNN?

- Assume that we are a given a **labeled** training dataset

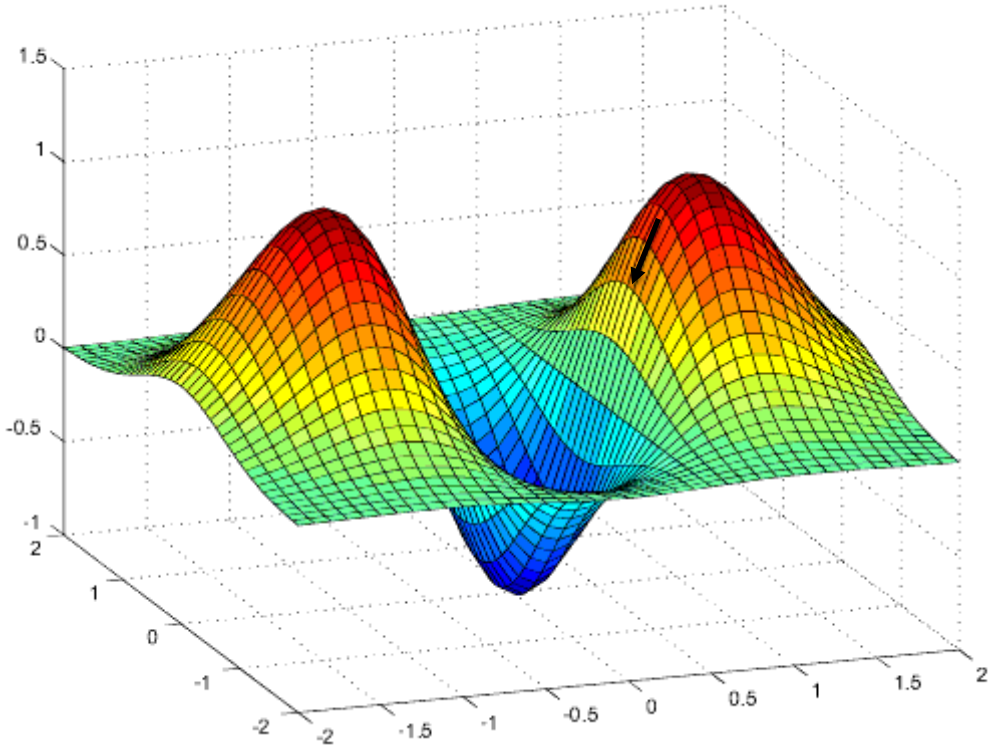
$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

- We want to adjust the parameters of a CNN such that CNN's predictions would be as close to true labels as possible.
- This is difficult to do because the learning objective is highly non-linear.

Gradient descent:

- Iteratively minimizes the objective function.
- The function needs to be differentiable.

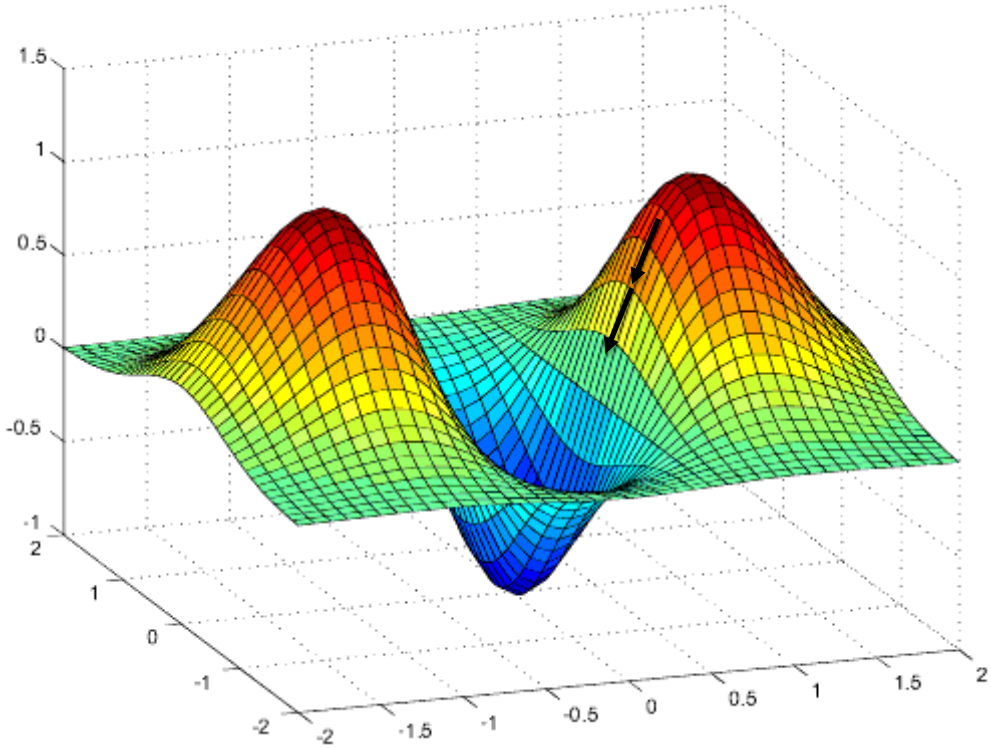
$$\theta = \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$



Gradient descent:

- Iteratively minimizes the objective function.
- The function needs to be differentiable.

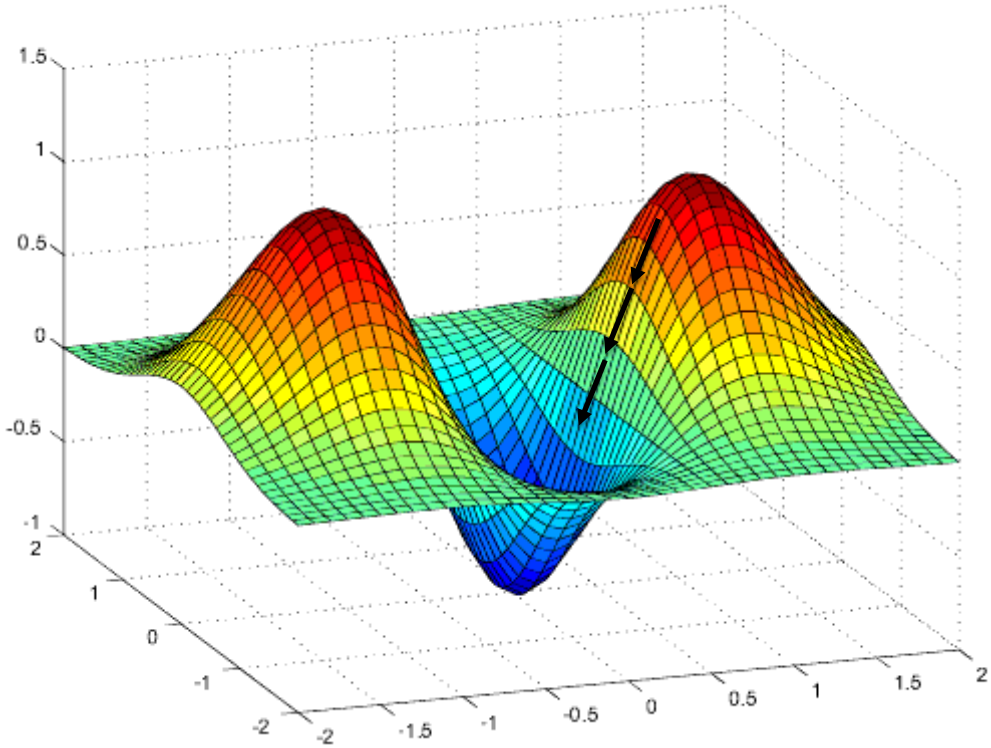
$$\theta = \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$



Gradient descent:

- Iteratively minimizes the objective function.
- The function needs to be differentiable.

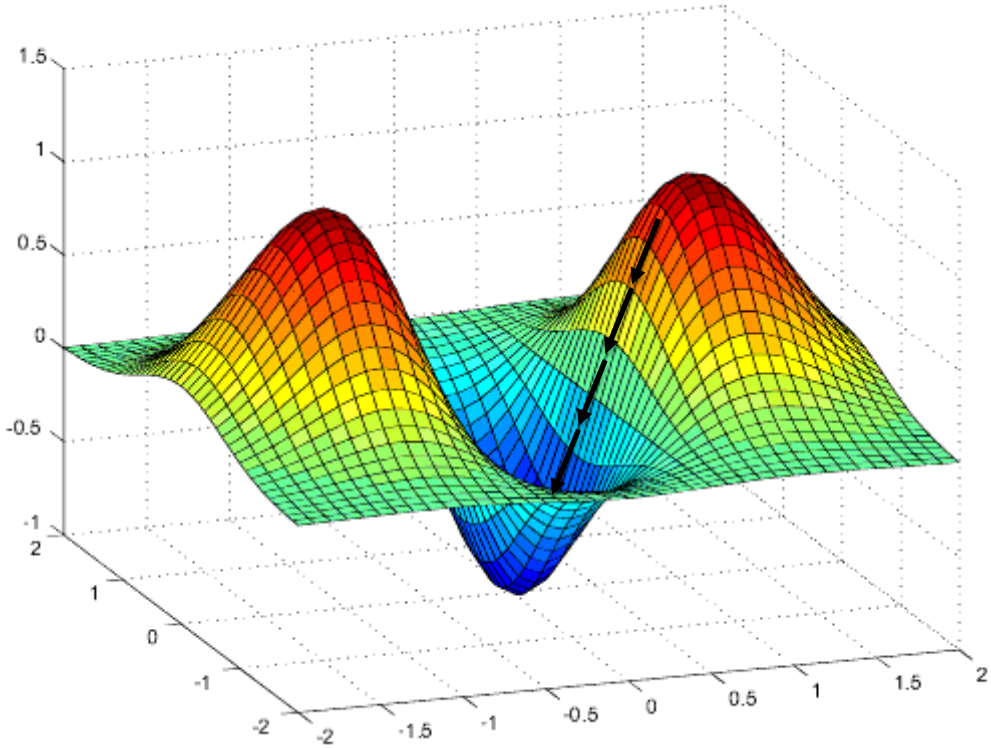
$$\theta = \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$



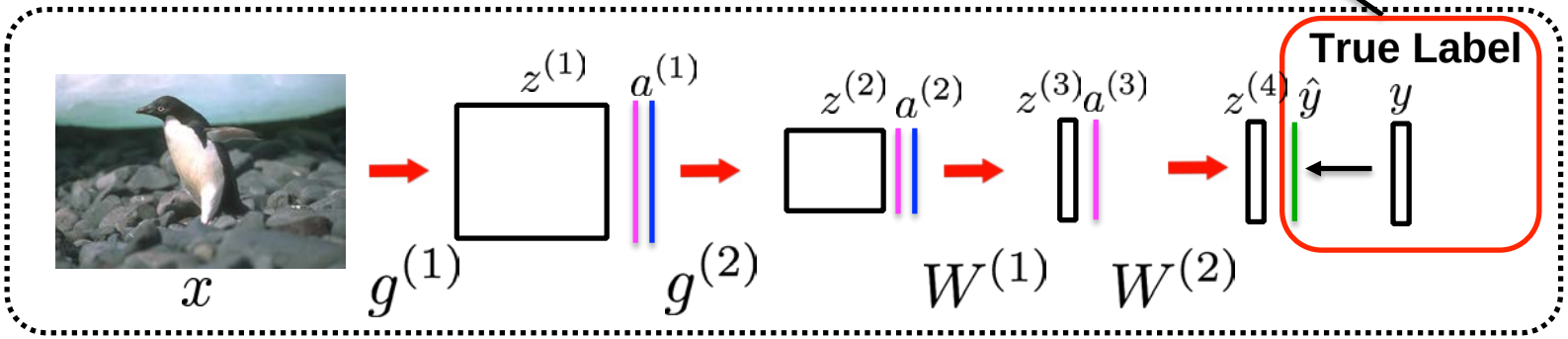
Gradient descent:

- Iteratively minimizes the objective function.
- The function needs to be differentiable.

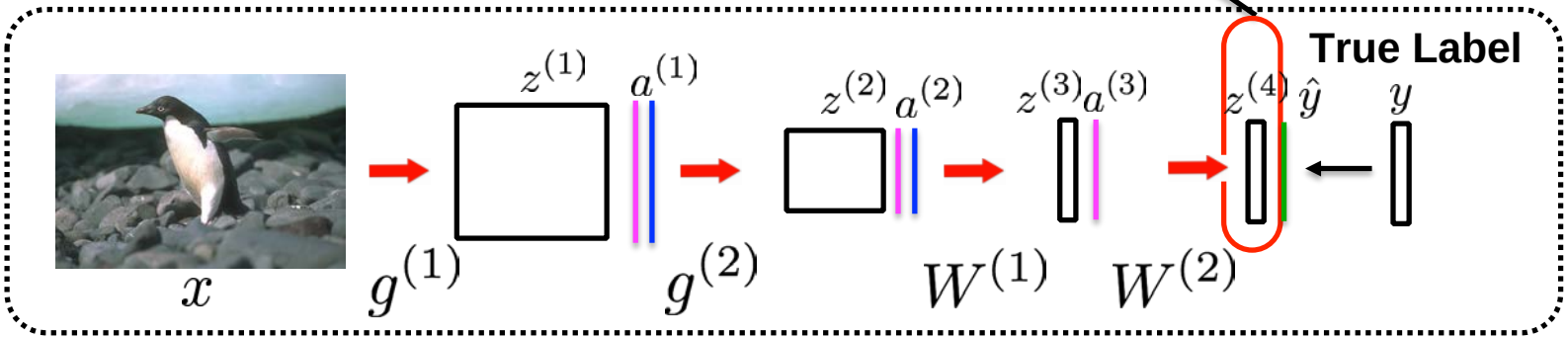
$$\theta = \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$

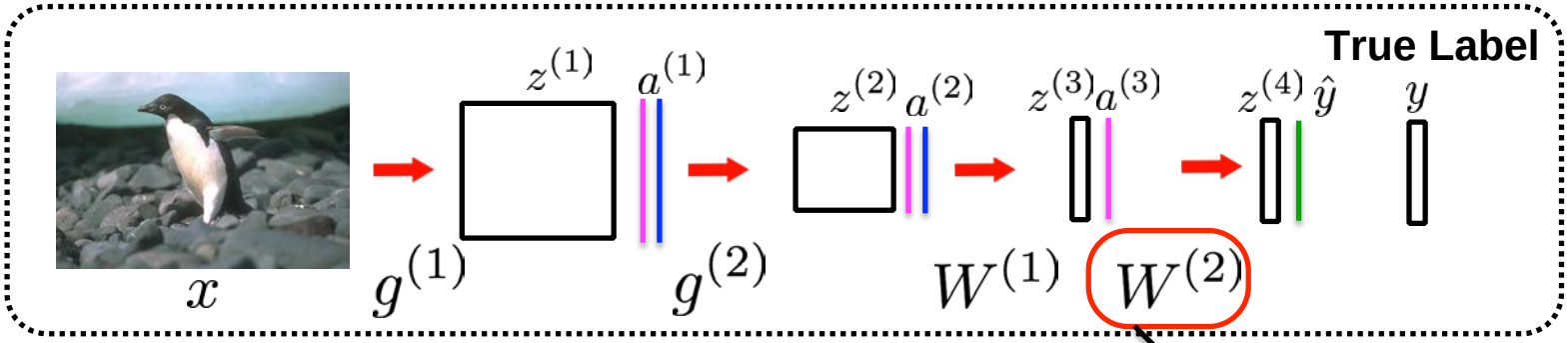


1. Compute the gradients of the overall loss w.r.t. to our predictions and propagate it back: $\frac{\partial L}{\partial \hat{y}}$



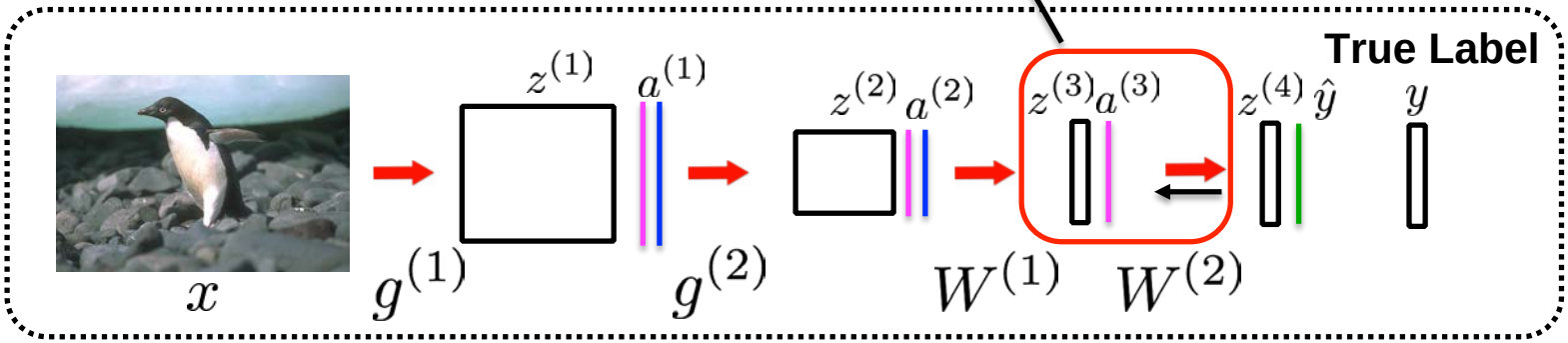
2. Compute the gradients of the overall loss and propagate it back: $\frac{\partial L}{\partial z^{(4)}}$

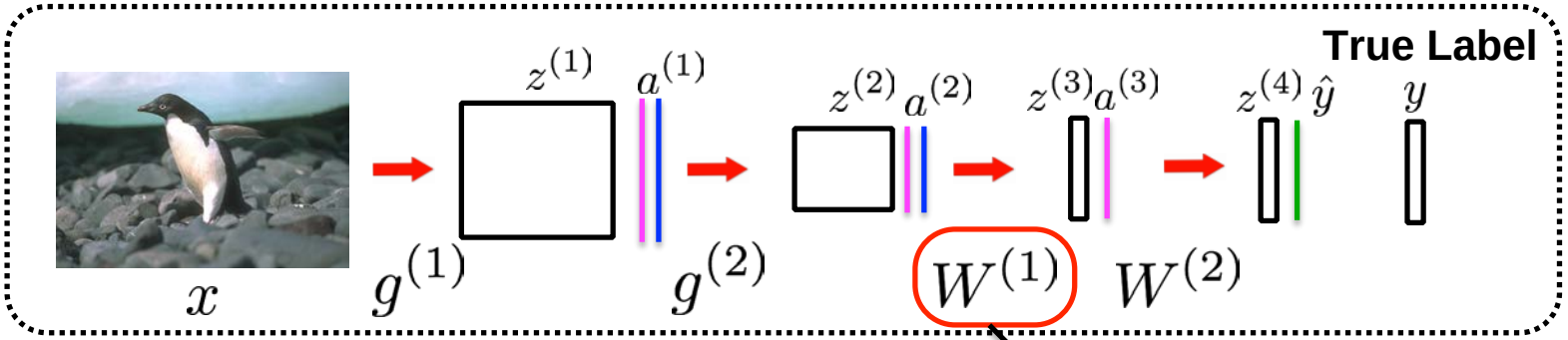




3. Compute the gradients to adjust the weights: $\dfrac{\partial L}{\partial W^{(2)}}$

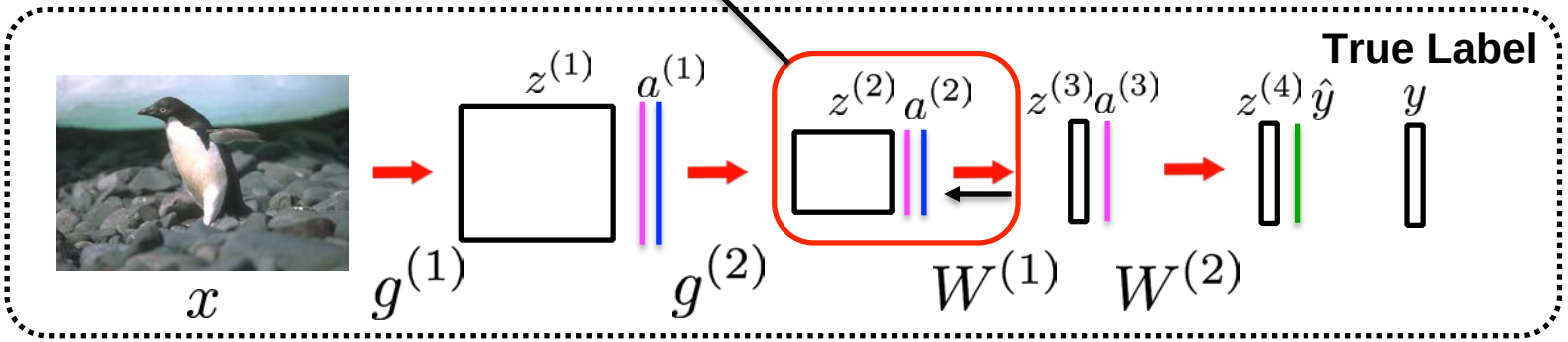
4. Backpropagate the gradients to previous layers: $\frac{\partial L}{\partial z^{(3)}}$

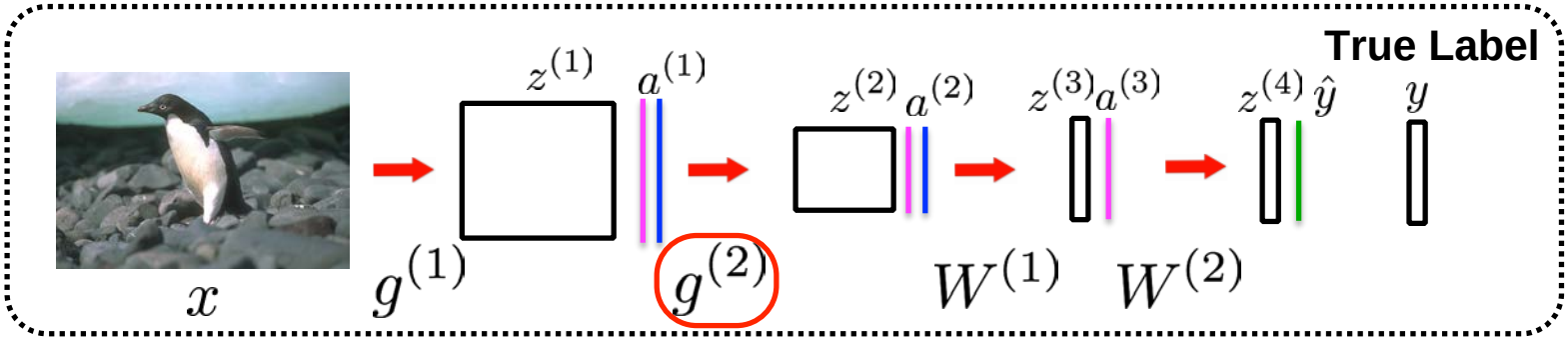




5. Compute the gradients to adjust the weights: $\dfrac{\partial L}{\partial W^{(1)}}$

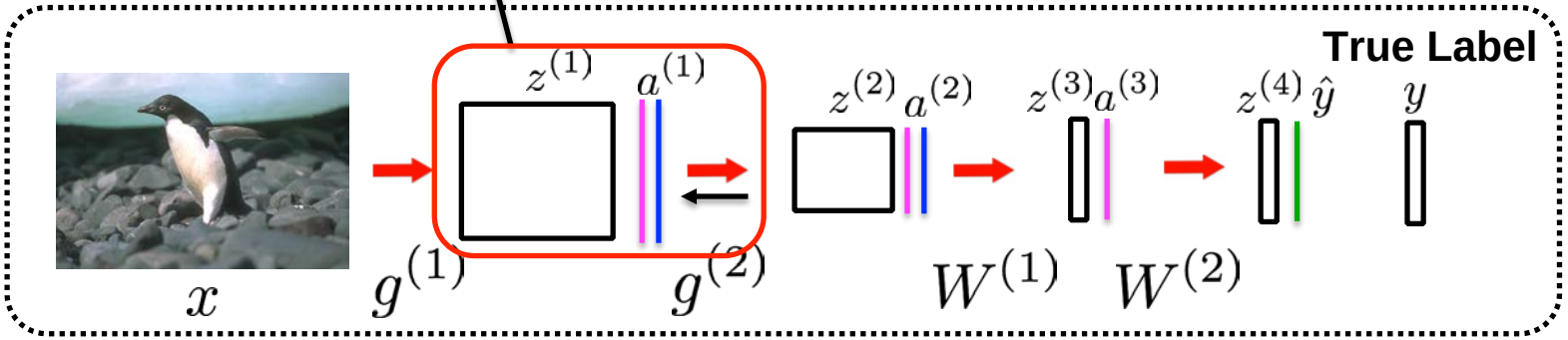
6. Backpropagate the gradients to previous layers: $\frac{\partial L}{\partial z^{(2)}}$

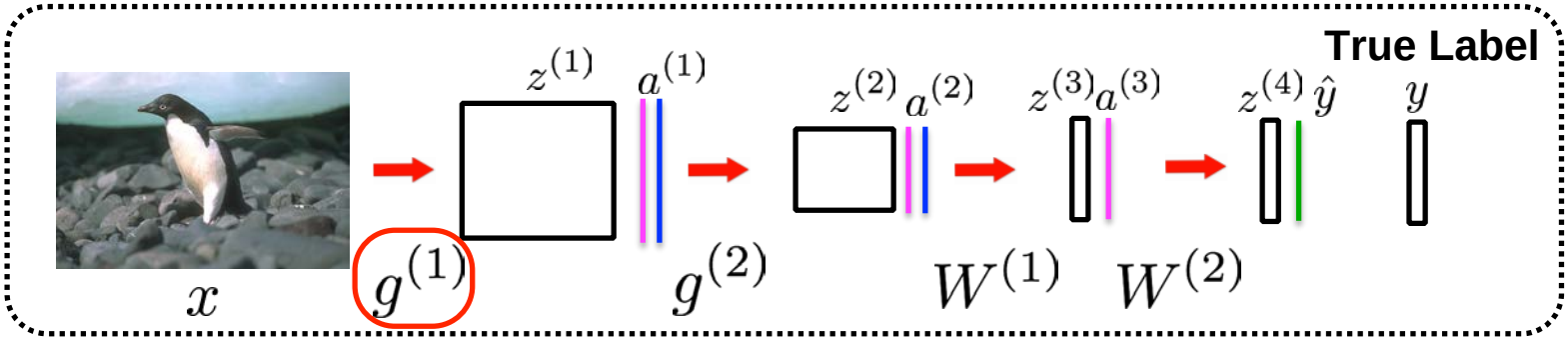




7. Compute the gradients to adjust the weights: $\dfrac{\partial L}{\partial g^{(2)}}$

8. Backpropagate the gradients to previous layers: $\frac{\partial L}{\partial z^{(1)}}$



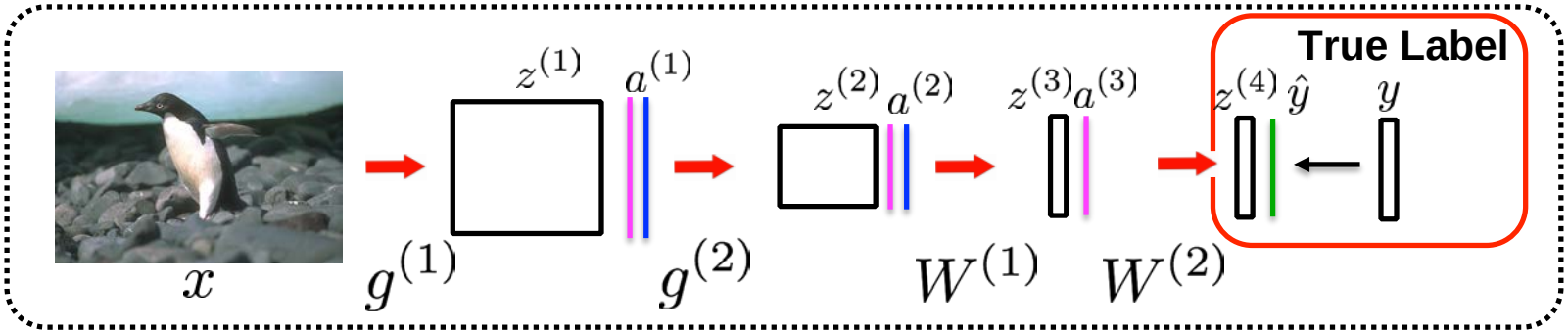


9. Compute the gradients $\frac{\partial L}{\partial g^{(1)}}$ to adjust the weights:

An Example of

Backpropagation

Convolutional Neural Networks

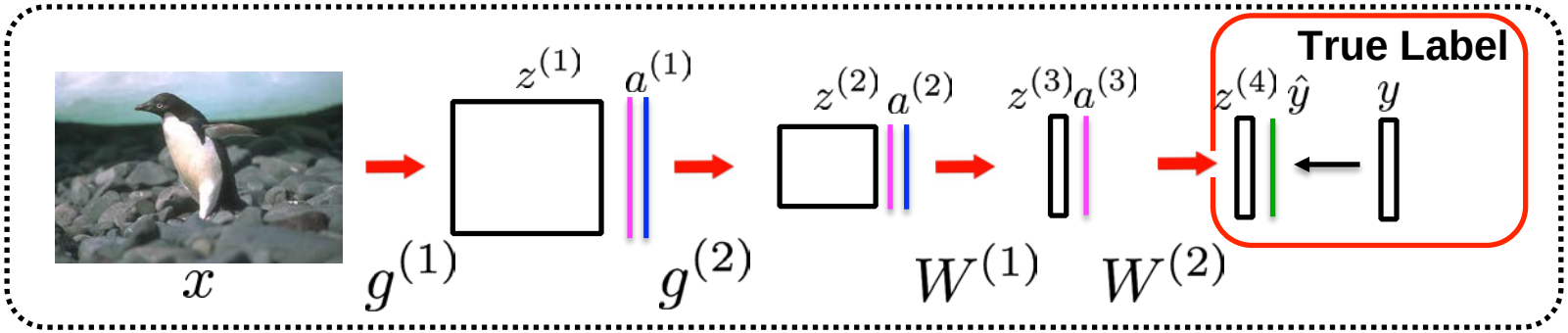


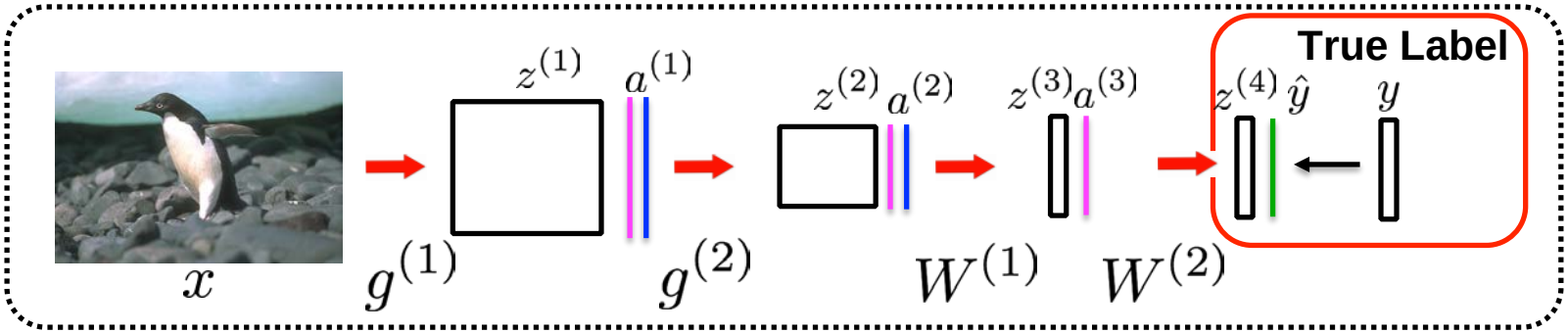
Assume that we have K=5 object classes:

Class 1: **Penguin**
Class 2: **Building**
Class 3: **Chair**
Class 4: **Person**
Class 5: **Bird**

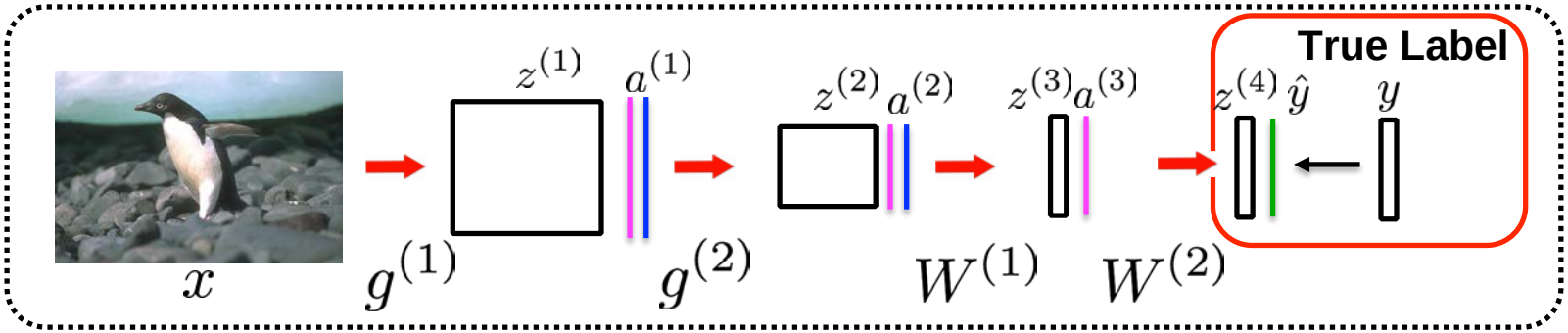
$$\hat{y} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$


True Label
x
$g^{(1)}$
$z^{(1)}$
$a^{(1)}$
$g^{(2)}$
$z^{(2)}$
$a^{(2)}$
$W^{(1)}$
$z^{(3)}$
$a^{(3)}$
$W^{(2)}$
$z^{(4)}$
$\hat{y}$
y

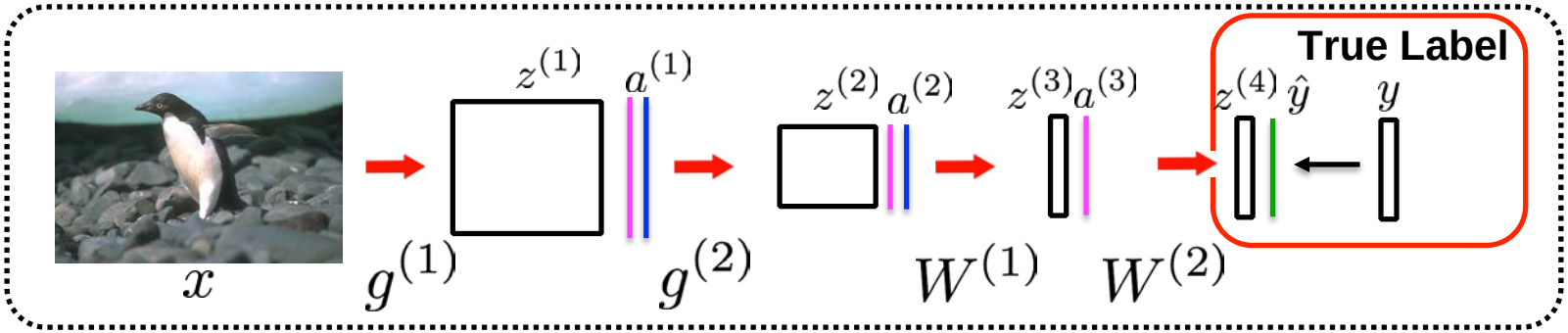


$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \quad \text{where} \quad \hat{y}_i = \frac{\exp(z_i^{(4)})}{\sum_{j=1}^{K} \exp(z_j^{(4)})}$$



$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \quad \text{where} \quad \hat{y}_i = \frac{\exp(z_i^{(4)})}{\sum_{j=1}^{K} \exp(z_j^{(4)})}$$

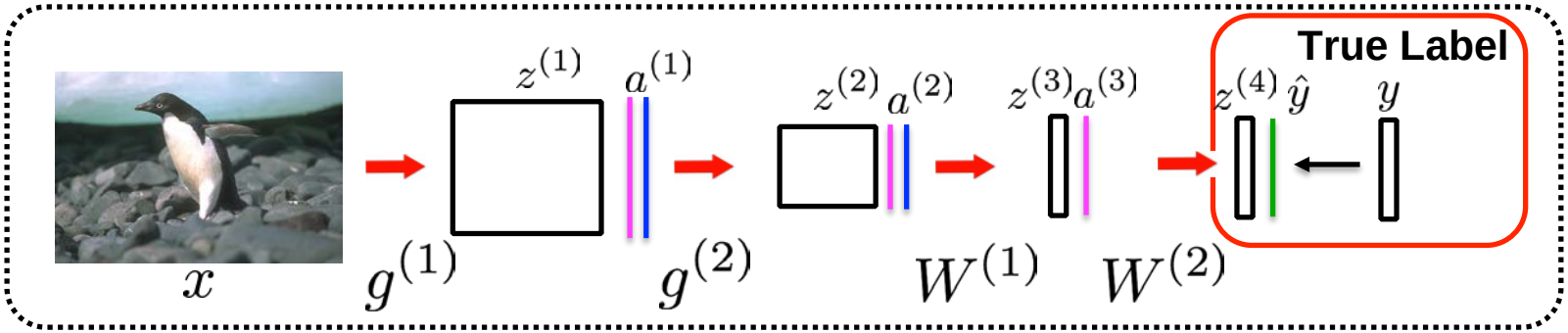
$$\frac{\partial L}{\partial z_i^{(4)}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$



$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \quad \text{where} \quad \hat{y}_i = \frac{\exp(z_i^{(4)})}{\sum_{j=1}^{K} \exp(z_j^{(4)})}$$

$$\frac{\partial L}{\partial z_i^{(4)}} = \boxed{\frac{\partial L}{\partial \hat{y}}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$

$$\boxed{\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}}$$

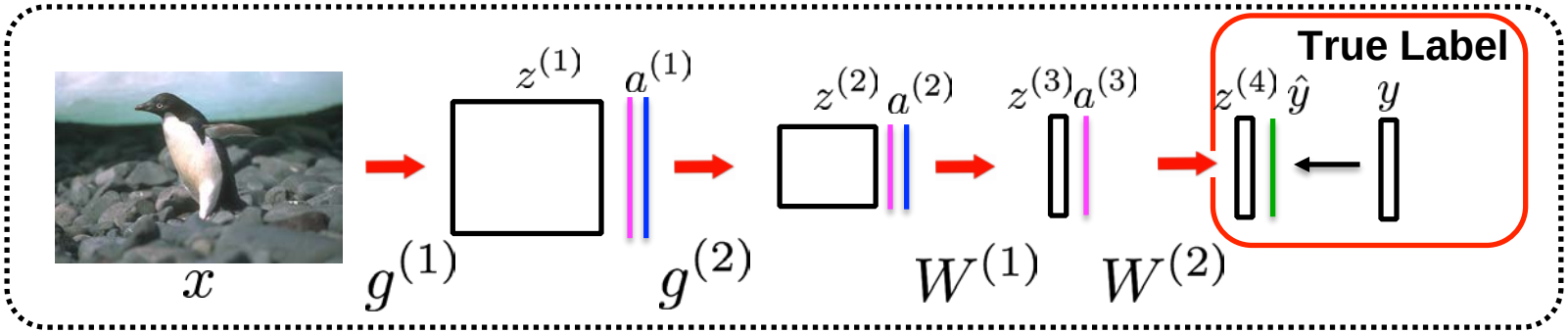


$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \quad \text{where} \quad \hat{y}_i = \frac{\exp(z_i^{(4)})}{\sum_{j=1}^{K} \exp(z_j^{(4)})}$$

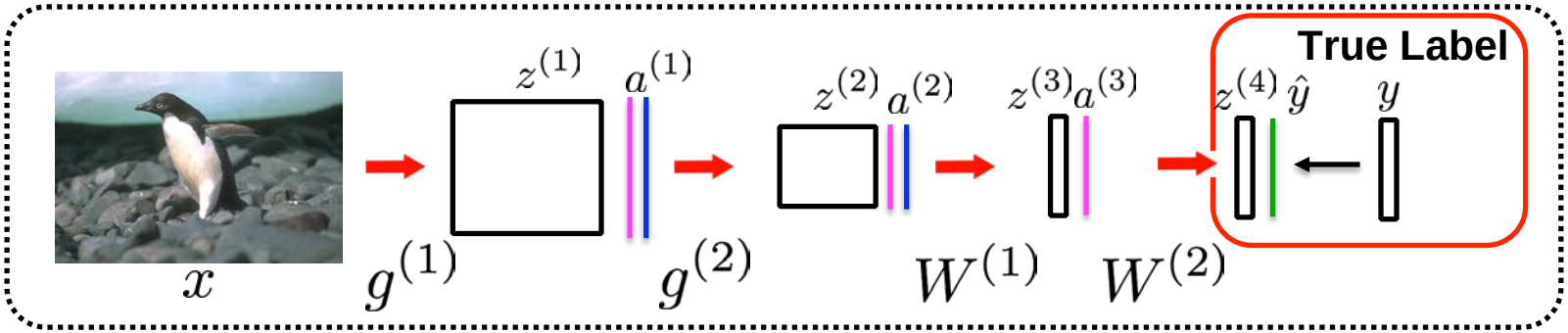
$$\frac{\partial L}{\partial z_i^{(4)}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

$$\frac{\partial \hat{y}_i}{\partial z_j^{(4)}} = \begin{cases} \hat{y}_i(1-\hat{y}_i), & \text{if} \quad i = j \\ -\hat{y}_i \hat{y}_j, & \text{if} \quad i \neq j \end{cases}$$

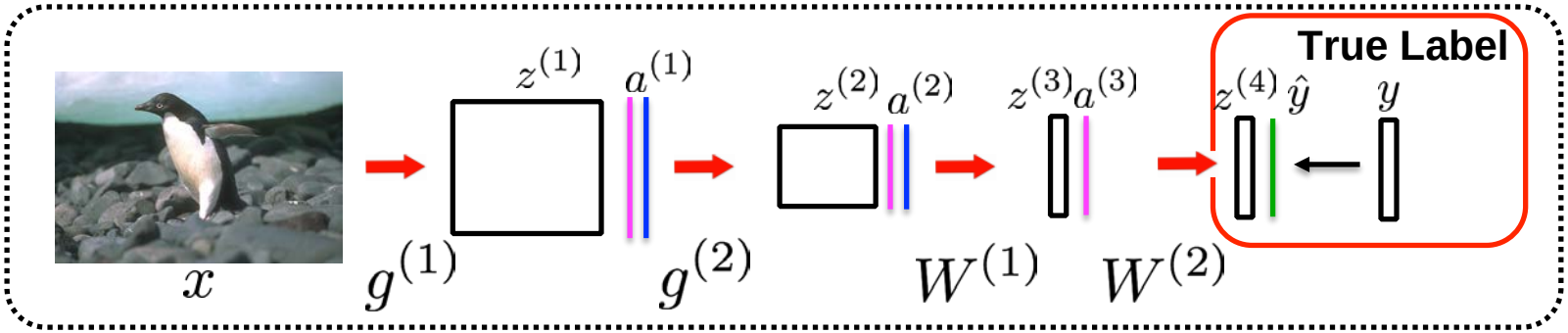


$$\frac{\partial L}{\partial z_i^{(4)}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$



$$\frac{\partial L}{\partial z_i^{(4)}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$

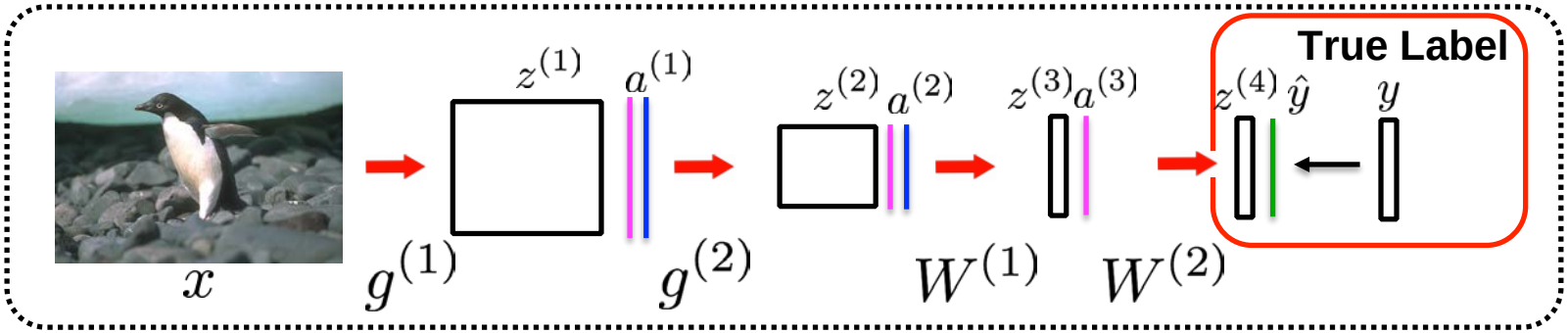
$$= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i^{(4)}} + \sum_{i \neq j} \frac{\partial L}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i^{(4)}}$$



$$\frac{\partial L}{\partial z_i^{(4)}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_i^{(4)}}$$

$$= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i^{(4)}} + \sum_{i \neq j} \frac{\partial L}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i^{(4)}}$$

$$\boxed{= \hat{y}_i - y_i}$$

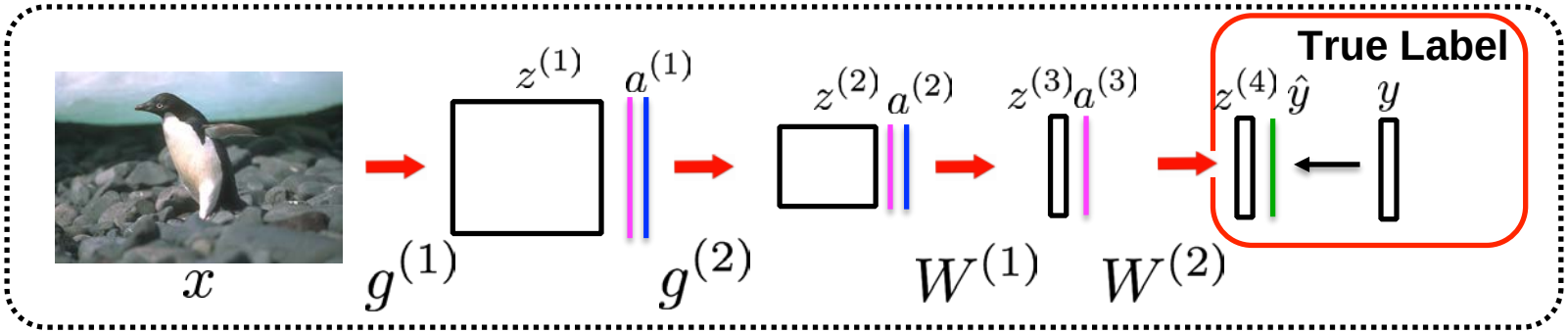


Assume that we have K=5 object classes:

Class 1: **Penguin**
Class 2: **Building**
Class 3: **Chair**
Class 4: **Person**
Class 5: **Bird**

$$\hat{y} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$



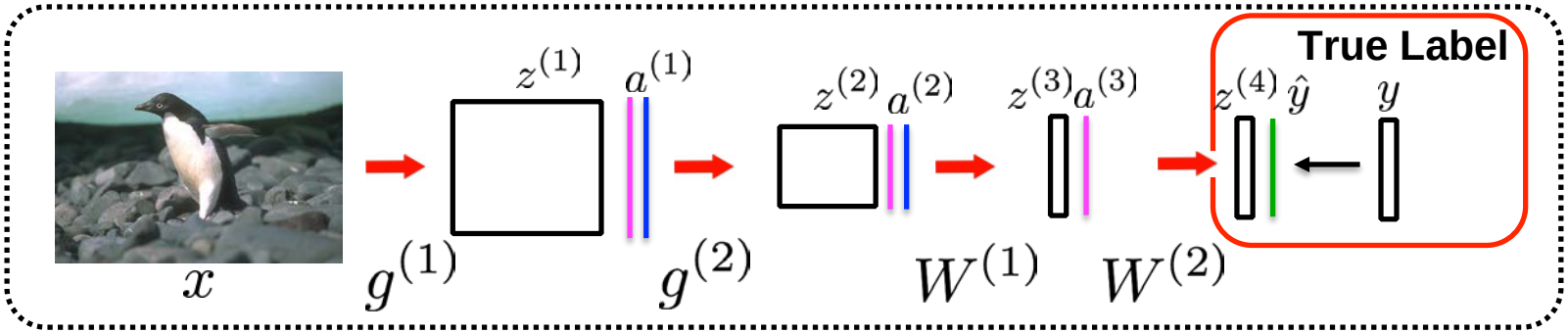
Assume that we have K=5 object classes:

Class 1: **Penguin**
Class 2: **Building**
Class 3: **Chair**
Class 4: **Person**
Class 5: **Bird**

$$\hat{y} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial z^{(4)}} = \begin{bmatrix} -0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$



Assume that we have K=5 object classes:

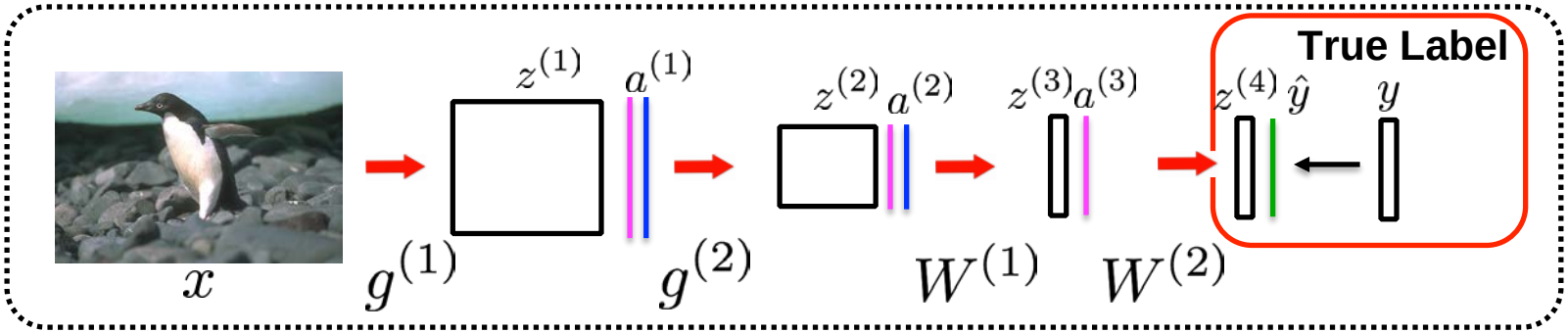
Class 1: **Penguin**
Class 2: **Building**
Class 3: **Chair**
Class 4: **Person**
Class 5: **Bird**

$$\hat{y} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial z^{(4)}} = \begin{bmatrix} -0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

Increasing the score corresponding to the true class decreases the loss.



Assume that we have K=5 object classes:

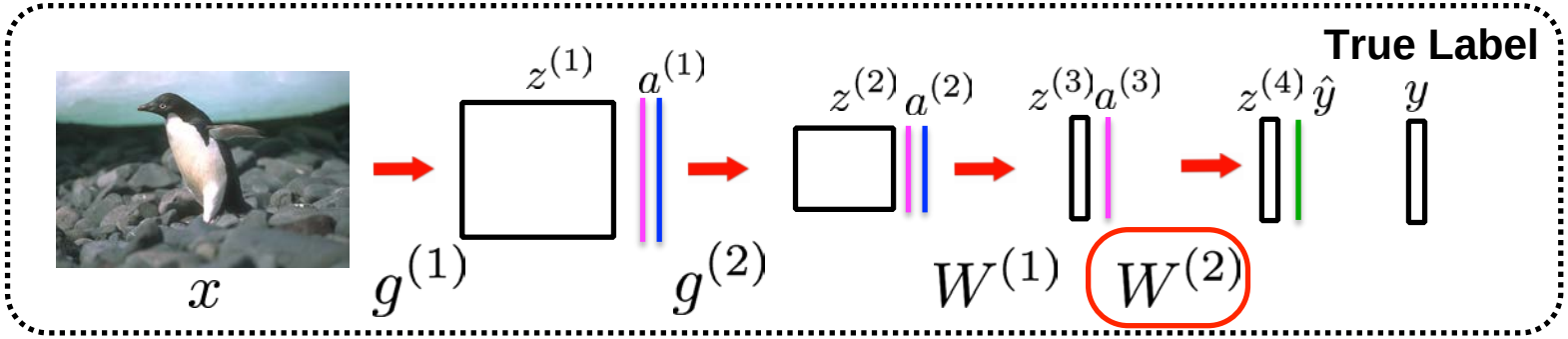
Class 1: **Penguin**
Class 2: **Building**
Class 3: **Chair**
Class 4: **Person**
Class 5: **Bird**

$$\hat{y} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

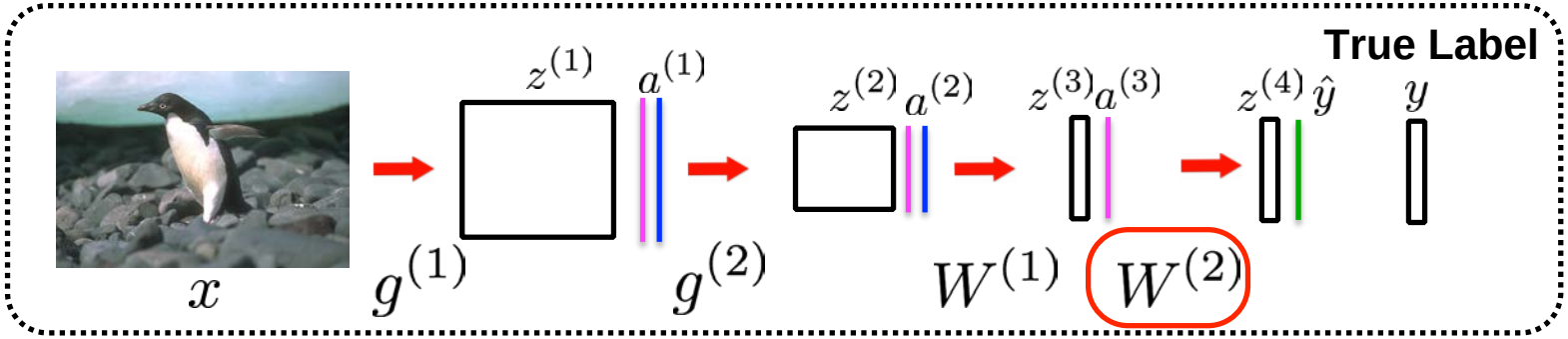
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial z^{(4)}} = \begin{bmatrix} -0.5 & 0 & 0.1 & 0.2 & 0.1 \end{bmatrix}$$

Decreasing the score of other classes also decreases the loss.

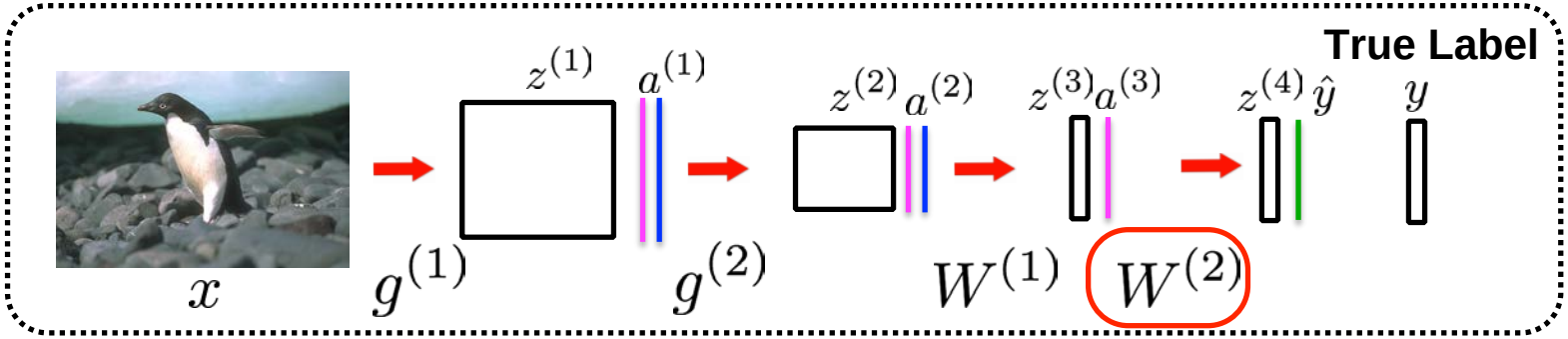


Adjusting the weights:



Adjusting the weights:

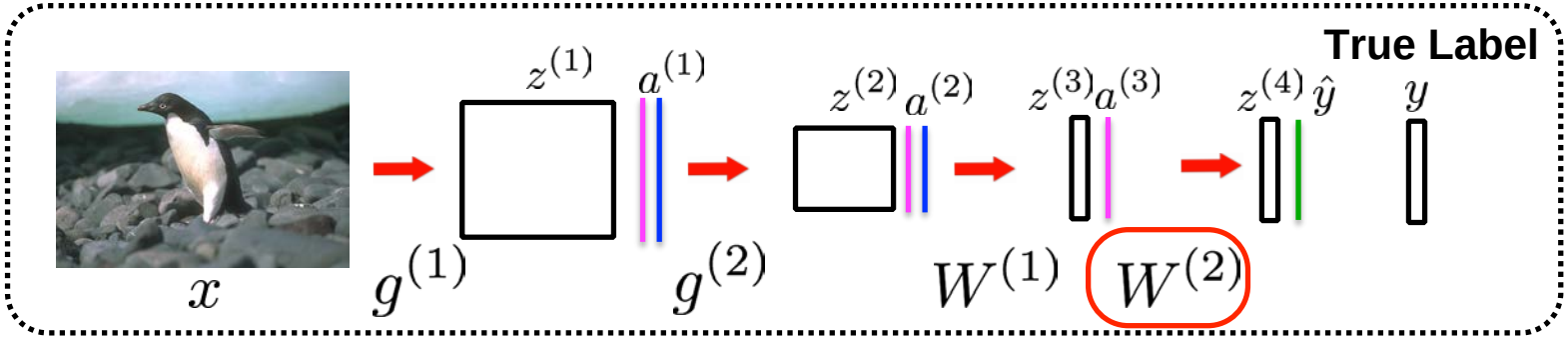
Need to compute the following gradient $$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial W^{(2)}}$$



Adjusting the weights:

Need to compute the following gradient

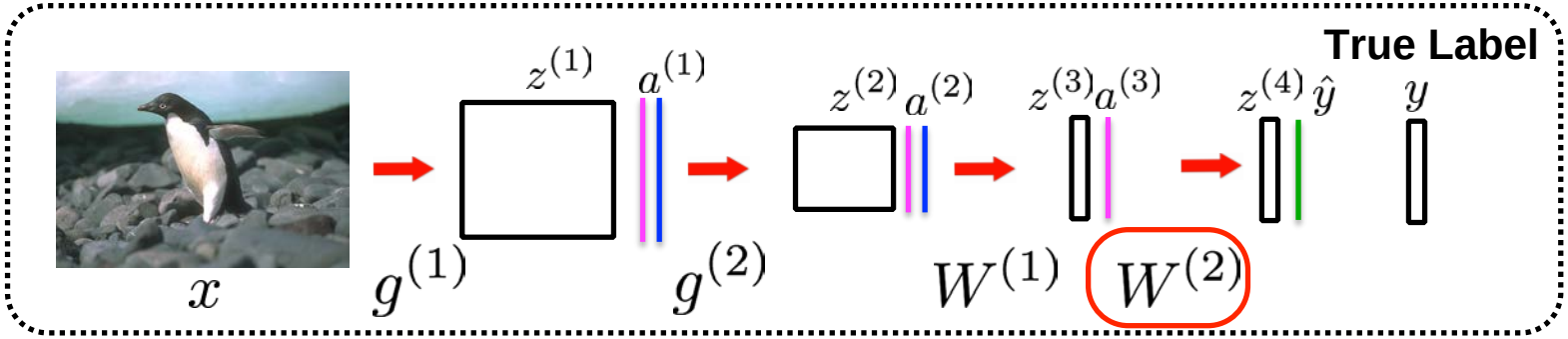
$$\frac{\partial L}{\partial W^{(2)}} = \boxed{\frac{\partial L}{\partial z^{(4)}}} \frac{\partial z^{(4)}}{\partial W^{(2)}}$$



Adjusting the weights:

Need to compute the following gradient $$\frac{\partial L}{\partial W^{(2)}} = \boxed{\frac{\partial L}{\partial z^{(4)}}} \frac{\partial z^{(4)}}{\partial W^{(2)}}$$

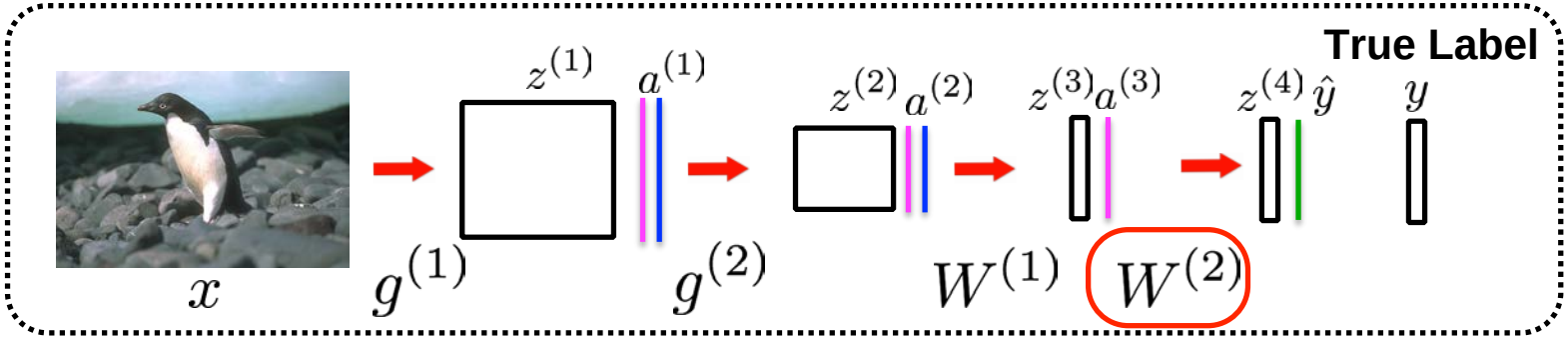
$\boxed{\frac{\partial L}{\partial z^{(4)}}}$ was already computed in the previous step



Adjusting the weights:

Need to compute the following gradient

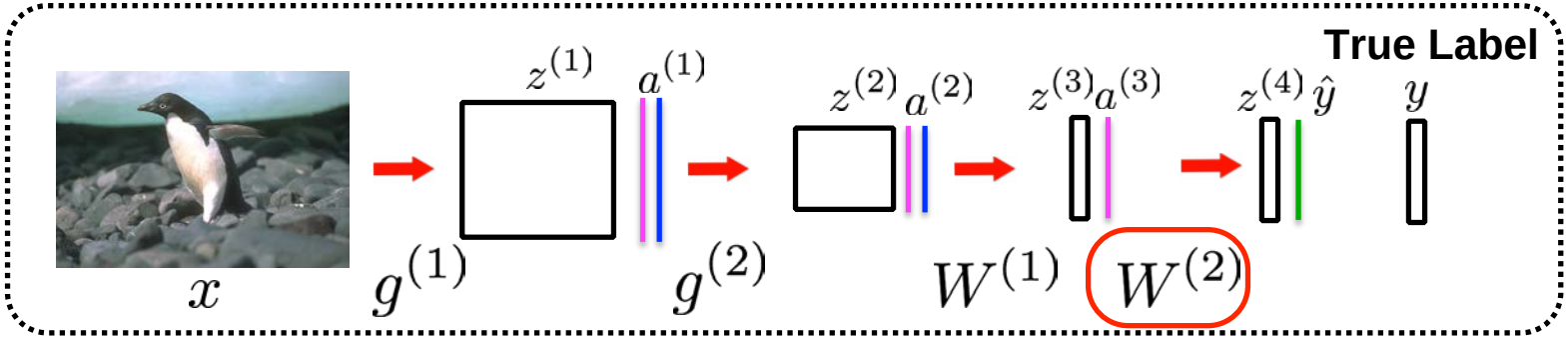
$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial W^{(2)}}}$$



Adjusting the weights:

Need to compute the following gradient $\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial W^{(2)}}}$

$$z_i^{(4)} = \sum_{k=1}^{N} W_{ik}^{(2)} f(z_k^{(3)})$$ where $f(z^{(3)}) = a^{(3)}$

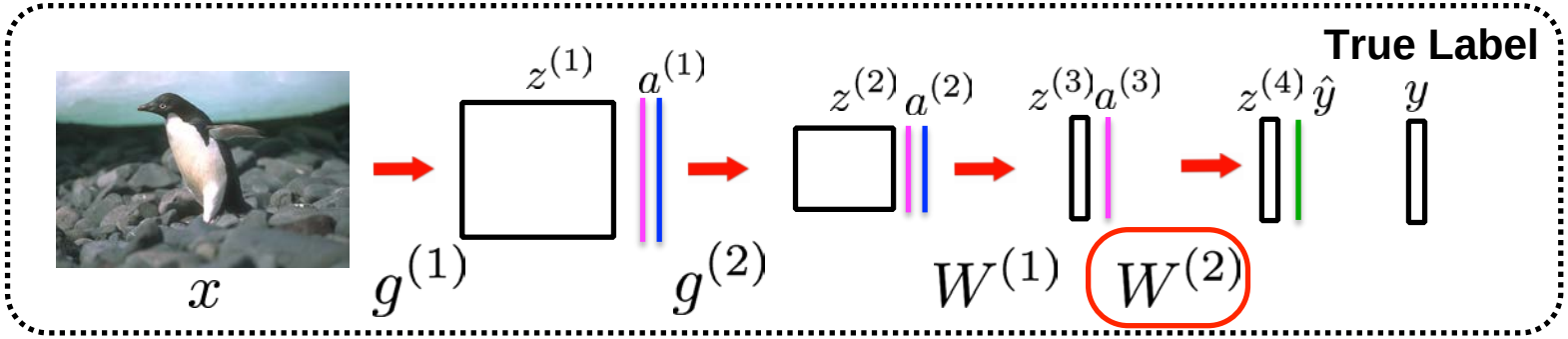


Adjusting the weights:

Need to compute the following gradient $\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial W^{(2)}}}$

$$z_i^{(4)} = \sum_{k=1}^{N} W_{ik}^{(2)} f(z_k^{(3)}) \quad \text{where} \quad f(z^{(3)}) = a^{(3)}$$

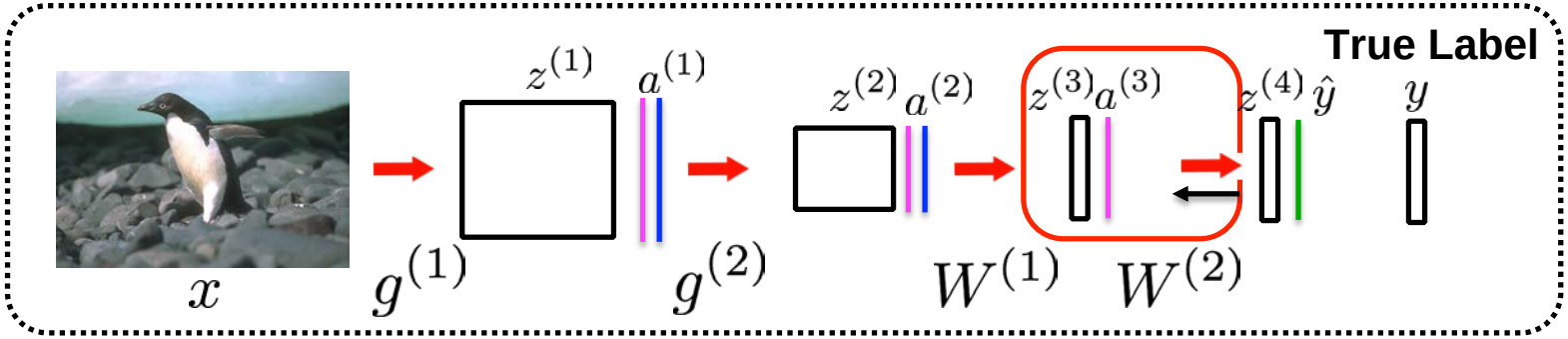
$$\boxed{\frac{\partial z_i^{(4)}}{\partial W_{ij}^{(2)}} = f(z_j^{(3)})}$$



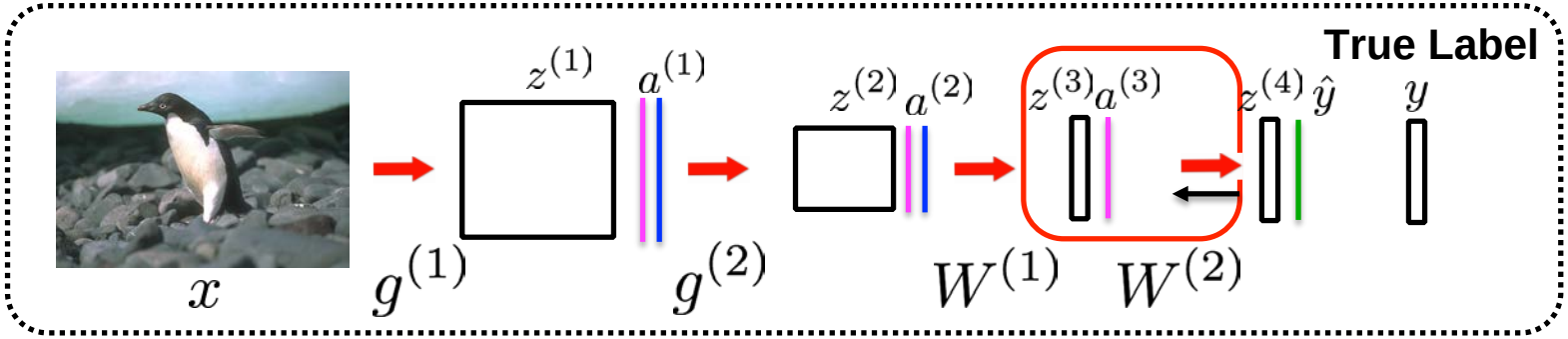
Adjusting the weights:

Need to compute the following gradient $$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial W^{(2)}}$$

Update rule: $$W_{ij}^{(2)} = W_{ij}^{(2)} - \alpha \frac{\partial L}{\partial W_{ij}^{(2)}}$$

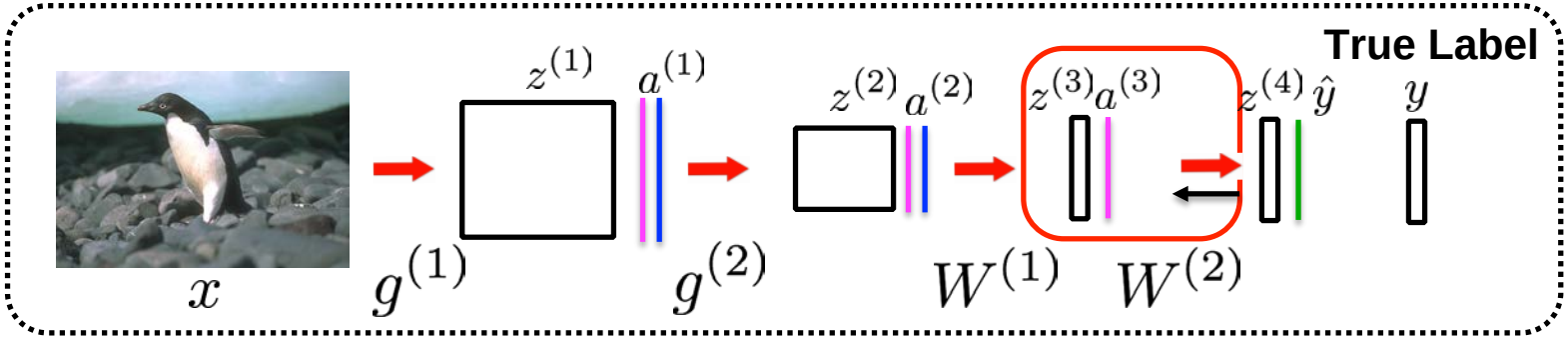


Backpropagating the gradients:



Backpropagating the gradients:

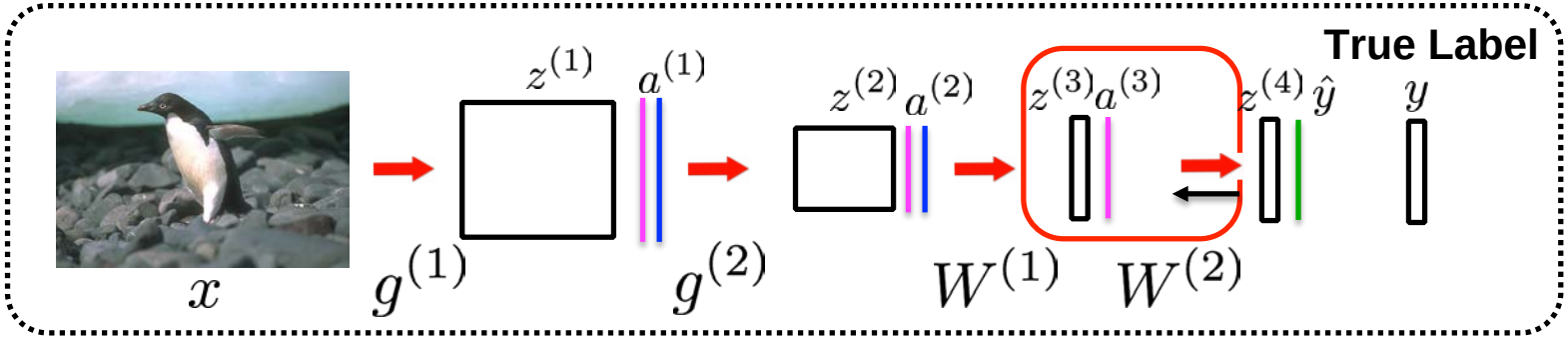
Need to compute the following gradient: $$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$



Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(3)}} = \boxed{\frac{\partial L}{\partial z^{(4)}}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$

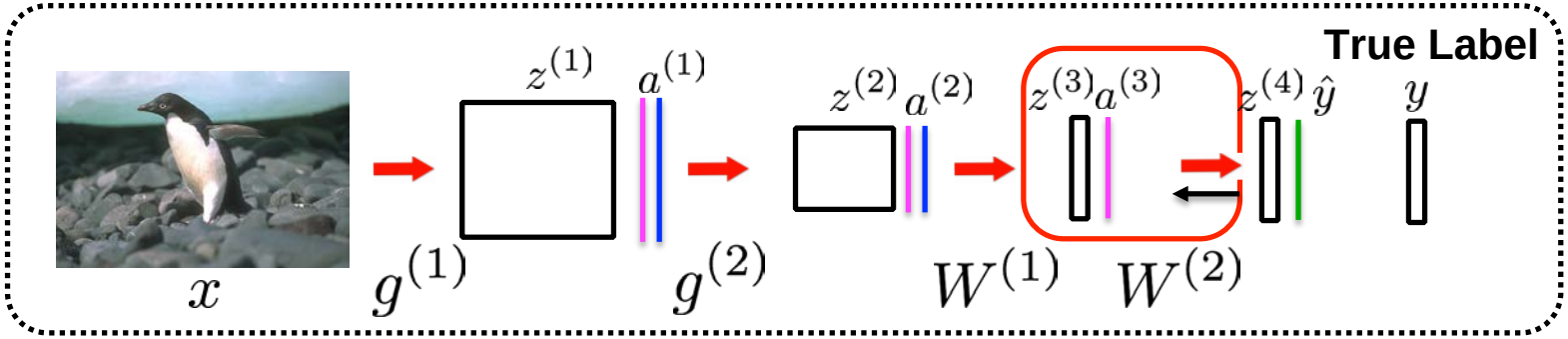


Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(3)}} = \boxed{\frac{\partial L}{\partial z^{(4)}}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$

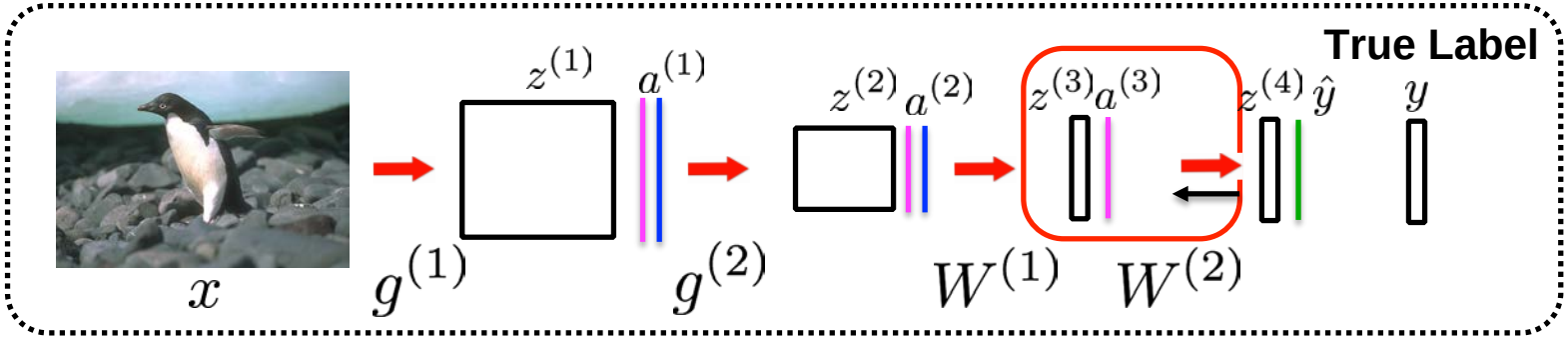
$\boxed{\frac{\partial L}{\partial z^{(4)}}}$ was already computed in the previous step



Backpropagating the gradients:

Need to compute the following gradient:

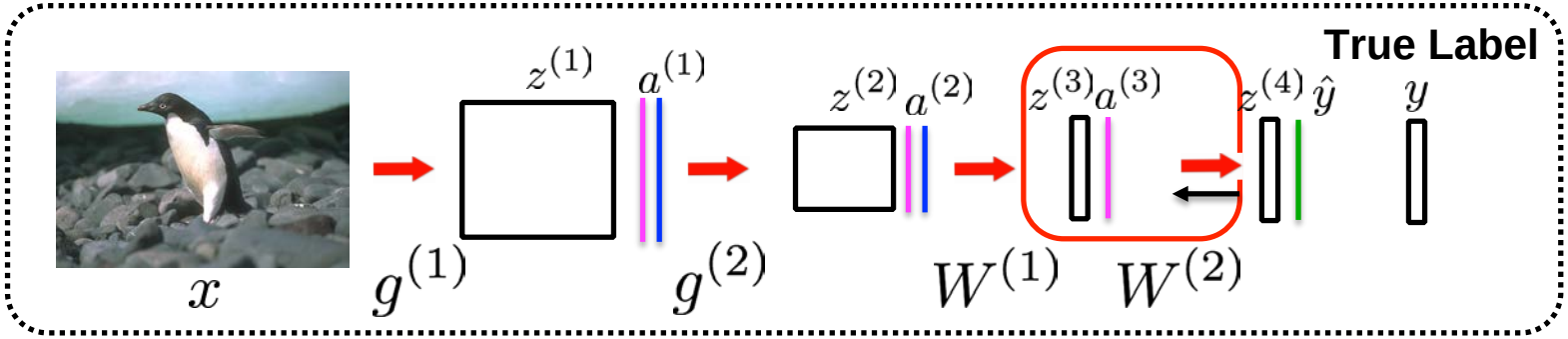
$$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial f(z^{(3)})}} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$



Backpropagating the gradients:

Need to compute the following gradient: $$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial f(z^{(3)})}} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$

$$z_i^{(4)} = \sum_{k=1}^{N} W_{ik}^{(2)} f(z_k^{(3)})$$ where $$f(z^{(3)}) = a^{(3)}$$

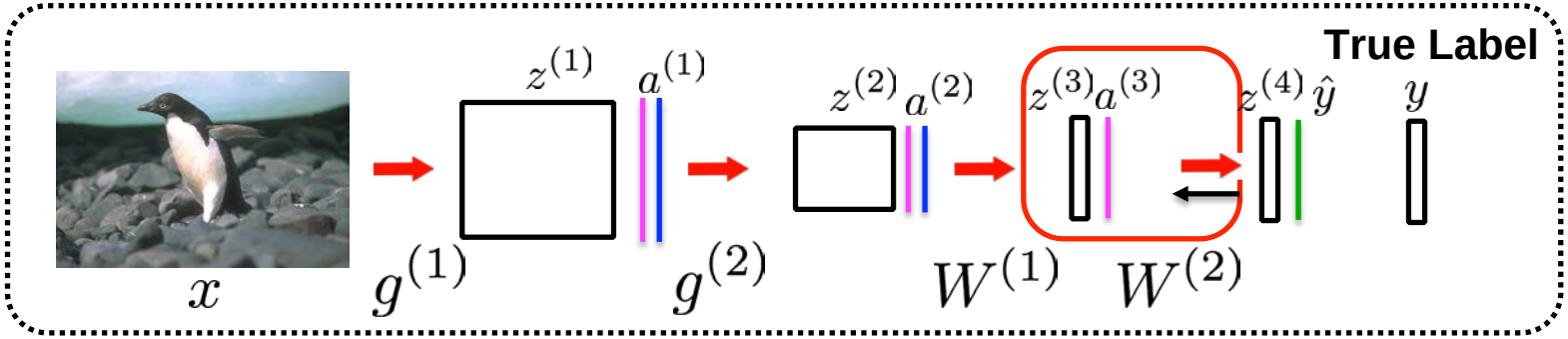


Backpropagating the gradients:

Need to compute the following gradient: $$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial f(z^{(3)})}} \frac{\partial f(z^{(3)})}{\partial z^{(3)}}$$

$$z_i^{(4)} = \sum_{k=1}^{N} W_{ik}^{(2)} f(z_k^{(3)}) \quad \text{where} \quad f(z^{(3)}) = a^{(3)}$$

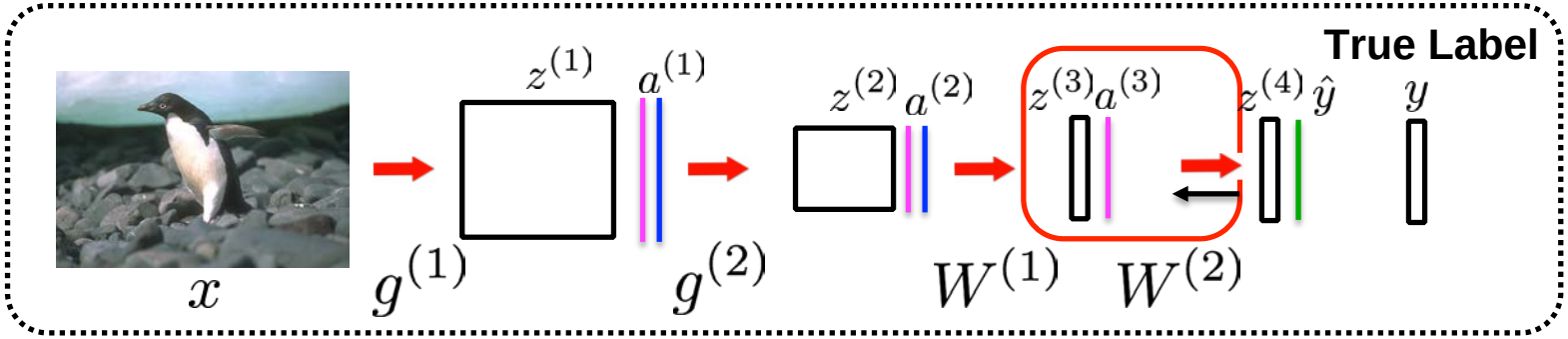
$$\boxed{\frac{\partial z_i^{(4)}}{\partial f(z_j^{(3)})} = W_{ij}^{(2)}}$$



Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \boxed{\frac{\partial f(z^{(3)})}{\partial z^{(3)}}}$$

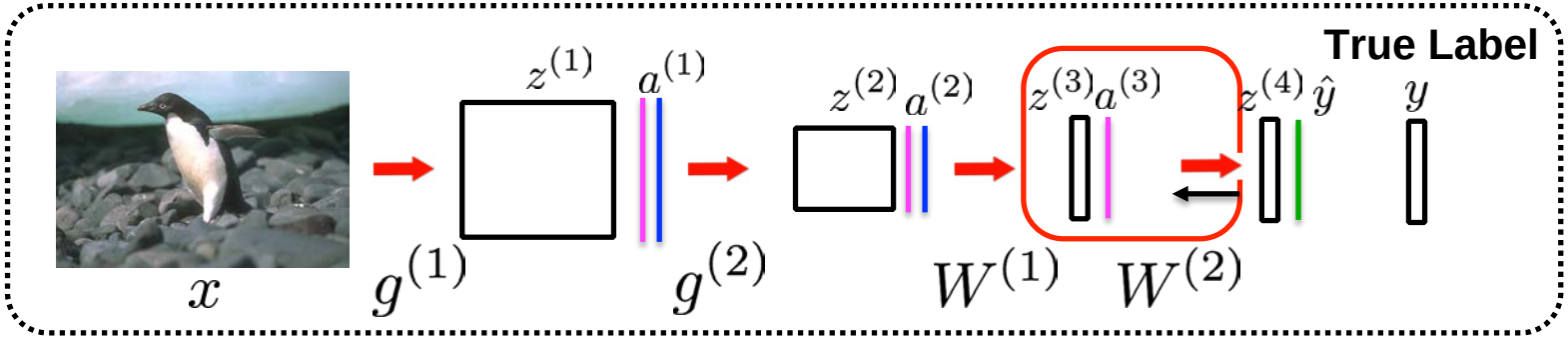


Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \boxed{\frac{\partial f(z^{(3)})}{\partial z^{(3)}}}$$

$$f(z^{(3)}) = \frac{1}{1 + \exp\left(-z^{(3)}\right)}$$

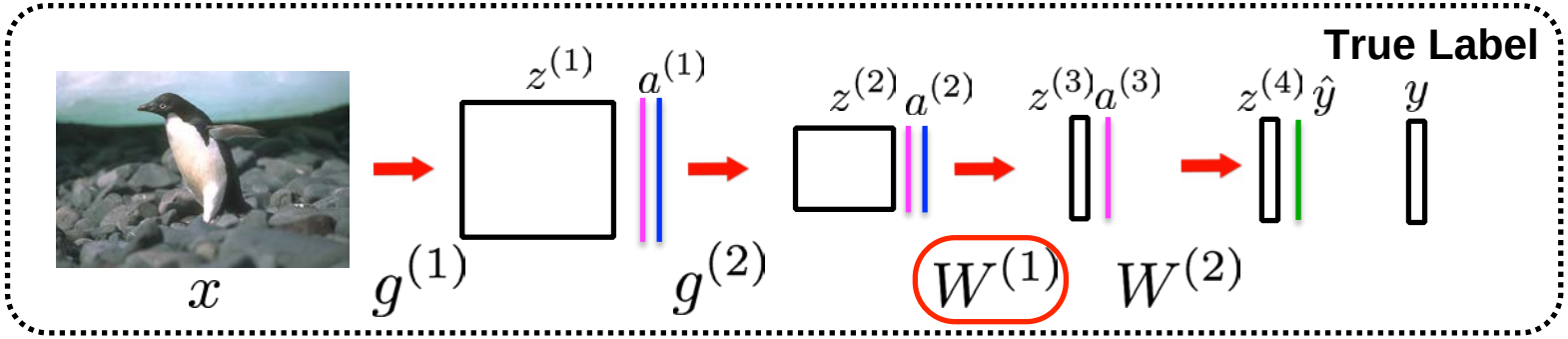


Backpropagating the gradients:

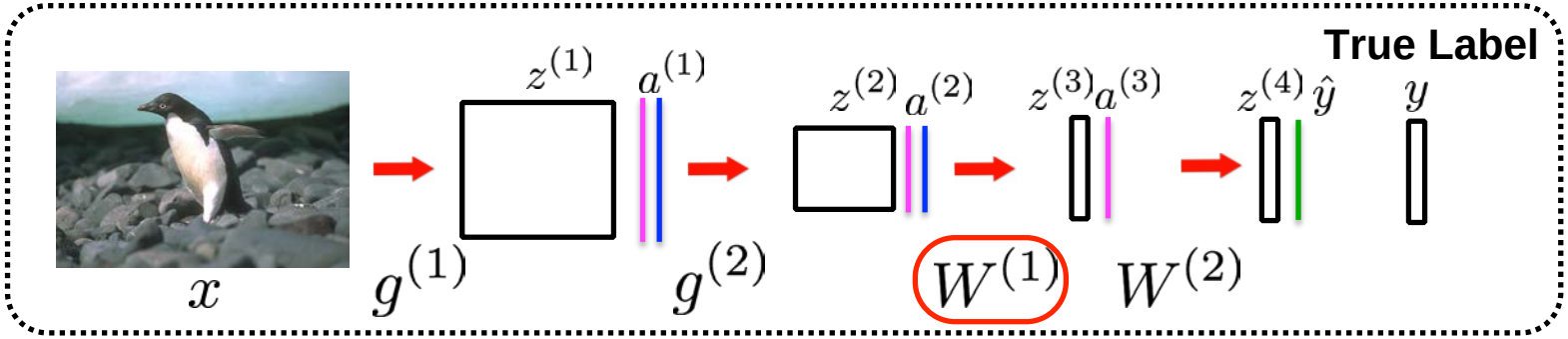
Need to compute the following gradient: $$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial L}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial f(z^{(3)})} \boxed{\frac{\partial f(z^{(3)})}{\partial z^{(3)}}}$$

$$f(z^{(3)}) = \frac{1}{1 + \exp\left(-z^{(3)}\right)}$$

$$\boxed{\frac{\partial f(z^{(3)})}{\partial z^{(3)}} = f(z^{(3)})(1 - f(z^{(3)}))}$$



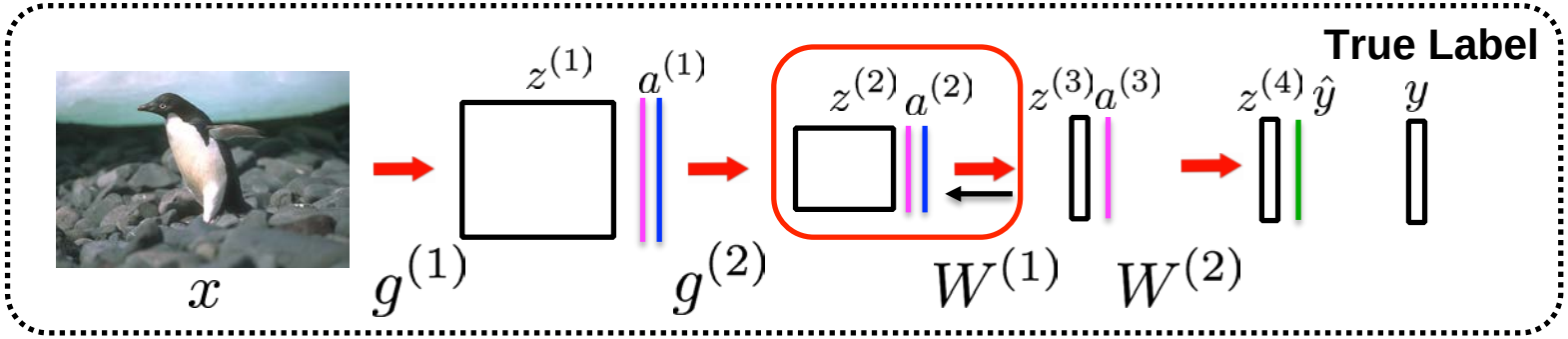
Adjusting the weights:



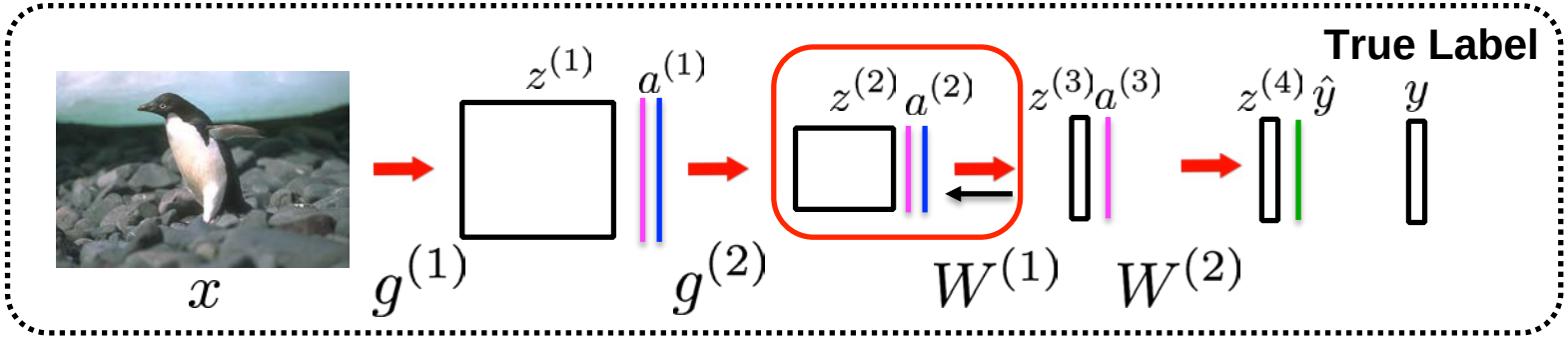
Adjusting the weights:

Need to compute the following gradient $$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

Update rule: $$W_{ij}^{(1)} = W_{ij}^{(1)} - \alpha \frac{\partial L}{\partial W_{ij}^{(1)}}$$



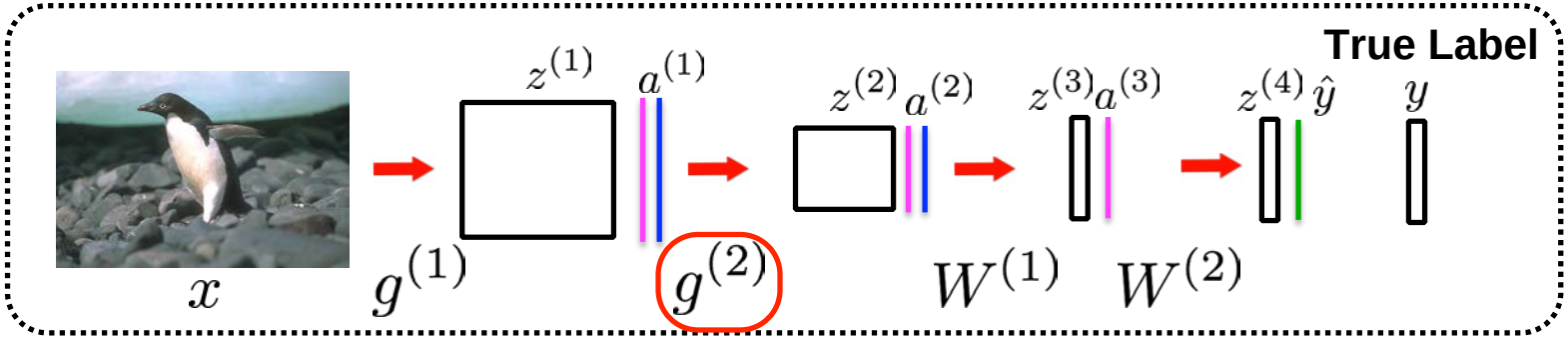
Backpropagating the gradients:



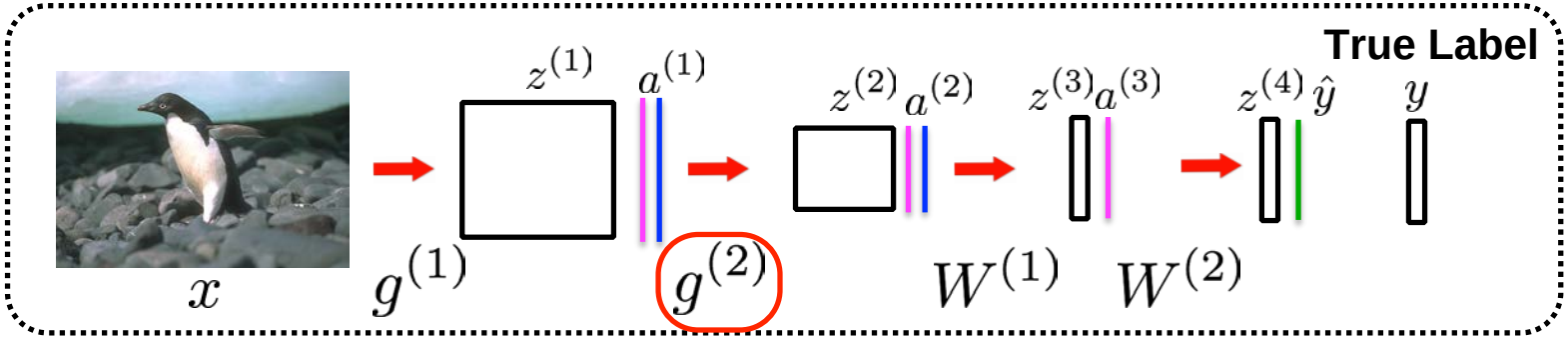
Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial f(z^{(2)})} \frac{\partial f(z^{(2)})}{\partial z^{(2)}}$$



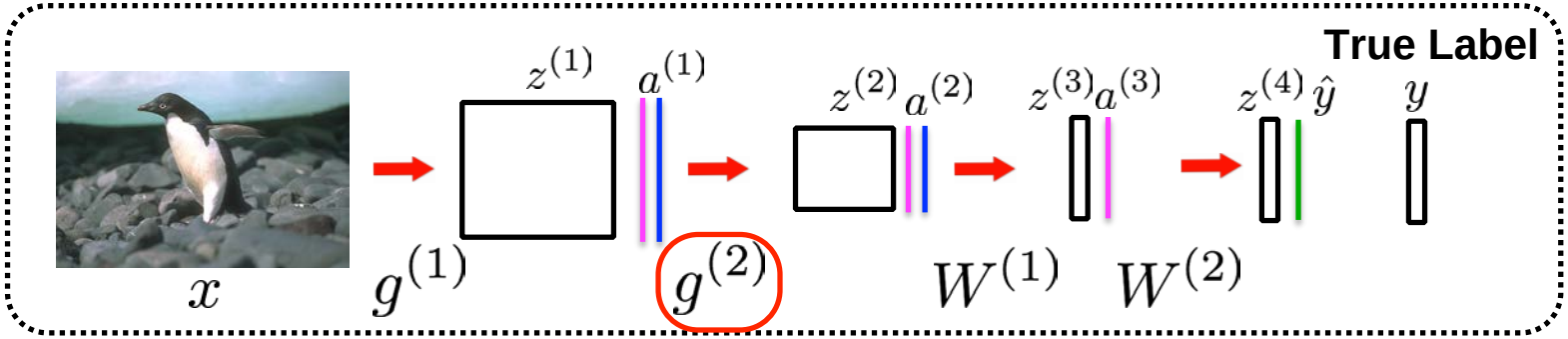
Adjusting the weights:



Adjusting the weights:

Need to compute the following gradient

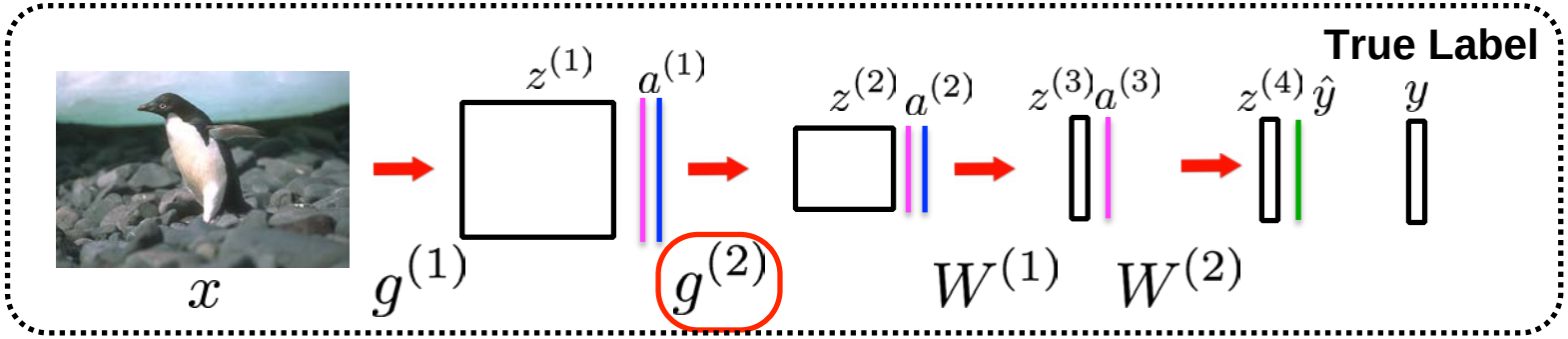
$$\frac{\partial L}{\partial g^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial g^{(2)}}$$



Adjusting the weights:

Need to compute the following gradient

$$\frac{\partial L}{\partial g^{(2)}} = \boxed{\frac{\partial L}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial g^{(2)}}$$

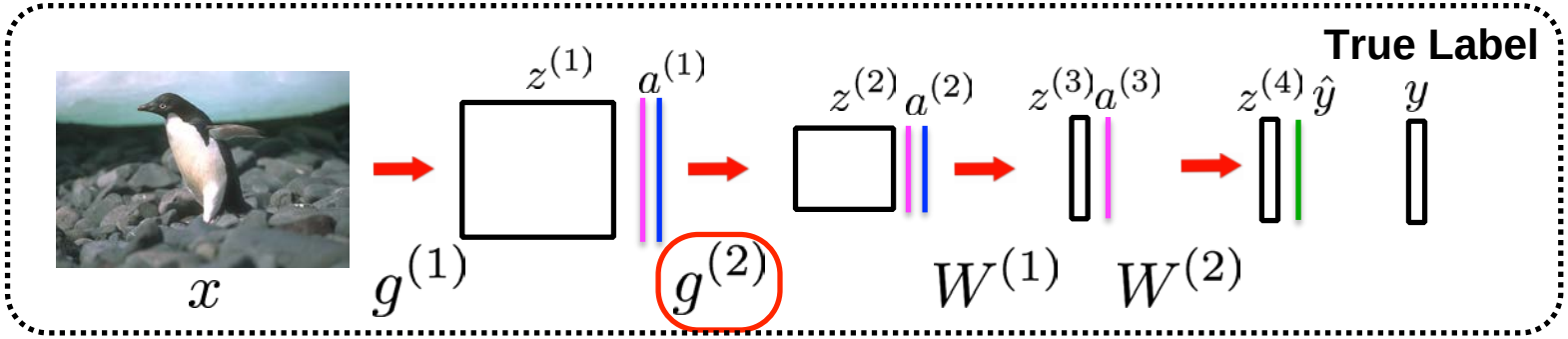


Adjusting the weights:

Need to compute the following gradient

$$\frac{\partial L}{\partial g^{(2)}} = \boxed{\frac{\partial L}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial g^{(2)}}$$

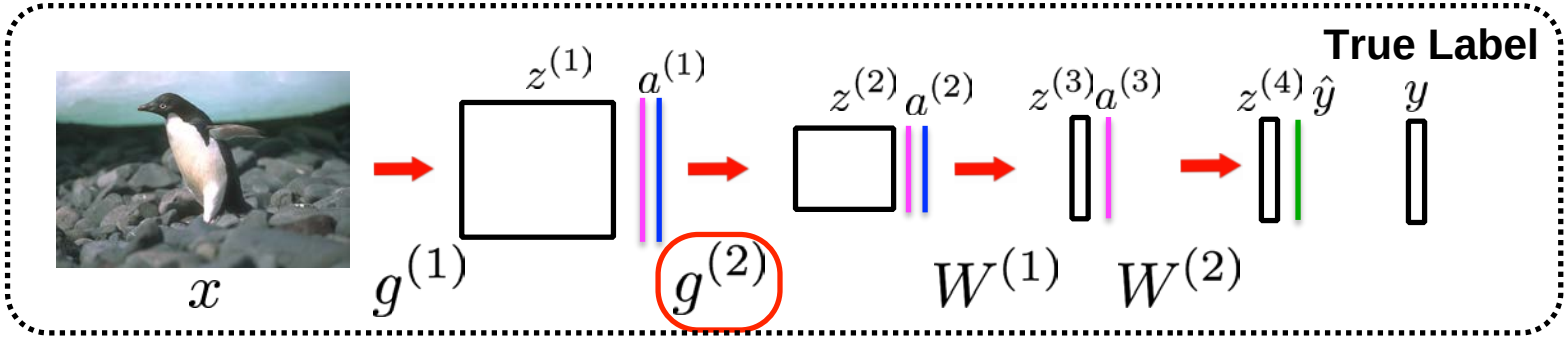
$\boxed{\frac{\partial L}{\partial z^{(2)}}}$ was already computed in the previous step



Adjusting the weights:

Need to compute the following gradient

$$\frac{\partial L}{\partial g^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial g^{(2)}}}$$

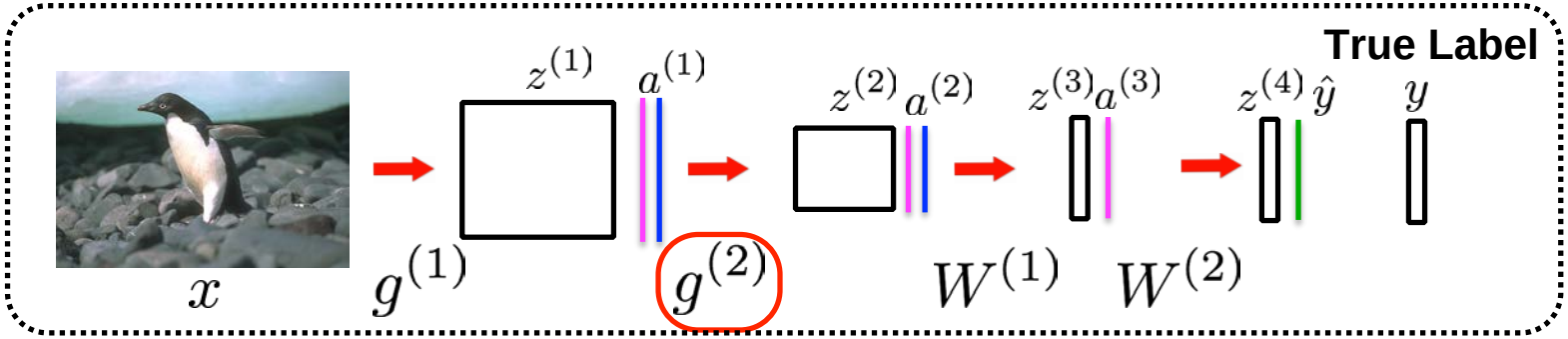


Adjusting the weights:

Need to compute the following gradient

$$\frac{\partial L}{\partial g^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial g^{(2)}}}$$

$$z_{ij}^{(2)} = \sum_{u=0}^{M} \sum_{v=0}^{N} g_{uv}^{(2)} a_{(i-u)(j-v)}^{(1)}$$ where $f\left(z^{(1)}\right) = a^{(1)}$



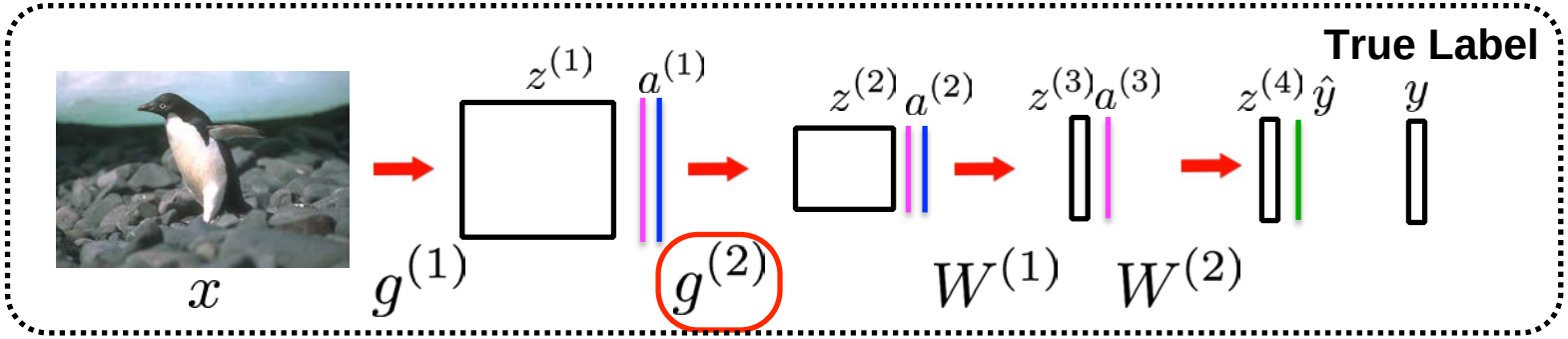
Adjusting the weights:

Need to compute the following gradient

$$\frac{\partial L}{\partial g^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial g^{(2)}}}$$

$$z_{ij}^{(2)} = \sum_{u=0}^{M} \sum_{v=0}^{N} g_{uv}^{(2)} a_{(i-u)(j-v)}^{(1)}$$ where $f(z^{(1)}) = a^{(1)}$

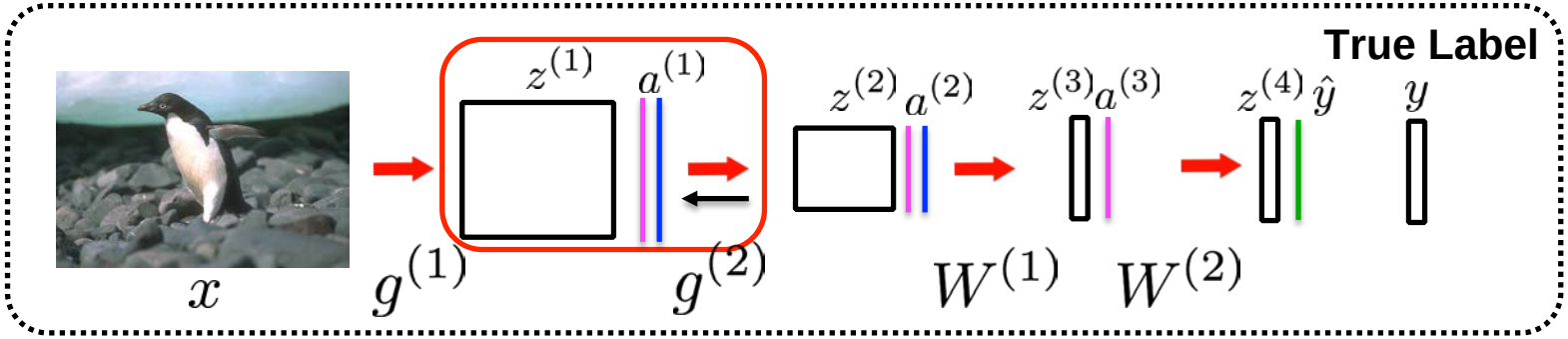
$$\boxed{\frac{\partial z_{ij}^{(2)}}{\partial g_{mn}^{(2)}} = a_{(i-m)(j-n)}^{(1)}}$$



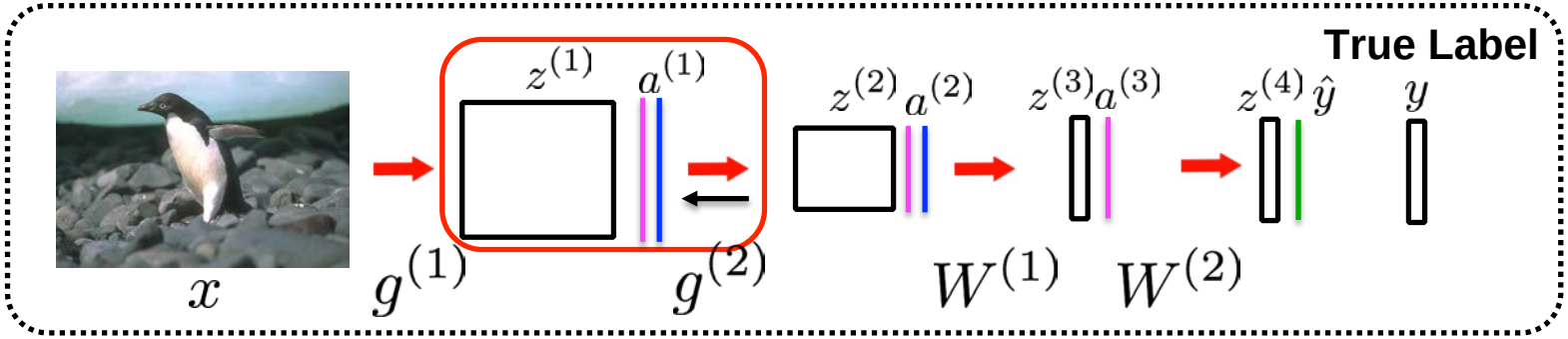
Adjusting the weights:

Need to compute the following gradient $$\frac{\partial L}{\partial g^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial g^{(2)}}$$

Update rule: $$g^{(2)}_{mn} = g^{(2)}_{mn} - \alpha \frac{\partial L}{\partial g^{(2)}_{mn}}$$



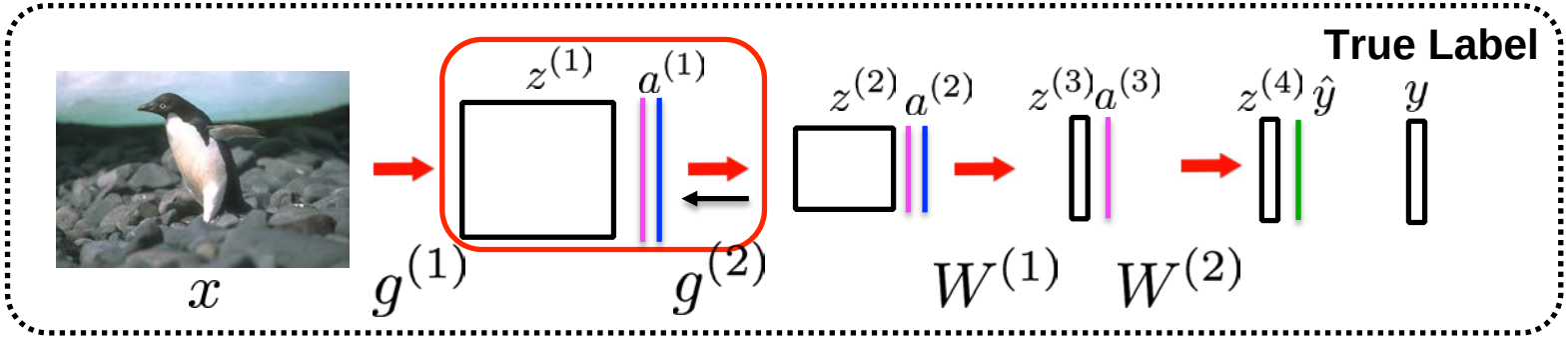
Backpropagating the gradients:



Backpropagating the gradients:

Need to compute the following gradient:

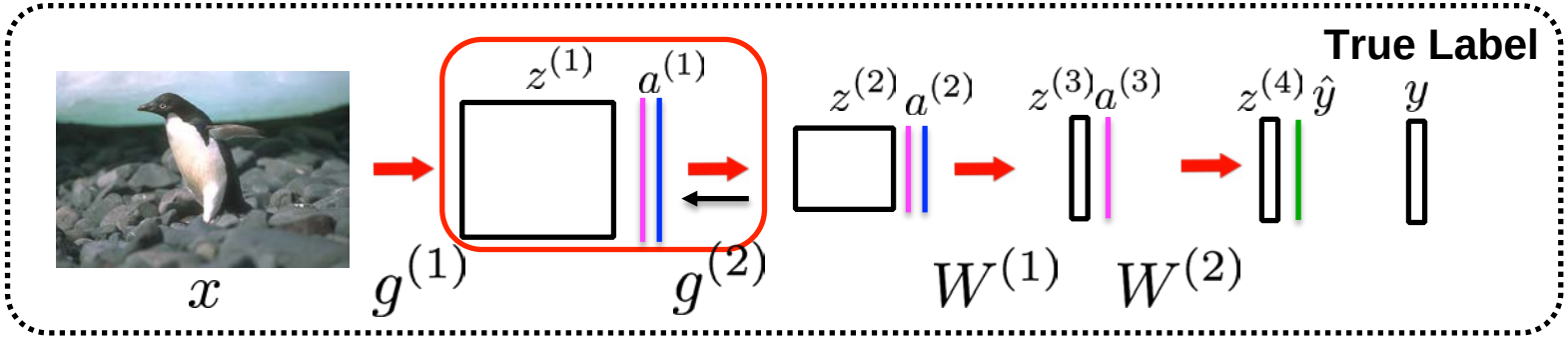
$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$



Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \boxed{\frac{\partial L}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$

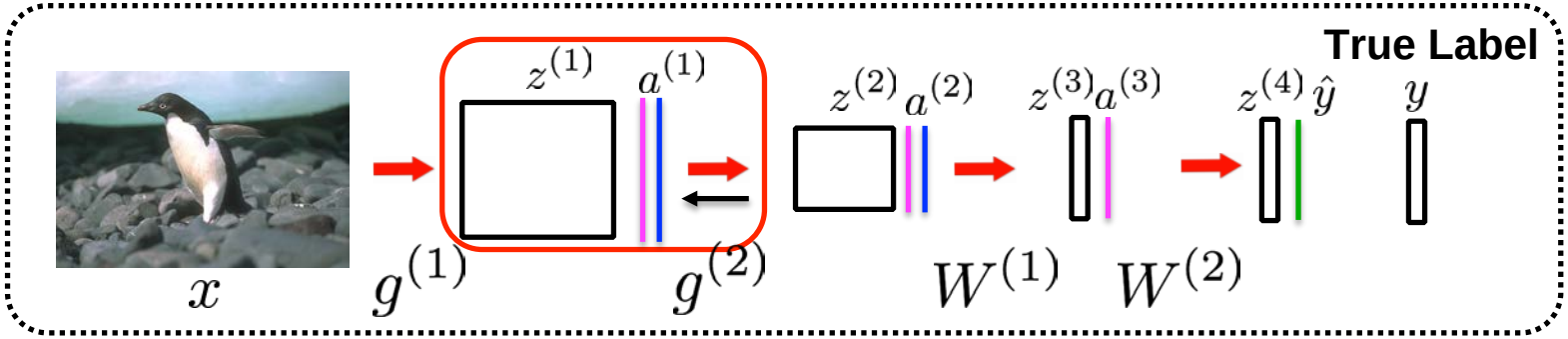


Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \boxed{\frac{\partial L}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$

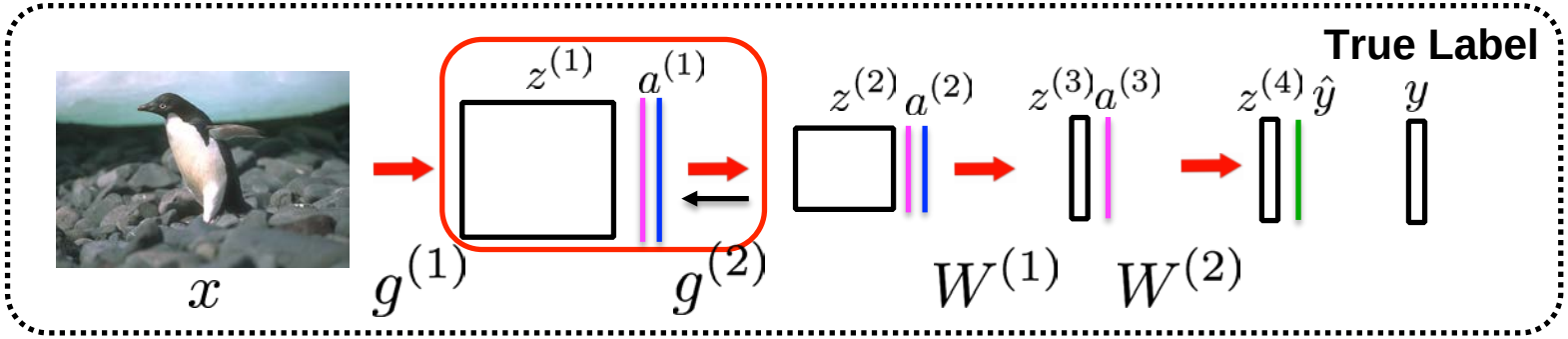
$\boxed{\dfrac{\partial L}{\partial z^{(2)}}}$ was already computed in the previous step



Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial f(z^{(1)})}} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$

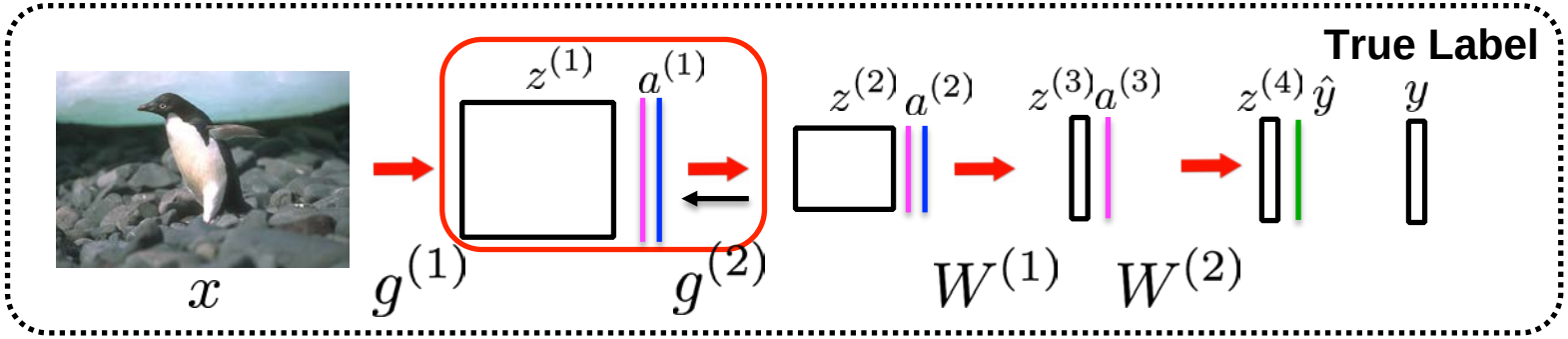


Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial f(z^{(1)})}} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$

$$z_{ij}^{(2)} = \sum_{u=0}^{M} \sum_{v=0}^{N} g_{uv}^{(2)} a_{(i-u)(j-v)}^{(1)} \quad \text{where} \quad f(z^{(1)}) = a^{(1)}$$



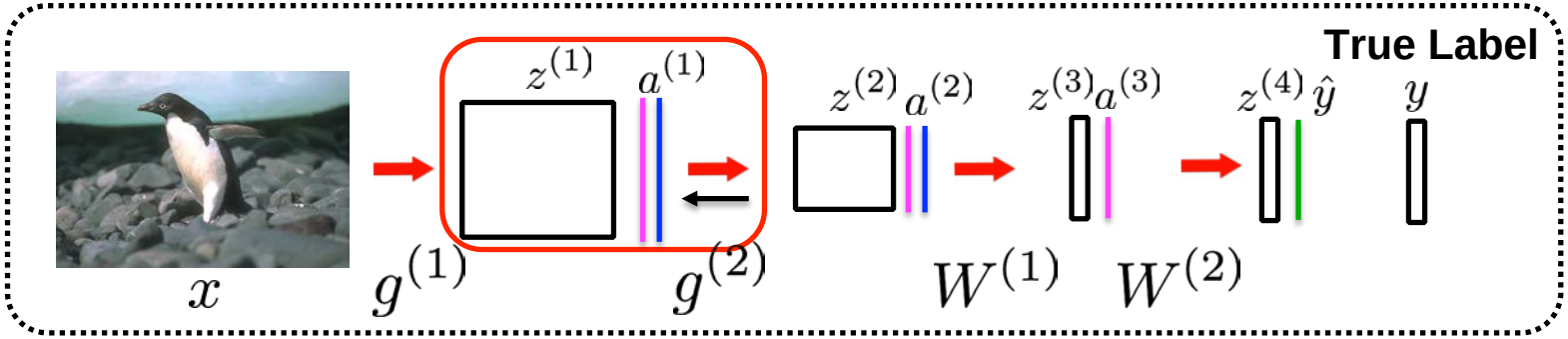
Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \boxed{\frac{\partial z^{(2)}}{\partial f(z^{(1)})}} \frac{\partial f(z^{(1)})}{\partial z^{(1)}}$$

$$z_{ij}^{(2)} = \sum_{u=0}^{M} \sum_{v=0}^{N} g_{uv}^{(2)} a_{(i-u)(j-v)}^{(1)} \quad \text{where} \quad f(z^{(1)}) = a^{(1)}$$

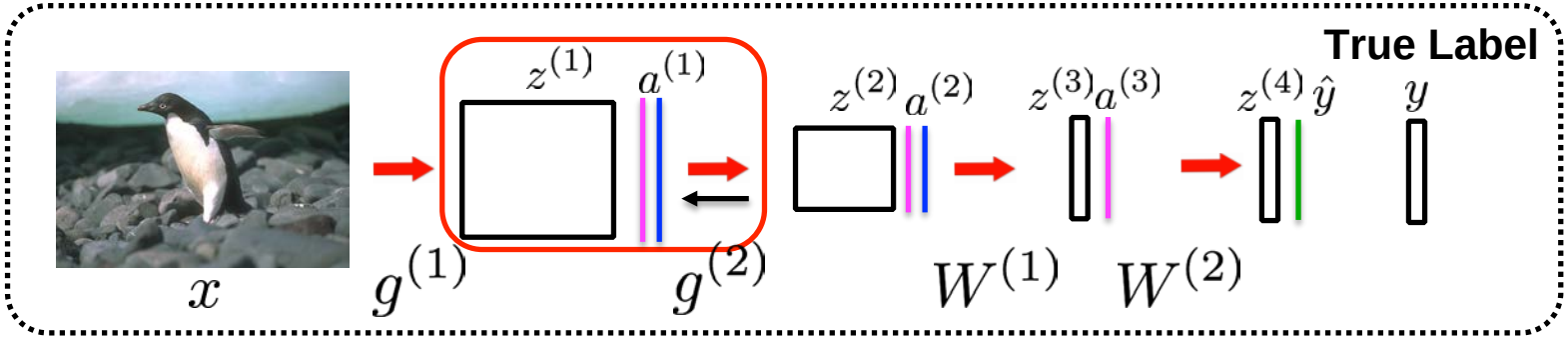
$$\boxed{\frac{\partial z_{ij}^{(2)}}{\partial a_{(i-m)(j-n)}^{(1)}} = g_{mn}^{(2)}}$$



Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \boxed{\frac{\partial f(z^{(1)})}{\partial z^{(1)}}}$$

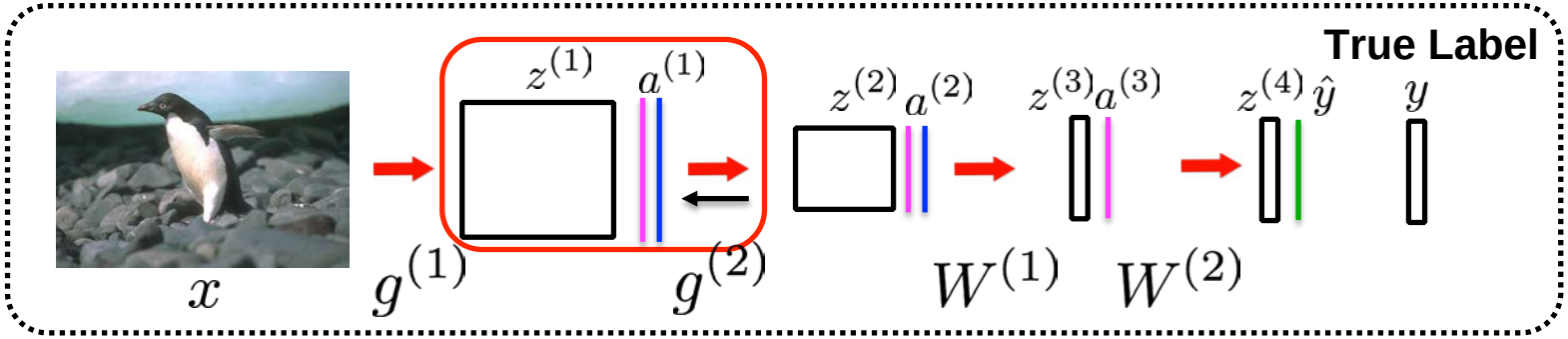


Backpropagating the gradients:

Need to compute the following gradient:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \boxed{\frac{\partial f(z^{(1)})}{\partial z^{(1)}}}$$

$$f(z^{(1)}) = \frac{1}{1 + \exp\left(-z^{(1)}\right)}$$



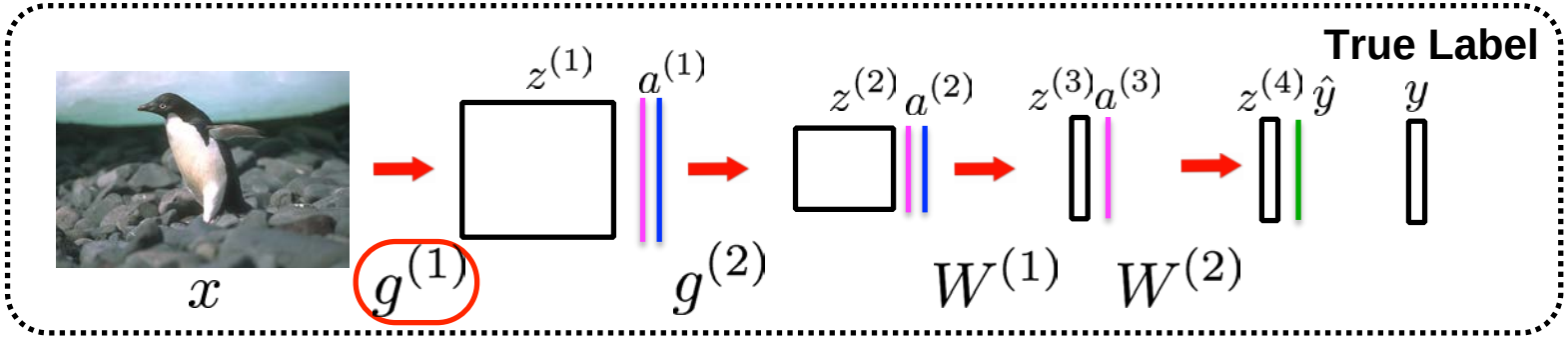
Backpropagating the gradients:

Need to compute the following gradient:

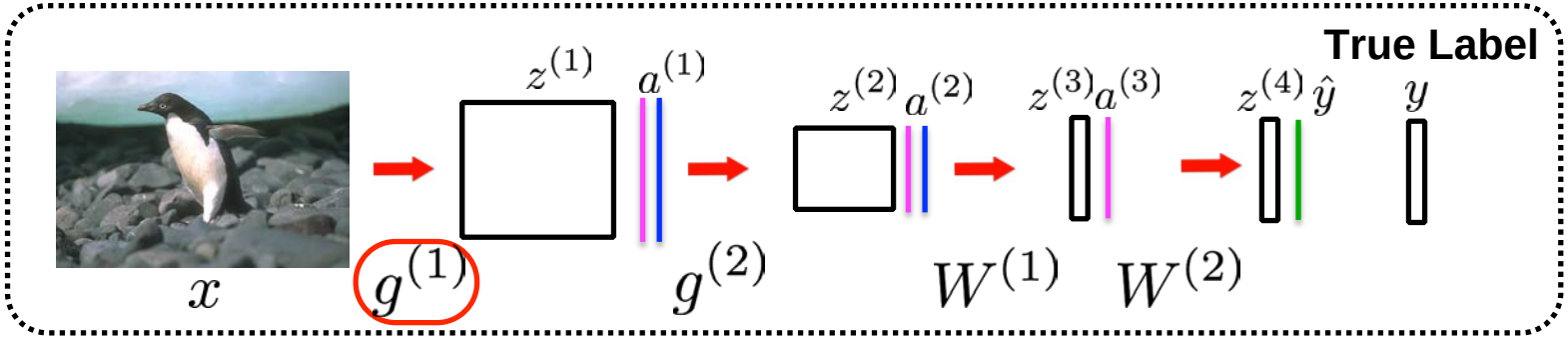
$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial f(z^{(1)})} \boxed{\frac{\partial f(z^{(1)})}{\partial z^{(1)}}}$$

$$f(z^{(1)}) = \frac{1}{1 + \exp\left(-z^{(1)}\right)}$$

$$\boxed{\frac{\partial f(z^{(1)})}{\partial z^{(1)}} = f(z^{(1)})(1 - f(z^{(1)}))}$$



Adjusting the weights:



Adjusting the weights:

Need to compute the following gradient $$\frac{\partial L}{\partial g^{(1)}} = \frac{\partial L}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial g^{(1)}}$$

Update rule: $$g^{(1)}_{mn} = g^{(1)}_{mn} - \alpha \frac{\partial L}{\partial g^{(1)}_{mn}}$$

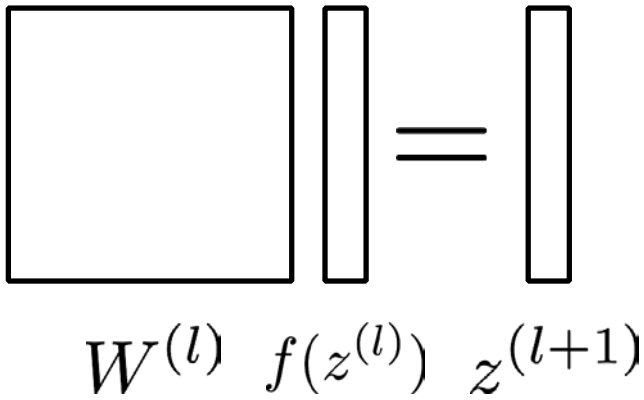
Visual illustration

Backpropagation

Convolutional Neural Networks

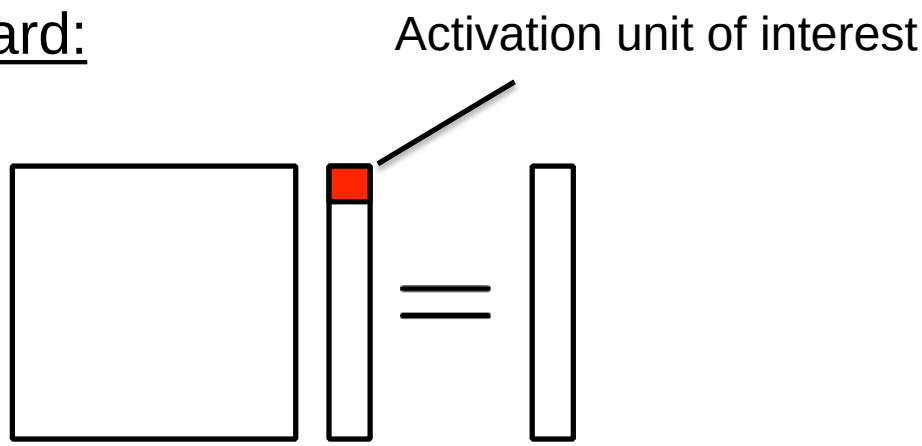
Fully Connected Layers:

Forward:



Fully Connected Layers:

Forward:

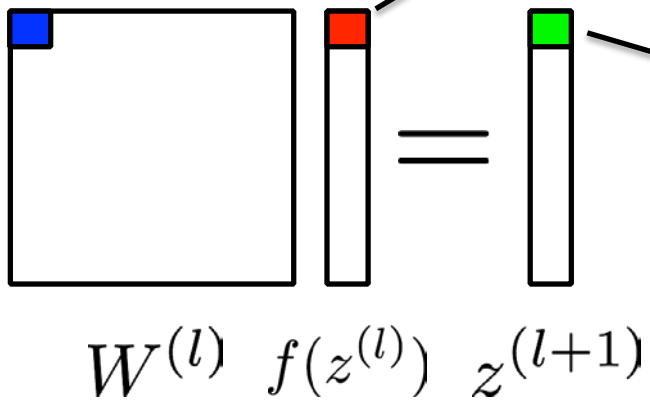


$$W^{(l)} \quad f(z^{(l)}) \quad z^{(l+1)}$$

Fully Connected Layers:

Forward:

Activation unit of interest

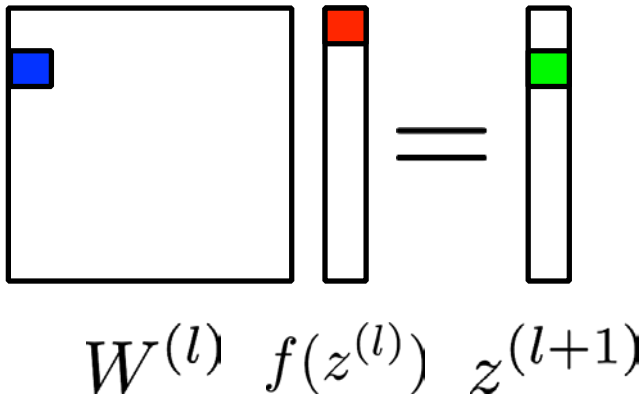


The output

The weight that is used in conjunction with the activation unit of interest

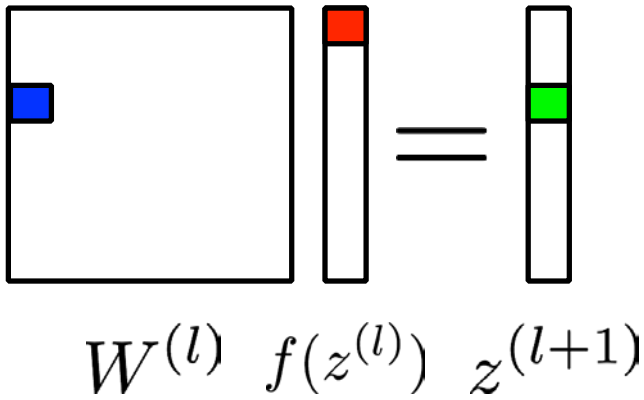
Fully Connected Layers:

Forward:



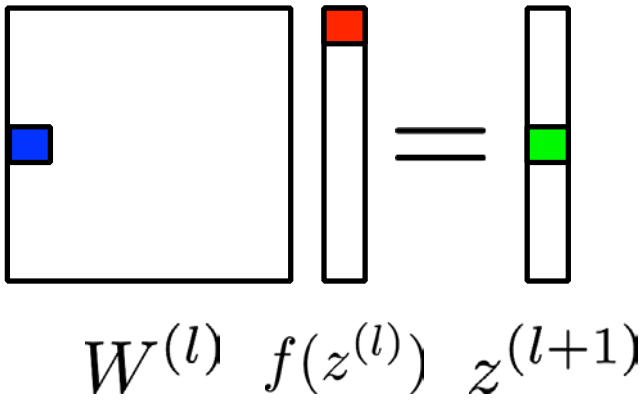
Fully Connected Layers:

Forward:



Fully Connected Layers:

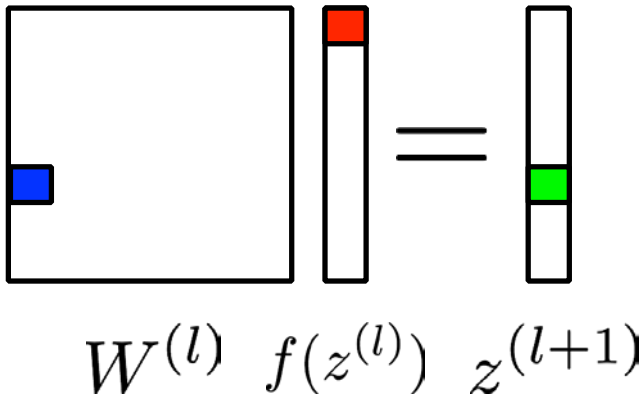
Forward:



Backpropagation

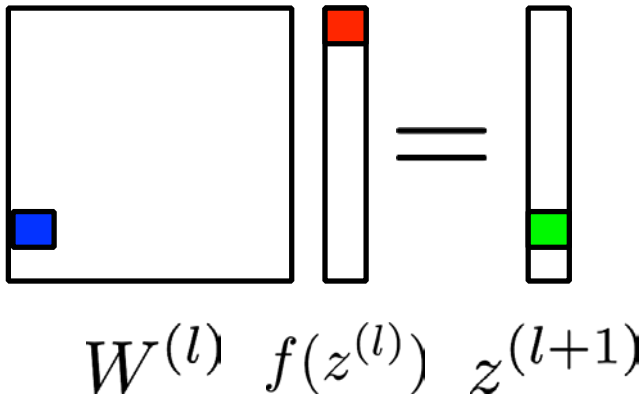
<u>Fully Connected Layers:</u>

<u>Forward:</u>



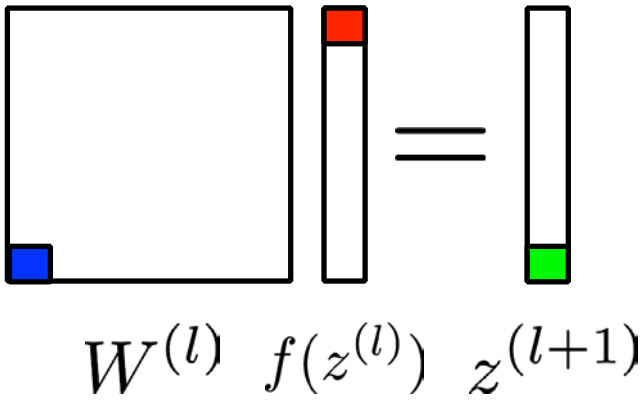
Fully Connected Layers:

Forward:



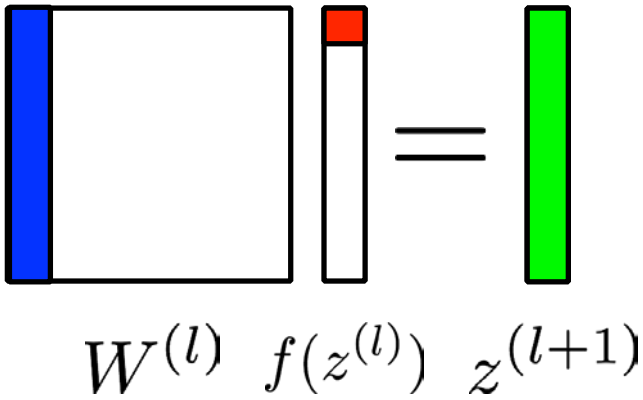
Fully Connected Layers:

Forward:



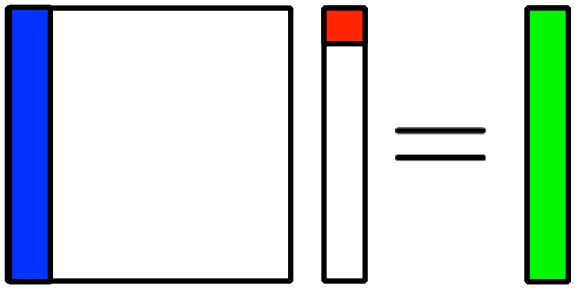
Fully Connected Layers:

Forward:



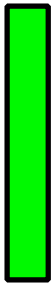
Fully Connected Layers:

Forward:



$W^{(l)}$ $f(z^{(l)})$ $z^{(l+1)}$

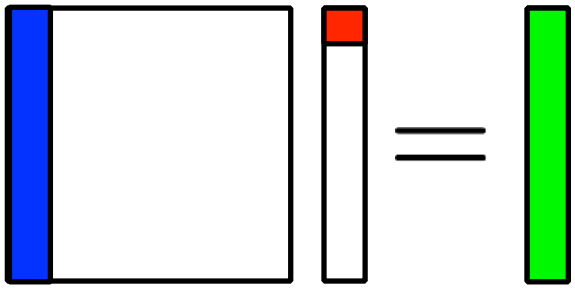
Backward:



$$\frac{\partial L}{\partial z^{(l+1)}}$$

Fully Connected Layers:

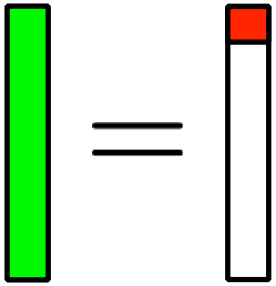
Forward:



$W^{(l)}$ $f(z^{(l)})$ $z^{(l+1)}$

Backward:

A measure how much an activation unit contributed to the loss

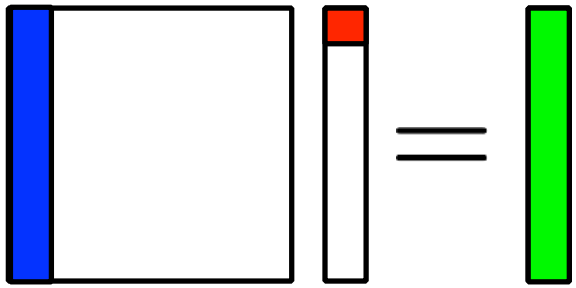


$\frac{\partial L}{\partial z^{(l+1)}}$ $\frac{\partial L}{\partial f(z^{(l)})}$

Fully Connected Layers:

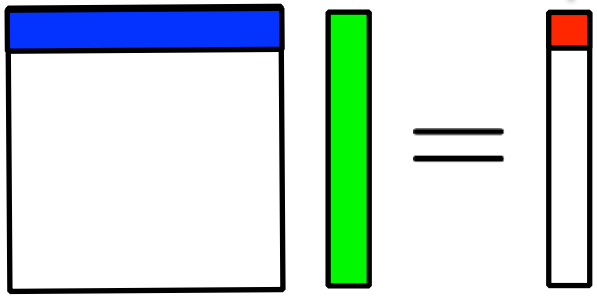
A measure how much an activation unit contributed to the loss

Forward:



$$W^{(l)} \quad f(z^{(l)}) \quad z^{(l+1)}$$

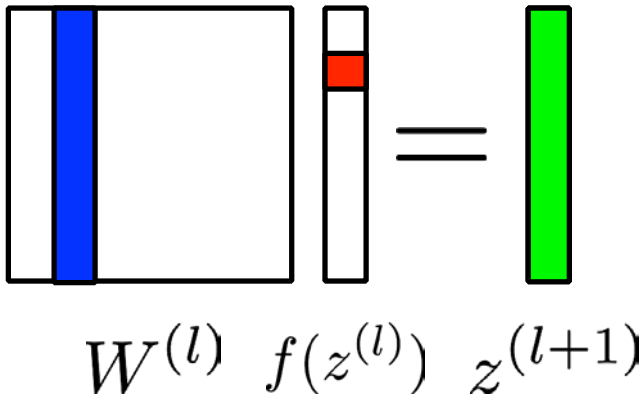
Backward:



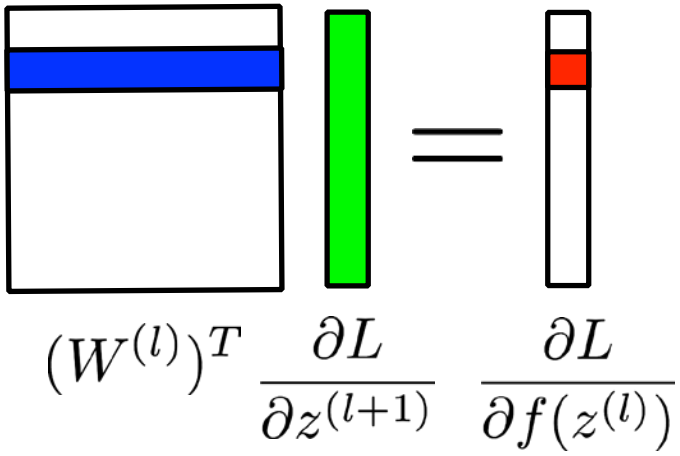
$$(W^{(l)})^T \quad \frac{\partial L}{\partial z^{(l+1)}} \quad \frac{\partial L}{\partial f(z^{(l)})}$$

Fully Connected Layers:

Forward:

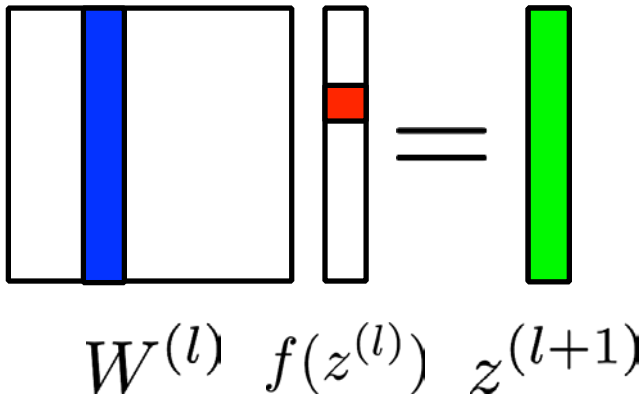


Backward:

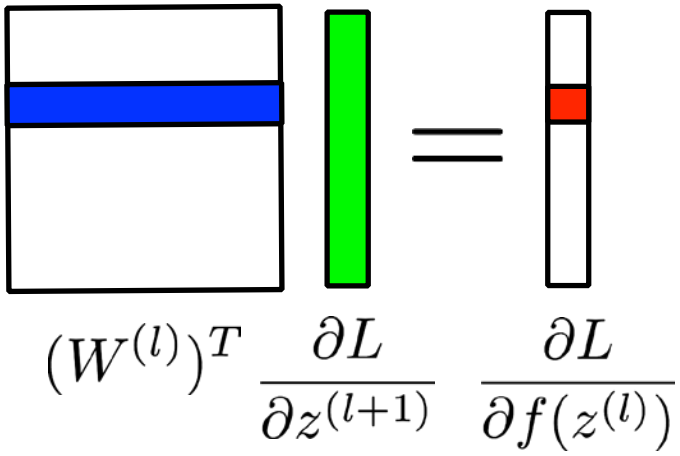


Fully Connected Layers:

Forward:

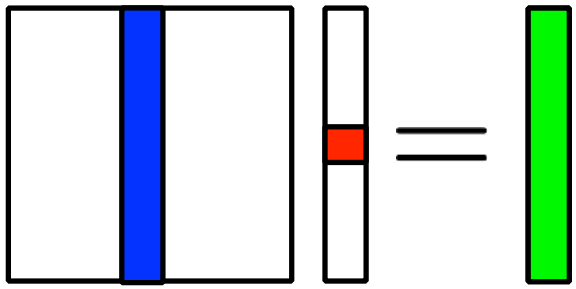


Backward:



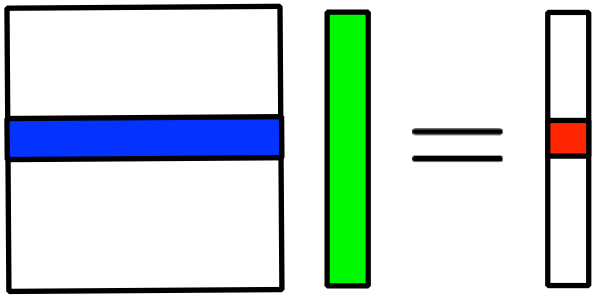
Fully Connected Layers:

Forward:



$W^{(l)}$ $f(z^{(l)})$ $z^{(l+1)}$

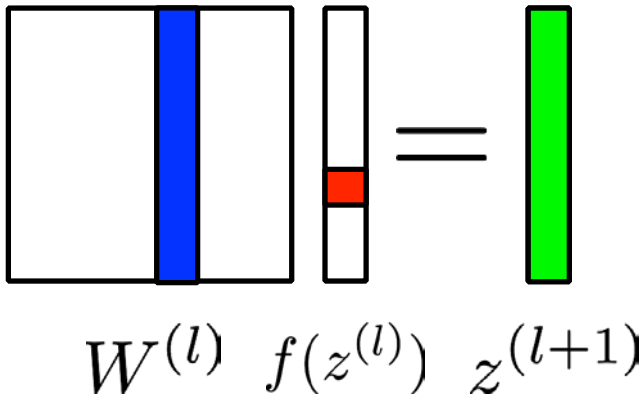
Backward:



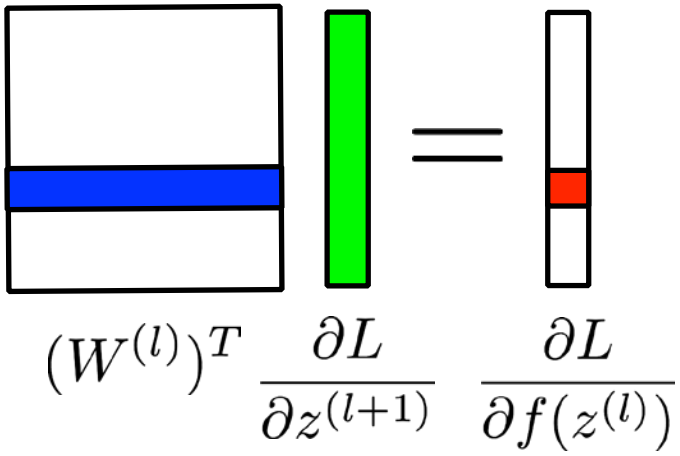
$(W^{(l)})^T$ $\frac{\partial L}{\partial z^{(l+1)}}$ $\frac{\partial L}{\partial f(z^{(l)})}$

Fully Connected Layers:

Forward:

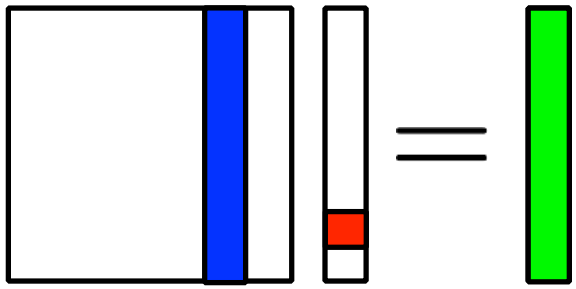


Backward:



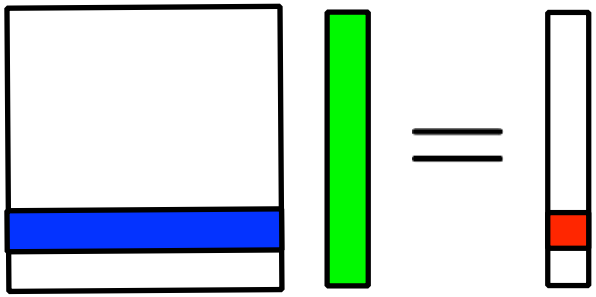
Fully Connected Layers:

Forward:



$$W^{(l)} \quad f(z^{(l)}) \quad z^{(l+1)}$$

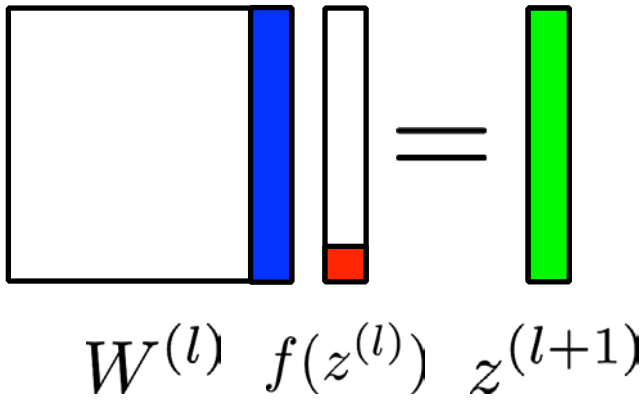
Backward:



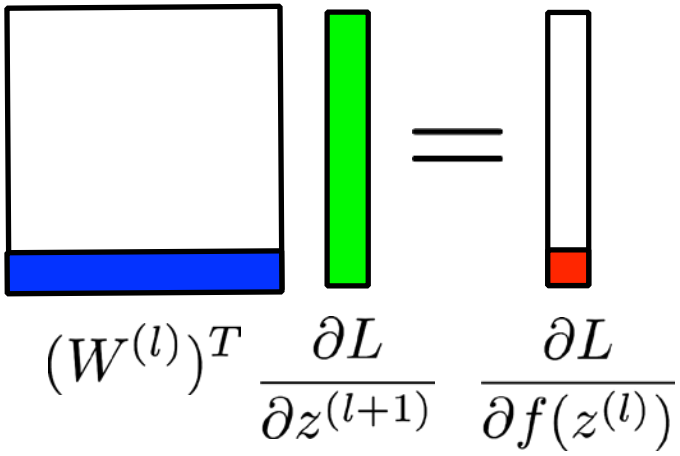
$$(W^{(l)})^T \quad \frac{\partial L}{\partial z^{(l+1)}} \quad \frac{\partial L}{\partial f(z^{(l)})}$$

Fully Connected Layers:

Forward:



Backward:



Summary for fully connected layers

Backpropagation

Convolutional Neural Networks

Summary:

1. Let $\frac{\partial L}{\partial z_i^{(n)}} = \hat{y}_i - y_i$, where n denotes the number of layers in the network.

Summary:

1. Let $\frac{\partial L}{\partial z_i^{(n)}} = \hat{y}_i - y_i$, where n denotes the number of layers in the network.

2. For each fully connected layer l :

   - For each node i in layer l set:

$$\frac{\partial L}{\partial z_i^{(l)}} = \left(\sum_{j=1}^{s^{l+1}} W_{ji}^{(l)} \frac{\partial L}{\partial z_j^{(l+1)}}\right) \frac{\partial f(z_i^{(l)})}{\partial z_i^{(l)}}$$

Summary:

1. Let $\frac{\partial L}{\partial z_i^{(n)}} = \hat{y}_i - y_i$, where n denotes the number of layers in the network.

2. For each fully connected layer l :

   - For each node i in layer l set:

$$\frac{\partial L}{\partial z_i^{(l)}} = \left(\sum_{j=1}^{s^{l+1}} W_{ji}^{(l)} \frac{\partial L}{\partial z_j^{(l+1)}}\right) \frac{\partial f(z_i^{(l)})}{\partial z_i^{(l)}}$$

   - Compute partial derivatives: $\frac{\partial L}{\partial W_{ij}^{(l)}} = f(z_j^{(l)}) \frac{\partial L}{\partial z_i^{(l+1)}}$

Summary:

1. Let $\frac{\partial L}{\partial z_i^{(n)}} = \hat{y}_i - y_i$, where n denotes the number of layers in the network.

2. For each fully connected layer l:

   - For each node i in layer l set:

$$\frac{\partial L}{\partial z_i^{(l)}} = \left(\sum_{j=1}^{s^{l+1}} W_{ji}^{(l)} \frac{\partial L}{\partial z_j^{(l+1)}}\right) \frac{\partial f(z_i^{(l)})}{\partial z_i^{(l)}}$$

   - Compute partial derivatives: $\frac{\partial L}{\partial W_{ij}^{(l)}} = f(z_j^{(l)}) \frac{\partial L}{\partial z_i^{(l+1)}}$

   - Update the parameters: $W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial L}{\partial W_{ij}^{(l)}}$

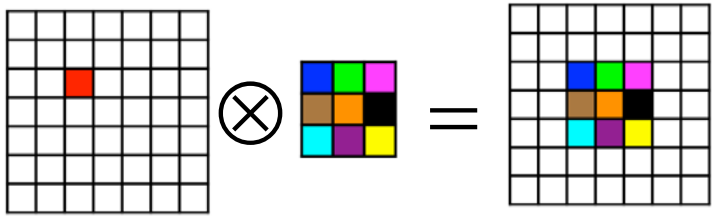
Visual illustration

Backpropagation

Convolutional Neural Networks

Convolutional Layers:

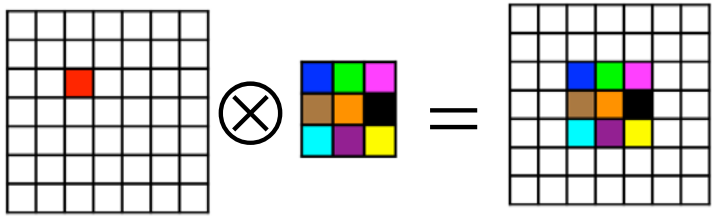
Forward:



$$a^{(l)} \otimes g^{(l)} = z^{(l+1)}$$

Convolutional Layers:

Forward:



$$a^{(l)} \otimes g^{(l)} = z^{(l+1)}$$

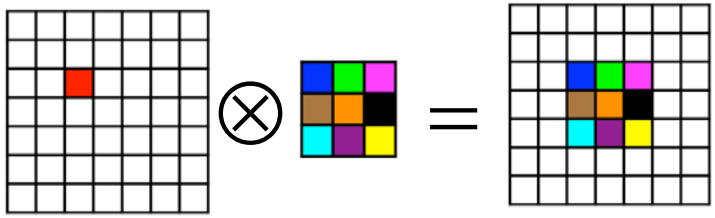
A measure how much an activation unit contributed to the loss

Backward:

$$\frac{\partial L}{\partial z^{(l+1)}} \qquad \frac{\partial L}{\partial f(z^{(l)})}$$

Convolutional Layers:

Forward:



$$a^{(l)} \otimes g^{(l)} = z^{(l+1)}$$

Backward:

$$sum\left(g^{(l)} \odot \frac{\partial L}{\partial z^{(l+1)}} \right) = \frac{\partial L}{\partial f(z^{(l)})}$$

Summary:

1. Let $\frac{\partial L}{\partial z^{(c)}}$, where c denotes the index of a first fully connected layer.

2. For each convolutional layer l:

   - For each node ij in layer l set

$$\frac{\partial L}{\partial z_{ij}^{(l)}} = \left(\sum_{m=0}^{M}\sum_{n=0}^{N} g_{mn}^{(l)} \frac{\partial L}{\partial z_{(i+m)(j+n)}^{(l+1)}}\right)\frac{\partial f(z_{ij}^{(l)})}{\partial z_{ij}^{(l)}}$$

Summary:

1. Let $\frac{\partial L}{\partial z^{(c)}}$, where c denotes the index of a first fully connected layer.

2. For each convolutional layer l :

   - For each node ij in layer l set

$$\frac{\partial L}{\partial z_{ij}^{(l)}} = \left(\sum_{m=0}^{M}\sum_{n=0}^{N} g_{mn}^{(l)} \frac{\partial L}{\partial z_{(i+m)(j+n)}^{(l+1)}}\right)\frac{\partial f(z_{ij}^{(l)})}{\partial z_{ij}^{(l)}}$$

   - Compute partial derivatives:

$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

Summary:

1. Let $\frac{\partial L}{\partial z^{(c)}}$, where c denotes the index of a first fully connected layer.

2. For each convolutional layer l :

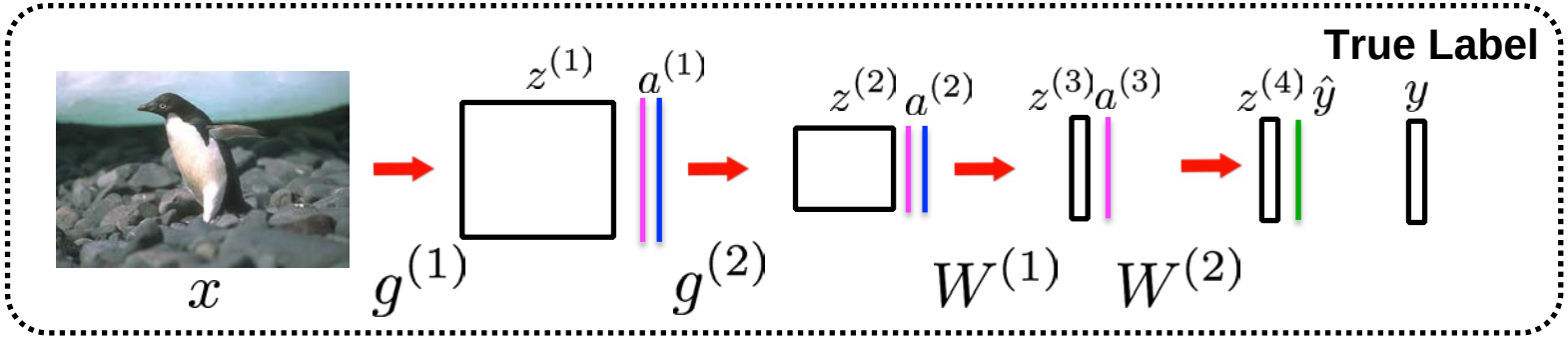
   - For each node ij in layer l set

$$\frac{\partial L}{\partial z_{ij}^{(l)}} = \left(\sum_{m=0}^{M}\sum_{n=0}^{N} g_{mn}^{(l)} \frac{\partial L}{\partial z_{(i+m)(j+n)}^{(l+1)}}\right)\frac{\partial f(z_{ij}^{(l)})}{\partial z_{ij}^{(l)}}$$

   - Compute partial derivatives:

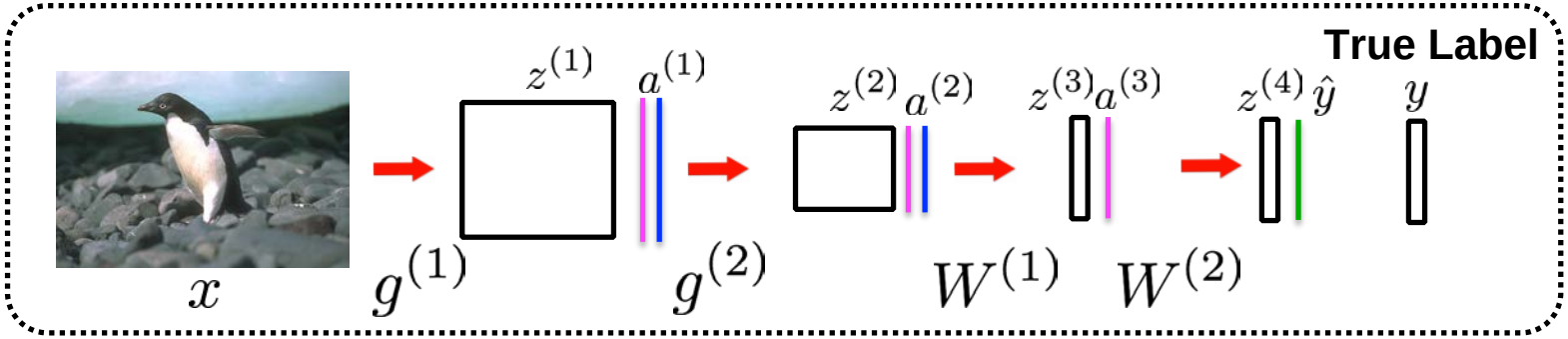
$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

   - Update the parameters: $g_{ij}^{(l)} = g_{ij}^{(l)} - \alpha \frac{\partial L}{\partial g_{ij}^{(l)}}$



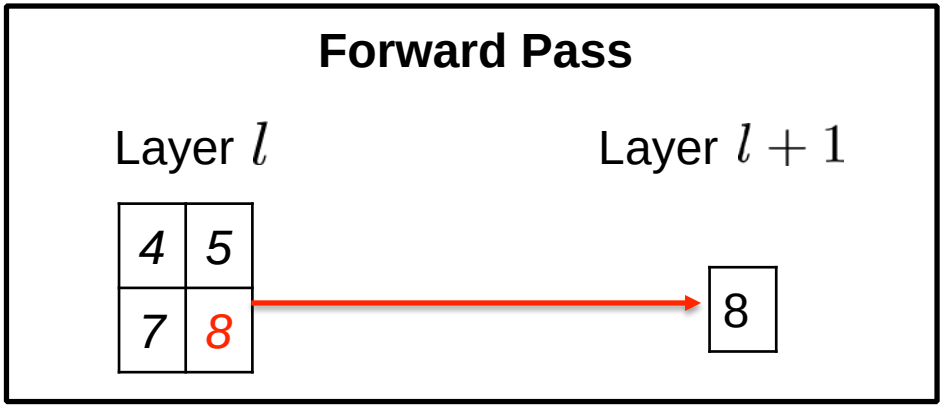
Gradient in pooling layers:

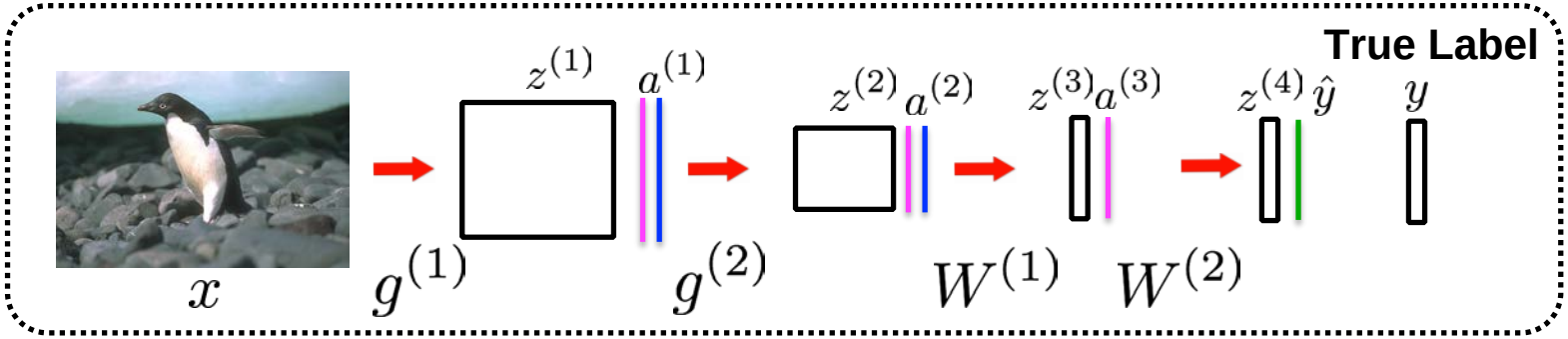
- There is no learning done in the pooling layers
- The error that is backpropagated to the pooling layer, is sent back from to the node where it came from.



Gradient in pooling layers:

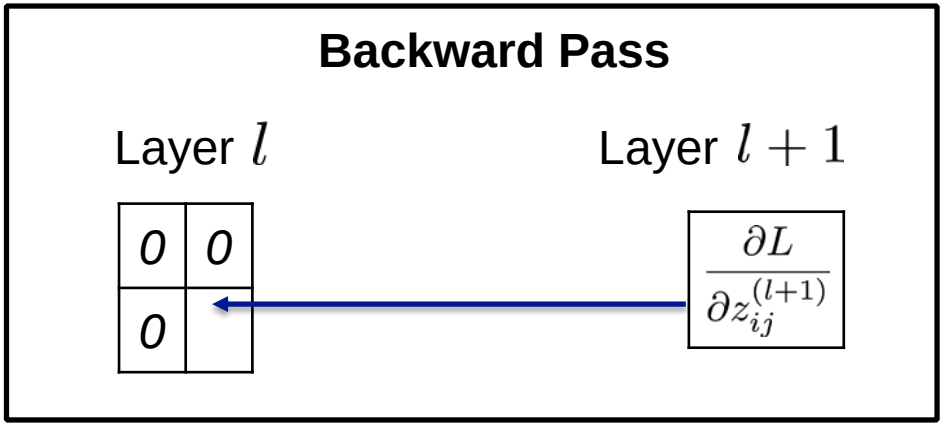
- There is no learning done in the pooling layers
- The error that is backpropagated to the pooling layer, is sent back from to the node where it came from.



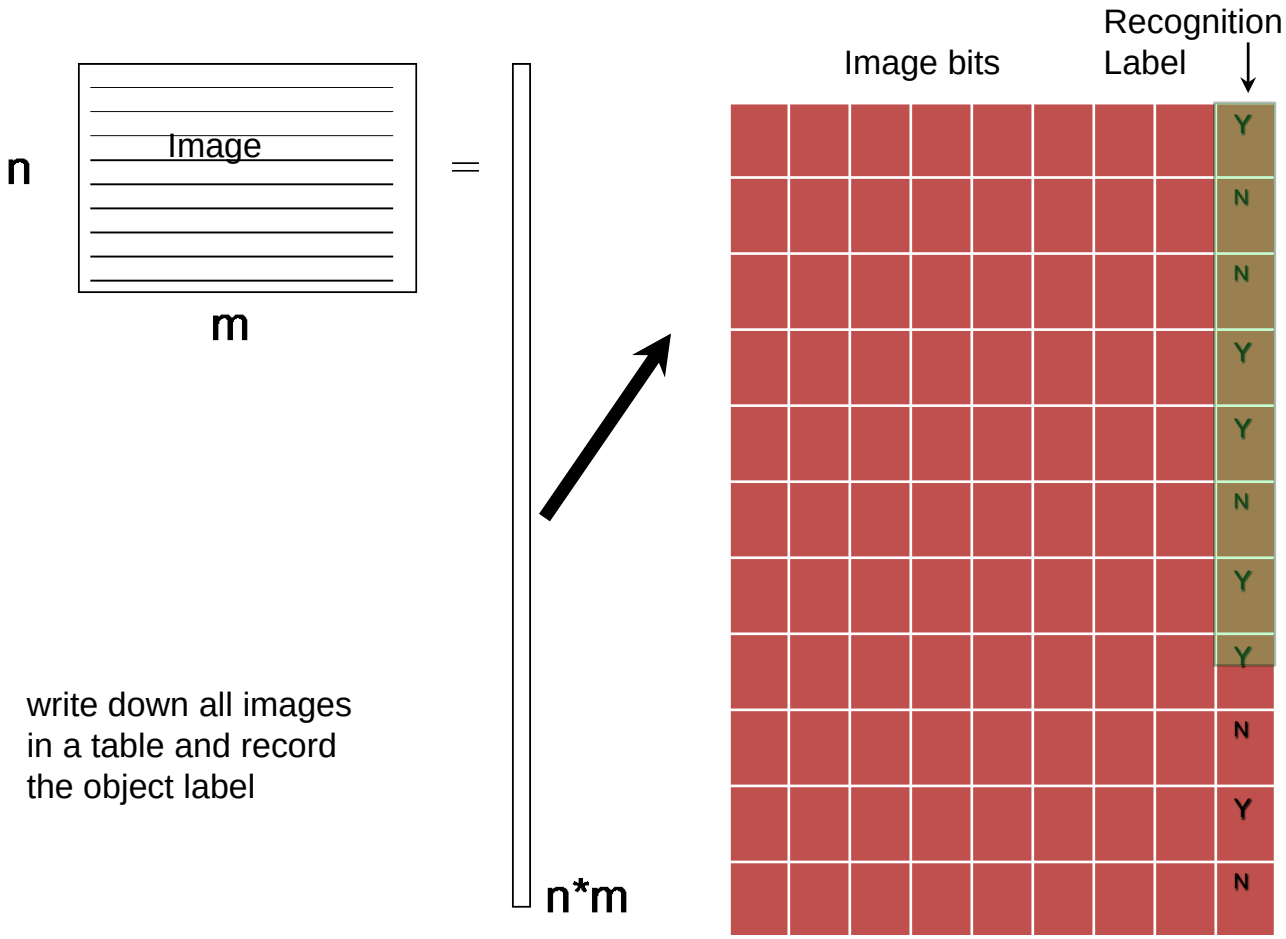


Gradient in pooling layers:

- There is no learning done in the pooling layers
- The error that is backpropagated to the pooling layer, is sent back from to the node where it came from.



Recognition as a big table lookup



write down all images
in a table and record
the object label

“How many images are there?”

An tiny image example (can you see what it is?)

3

18

Each pixel has $2^8 = 256$ values

3x18 image above has 54 pixels

Total possible 54 pixel images = $256^{54} = 1.1 \times 10^{130}$

Prof. Kanade’s Theorem: we have not seen anything yet!

3

18

Total possible 54 pixel images $= 1.1 \times 10^{130}$

Compared

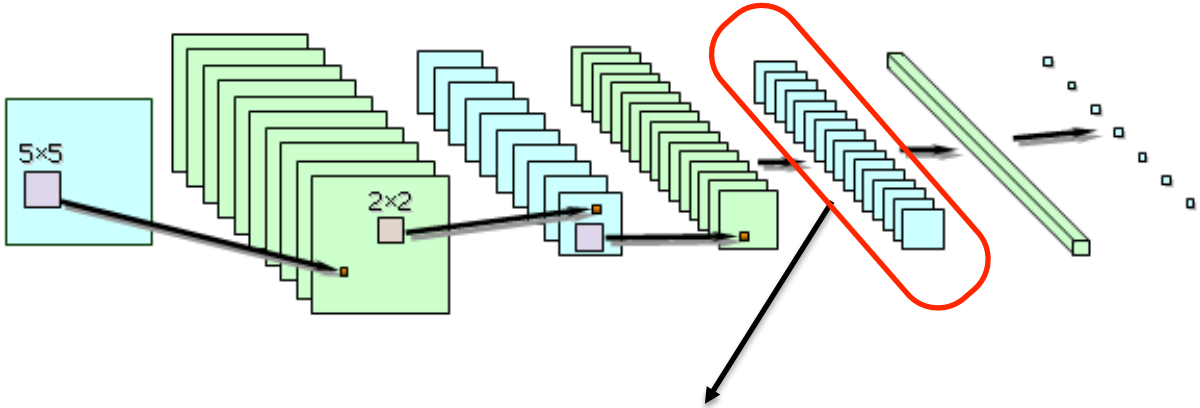
number of images seen by all humans ever:

10 billion x 1000 x 100 x 356 x 24 x 60 x 60 x 30 = 10^{24}

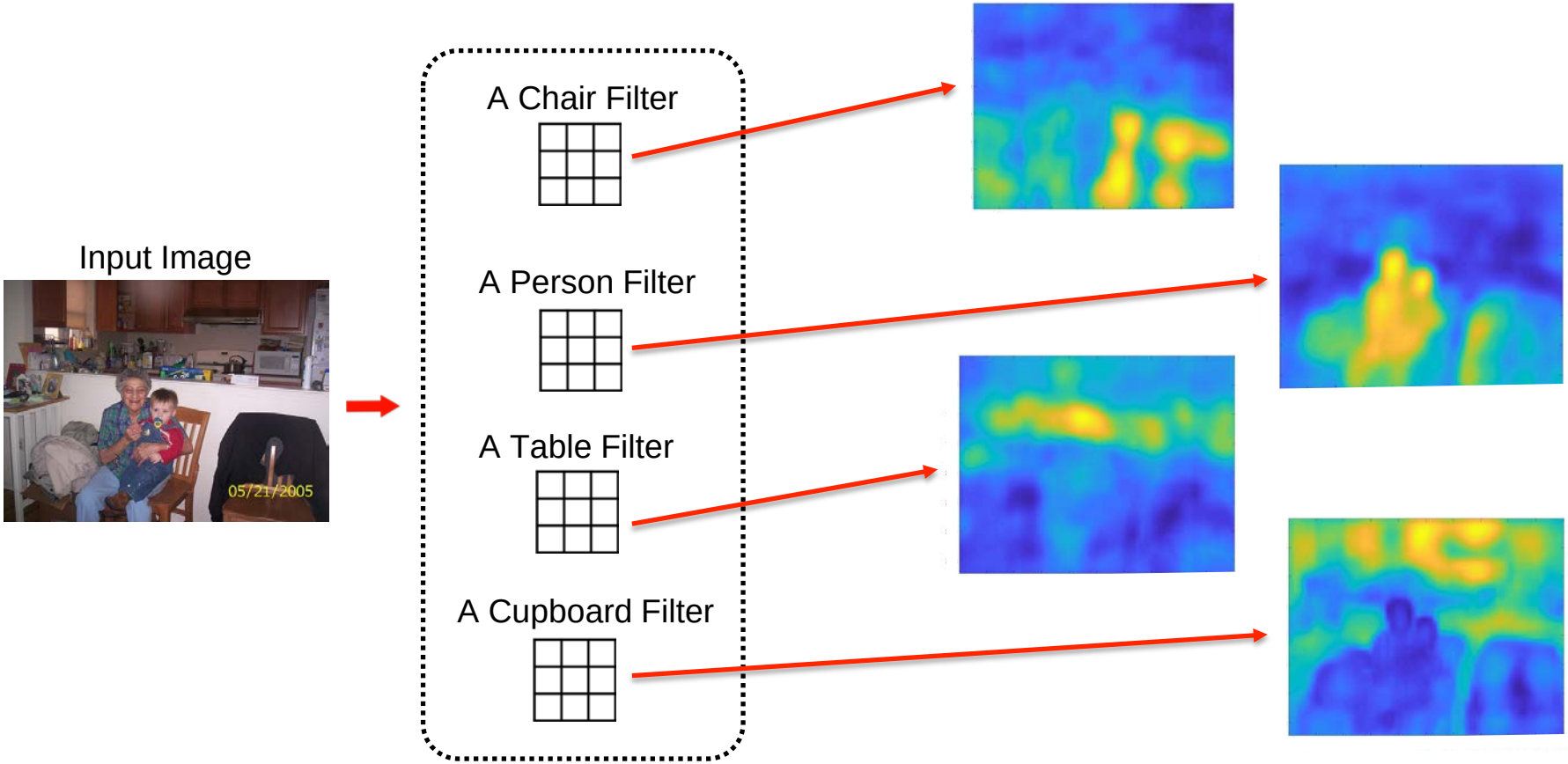
Total population years hours frame rate

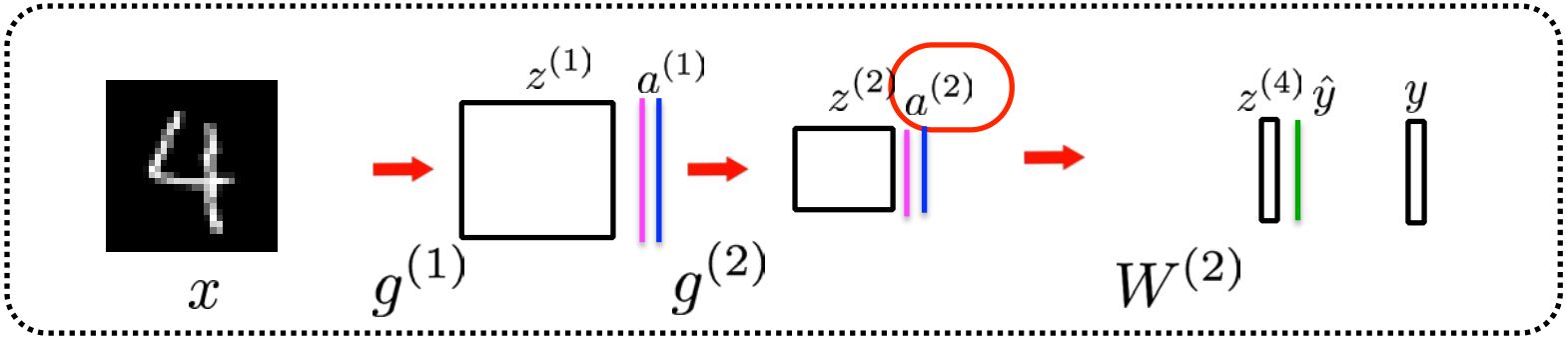
generations days min/sec

We have to be clever in writing down this table!

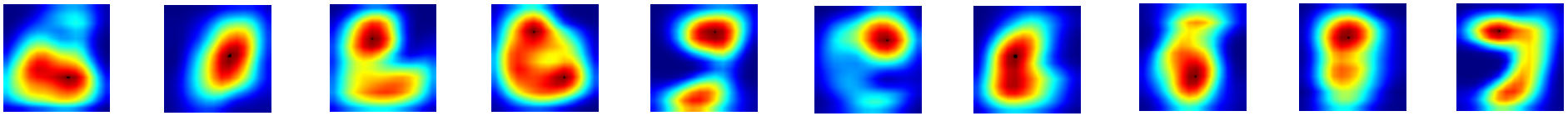


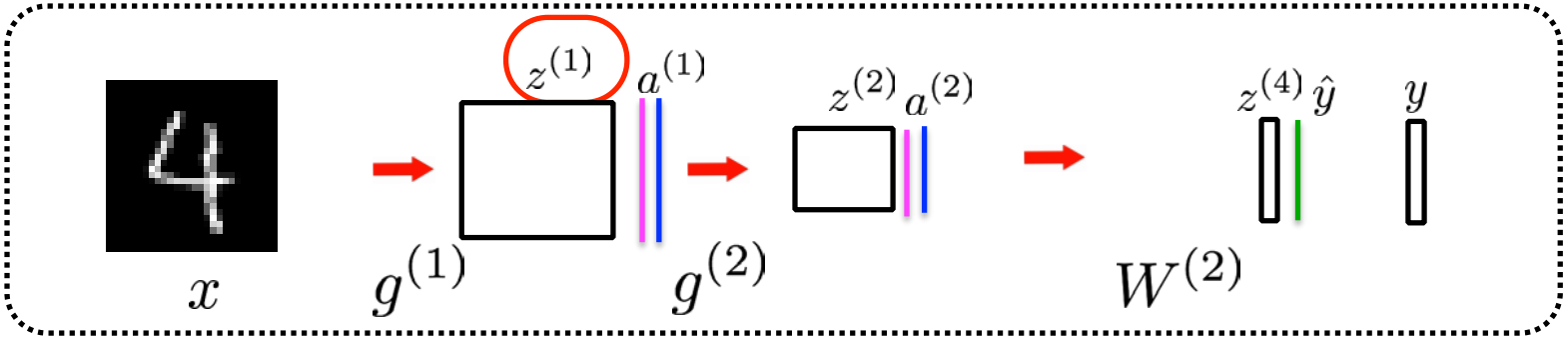
A Closer Look inside the Convolutional Layer





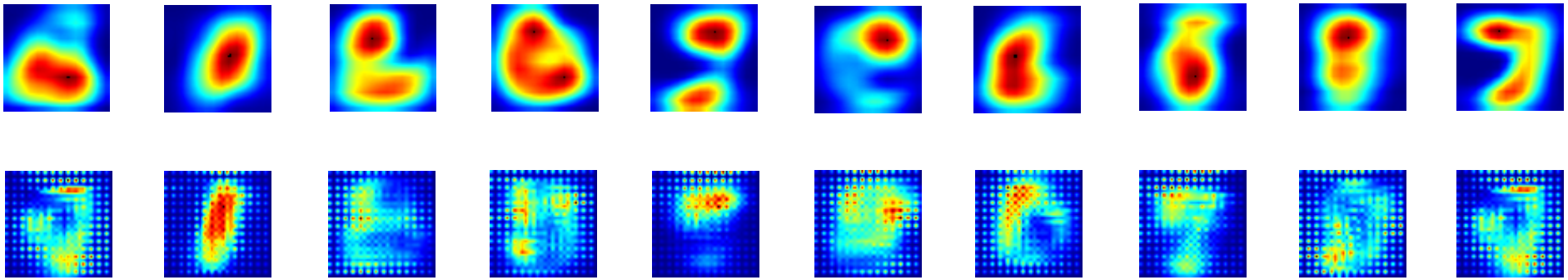
A Closer Look inside the Convolutional Layer :

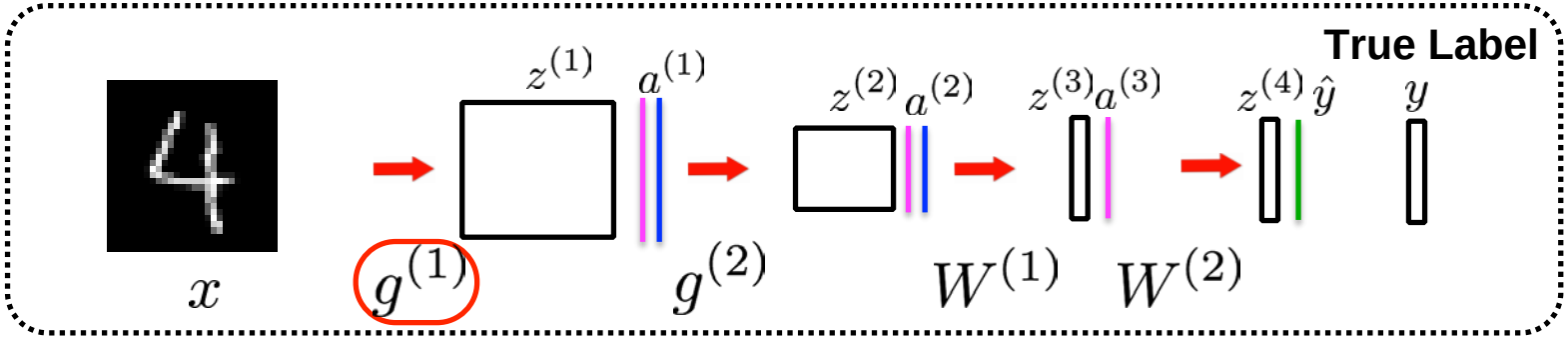




A Closer Look inside the Back Propagation Convolutional Layer :

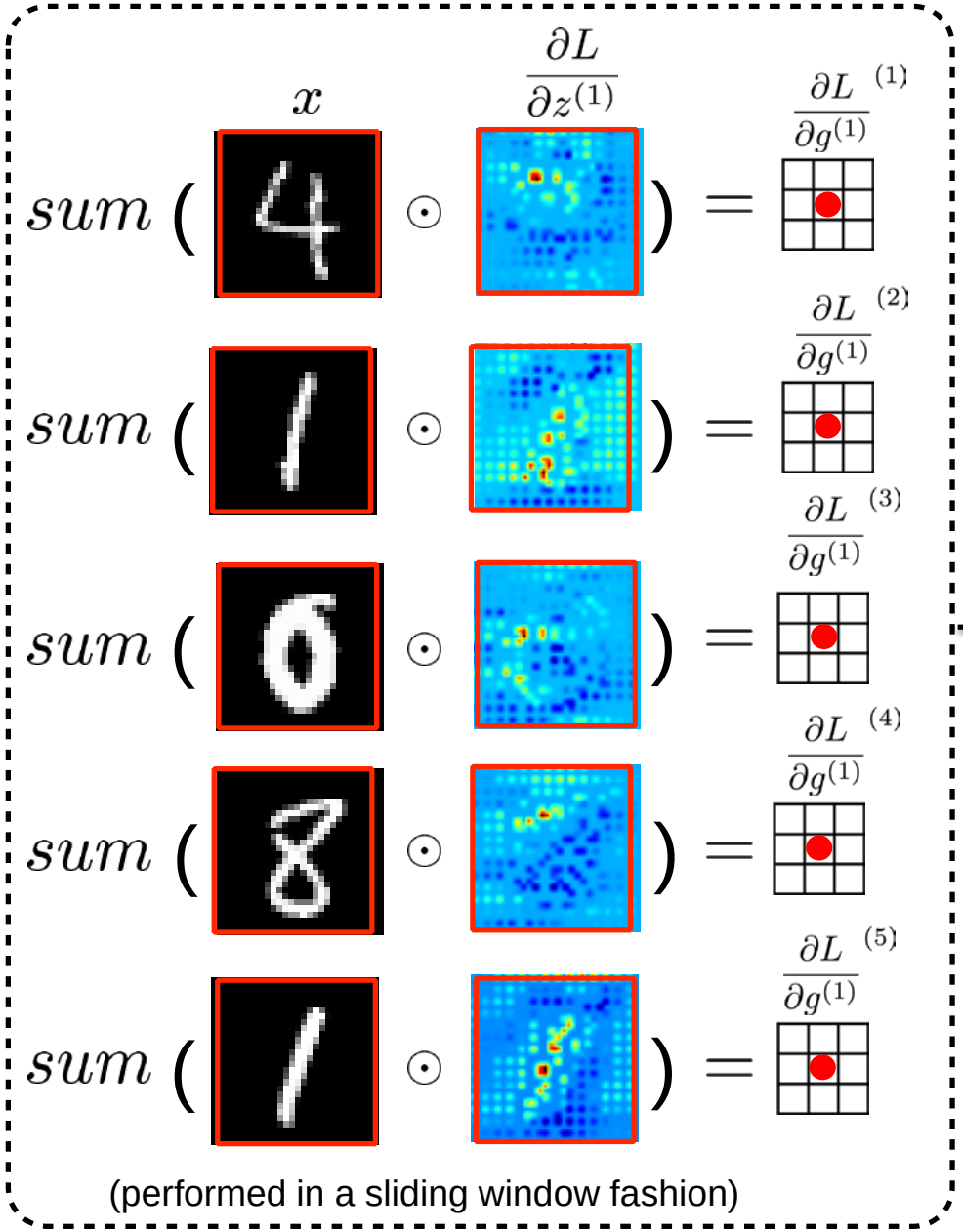
$$\frac{\partial L}{\partial z_{ij}^{(l)}} = \left(\sum_{m=0}^{M}\sum_{n=0}^{N} g_{mn}^{(l)} \frac{\partial L}{\partial z_{(i+m)(j+n)}^{(l+1)}}\right)\frac{\partial f(z_{ij}^{(l)})}{\partial z_{ij}^{(l)}}$$





Adjusting the weights:

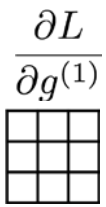
Training Batch



⊙ - elementwise multiplication

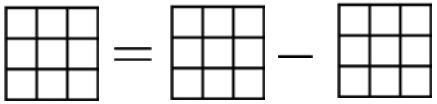
$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

Average the Gradient Across the Batch

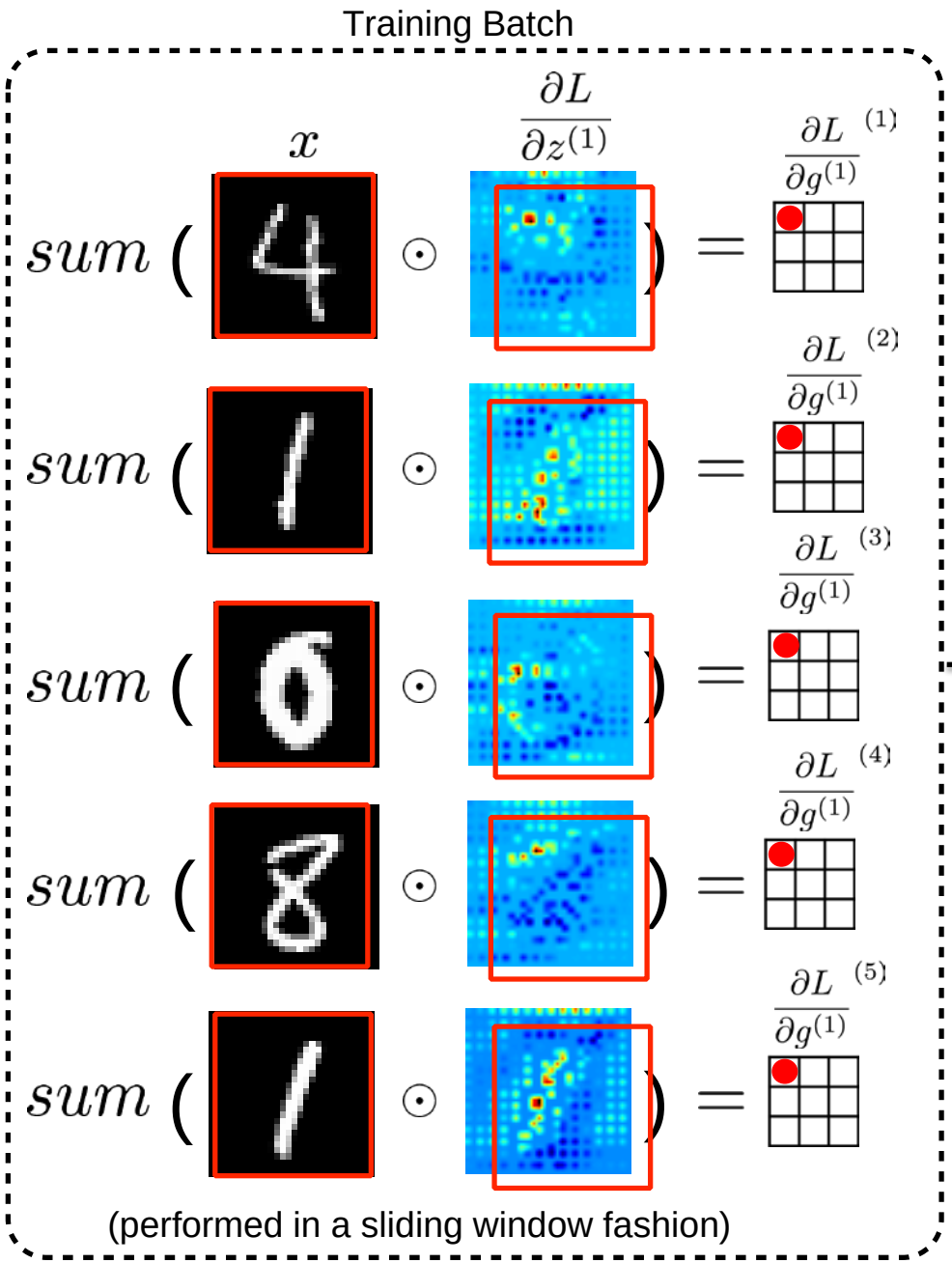


Parameter Update

$$g^{(1)} = g^{(1)} - \frac{\partial L}{\partial g^{(1)}}$$



new $g^{(1)}$ old $g^{(1)}$ $\frac{\partial L}{\partial g^{(1)}}$



$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

Average the Gradient Across the Batch

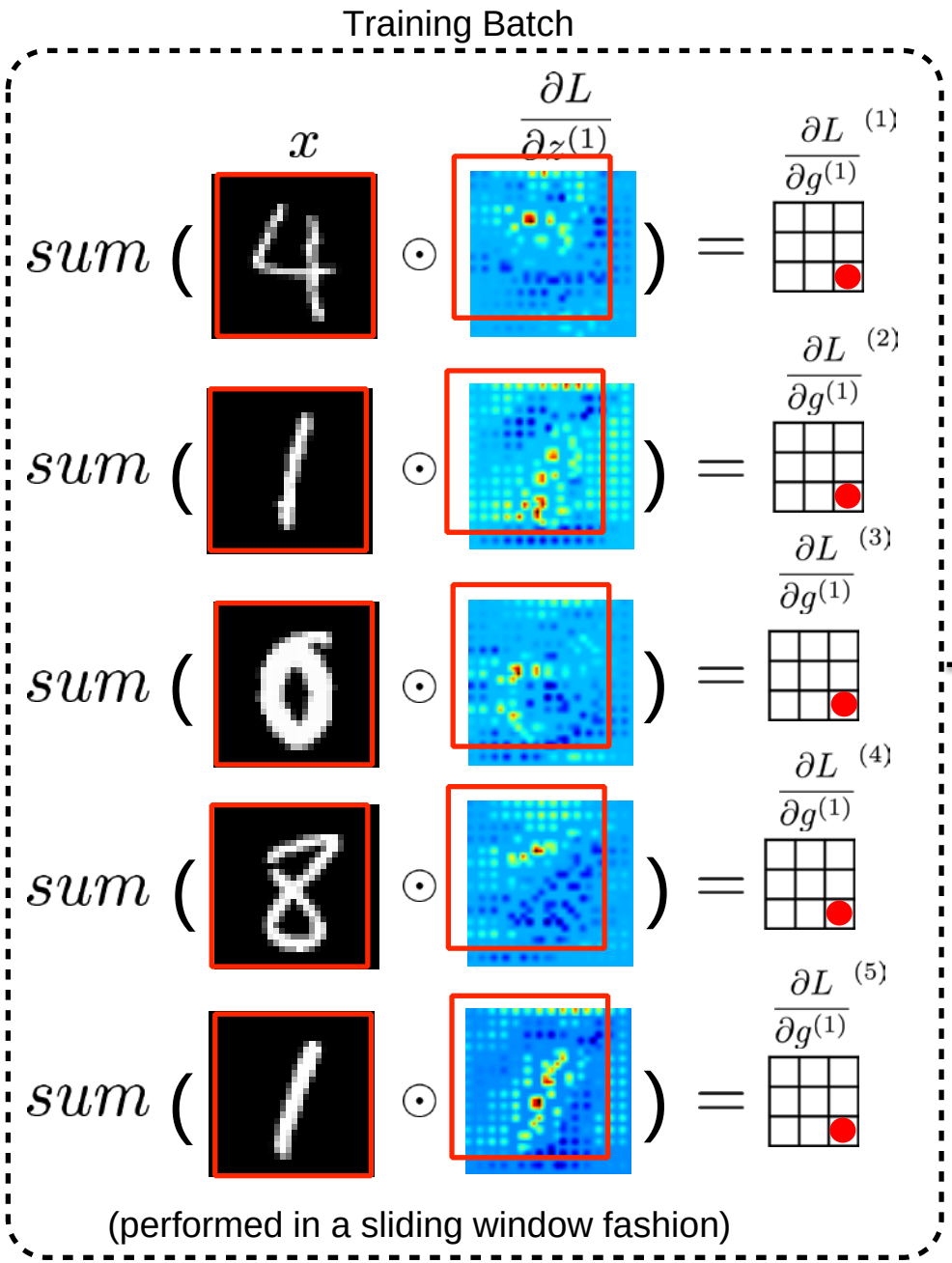
$\frac{\partial L}{\partial g^{(1)}}$

Parameter Update

$$g^{(1)} = g^{(1)} - \frac{\partial L}{\partial g^{(1)}}$$

new $g^{(1)}$ = old $g^{(1)}$ − $\frac{\partial L}{\partial g^{(1)}}$

$\odot$ - elementwise multiplication



$\odot$ - elementwise multiplication

$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

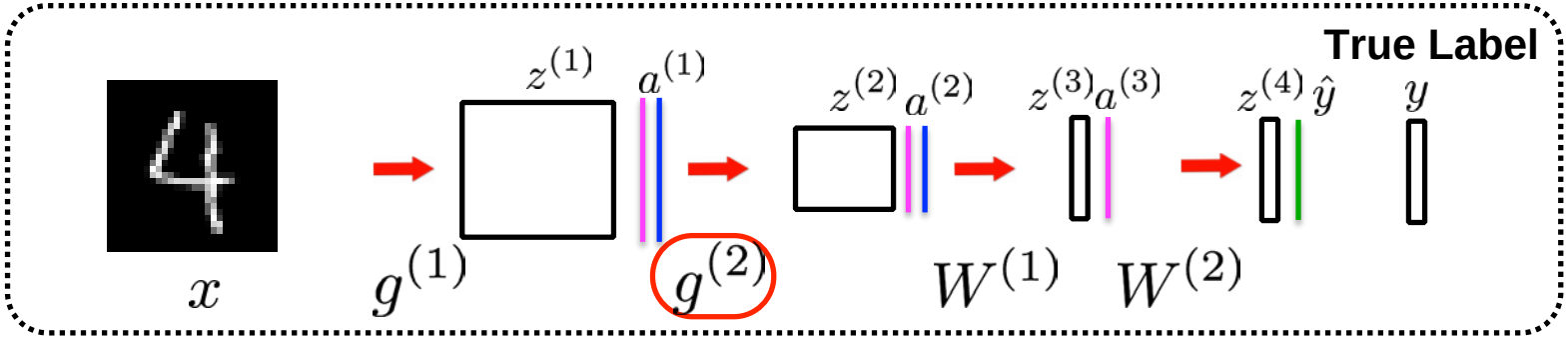
Average the Gradient Across the Batch

$\frac{\partial L}{\partial g^{(1)}}$

Parameter Update

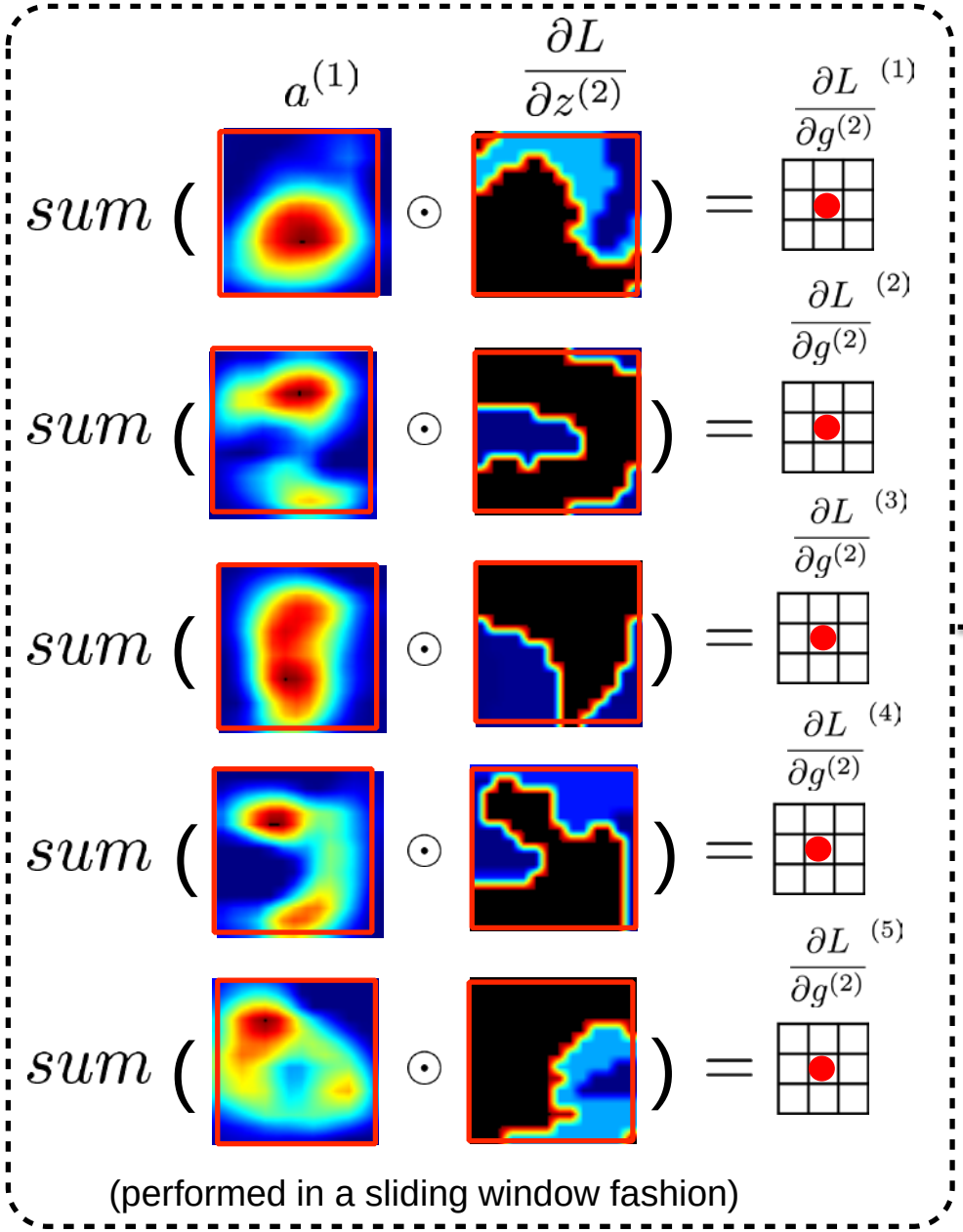
$$g^{(1)} = g^{(1)} - \frac{\partial L}{\partial g^{(1)}}$$

new $g^{(1)}$ = old $g^{(1)}$ − $\frac{\partial L}{\partial g^{(1)}}$



Adjusting the weights:

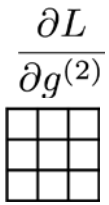
Training Batch



$\odot$ - elementwise multiplication

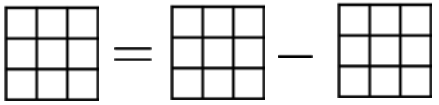
$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

Average the Gradient Across the Batch

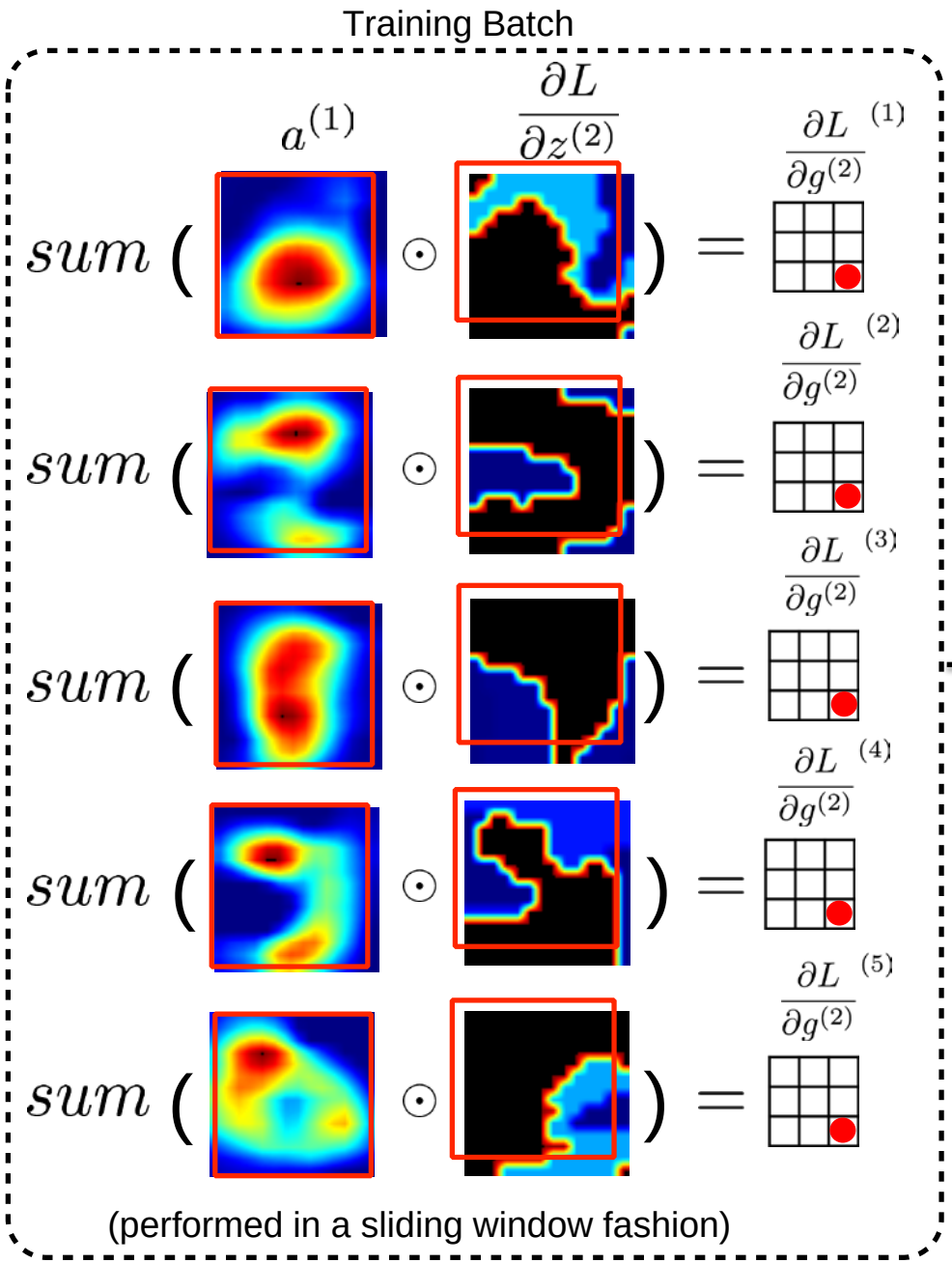


Parameter Update

$$g^{(2)} = g^{(2)} - \frac{\partial L}{\partial g^{(2)}}$$



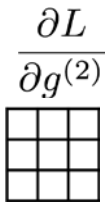
new $g^{(2)}$ old $g^{(2)}$ $\frac{\partial L}{\partial g^{(2)}}$



$\odot$ - elementwise multiplication

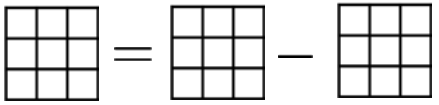
$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H} \sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$

Average the Gradient Across the Batch

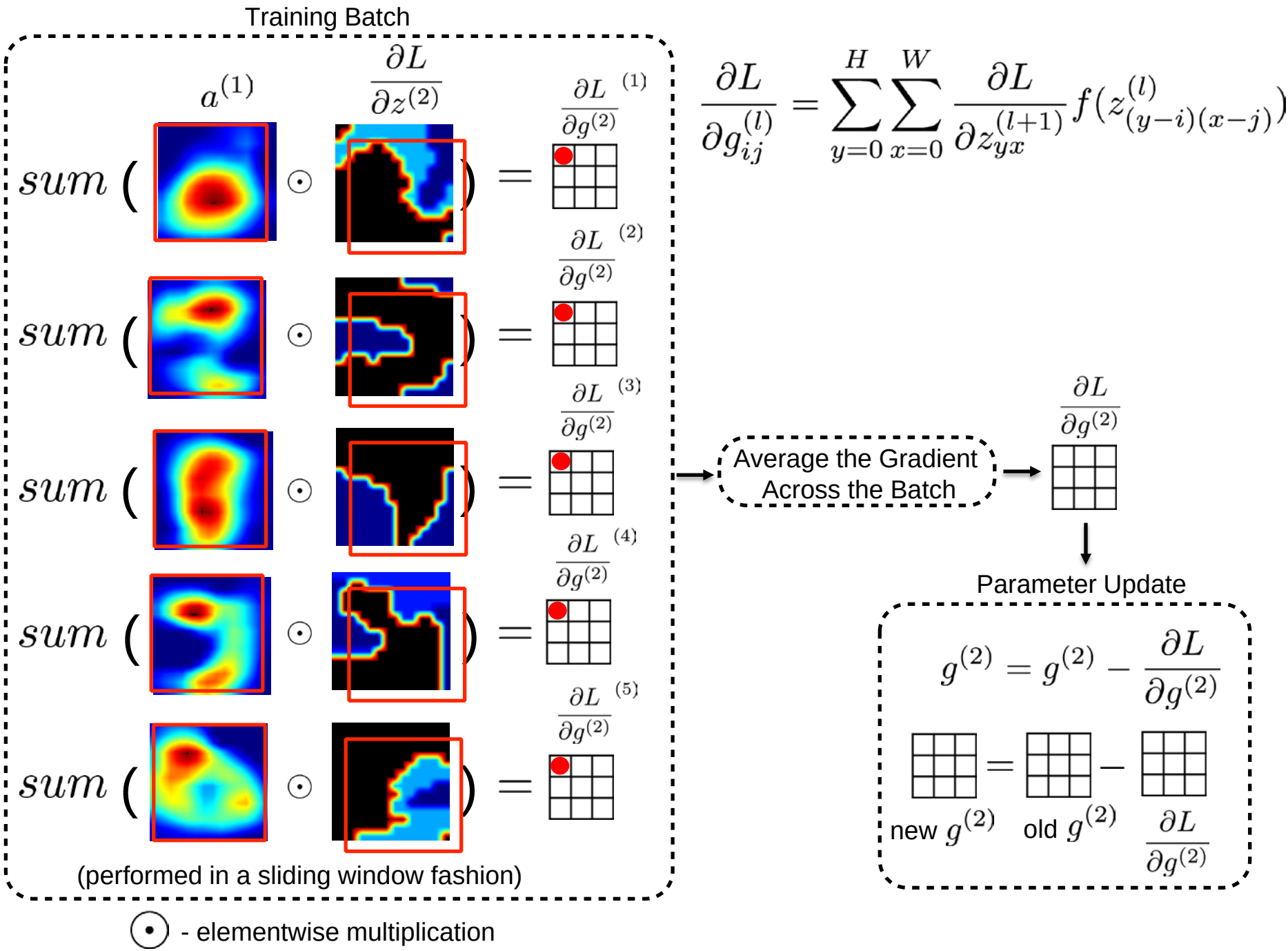


Parameter Update

$$g^{(2)} = g^{(2)} - \frac{\partial L}{\partial g^{(2)}}$$



new $g^{(2)}$ old $g^{(2)}$ $\frac{\partial L}{\partial g^{(2)}}$


Training Batch
$a^{(1)}$
$\frac{\partial L}{\partial z^{(2)}}$
$sum\,(\ \odot\ ) =$
$\frac{\partial L}{\partial g^{(2)}}^{(1)}$
$\frac{\partial L}{\partial g^{(2)}}^{(2)}$
$\frac{\partial L}{\partial g^{(2)}}^{(3)}$
$\frac{\partial L}{\partial g^{(2)}}^{(4)}$
$\frac{\partial L}{\partial g^{(2)}}^{(5)}$
(performed in a sliding window fashion)
$\odot$ - elementwise multiplication
$$\frac{\partial L}{\partial g_{ij}^{(l)}} = \sum_{y=0}^{H}\sum_{x=0}^{W} \frac{\partial L}{\partial z_{yx}^{(l+1)}} f(z_{(y-i)(x-j)}^{(l)})$$
Average the Gradient Across the Batch
$\frac{\partial L}{\partial g^{(2)}}$
Parameter Update
$$g^{(2)} = g^{(2)} - \frac{\partial L}{\partial g^{(2)}}$$
new $g^{(2)}$
old $g^{(2)}$
$\frac{\partial L}{\partial g^{(2)}}$