
University of Colorado
Boulder

Deep Learning Applications for Computer Vision

Lecture 13: The Image Classification Pipeline

Neural Networks for Image Classification

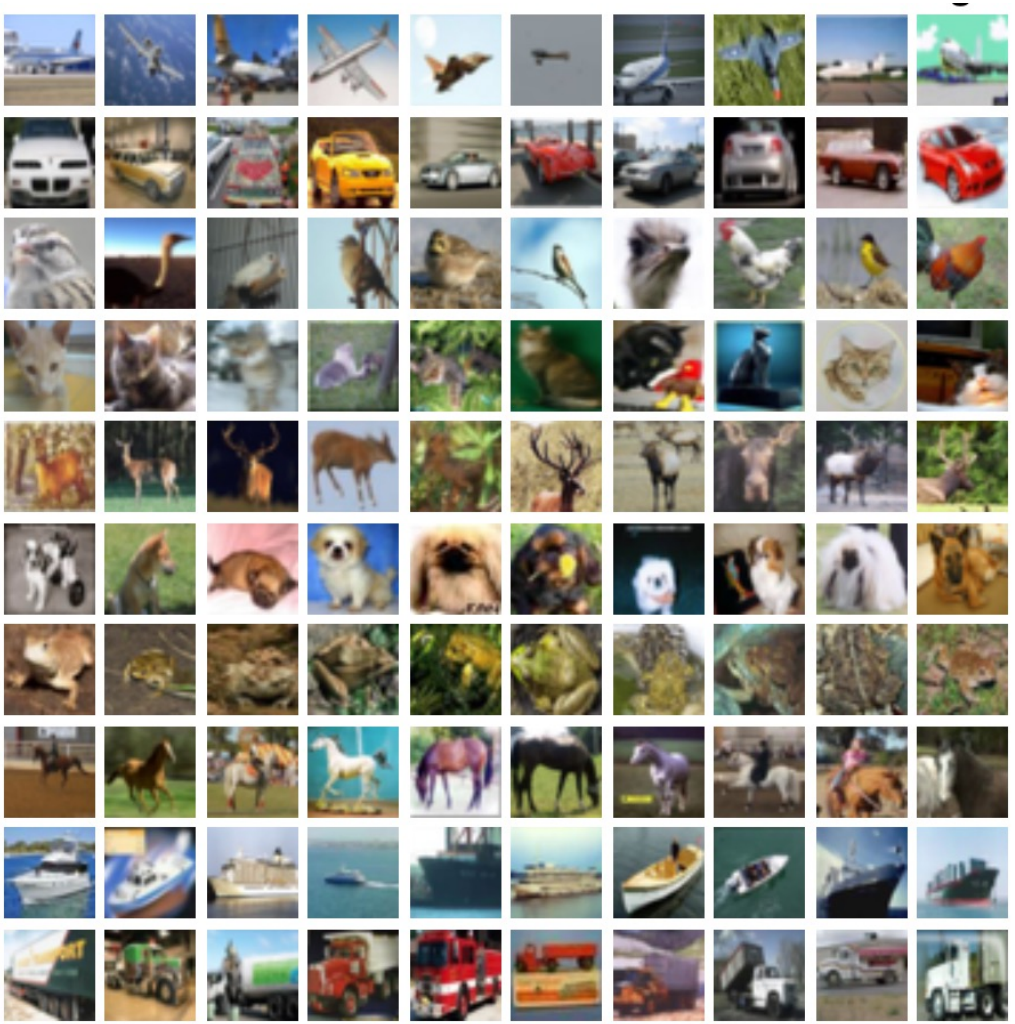
Last time: How do we train a neural network?

Today: Krizhevski, A: "*Learning multiple layers of features from tiny images*", 2009

Image Classification with the CIFAR-10 dataset

(https://www.cs.toronto.edu/~kriz/cifar.html)

[C-FAR]



Neural Networks for Image Classification

Today:

- Example: Image Classification with 10 categories
- CIFAR-10 dataset: "airplane, automobile (but not truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (but not pickup truck)"
  - 50 000 training images (5000 per class)
  - 10 000 test images
  - image size: 32 x 32 pixels

60k

Krizhevski, A, "*Learning multiple layers of features from tiny images*", 2009

Network parameters

- Assume only one layer
- Output layer: 10 neurons = 10 labels

Input

Data
Raw data
Image data

32 x 32 x 3

x: 3072 (0 .. 1)

10 Functions
Output Layer

$F_1(x, W_1) = W_1 \cdot x + b_1 = s_1$ (3072 | 3072 x 1)

$F_2 \quad W_2 \quad + b_2 = s_2$

$F_{10} \quad W_{10} \quad + b_{10} = s_{10}$

10 x 3072 | 3072 x 1 | 10

Total # parameters: 30720 + 10 = 30 730

Even Smaller Example

- Image w/ 4 pixel-values
- Classifier w/ 3 labels: frog, dog, cat

	W_2				x_i		b		S	
	1		3	4						
1)	0.2	0.3	-0.7	0.2	0		-3.5		5.6	s_1
2)	-0.2	0.12	-0.3	-0.2	255	+	8.7	=	-11.6	s_2
3)	1.2	0.3	0.01	-0.9	125		-8		9.45	s_3
					67					

What happens with the weighted sum?

Cost function, error function, objective function

- Loss function: how happy are we with the scores?

known label ↑ assigned ↗

$$P(Y=k \mid X=x_i) = \frac{e^{s_k}}{\sum e^{s_k}}$$

- Example: Softmax classifier
  - takes a vector of arbitrary scores and turns them into values from 0 to 1, which sum to 1
  - uses cross-entropy loss: minimize the negative log likelihood of the correct class choice

$$L_i = -\log\left(P(Y=k \mid X=x_i)\right)$$

$s = f(x, W)$

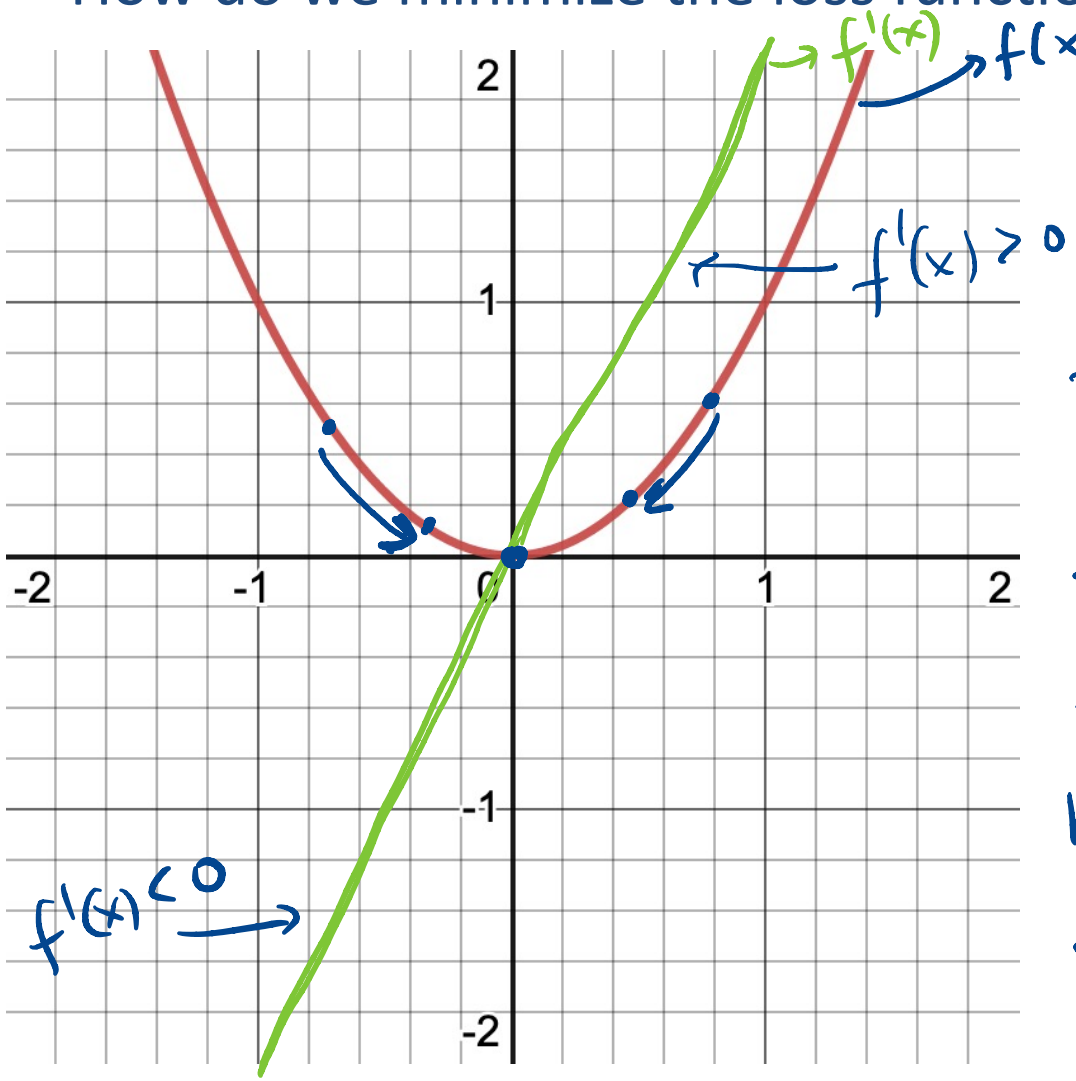
Sum to 1 Between 0 and 1

		e^{s_k} →		$/\sum e^{s_k}$ →	
s_1	5.6		270.43		0.02
s_2	-11.6		9.17e-6		7.06e-10
s_3	9.45		12708.17		0.979

↑ → $-\log(0.979) = 0.0009$

Backpropagation and Gradient Descent

- How do we minimize the loss function?



$f(x) = x^2$

$f'(x) = \frac{dy}{dx} = 2x$

$f(x+\varepsilon) \approx f(x) + \boxed{\varepsilon * f'(x)}$

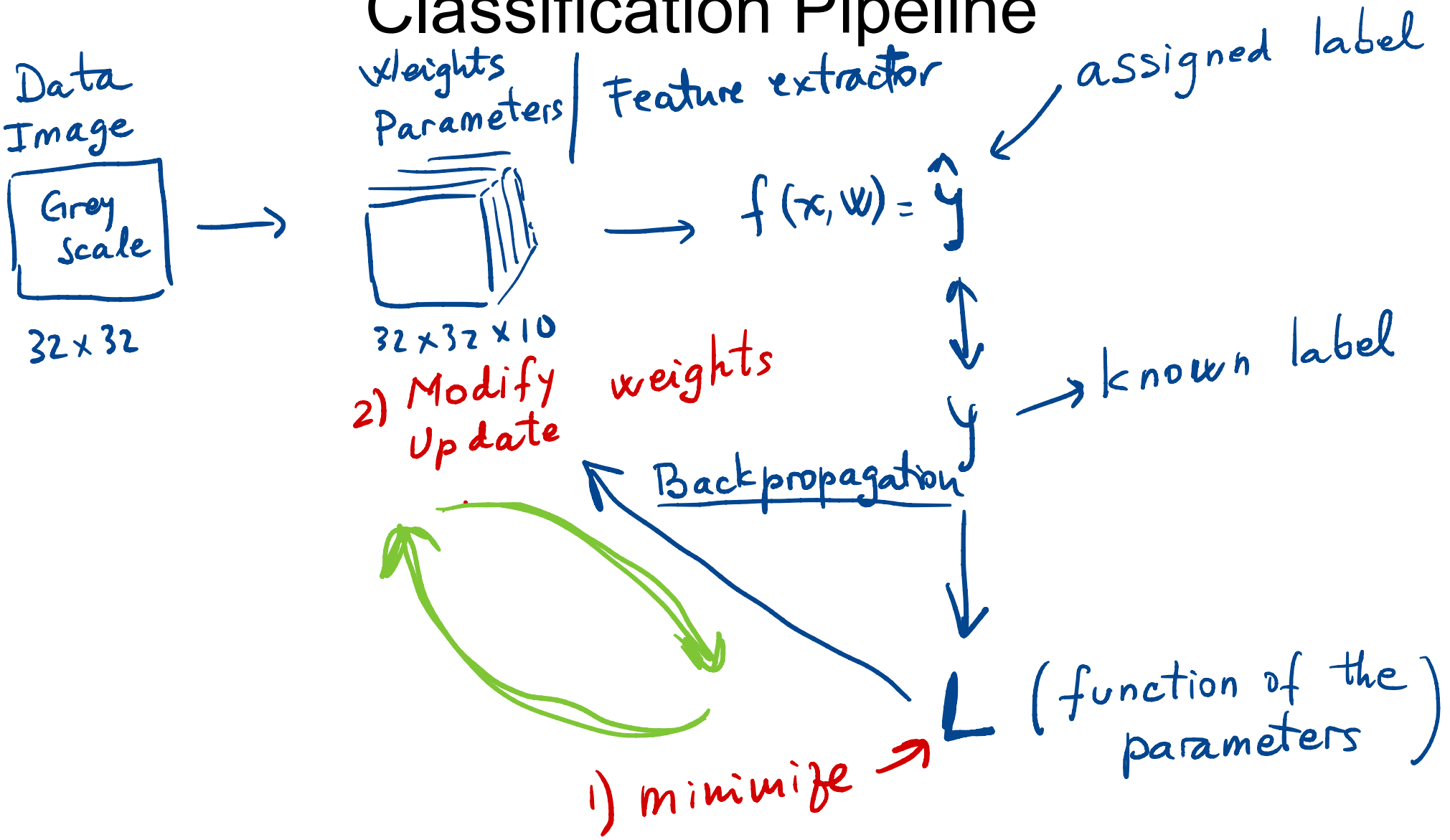
< 0

→ if $f'(x) < 0 \Rightarrow \varepsilon > 0$

→ if $f'(x) > 0 \Rightarrow \varepsilon < 0$

Keep moving x in small steps with the opposite sign of $f'(x)$ = Gradient Descent

Classification Pipeline



System performance

- Time to train
- Overall accuracy
- Best/worst class accuracy