English for Computer Science

He Tang （唐赫）
School of Software
hetang@hust.edu.cn

Lecture3:
Image Classification—
Linear Classification

k-Nearest Neighbor on images **never used.**

- The classifier must *remember* all of the training data and store it for future comparisons with the test data. This is space inefficient because datasets may be very large.
- Classifying a test image is expensive since it requires a comparison to all training images.

k-Nearest Neighbor on images **never used.**

- Distance metrics on pixels are not informative

Original

1

Boxed

2

Shifted

3

Tinted

4

(all 3 images have same L2 distance to the one on the left)

Second Classifier: Linear Classifier

Neural Network

Linear classifiers

Recall CIFAR10

50,000 training images

each image is **32x32x3**

10,000 test images.

Parametric Approach

***Score Function*:**

Image

Array of **32x32x3** numbers
(3072 numbers total)

f(**x**,**W**)

W

parameters
or weights

10 numbers giving
class scores

Parametric Approach: Linear Classifier

Score Function: $f(x,W) = Wx$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$

10 numbers giving class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Score Function: $\boxed{f(x,W)} = \boxed{W}\boxed{x}$

$f(x,W)$: **10x1** $\quad$ W: **10x3072** $\quad$ x: **3072x1**

Image

Array of **32x32x3** numbers
(3072 numbers total)

$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$ **10** numbers giving class scores

$\uparrow$

W

parameters
or weights

Parametric Approach: Linear Classifier

3072x1

Score Function: $\boxed{f(x,W)} = \boxed{W}\boxed{x} + \boxed{b}$ 10x1

10x1 10x3072

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W}) \longrightarrow$ **10** numbers giving class scores

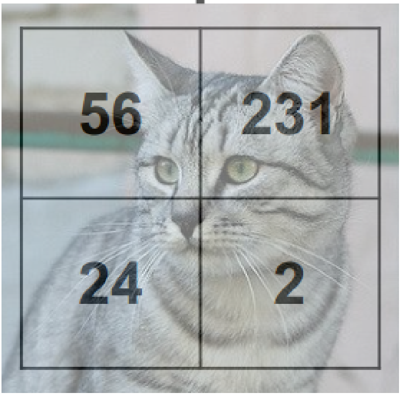
W

parameters
or weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Stretch pixels into column



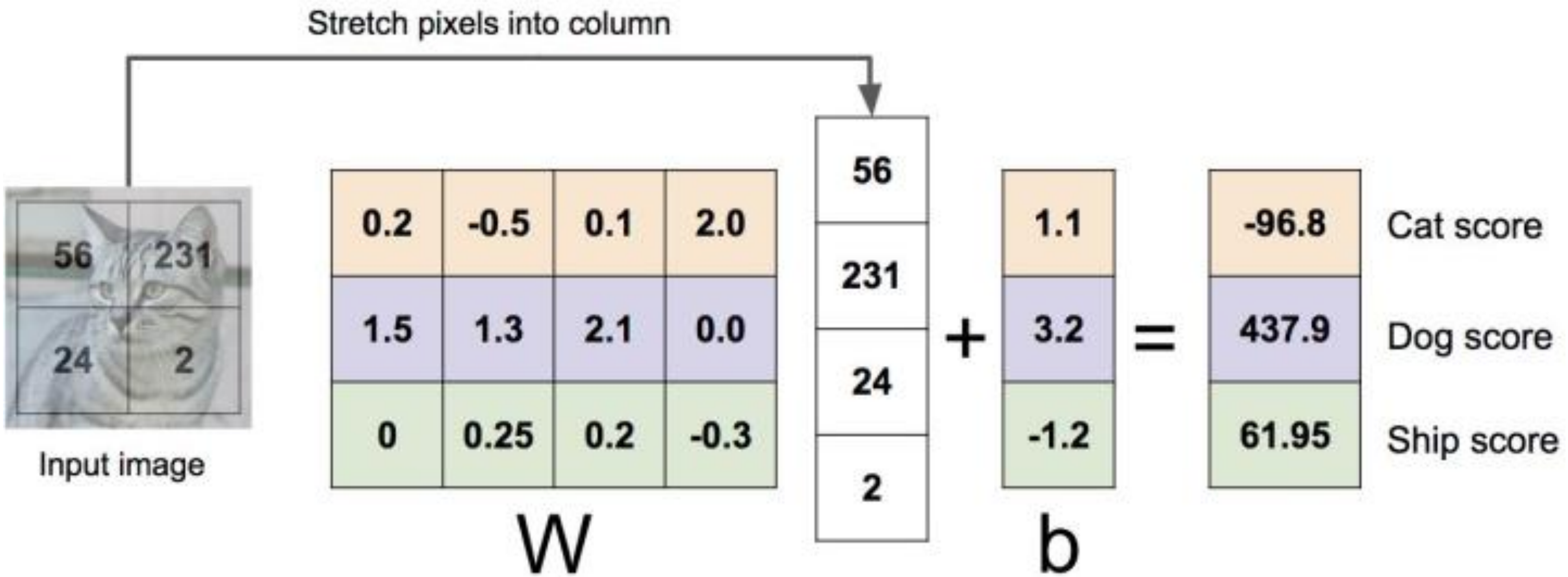
Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{array}{c} \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \end{array} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Algebraic Viewpoint

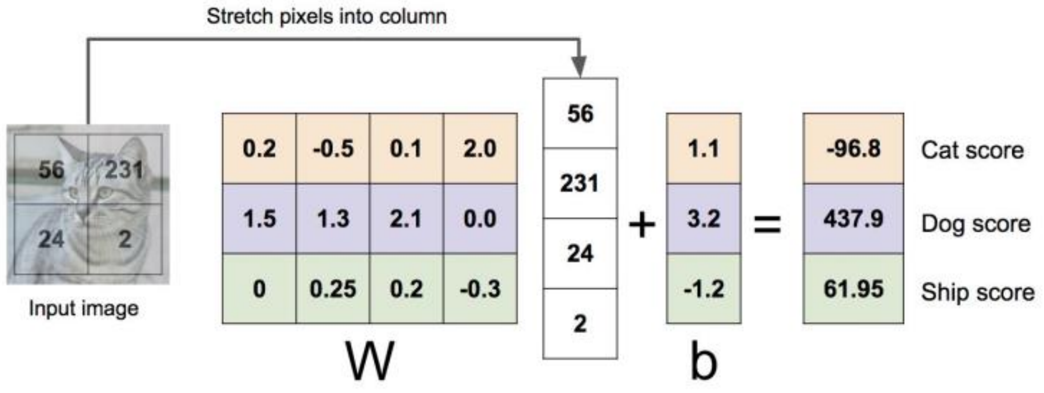
$$f(x,W) = Wx$$

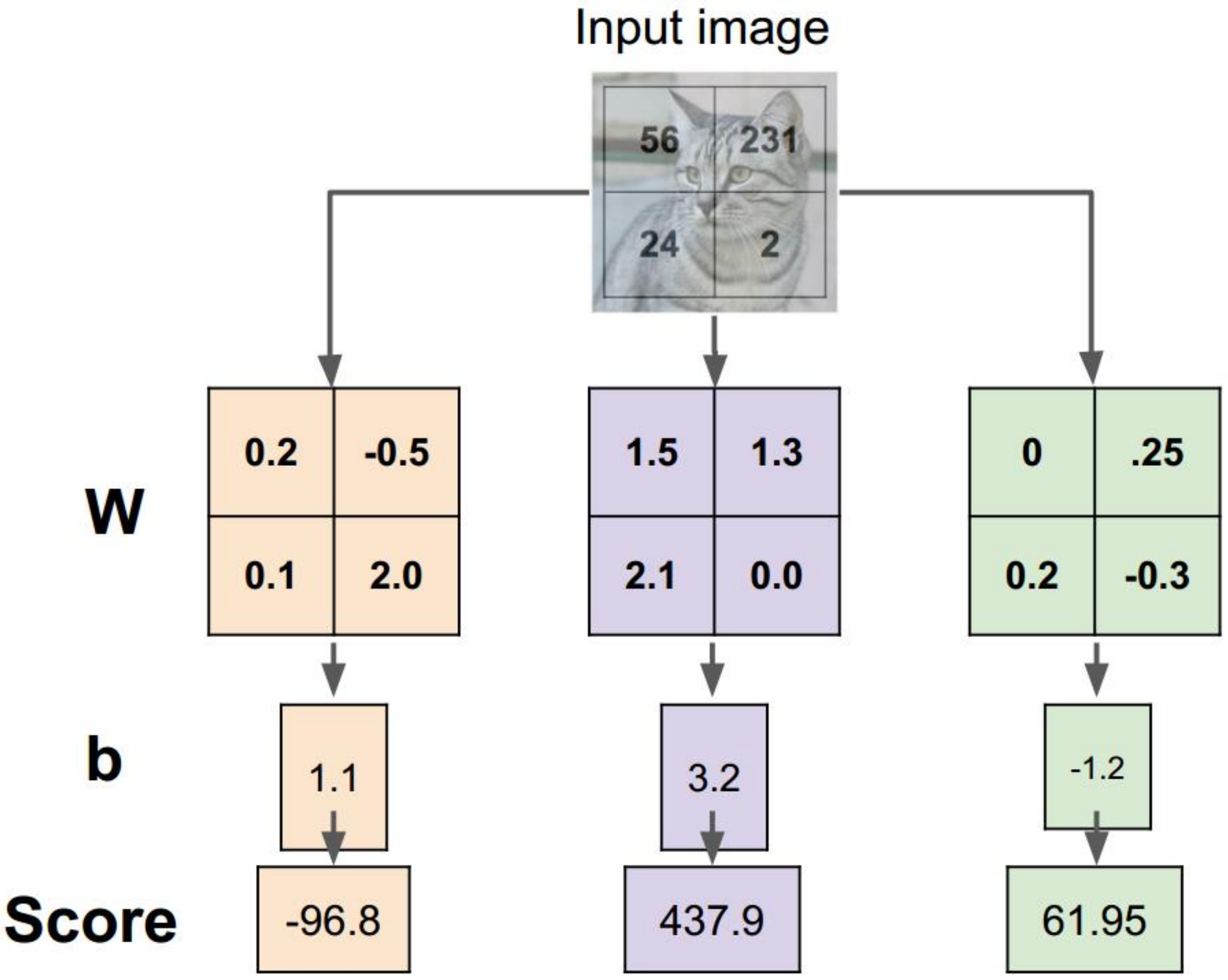


Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Algebraic Viewpoint

$$f(x,W) = Wx$$





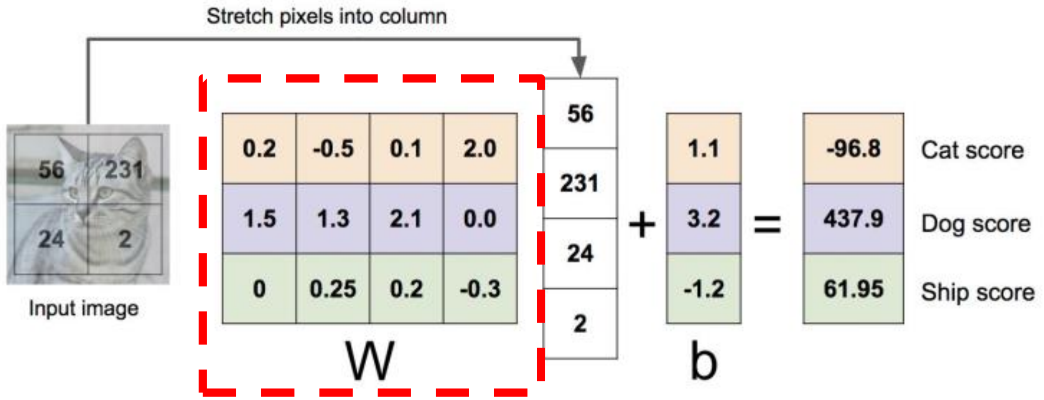
There are a few things to note:

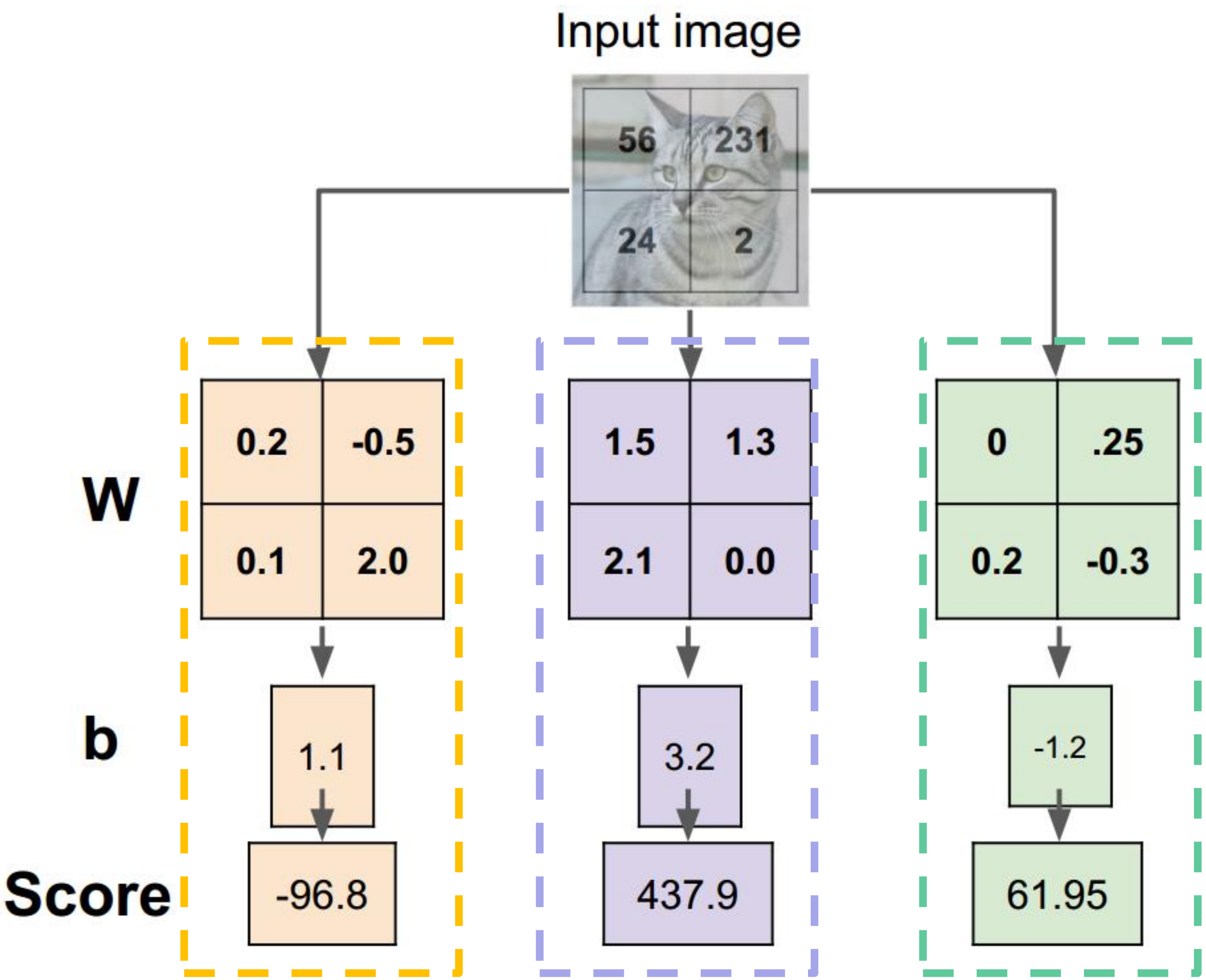
- First, note that the single matrix multiplication Wx is effectively evaluating 10 separate classifiers in parallel (one for each class), where each classifier is a row of W.
- Notice also that we think of the input data (x, y) as given and fixed, but we have control over the setting of the parameters W, b. Our goal will be to set these in such way that the computed scores match the ground truth labels across the whole training set. Intuitively we wish that the *correct* class has a score that is higher than the scores of *incorrect* classes.
- An advantage of this approach is that the training data is used to learn the *parameters* W, b, but once the learning is complete we can discard the entire training set and only keep the learned parameters. That is because a new test image can be simply forwarded through the function and classified based on the computed scores.
- Lastly, note that classifying the test image involves a single *matrix multiplication* and *addition*, which is significantly faster than comparing a test image to all training images.

Classify in parallel

Algebraic Viewpoint

$$f(x,W) = Wx$$

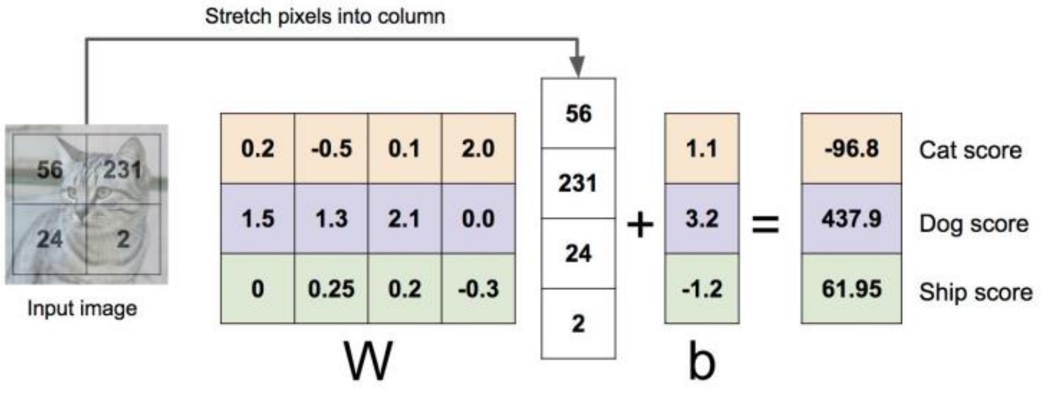


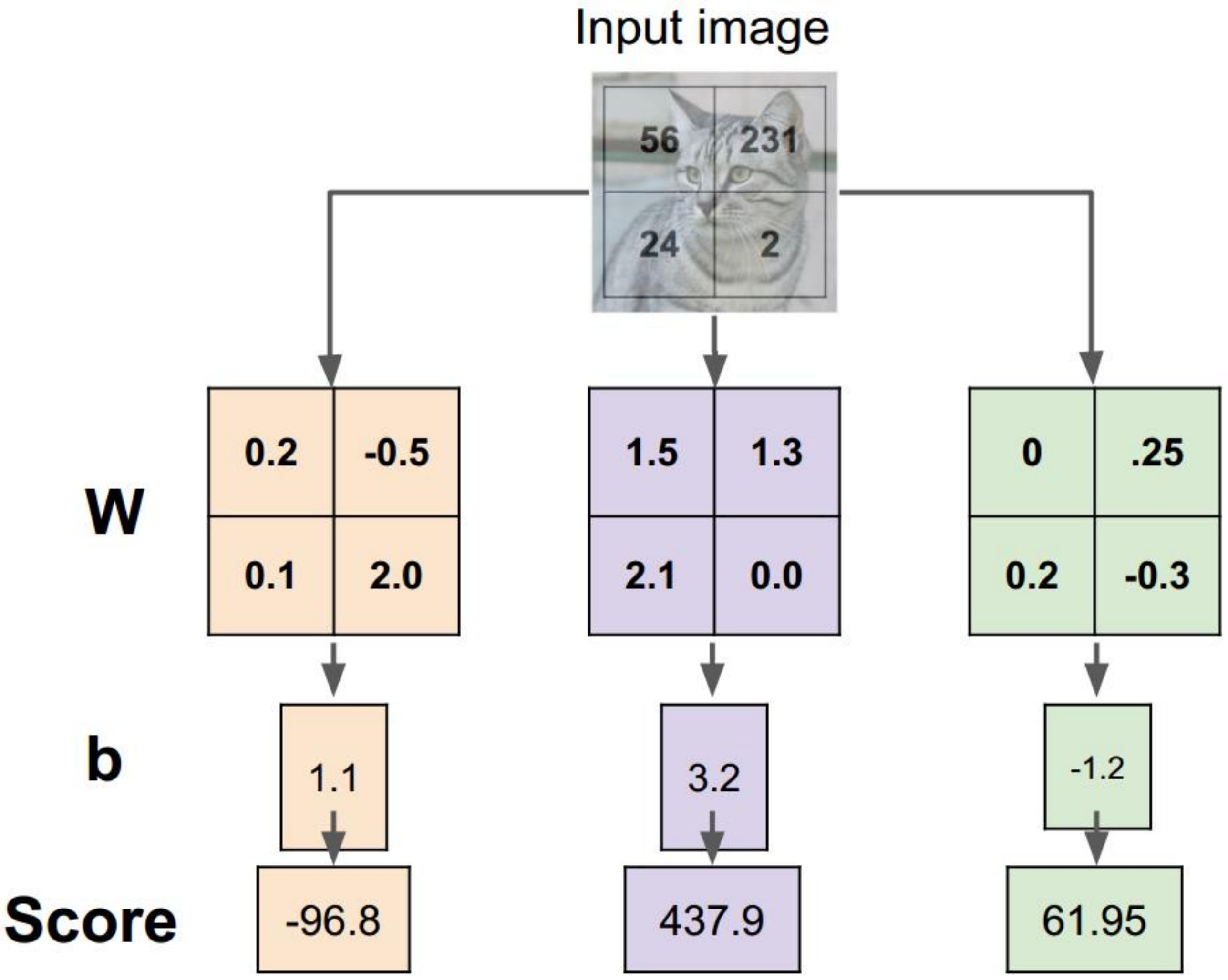


Ground truth label

Algebraic Viewpoint

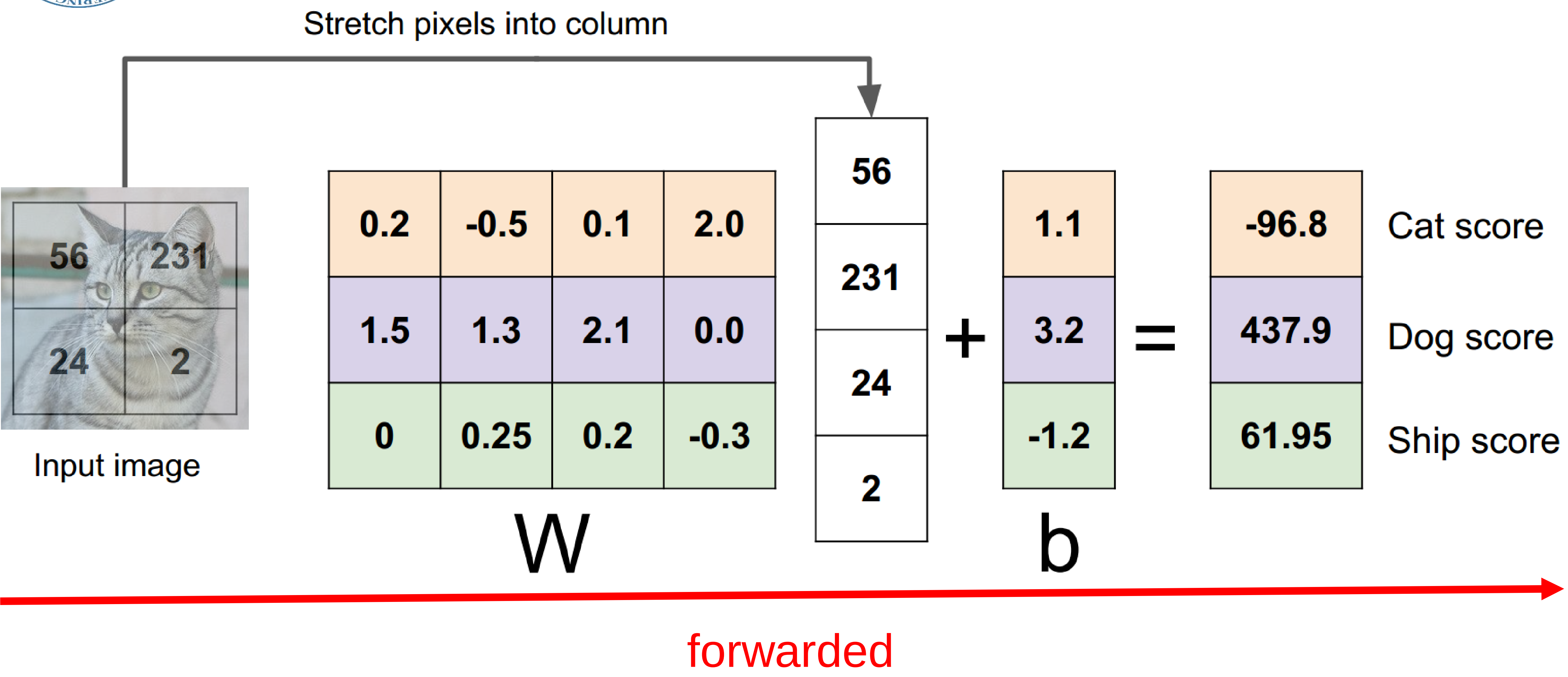
$$f(x,W) = Wx$$



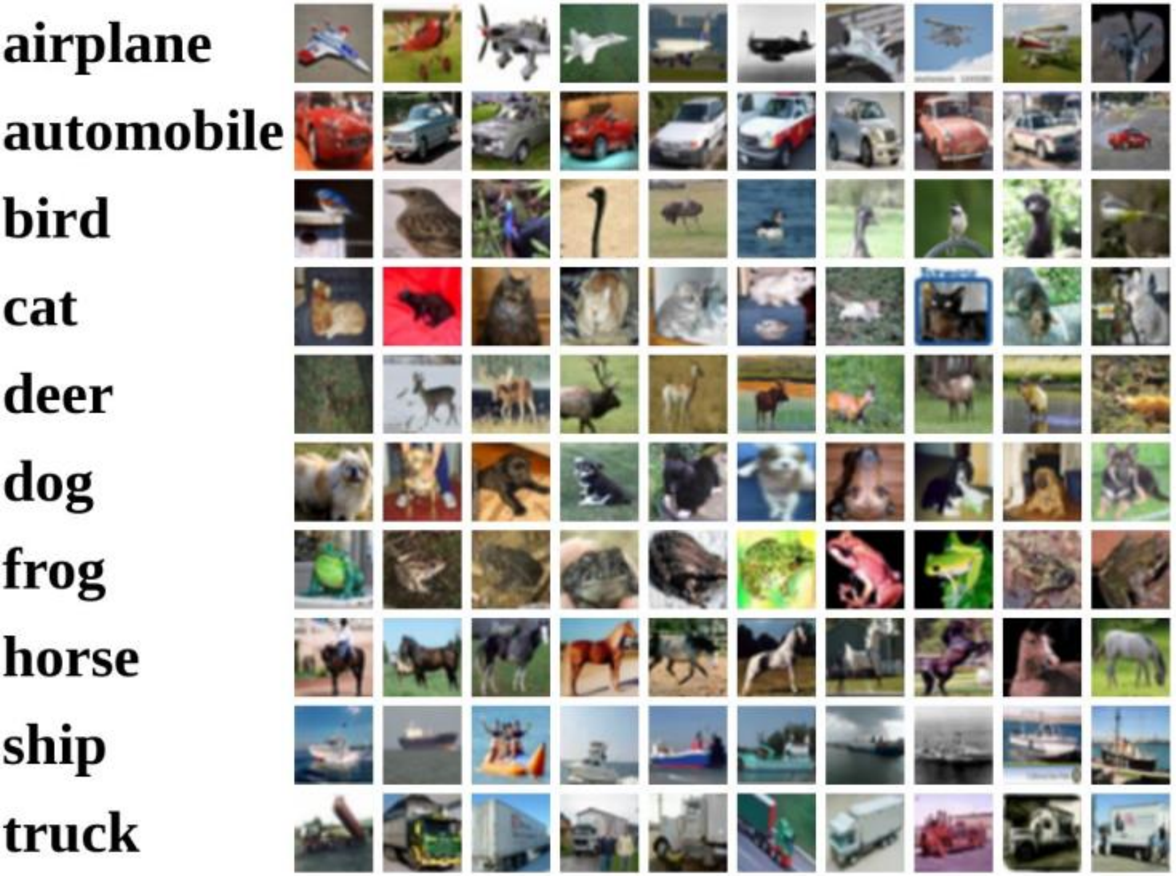


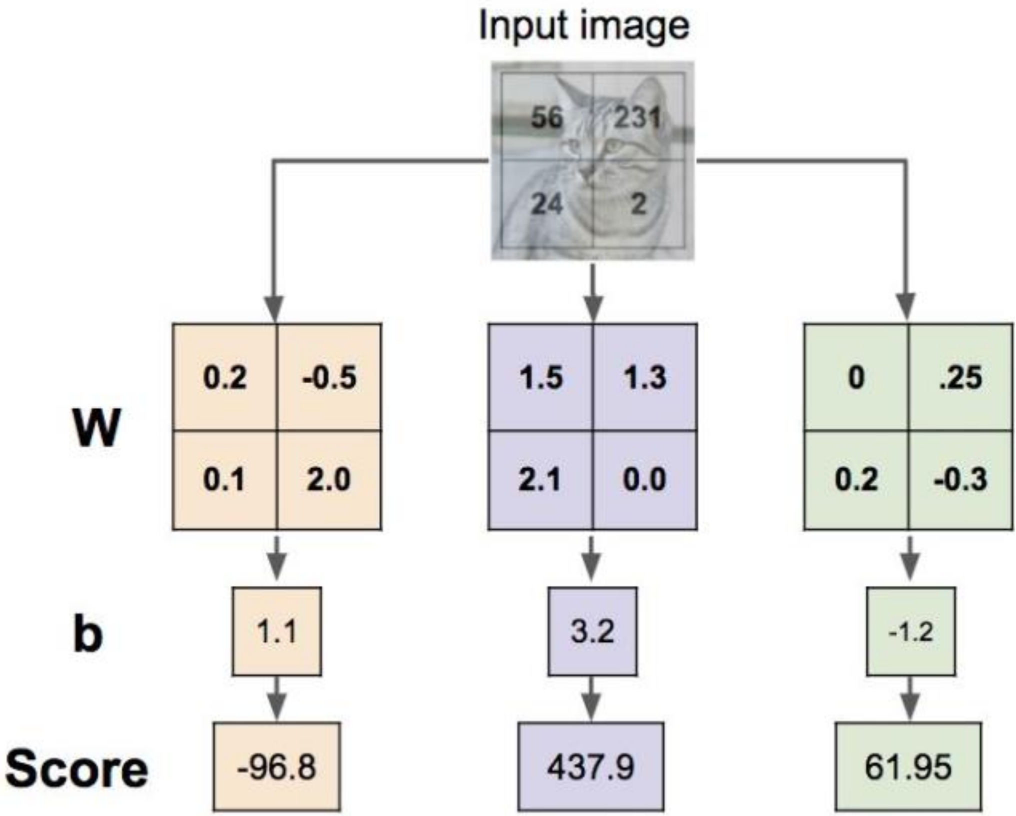
Ground truth label **[1 0 0]**

Prediction by forwarded

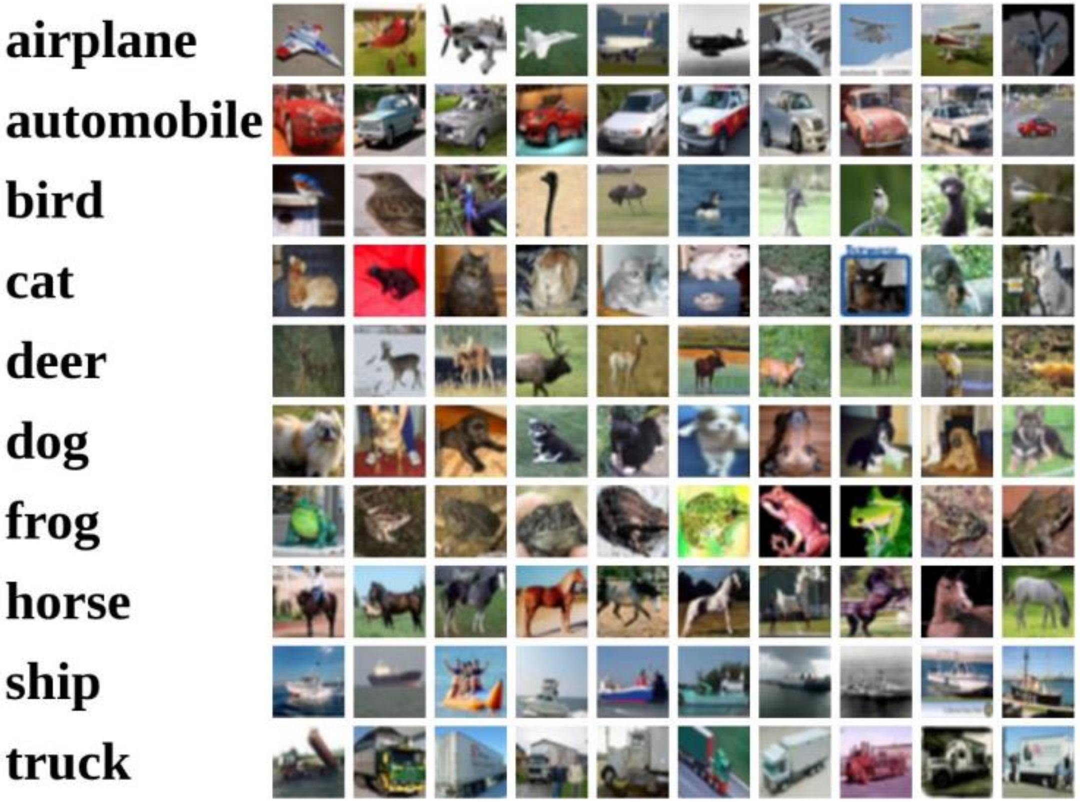


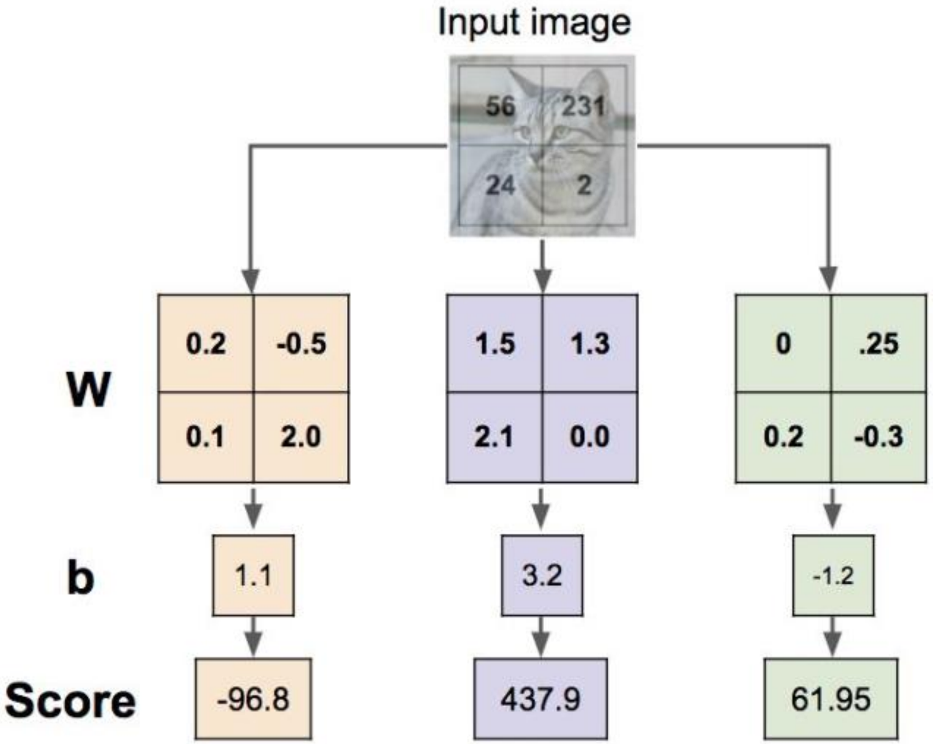
Interpreting a Linear Classifier



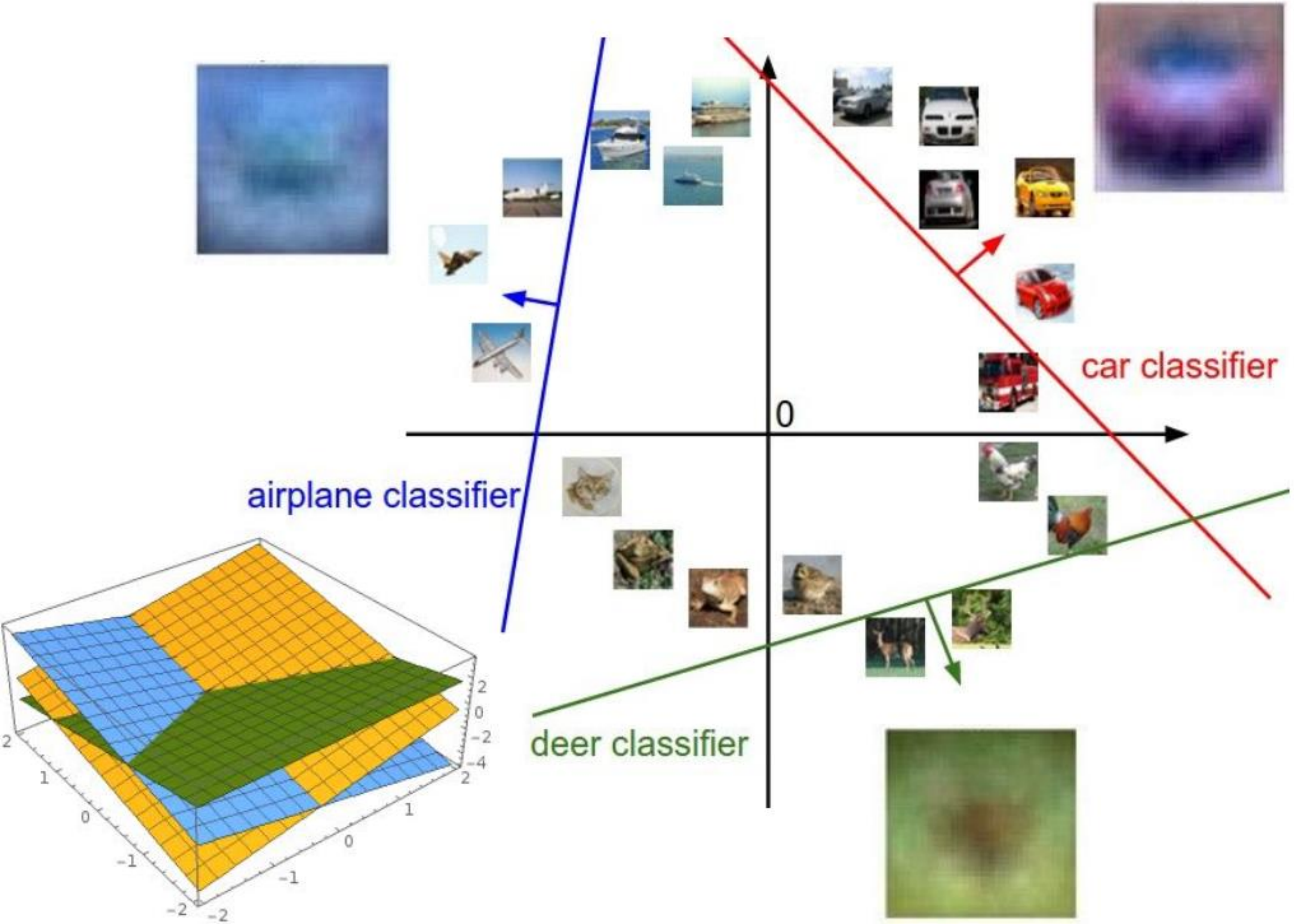


Interpreting a Linear Classifier: Visual Viewpoint





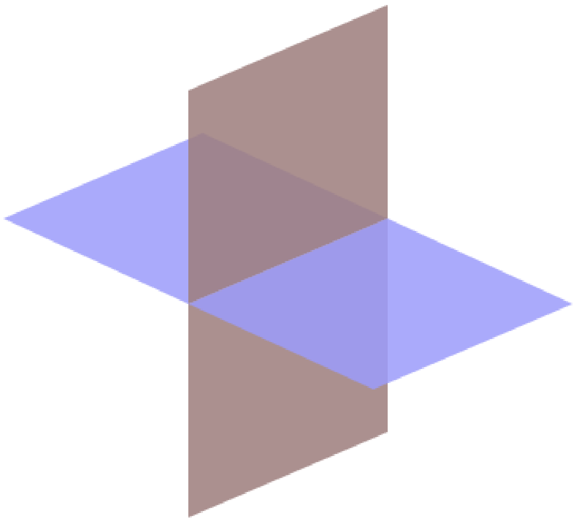
Interpreting a Linear Classifier: Geometric Viewpoint

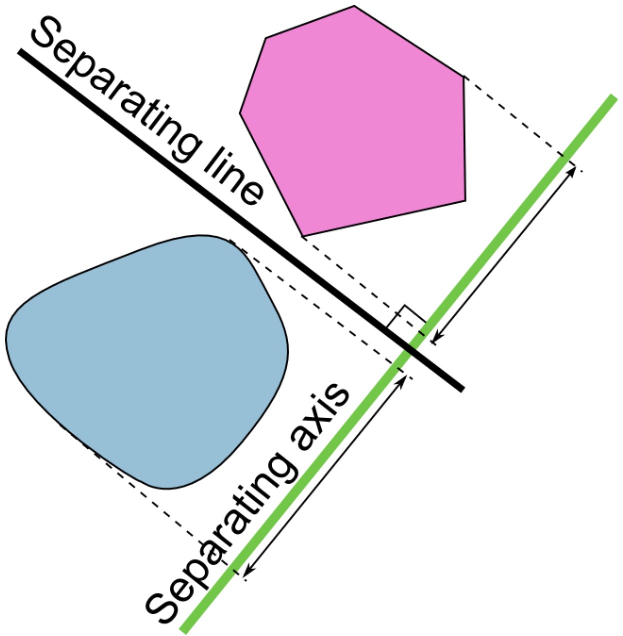


$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers (3072 numbers total)

Hyperplane





Let w_1, w_2, ..., w_n be scalars not all equal to 0. Then the set X consisting of all vectors:

$$X = [x_1 x_2 \ldots xn]^{\mathrm{T}}$$

In $\mathbf{R}^n$ such that

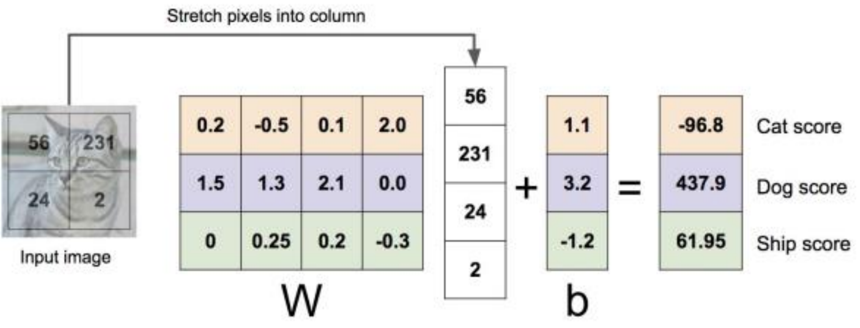
$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = b$$

For b a constant is a subspace of $\mathbf{R}^n$ called a hyperplane

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

$$f(x,W) = Wx$$

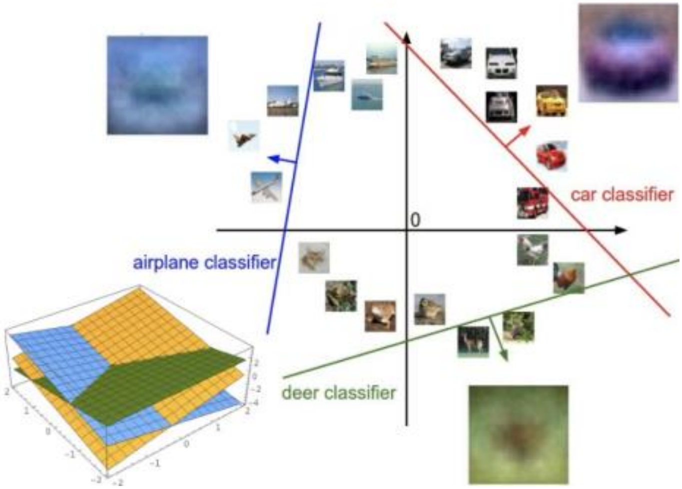


Visual Viewpoint

One template per class

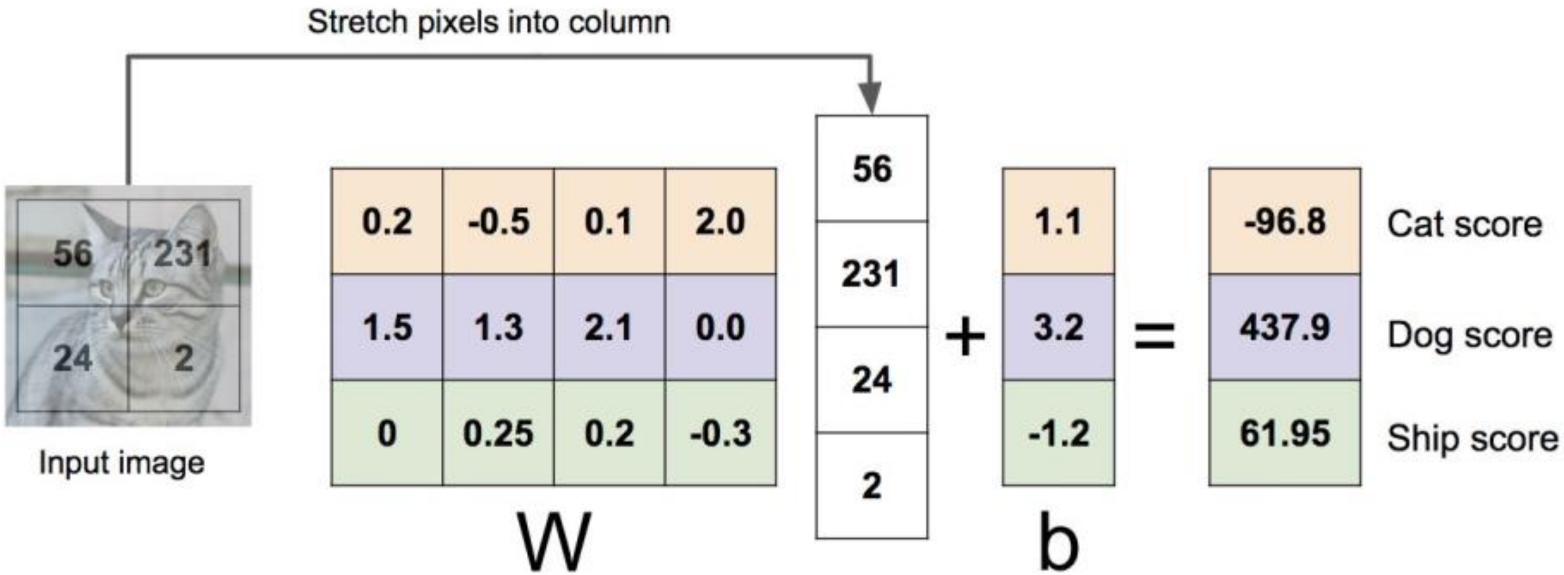
Geometric Viewpoint

Hyperplanes cutting up space



Algebraic Viewpoint

$$f(x,W) = Wx$$

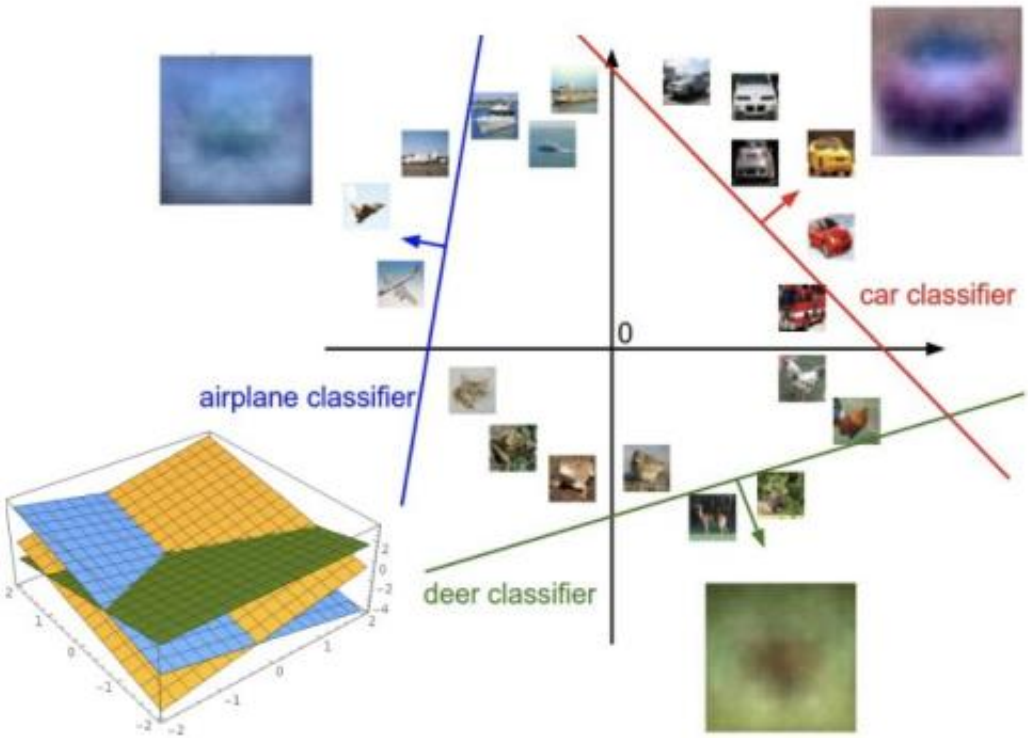


Visual Viewpoint

One template
per class

Geometric Viewpoint

Hyperplanes cutting up space



So far: Defined a (linear) score function $f(x,W) = Wx + b$

Example class scores for 3 images for some W:

How can we tell whether this W is good or bad?



airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

$$f(x,W) = Wx + b$$

Coming up:

- Loss function (quantifying what it means to have a “good” W)
- Optimization (start with random W and find a W that minimizes the loss)
- ConvNets! (tweak the functional form of f)