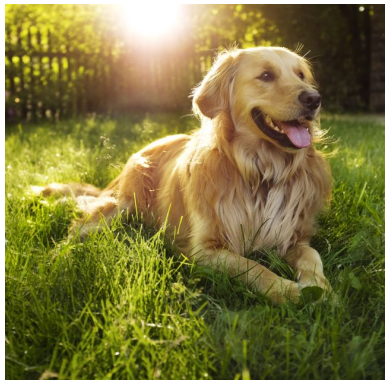
Machine Learning

EECS 442

Fall 2020, University of Michigan

Image Classification

Input Image



Output Class

dog
cat
mountain
ship
.
.

dog
cat
mountain
ship
.
.

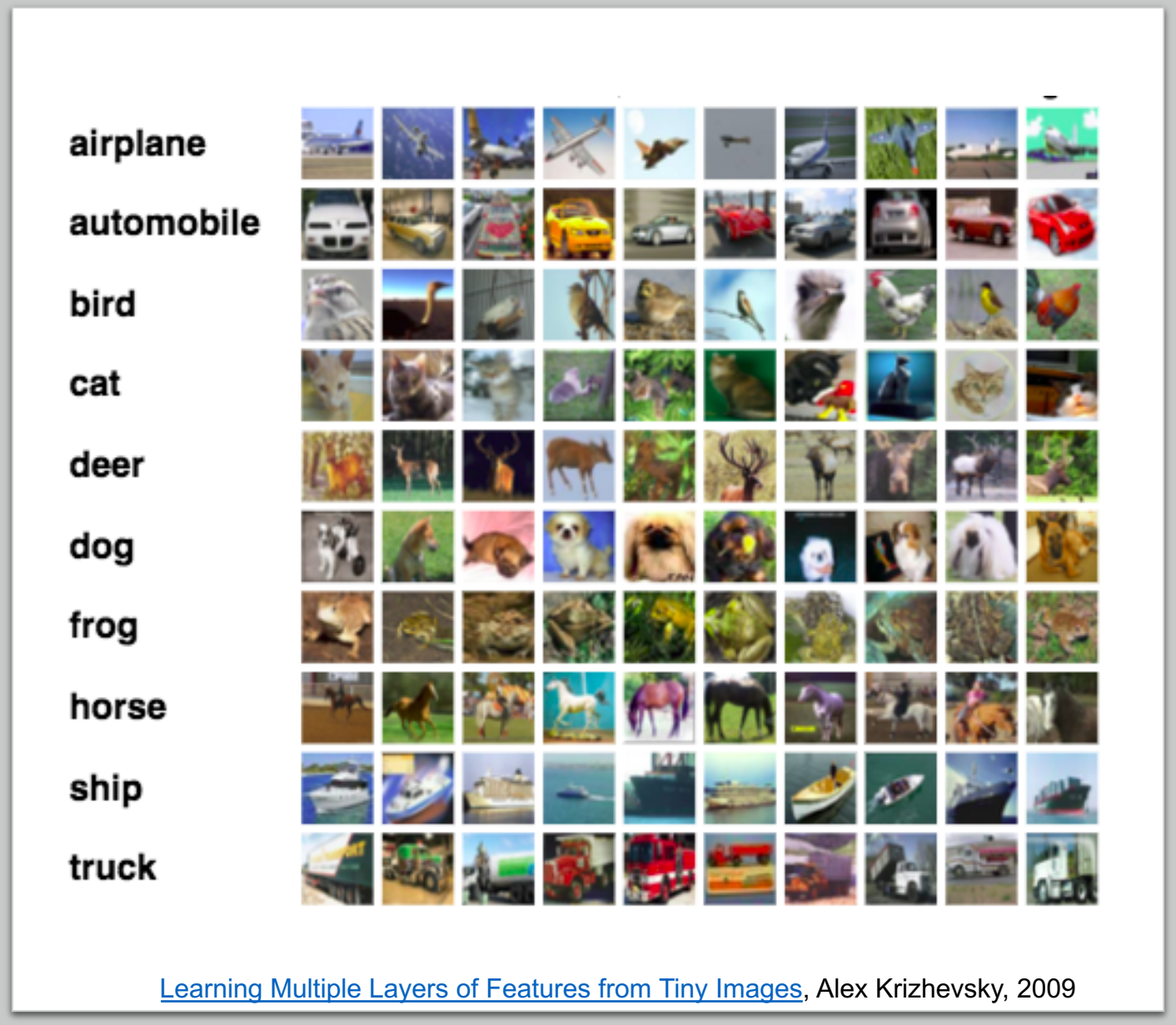
Example dataset: CIFAR10

Images: 32x32x3 (RGB)

10 classes

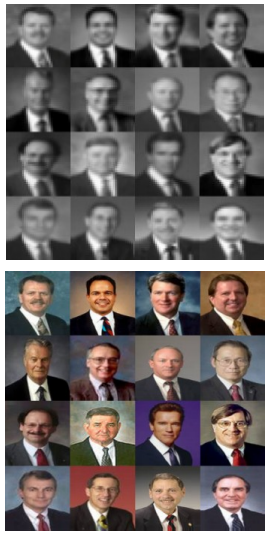
Training set: 5k images per class

Test set: 1k images per class



Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009

K-Nearest Neighbor Classification



How do we calculate the distance between two images?

$$\left\| \begin{array}{|c|c|}\hline 25 & 243 \\ \hline 13 & 9 \\ \hline \end{array} - \begin{array}{|c|c|}\hline 10 & 190 \\ \hline 20 & 52 \\ \hline \end{array} \right\|_2$$

Square of difference

225	2809
49	1849

Add and take square root

$= \sqrt{225 + 2809 + 49 + 1849}$

$= 70.23$

Called the L2 Distance between two images.

L1 distance between two images

-

Absolute differences

15	53
7	43

Add the absolute differences

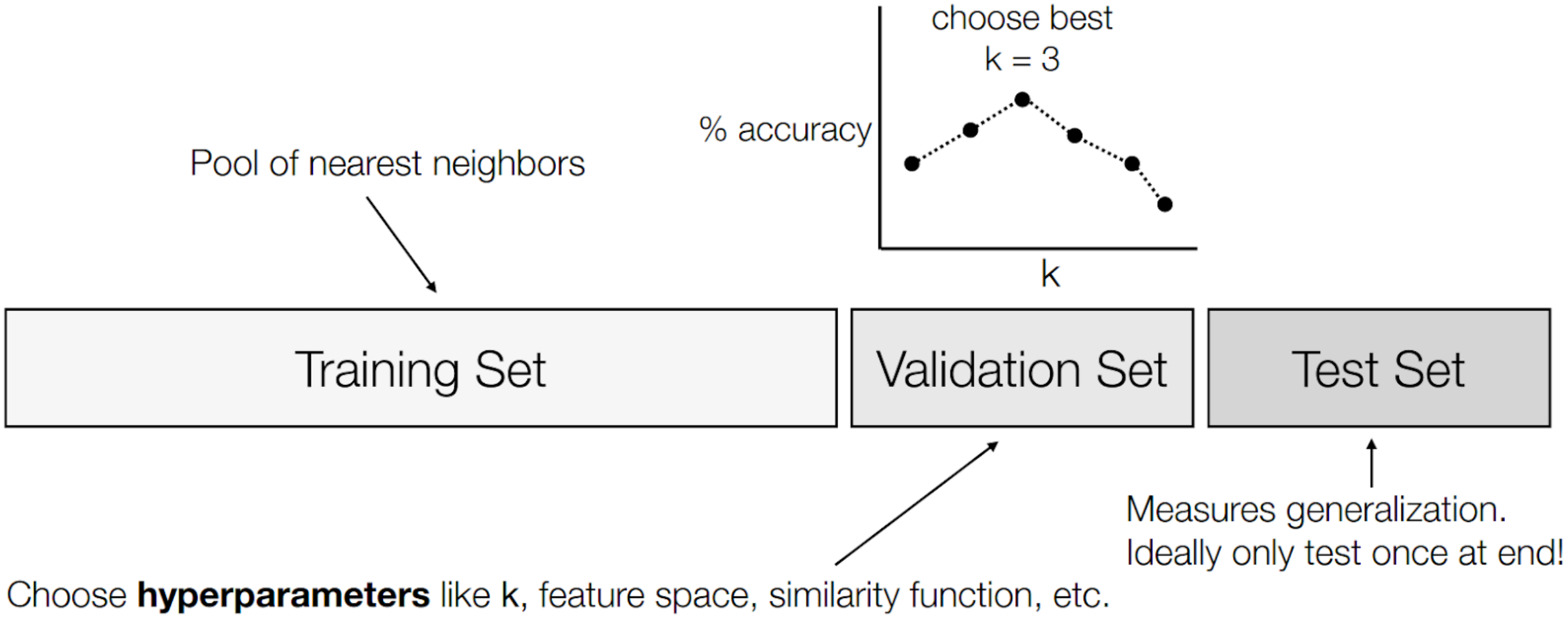
= 15 + 53 + 7 + 43
= 118

Called the L1 Distance between two images.

Recap: How do we choose k?

- The goal of a machine learning system is for the system to generalize in the wild by training on a training dataset which is hopefully representative of the samples to which we want to apply our system.
- How do we get a measure of how good our system will be?
- We calculate its accuracy on a “test set” which is done **only once** at the end of training.
- How do we tune our hyperparameters (like ‘k’) then?

Training, Validation and Test Set

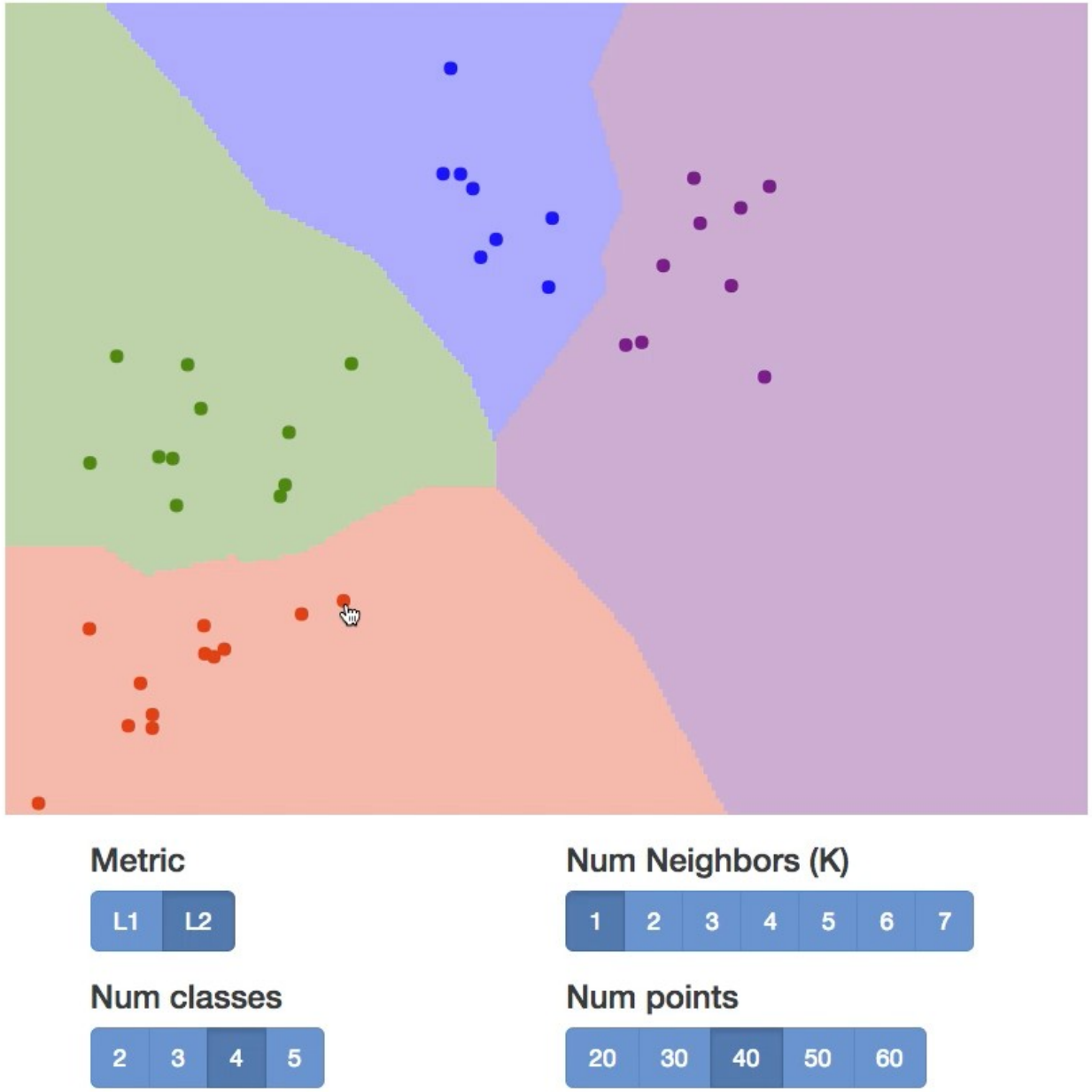


K-Nearest Neighbors: Web Demo

Interactively move points around and see decision boundaries change

Play with L1 vs L2 metrics

Play with changing number of training points, value of K

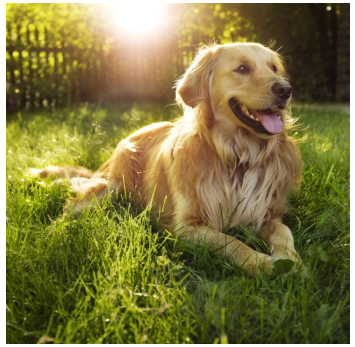


http://vision.stanford.edu/teaching/cs231n-demos/knn/

Slide credits: Justin Johnson

Linear classifiers

Linear Classifier



x : (32x32x3)

x : (3072,)

Linear Classifier
f(x) = Wx + b

W: (10, 3072)
b: (10,)

250.75	dog
30.5	cat
67.325	mountain
-20.25	ship
.	.
.	.

Example 2x2 image, 3 classes (dog, cat, mountain)

Flatten the image / Vectorize the matrix

Input Image
(2,2)

0.5	0.2	1.5	-1.0
2.6	-1.2	0.6	1.8
0.25	0.0	3.2	0.7

W
(3, 4)

25
243
13
9

x
(4,)

+

1.5
1.0
-2.5

b
(3,)

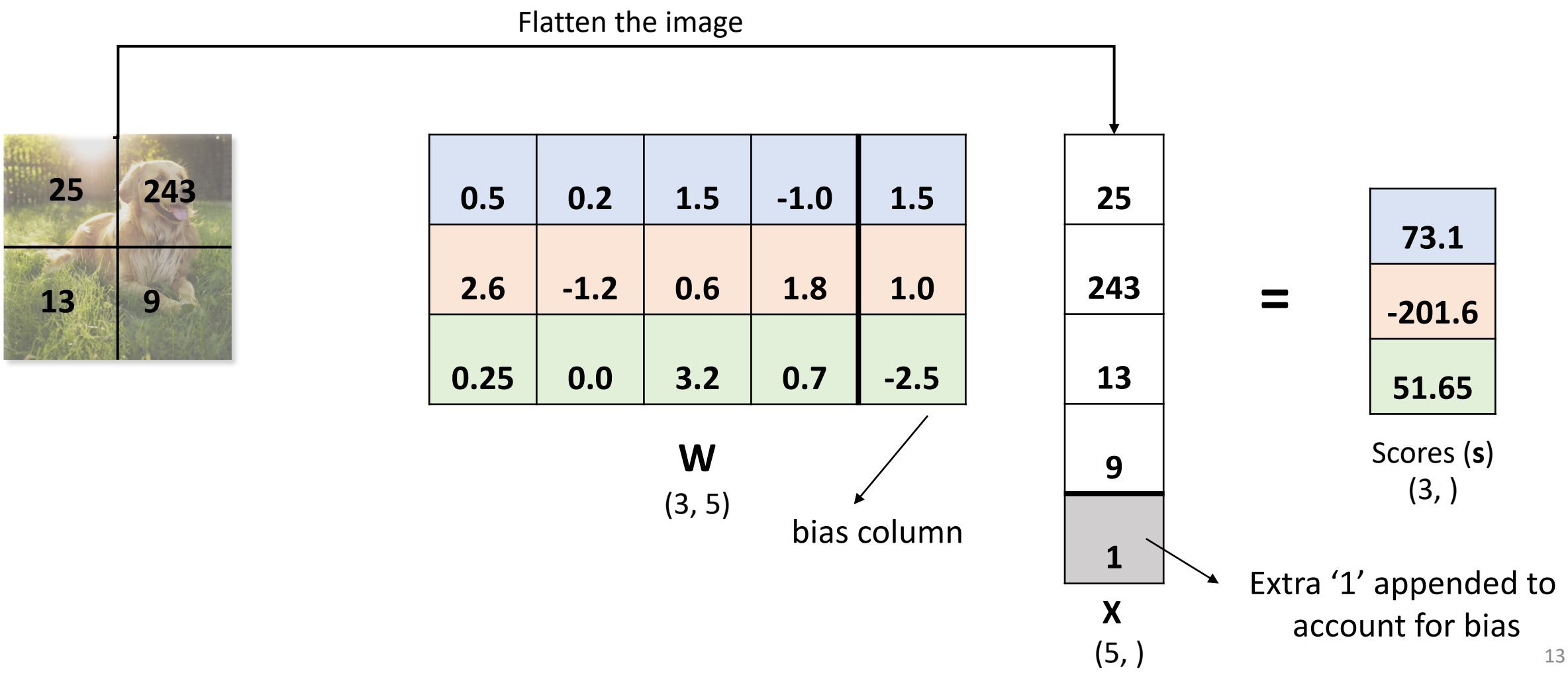
=

73.1
-201.6
51.65

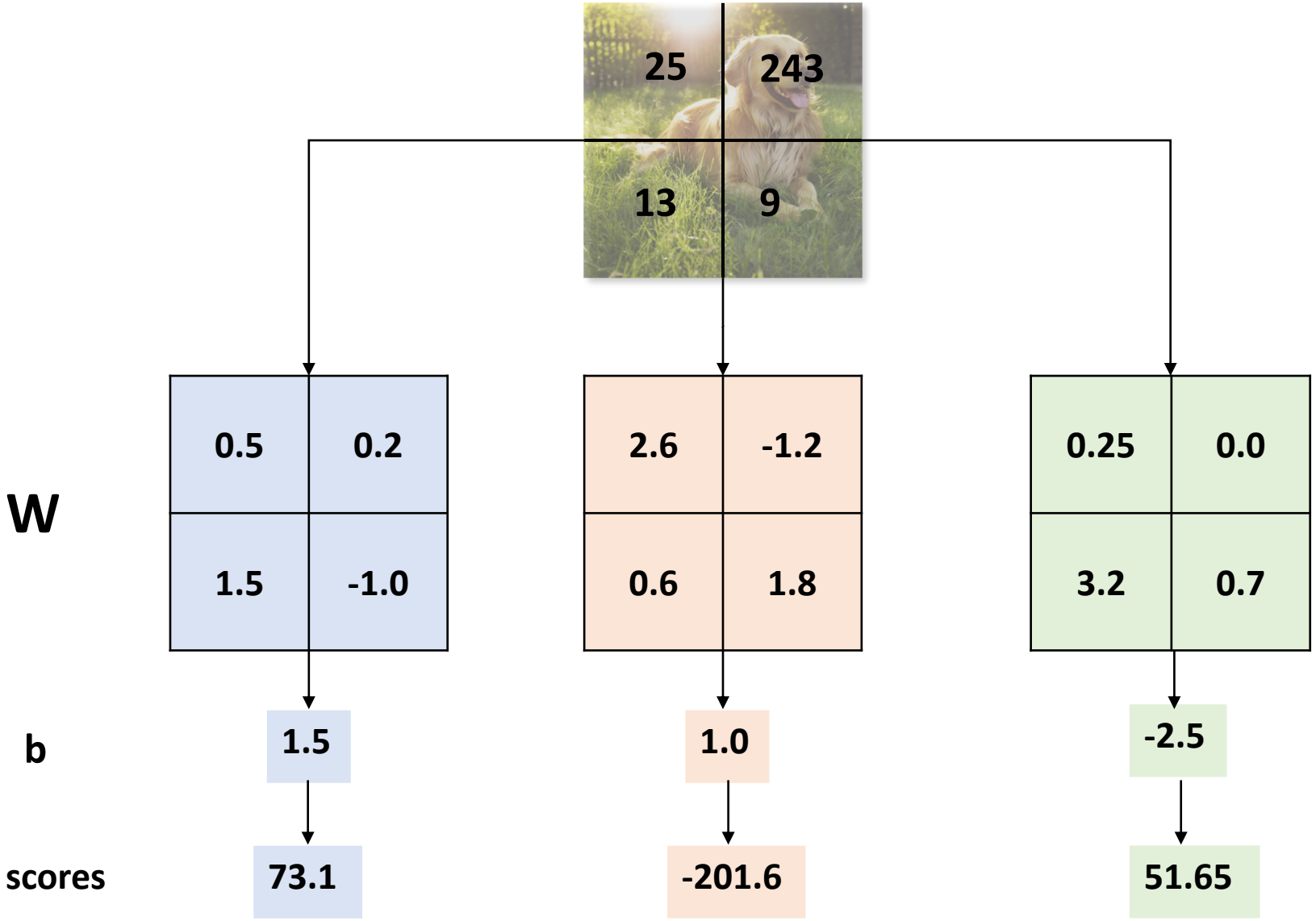
Scores (**s**)
(3,)

Idea credit: Justin Johnson

Bias Trick


Flatten the image
25
243
13
9
0.5
0.2
1.5
-1.0
1.5
2.6
-1.2
0.6
1.8
1.0
0.25
0.0
3.2
0.7
-2.5
W
(3, 5)
bias column
25
243
13
9
1
X
(5,)
Extra '1' appended to account for bias
=
73.1
-201.6
51.65
Scores (s)
(3,)

Visualizing the weights



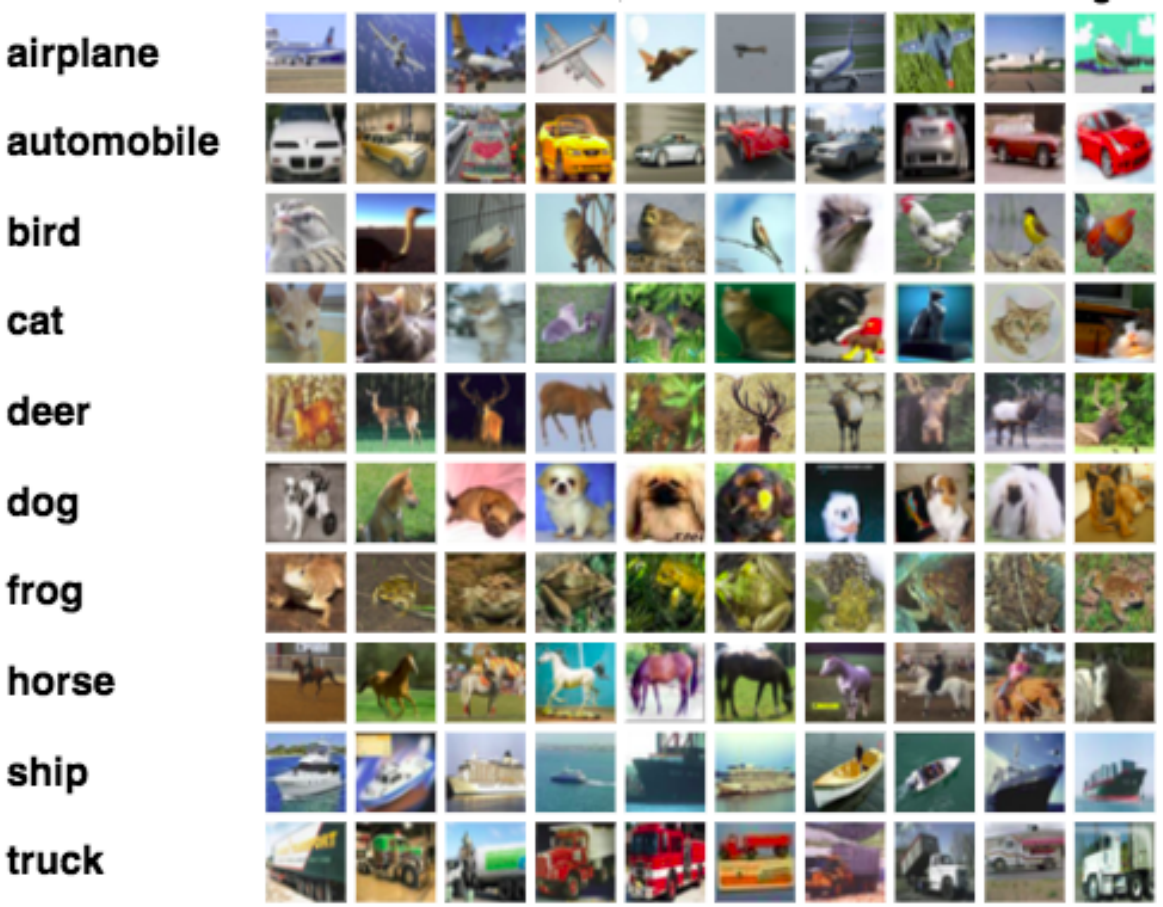
Idea credits: Justin Johnson

Visualizing weights for CIFAR10

The ‘unflattened’ weights can be visualized as an image

Linear classifiers have a “single” template per category

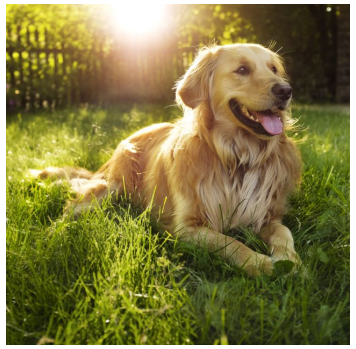
You will get to see similar visualizations after training your own linear classifier in PS3



Mean images from training set of CIFAR10

Learned weights for CIFAR10

Softmax loss / Multinomial logistic loss



For a single image, **x** with label 'y'

$$L(s, y) = -\log \frac{e^{s_y}}{\sum_{j=1}^{C} e^{s_j}}$$

			Probabilities
Dog	**2.5**	**12.18**	**0.06**
Cat	-1.3	0.27	0.0
Mountain	5.2	181.27	0.94

exp

normalize

In this case, class y = 1 (Dog)

$$L(s, 1) = -\log 0.06 = 2.81$$

Scores / Logits, **s** = Wx

Note: In class slides, logits Are denoted by 'z'.

$$e^{s_k}$$

k = 1, 2, 3

$$\frac{e^{s_k}}{\sum_{j=1}^{3} e^{s_j}}$$

Softmax loss properties

$$L(s, y) = -\log \frac{e^{s_y}}{\sum_{j=1}^{C} e^{s_j}}$$

C: Number of classes

What is the min/max possible Value of L(s, y) ? → Min 0, Max +infinity

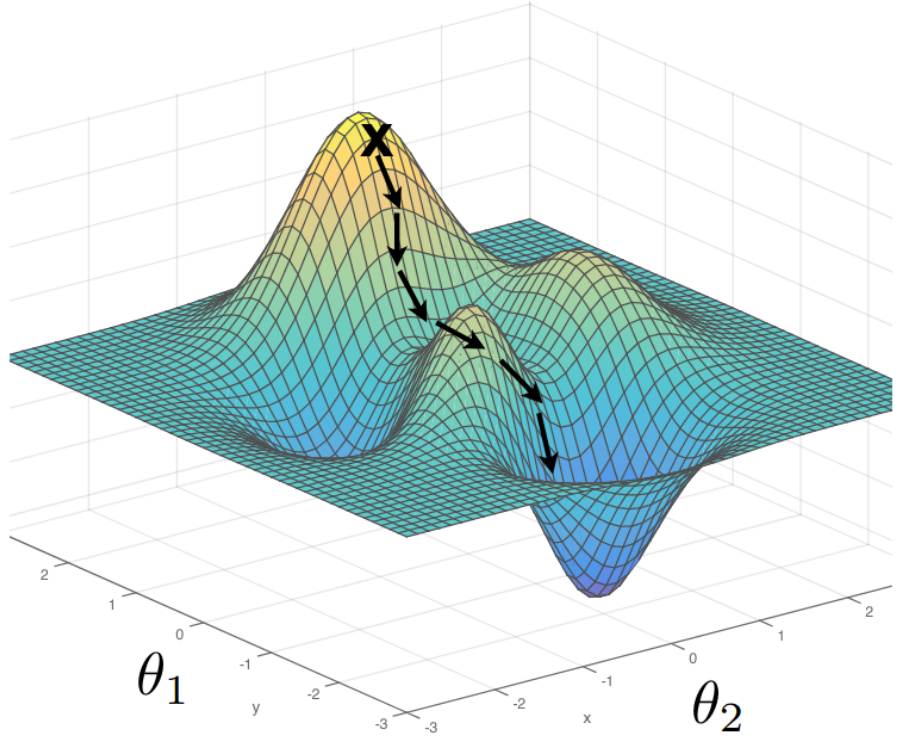
If all scores were really small and thus approximately the same, What would be the value of L(s, y)? → log(C)

How do we learn optimal parameters?

Review from Class

$$\theta^* = \arg\min_{\theta} \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

$J(\theta)$



$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$$

Recap: Stochastic Gradient Descent

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla L(x_i, y_i, \theta)$$

Vanilla Gradient Descent

We average the gradients over the whole training set which Can be expensive for a large training set. (Large N).

- Stochastic Gradient descent: Approximate this sum using a minibatch of training examples, which are a random subset of a fixed size **(B)**.

$$\nabla J(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \nabla L(x_i, y_i, \theta)$$

B – Batch Size

Calculating gradients using finite differences

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

- Called **numeric gradient**.

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25322

gradient dL/dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

Slide credits: Justin Johnson

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradient dL/dW:

[-2.5,
?,
?,

(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

?,
?,...]

Slide credits: Justin Johnson

How to analytically calculate gradients ?

- Numeric gradients are easy to compute.
- But they are slow and Approximate.
- Useful for gradient checking. We have implemented a function which does this and checks if the gradients implemented by you are correct.
- How do we efficiently calculate gradients analytically? We do it using a method called Backpropagation. Stay tuned to learn about this next week in class!