Detecting Improper Mask Wearing Using Convolutional Neural Networks

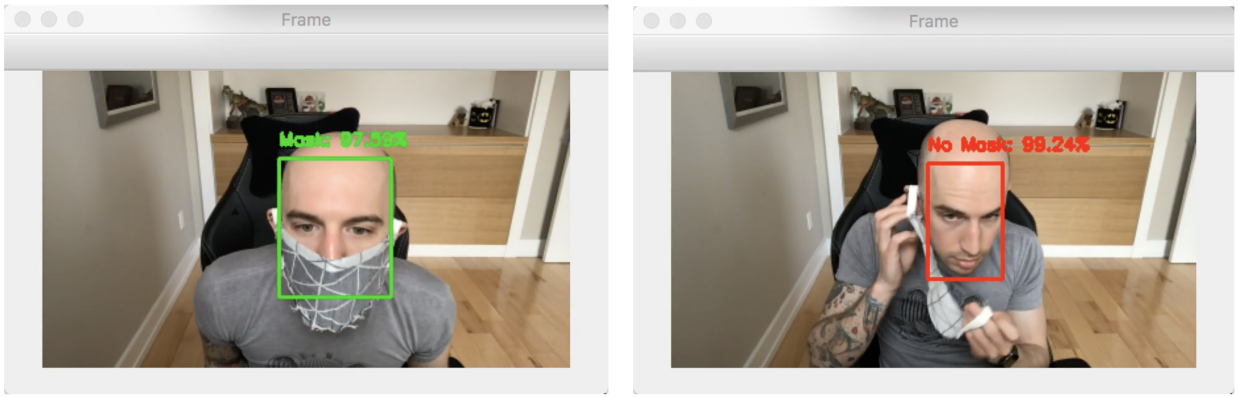
Jung Ho Park, Matt Zhang, Yao Zhang, Julie Zhu

Emory University

January 5, 2022

Why Classification?

- Identify if individuals wear masks!

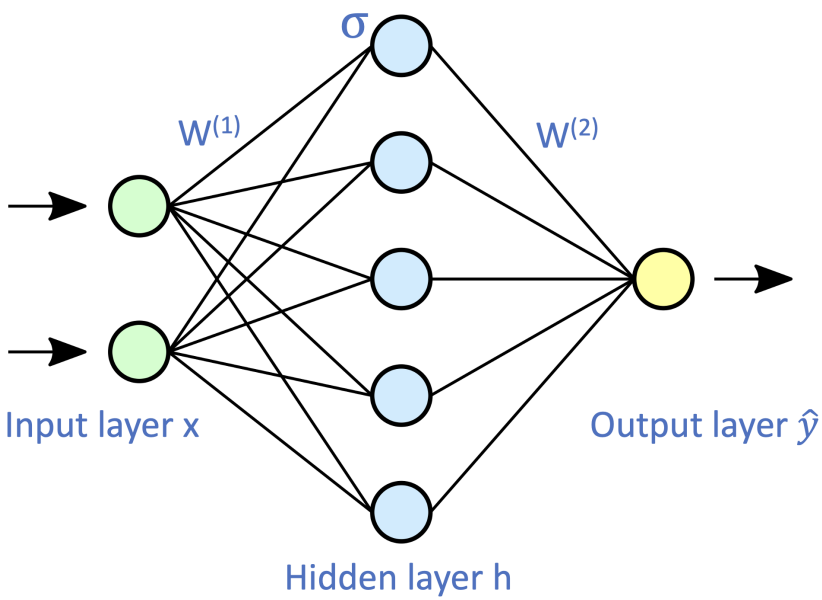


- Detect facial mask wearing at entrance of public spaces
- Protect public health

A. Rosebrock. *Covid-19: Face mask detector with opencv, Keras/TensorFlow, and Deep Learning*. 2021. URL: https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

Neural network setup

Training set data $(x^{(i)}, y^{(i)})$ where $i = 1, 2, \ldots N$



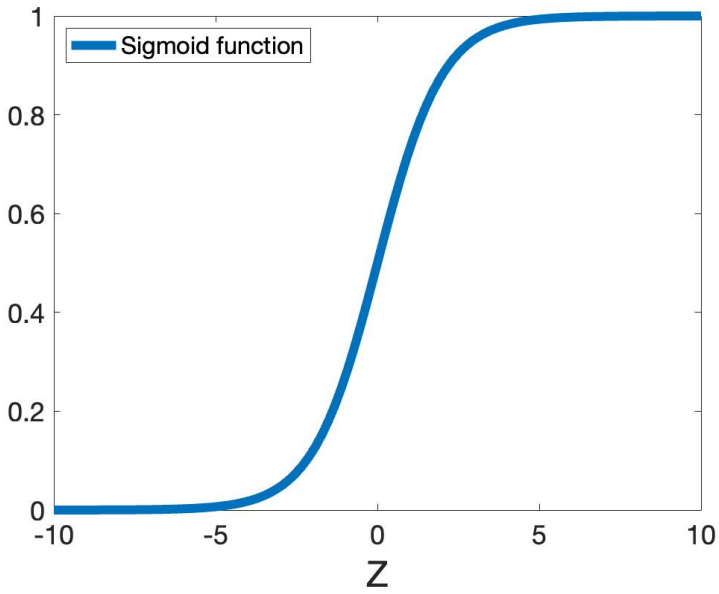
$$z^{(i)} = W^{(1)} x^{(i)}$$
$$h^{(i)} = \sigma(z^{(i)})$$
$$\hat{y}^{(i)} = W^{(2)} h^{(i)}$$

Sigmoid function $\sigma(z)$

- Adding sigmoid activation function to form a nonlinear model

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Sigmoid function is differentiable everywhere.



Our nonlinear optimization problem: minimize J

- Goal: Minimized the objective function J = L + S

$$\min_{W^{(1)},W^{(2)}} J(W^{(1)}, W^{(2)}) = \frac{1}{N}\sum_{i=1}^{N} l(\hat{y}^{(i)}, y^{(i)}) + S$$

where

$$z^{(i)} = W^{(1)}x^{(i)}$$
$$h^{(i)} = \sigma(z^{(i)})$$
$$\hat{y}^{(i)} = W^{(2)}h^{(i)}$$

- Add the regularization term S to J in order to deal with overfitting

$$S = \frac{\lambda}{2}(||W^{(1)}||_F^2 + ||W^{(2)}||_F^2)$$

Methods - Stochastic gradient descent algorithm

- Randomize the weights $W^{(1)}$ and $W^{(2)}$
- For i $= 1 :$ designated maximum epoch
  Shuffle the data set and break into B mini-batches
  For b $= 1 :$ B
  a. Calculate the objective function J_b with the mini-batch b
  b. Normal gradient descent with the selected mini-batch

$$\begin{pmatrix} W^{(1)}_{updated} \\ W^{(2)}_{updated} \end{pmatrix} = \begin{pmatrix} W^{(1)}_{prev} \\ W^{(2)}_{prev} \end{pmatrix} - \eta * \begin{pmatrix} \frac{\partial J_b}{\partial W^{(1)}_{prev}} \\ \frac{\partial J_b}{\partial W^{(2)}_{prev}} \end{pmatrix}$$

  end for
end for

Methods - Stochastic gradient descent

Why stochastic gradient descent instead of normal gradient descent?

- Add randomness to training data
- Avoid getting stuck at local minimum
- Reduce computation cost

Methods - Chain Rule to Calculate $\frac{\partial L}{\partial W^{(2)}}$ and $\frac{\partial L}{\partial W^{(1)}}$

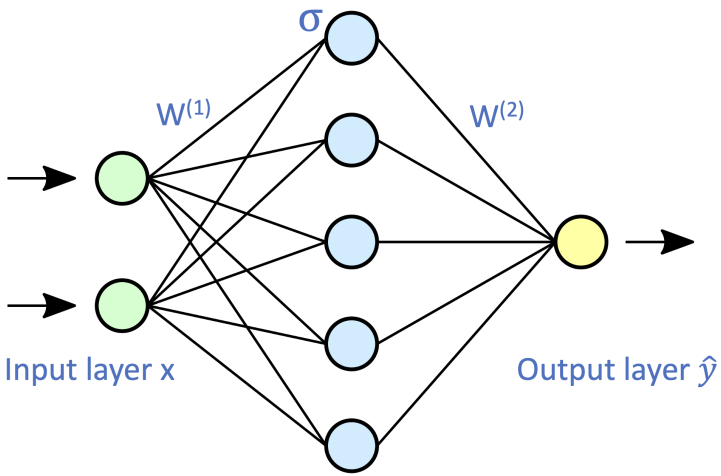
Forward Propagation

$$Z = W^{(1)}x^{(i)}$$
$$h = \sigma(Z)$$
$$\hat{y} = W^{(2)}h$$
$$L = l(\hat{y}, y)$$

Backward Propagation

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial W^{(2)}}$$

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h} * \frac{\partial h}{\partial z} * \frac{\partial z}{\partial W^{(1)}}$$



Intermediate steps - chain rule to calculate $\frac{\partial J}{\partial W^{(2)}}$ and $\frac{\partial J}{\partial W^{(1)}}$

- Recall that our objective function is to minimize J = L + S
- Our goal is to calculate $\frac{\partial J}{\partial W^{(2)}}$ and $\frac{\partial J}{\partial W^{(1)}}$
- Keep using chain rule, we have

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial L}\frac{\partial L}{\partial W^{(2)}} + \frac{\partial J}{\partial S}\frac{\partial S}{\partial W^{(2)}}$$

- Similarly, we have

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial L}\frac{\partial L}{\partial W^{(1)}} + \frac{\partial J}{\partial S}\frac{\partial S}{\partial W^{(1)}}$$

Intermediates steps - calculate $\frac{\partial S}{\partial W^{(2)}}$, $\frac{\partial S}{\partial W^{(1)}}$, and $\frac{\partial L}{\partial \hat{y}}$

- Our regularization S is $S = \frac{\lambda}{2}(||W^{(1)}||_F^2 + ||W^{(2)}||_F^2)$ and we compute

$$\frac{\partial S}{\partial W^{(1)}} = \lambda W^{(1)}$$

$$\frac{\partial S}{\partial W^{(2)}} = \lambda W^{(2)}$$

- Our loss function is $L = \frac{1}{2}||\hat{y} - y||_2^2$ and we compute

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

Specific procedures - calculate $\frac{\partial L}{\partial W^{(2)}}$

We can express the partial derivative of J w.r.t $W^{(2)}$ explicitly using the chain rule:

$$\begin{aligned}\frac{\partial J}{\partial W^{(2)}} &= \frac{\partial J}{\partial \hat{y}} h^T + \lambda W^{(2)} \\ &= \frac{\partial L}{\partial \hat{y}} h^T + \lambda W^{(2)} \\ &= (\hat{y} - y) h^T + \lambda W^{(2)}\end{aligned}$$

A. Zhang et al. *Dive into Deep Learning*. 2000. URL: https://d2l.ai/index.html

Specific procedures - calculate $\frac{\partial L}{\partial W^{(1)}}$

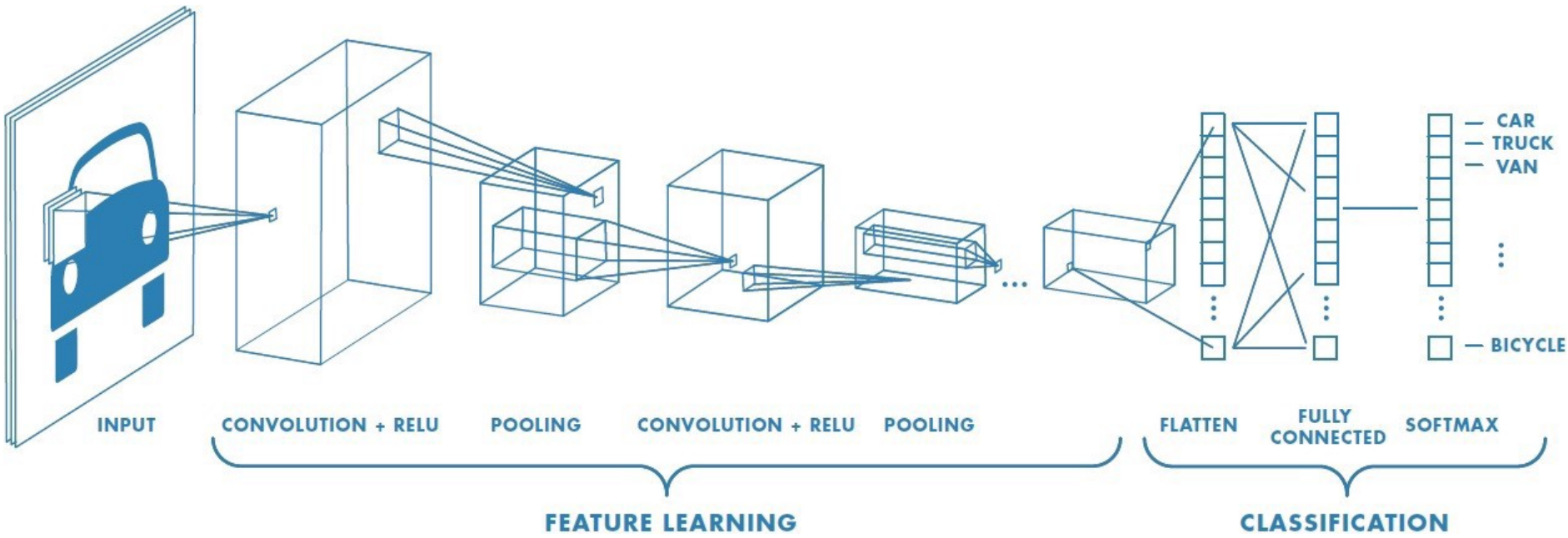
Similarly, we can express the partial derivative of J w.r.t $W^{(1)}$ explicitly using the chain rule:

$$
\begin{aligned}
\frac{\partial J}{\partial W^{(1)}} &= \frac{\partial J}{\partial z} x^T + \lambda W^{(1)} \\
&= \frac{\partial J}{\partial h} \odot \sigma'(z) x^T + \lambda W^{(1)} \\
&= \frac{\partial J}{\partial \hat{y}} W^{(2)\,T} \odot \sigma'(z) x^T + \lambda W^{(1)} \\
&= \frac{\partial L}{\partial \hat{y}} W^{(2)\,T} \odot \sigma'(z) x^T + \lambda W^{(1)} \\
&= (\hat{y} - y) W^{(2)\,T} \odot \sigma'(z) x^T + \lambda W^{(1)} \\
&= (\hat{y} - y) W^{(2)\,T} \odot \sigma(z)(1 - \sigma(z)) x^T + \lambda W^{(1)}
\end{aligned}
$$

Zhang et al., *Dive into Deep Learning*

Introduction to CNN

- Image filtering: scan the image with kernels
- Successive combination of convolution and pooling



S. Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Drawback of MLP and advantages of CNN

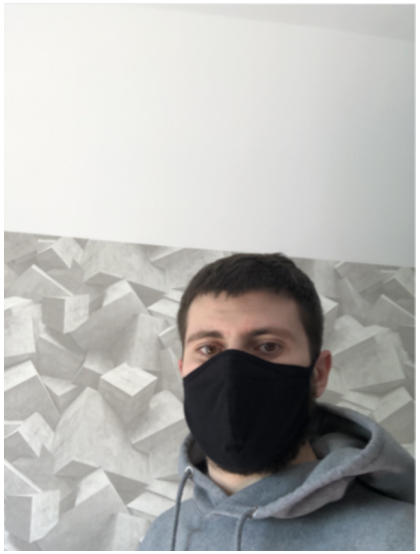
- Disadvantages of Multi-Layer Perceptron (Fully-Connected Layer): Computationally Costly
- Advantages of CNN (Convolutional Neural Network):
  1. Translation invariance
  – Same feature response to our filter everywhere in the image
  2. Locality
  – Proximity of pixels to each other is incorporated

Zhang et al., *Dive into Deep Learning*

Experiment: data sets

A total of 6877 images obtained from two data sets with corresponding Json file storing the labels for:

Properly Wearing



Improperly Wearing

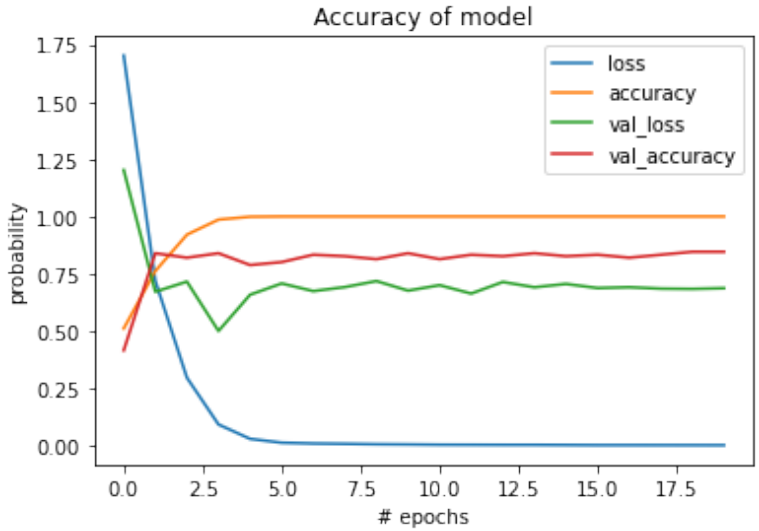
Not Wearing

Larxel. *Face Mask Detection*. 2019. URL: https://www.kaggle.com/andrewmvd/face-mask-detection

Medical Mask Dataset. URL: https://humansintheloop.org/resources/datasets/mask-dataset-download/?submissionGuid=137a4a93-d399-45c1-97a9-6339c945646a

Experiment: results

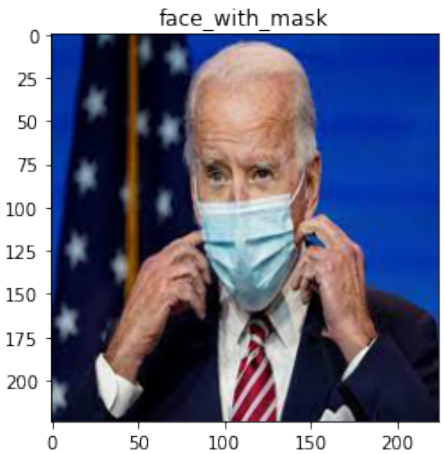
- Graph showing the accuracy of model after 20 epochs

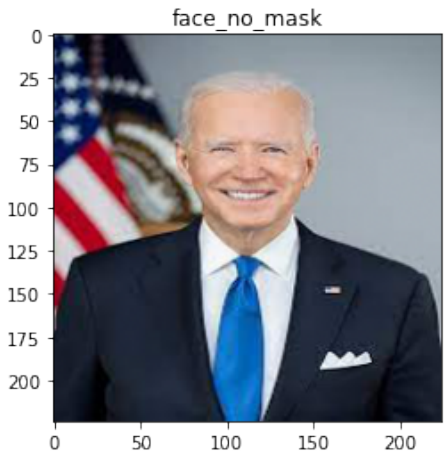


- Loss and accuracy of five epochs

```
Epoch 16/20
146/146 [==============================] - 439s 3s/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.6881 - val_accuracy: 0.
8333
Epoch 17/20
146/146 [==============================] - 439s 3s/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.6917 - val_accuracy: 0.
8205
Epoch 18/20
146/146 [==============================] - 453s 3s/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.6855 - val_accuracy: 0.
8333
Epoch 19/20
146/146 [==============================] - 466s 3s/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.6844 - val_accuracy: 0.
8462
Epoch 20/20
146/146 [==============================] - 465s 3s/step - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.6874 - val_accuracy: 0.
8462
```

Conclusion





- Our model is able to classify mask wearing with 85% accuracy
- Image localization could be added to improve the model
- Our project is meaningful in terms of recognizing improper mask wearing during the pandemic

References


Larxel. *Face Mask Detection*. 2019. URL: https://www.kaggle.com/andrewmvd/face-mask-detection.


Medical Mask Dataset. URL: https://humansintheloop.org/resources/datasets/mask-dataset-download/?submissionGuid=137a4a93-d399-45c1-97a9-6339c945646a.


Rosebrock, A. *Covid-19: Face mask detector with opencv, Keras/TensorFlow, and Deep Learning*. 2021. URL: https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/.


Saha, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.


Zhang, A. et al. *Dive into Deep Learning*. 2000. URL: https://d2l.ai/index.html.

Nonlinear Optimization problem: minimize J

Forward propagation:

$Z = W^{(1)}x^{(i)}$ where $W^{(1)} \in^{h \times d}$ and $x \in^{d}$

$h = \sigma(Z)$

$\hat{y} = W^{(2)}h$ where $W^{(2)} \in^{q \times h}$ and $h \in^{h}$

$L = l(\hat{y}, y)W^{(2)}h$ where $\hat{y} \in^{q}$ and $y \in^{q}$

$S = \frac{\lambda}{2}||W^{(1)}||_F^2 + ||W^{(2)}||_F^2$

Zhang et al., *Dive into Deep Learning*

Nonlinear Optimization problem: minimize J

Backpropagation (Chain Rule):

1. Compute the gradient of the objective function J w.r.t L and S
$\frac{\partial J}{\partial L} = 1$ and $\frac{\partial J}{\partial S} = 1$
2. Compute the gradient of J w.r.t output layer $\hat{y}$ by the chain rule
$\frac{\partial J}{\partial \hat{y}} = \text{prod}(\frac{\partial J}{\partial L}, \frac{\partial L}{\partial \hat{y}}) = \frac{\partial L}{\partial \hat{y}}$

Zhang et al., *Dive into Deep Learning*

Nonlinear Optimization problem: minimize J

Backpropagation (Chain Rule):

3. Compute the gradients of the regularization term w.r.t both $W^{(1)}$ and $W^{(2)}$

$\frac{\partial S}{\partial W^{(1)}} = \lambda W^{(1)}$ and $\frac{\partial S}{\partial W^{(2)}} = \lambda W^{(2)}$

4. Compute the gradient of J w.r.t $W^{(2)}$

$\frac{\partial J}{\partial W^{(2)}} = \text{prod}(\frac{\partial J}{\partial \hat{y}}, \frac{\partial \hat{y}}{\partial W^{(2)}}) + \text{prod}(\frac{\partial J}{\partial S}, \frac{\partial S}{\partial W^{(2)}}) = \frac{\partial J}{\partial \hat{y}} h^T + \lambda W^{(2)}$

Zhang et al., *Dive into Deep Learning*

Nonlinear Optimization problem: minimize J

Backpropagation (Chain Rule):

5. Compute the gradient of J w.r.t h

$\frac{\partial J}{\partial h} = \text{prod}(\frac{\partial J}{\partial \hat{y}}, \frac{\partial \hat{y}}{\partial h}) = W^{(2)T} \frac{\partial J}{\partial \hat{y}}$

6. Compute the gradient of J w.r.t z (since the activation function σ applies to z elementwise, we need to do the elementwise product)

$\frac{\partial J}{\partial z} = \text{prod}(\frac{\partial J}{\partial h}, \frac{\partial h}{\partial z}) = \frac{\partial J}{\partial h} \odot \sigma'(z)$

Zhang et al., *Dive into Deep Learning*

Nonlinear Optimization problem: minimize J

Backpropagation (Chain Rule):

7. Compute the gradient of J w.r.t $W^{(1)}$

$$\frac{\partial J}{\partial W^{(1)}} = \text{prod}(\frac{\partial J}{\partial z}, \frac{\partial z}{\partial W^{(1)}}) + \text{prod}(\frac{\partial J}{\partial S}, \frac{\partial S}{\partial W^{(1)}}) = \frac{\partial J}{\partial z} x^T + \lambda W^{(1)}$$

Zhang et al., *Dive into Deep Learning*