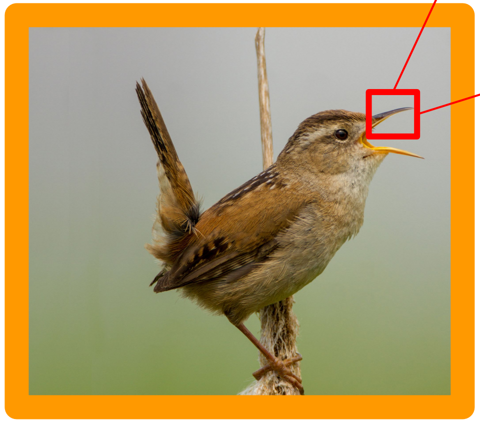
DATA 442: Neural Networks & Deep Learning

Dan Runfola – danr@wm.edu

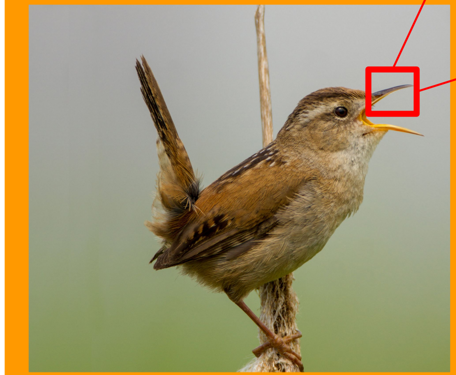
icss.wm.edu/data442/

34	40	34	3	8	30	50	26	16	28	41	6	23	2	24	0
14	37	19	25	6	9	14	17	4	46	20	7	38	6	29	28
4	39	0	29	15	6	50	2	21	10	8	45	150	145	106	46
42	10	15	19	24	18	111	123	118	104	119	122	117	140	138	28
21	35	19	30	14	143	146	147	142	103	109	127	108	148	20	23
30	105	147	102	126	118	108	101	140	131	124	136	47	27	26	38
135	133	137	108	140	144	135	120	118	137	125	43	8	31	45	10
106	142	108	138	137	111	38	36	32	1	19	44	34	4	38	49
122	142	127	131	143	8	47	4	0	31	39	18	46	1	50	25
149	137	122	36	50	19	24	45	16	30	2	47	2	35	29	50
147	115	3	29	10	2	13	1	48	3	45	28	39	14	14	20



34	40	34	3	8	30	50	26	16	28	41	6	23	2	24	0
14	37	19	25	6	9	14	17	4	46	20	7	38	6	29	28
4	39	0	29	15	6	50	2	21	10	8	45	150	145	106	46
42	10	15	19	24	18	111	123	118	104	119	122	117	140	138	28
21	35	19	30	14	143	146	147	142	103	109	127	108	148	20	23
30	105	147	102	126	118	108	101	140	131	124	136	47	27	26	38
135	133	137	108	140	144	135	120	118	137	125	43	8	31	45	10
106	142	108	138	137	111	38	36	32	1						
122	142	127	131	143	8	47	4	0	31						
149	137	122	36	50	19	24	45	16	30						
147	115	3	29	10	2	13	1	48	3						

34	40	34	3	8	30	50	26	16	28	41	6	23	2	24	0
14	37	19	25	6	9	14	17	4	46	20	7	38	6	29	28
28	6	31	39	0	23	36	34	21	10	8	45	6	29	45	46
71	6	9	44	41	23	36	61	4	104	119	122	117	140	138	28
8	60	45	11	12	165	122	115	142	103	109	127	108	148	100	23
2	94	156	88	174	160	62	59	140	131	124	136	127	150	145	38
183	94	160	101	108	163	135	119	118	137	125	43	8	31	45	10
128	179	74	122	89	140	59	22	32	1	19	44	34	4	38	49
165	100	106	172	110	41	58	11	0	31	39	18	46	1	50	25
154	175	158	76	53	58	61	69	16	30	2	47	2	35	29	50
153	138	43	12	19	40	62	26	48	3	45	28	39	14	14	20



Intra-Class Differences

Viewpoint

Intra-Class Differences

Viewpoint

Intra-Class
Differences

Background

Viewpoint

Intra-Class
Differences

Viewpoint

Intra-Class Differences

Deformation

Viewpoint

Intra-Class Differences

Deformation

Occlusion

Recap: KNN

	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	
				1	1				
				1	1				
				1	1				
				1	1				
				1	1				
				1	1				
				1	1				
				1	1				

T from Training Data

−

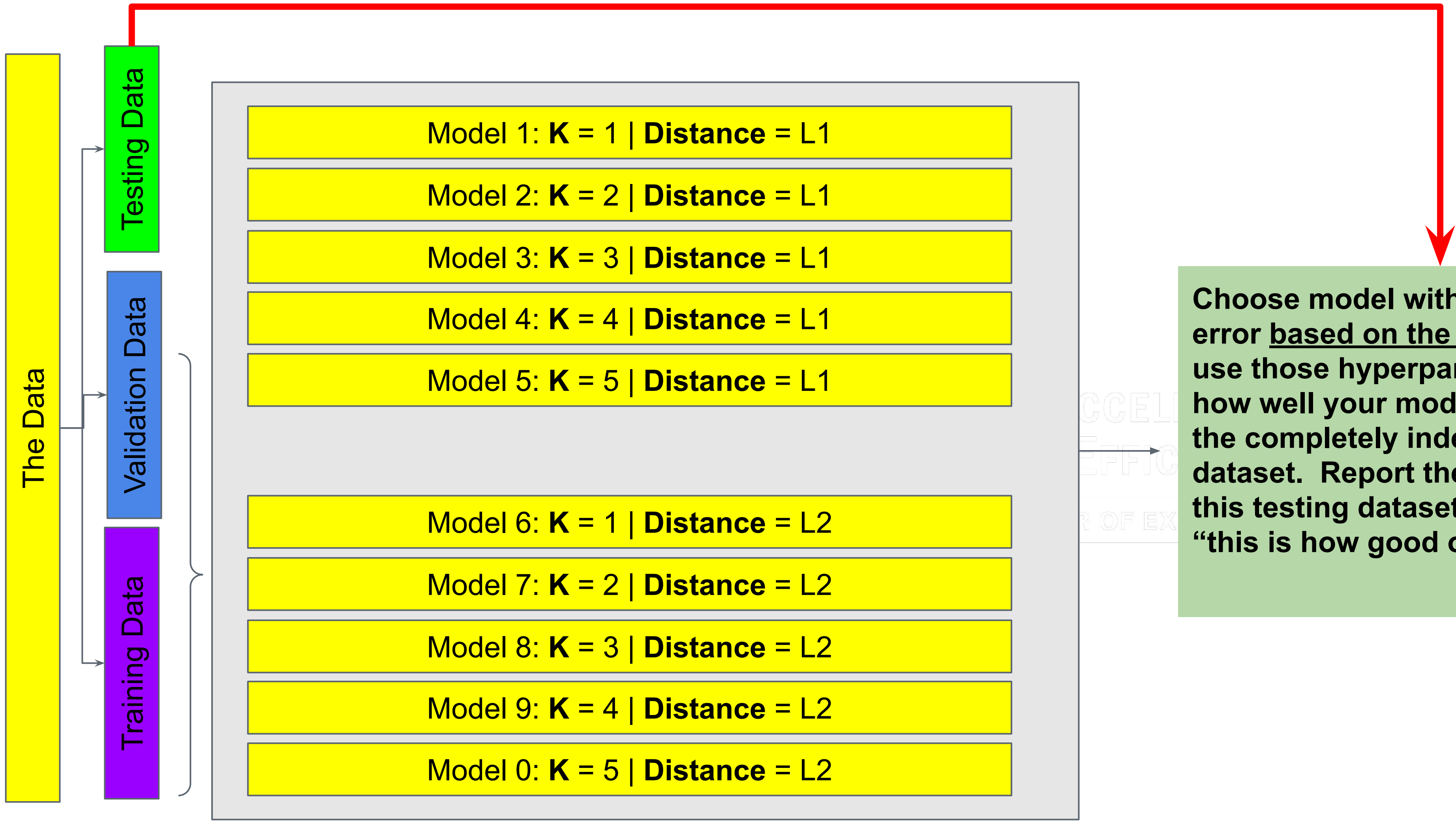
		1	1	1	1	1	1		
		1	1	1	1	1	1		
		1		1	1		1		
				1	1				
				1	1				
				1	1				
				1	1				
				1	1				
			1	1	1	1			

T we want to Recognize

=

Sum of Absolute Difference: 10

	1							1	
	1							1	
		1					1		
			1			1			
				1	1				



Choose model with lowest overall error based on the test data only, use those hyperparameters to test how well your model performs on the completely independent testing dataset. Report the accuracy from this testing dataset as your final "this is how good our model is".

Building Blocks of Neural Nets: Linear Classification

- **Parametric vs. Non Parametric**
- **Interpreting Linear Classifiers**
- **Limitations of Linear Classifiers**
- **Segway into Loss Functions**

CIFAR10 Dataset

(random examples generated from lab 1 code -->)

Goal: Given a new image, identify the correct class.

KNN approach: Record all of the images, and when a new image comes compare it to all images and select the most similar. Classify accordingly.

nn.predict(image)

	Probability
Bird	0.2
Dog	0.1
...	...
Cat	0.15
Plane	0.19

nn.predict(image, **W**)

Parameters (generally referred to as **Weights**)

	Probability
Bird	0.2
Dog	0.1
...	...
Cat	0.15
Plane	0.19

```
def predict(image, W):
    W*image
```

nn.predict(image, **W**)

	Probability
Bird	0.2
Dog	0.1
...	...
Cat	0.15
Plane	0.19

CIFAR10 Bird Example

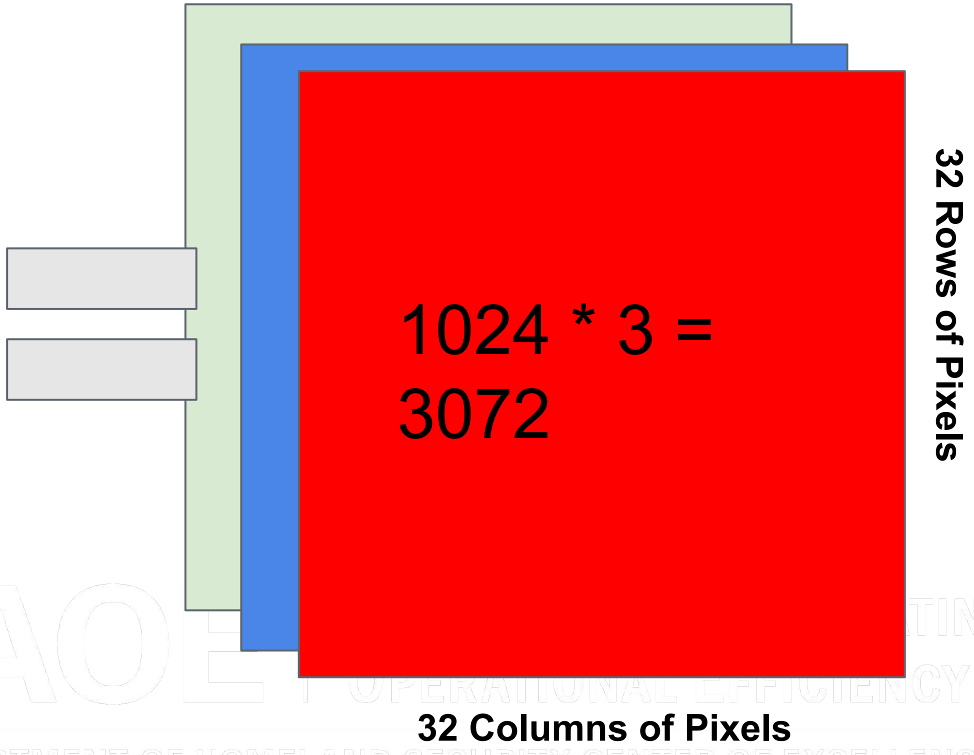
=

32 * 32 =
1024 Pixels

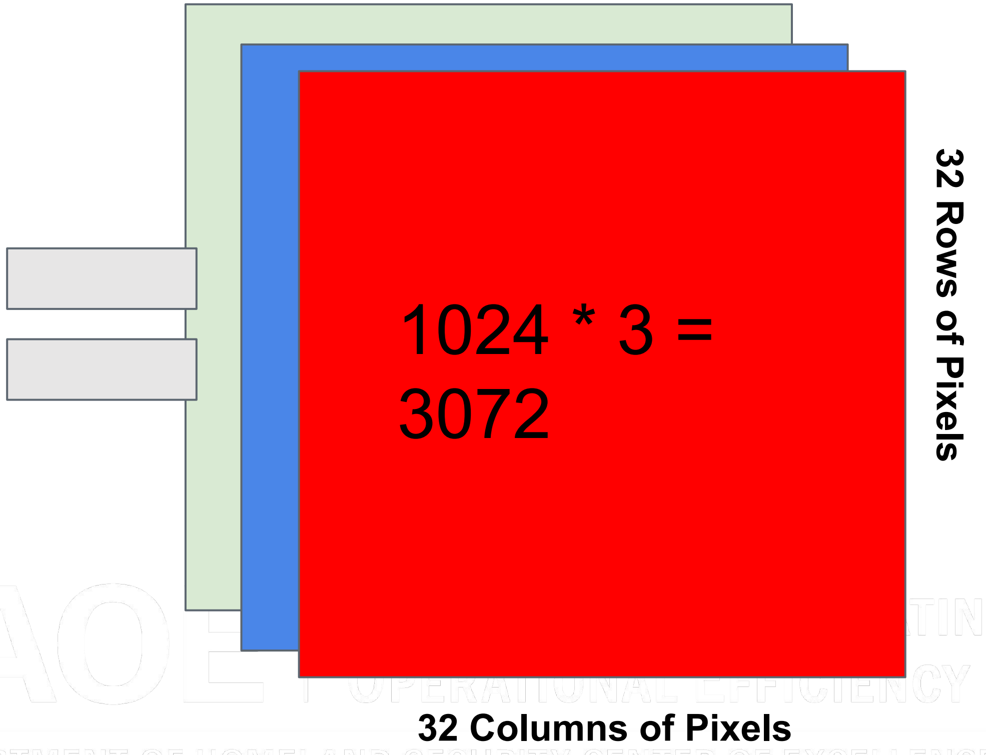
32 Rows of Pixels

32 Columns of Pixels

CIFAR10 Bird Example



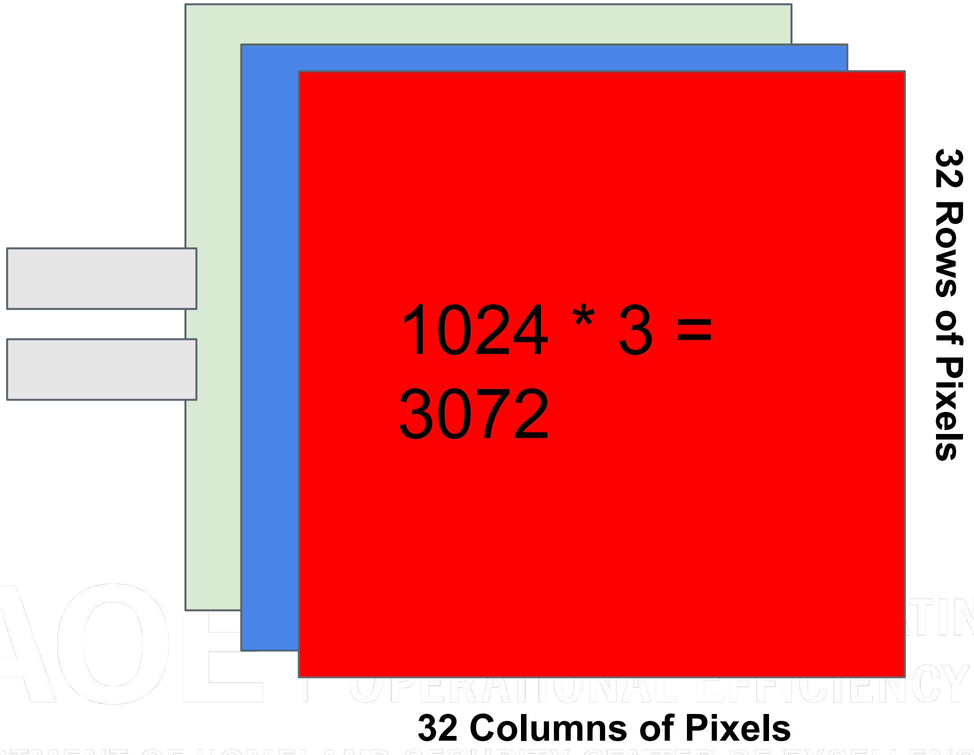
CIFAR10 Bird Example



def predict(image, W):
    W*image

image: A vector of length 3072 - [0,12,3,2, …. 392] - where each value represents a pixel in one of the three color bands.

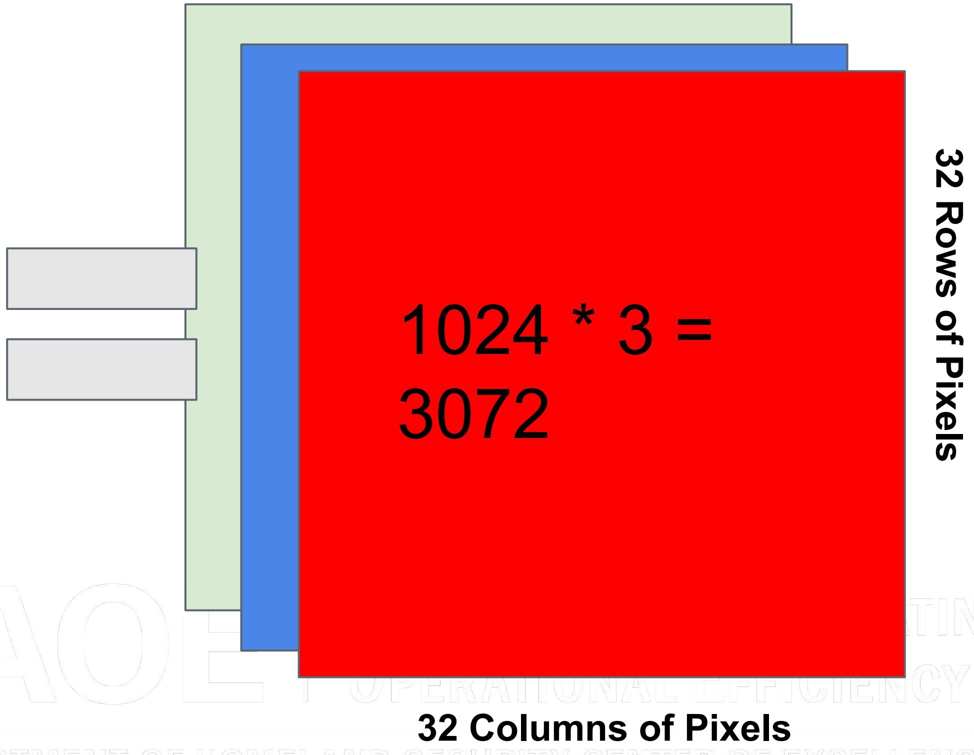
CIFAR10 Bird Example



def predict(image, $\boxed{\text{W}}$):
    W*image

W: A 10x3072 matrix, with each of ten “columns” indicating the value to multiply by each pixel to generate a probability.

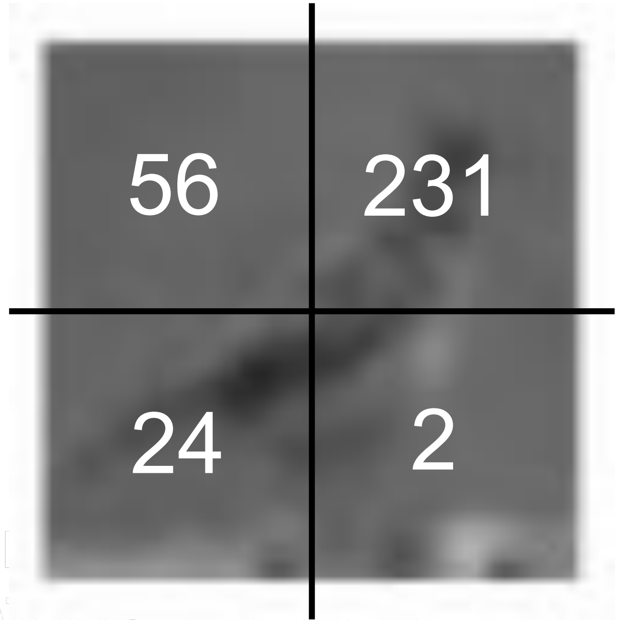
CIFAR10 Bird Example

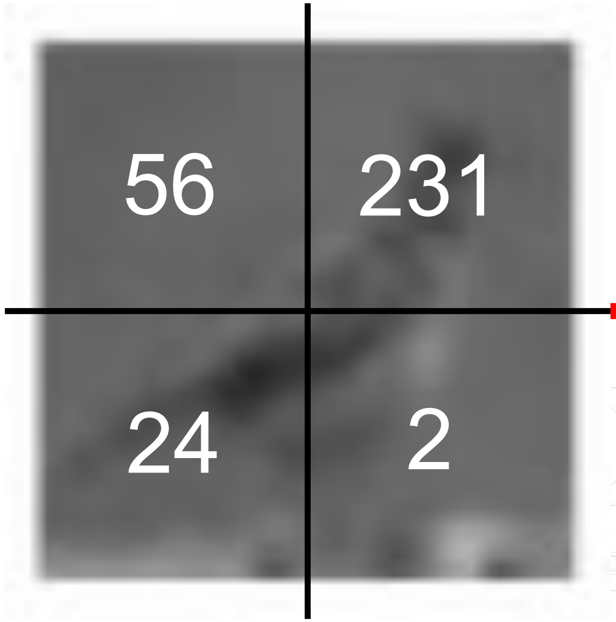



```
def predict(image, W):
    W*image
```

W*image: A 10 x 1 matrix in which each value is the probability of class inclusion.

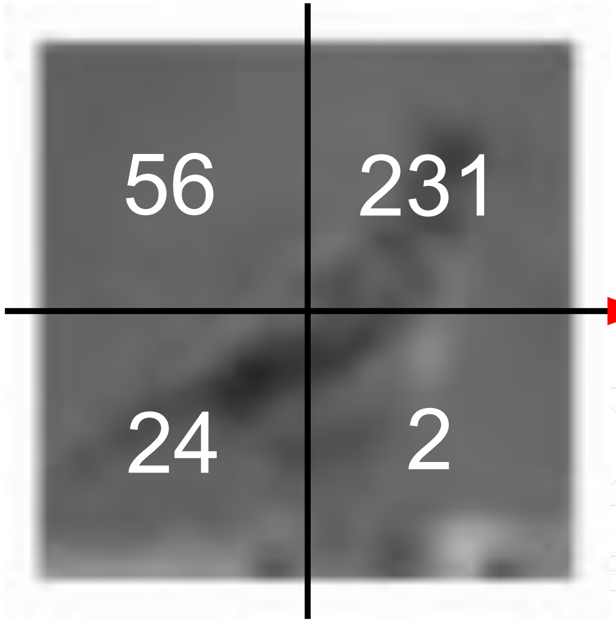
CIFAR10 Bird Example





56
231
24
2

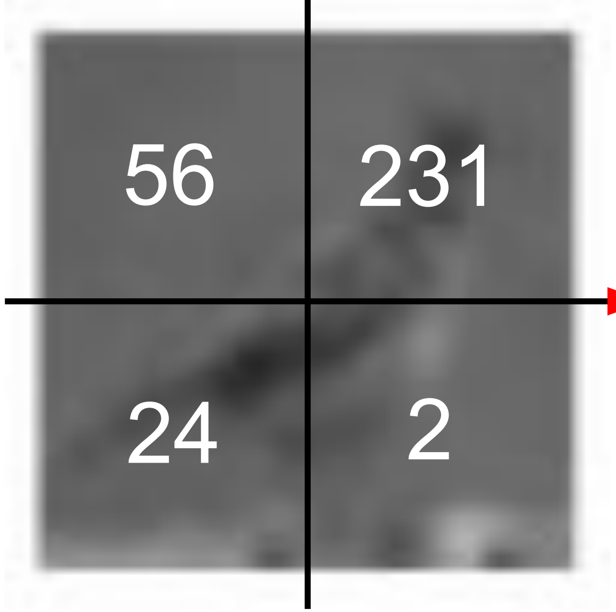

```
def predict(image, W):
    W*image
```



56
231
24
2

0.2	-0.5	0.1	2.0	Cat
1.5	1.3	2.1	0.0	Bird
0	0.25	0.2	-0.3	Plane


```
def predict(image, W):
    W*image
```

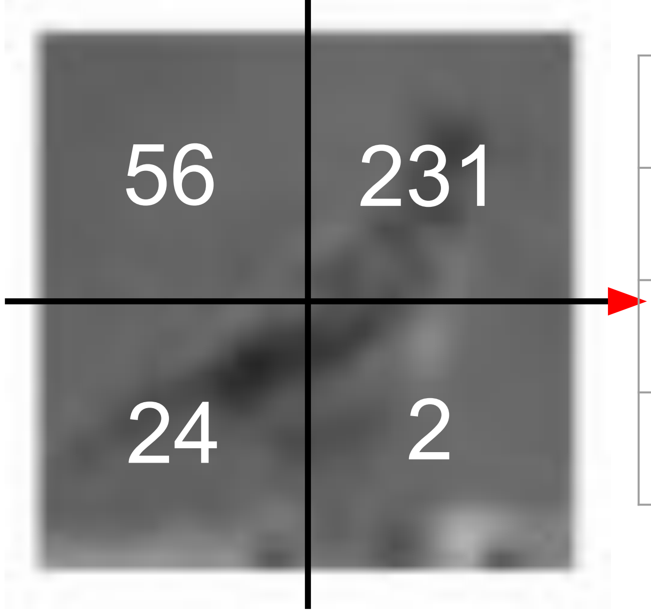


56
231
24
2

0.2	-0.5	0.1	2.0	Cat
1.5	1.3	2.1	0.0	Bird
0	0.25	0.2	-0.3	Plane

Cat Score = (56 * 0.2) + (231 * -0.5) + (24 * 0.1) + (2 * 2.0) = -97.9


```
def predict(image, W):
    W*image
```

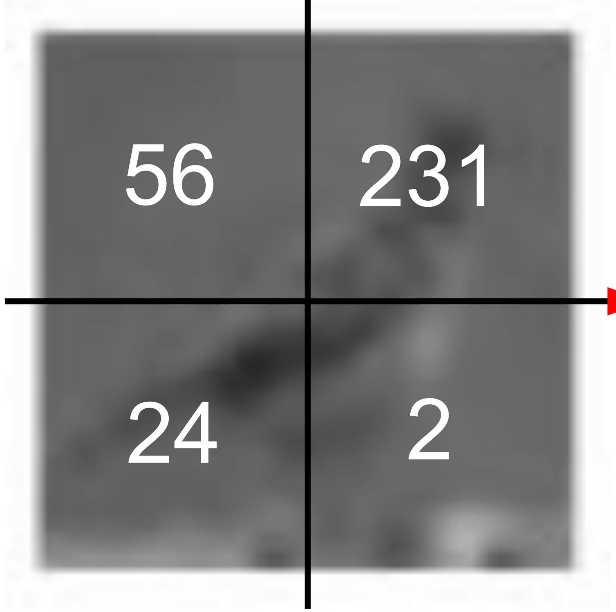


56
231
24
2

0.2	-0.5	0.1	2.0	Cat
1.5	1.3	2.1	0.0	Bird
0	0.25	0.2	-0.3	Plane

Cat Score = (56 * 0.2) + (231 * -0.5) + (24 * 0.1) + (2 * 2.0) = -97.9

```
def predict(image, W):
    W*image
```



56
231
24
2

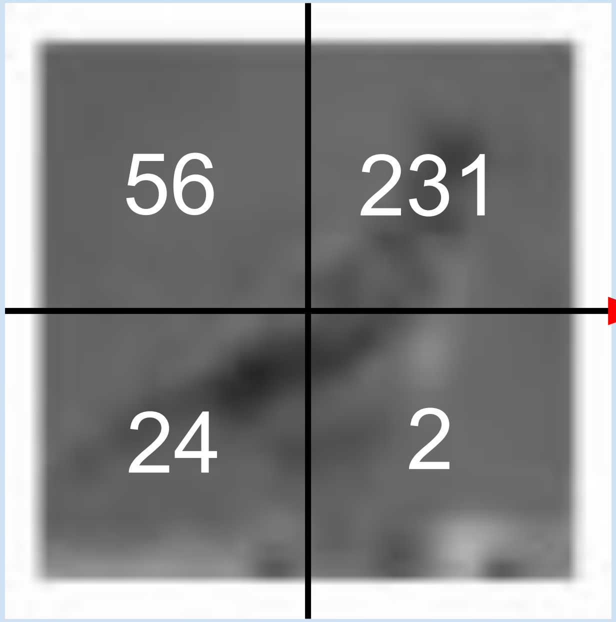
0.2	-0.5	0.1	2.0	Cat
1.5	1.3	2.1	0.0	Bird
0	0.25	0.2	-0.3	Plane

Cat Score = -97.9

Bird Score = 434.7

Plane Score = 63.15

Image (Matrix) to Vector



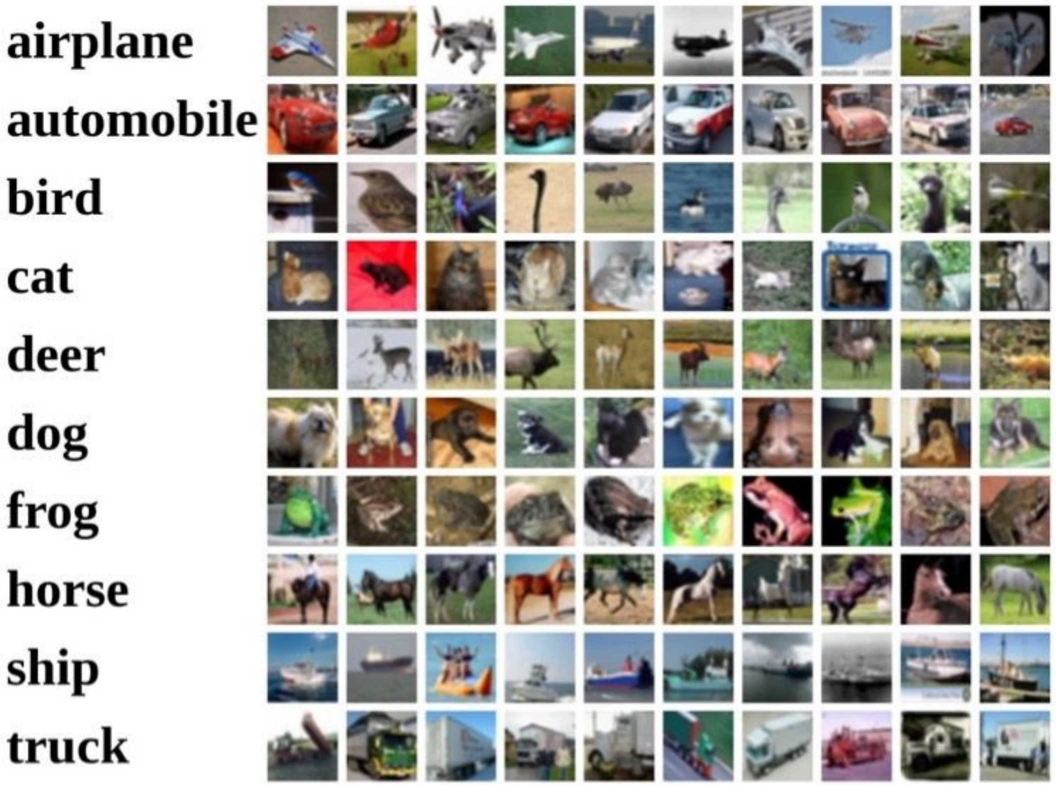
56
231
24
2

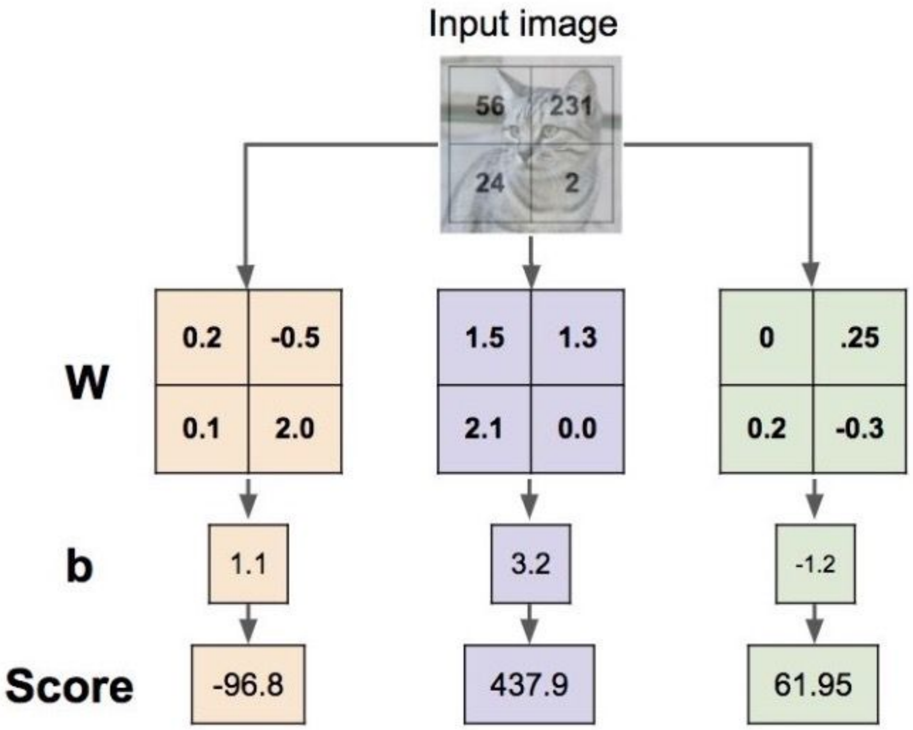
Weights Vector to Image

0.2	-0.5	0.1	2.0

Cat

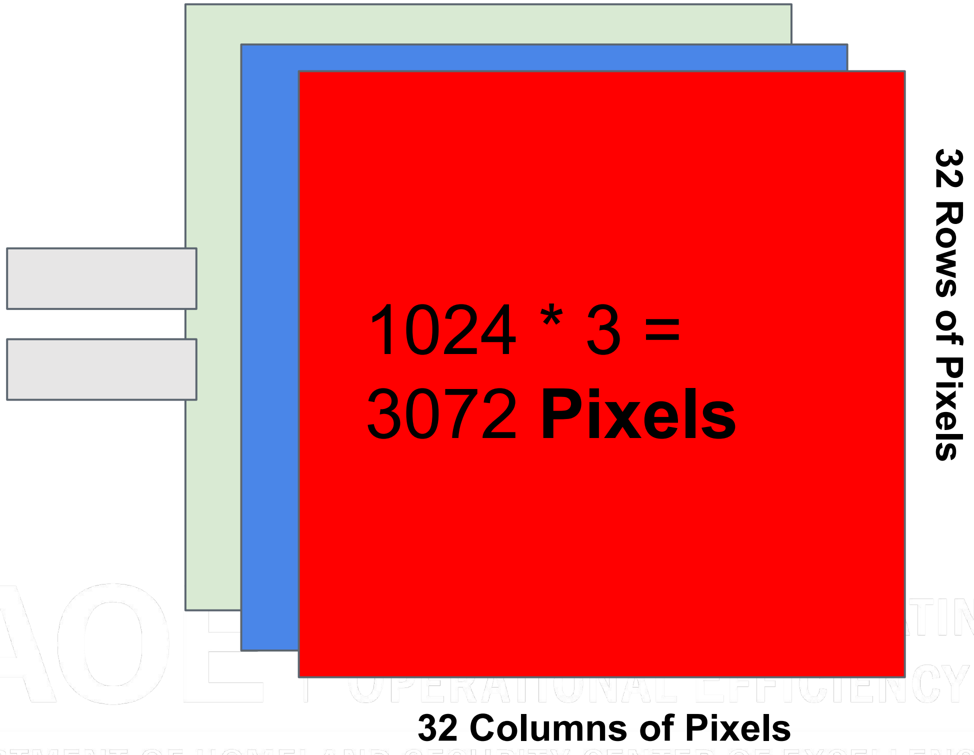
0.2	-0.5
0.1	2.0





http://cs231n.stanford.edu/

CIFAR10 Bird Example



Loss Function

A single score that quantifies how bad a classification is.

Cat Score = -97.9

Bird Score = 3.5

Plane Score = 63.15

1024 * 3 =
3072 **Pixels**

So you need 3072 weights per class (in a linear classifier)!

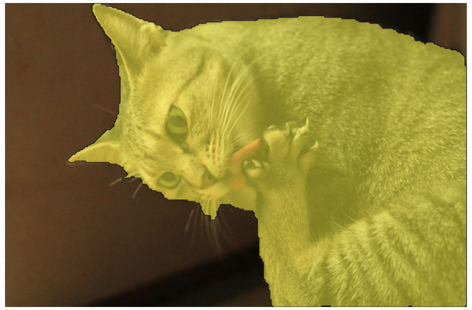
32 Rows of Pixels

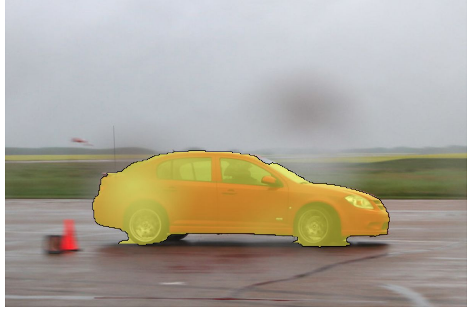
32 Columns of Pixels

Optimization Strategy

Finding the Weights that minimize the loss function.

Cat	**3.2**	**1.3**	**2.2**
Car	**5.1**	**4.9**	**2.5**
Frog	**-1.7**	**2.0**	**-3.1**

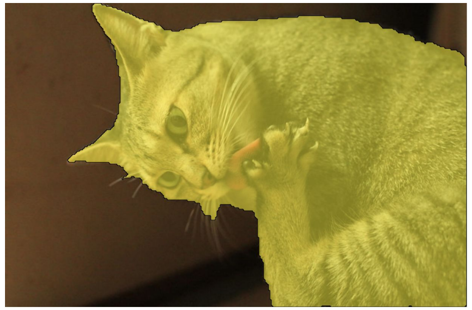
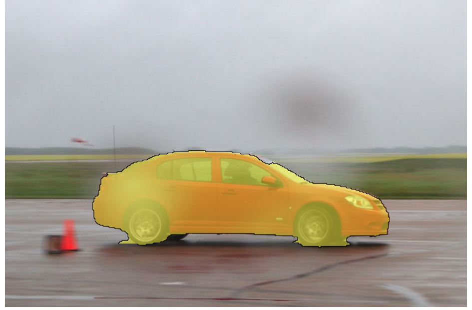




Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

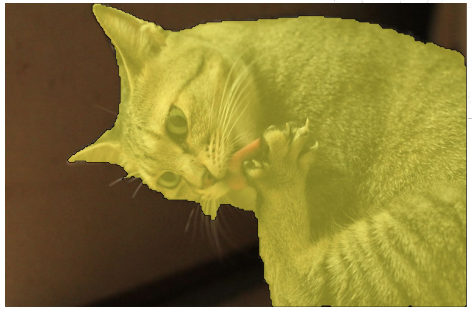




Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\{(x_i, y_i)\}_{i=1}^{N=3}$$

3 images (indexed i=1, i=2, i=3).
Each image has image data (xi)
and a label (yi).

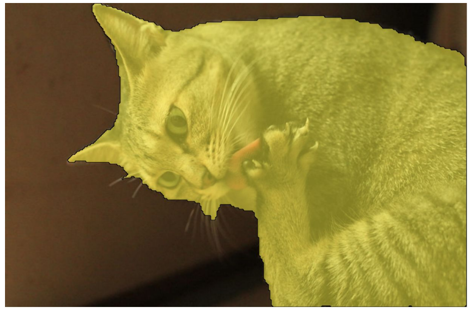
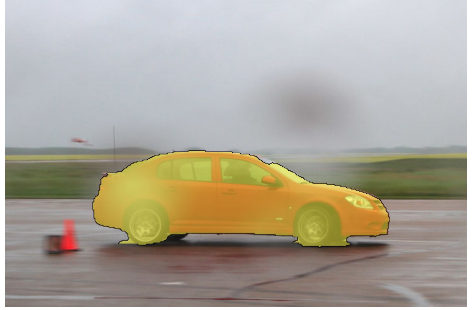
For example:

x1 = 

y1 = "Cat"

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

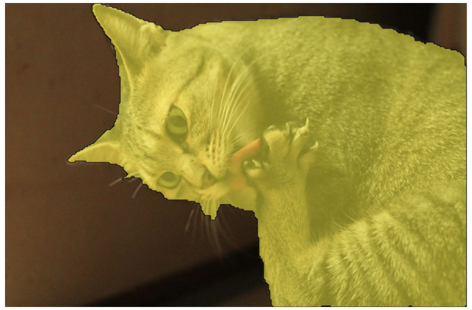
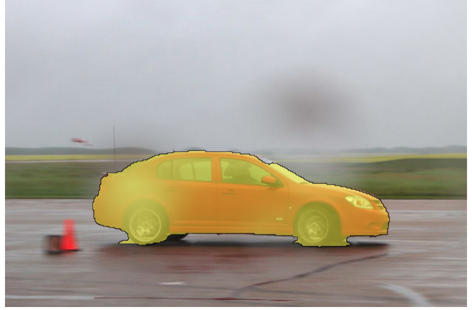

Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} Loss_i(f(x_i, W), y_i)$$

where ***N*** is the total number of images (i.e., 3), ***i*** is a unique index for each image, ***x_i*** is the image itself, ***y_i*** is the image label, ***Loss_i*** is the loss for that image, and **W** is the weights being tested.

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

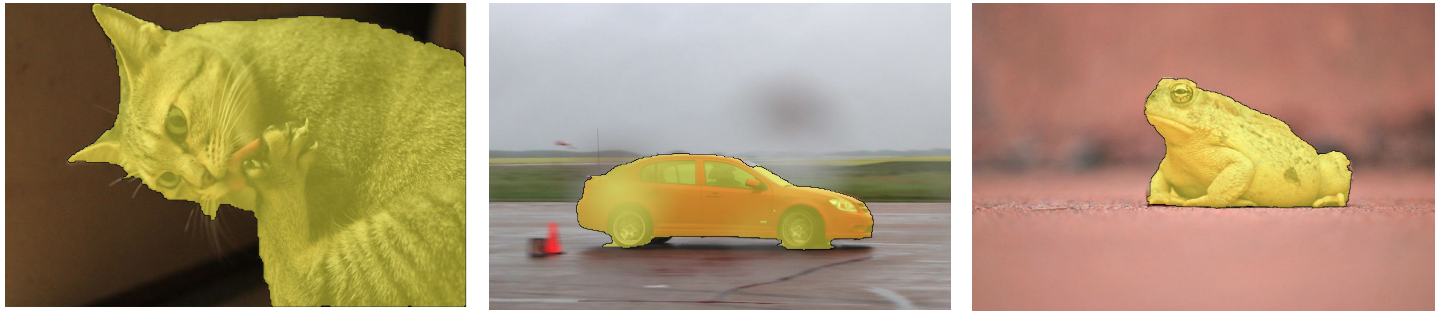

Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} Loss_i(\boxed{f(x_i, W)}, y_i)$$

where ***N*** is the total number of images (i.e., 3), ***i*** is a unique index for each image, ***x_i*** is the image itself, ***y_i*** is the image label, ***Loss_i*** is the loss for that image, and **W** is the weights being tested.

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1



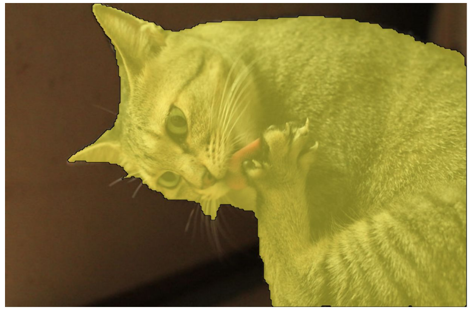
Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

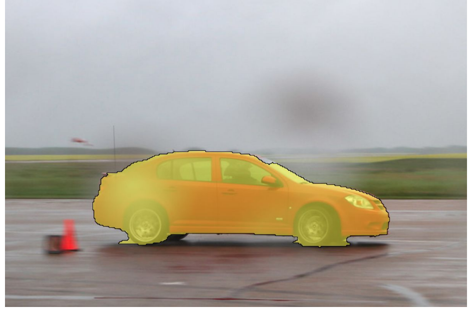
$$\text{Total Loss} = \frac{1}{N}\sum_i^N Loss_i(f(x_i, W)\boxed{, y_i})$$

where ***N*** is the total number of images (i.e., 3), ***i*** is a unique index for each image, ***x_i*** is the image itself, ***y_i*** is the image label, ***Loss_i*** is the loss for that image, and **W** is the weights being tested.

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1





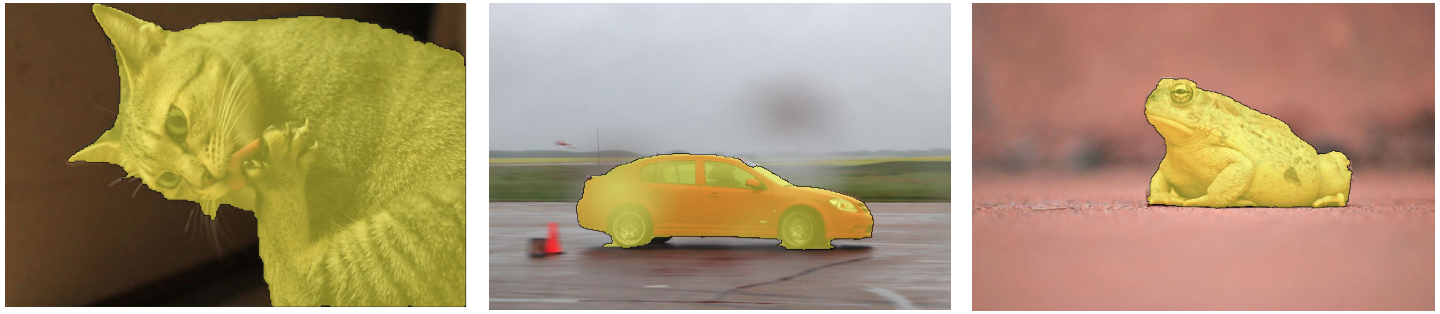
Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} \boxed{Loss_i(f(x_i, W), y_i)}$$

where ***N*** is the total number of images (i.e., 3), ***i*** is a unique index for each image, ***x_i*** is the image itself, ***y_i*** is the image label, ***Loss_i*** is the loss for that image, and **W** is the weights being tested.

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1



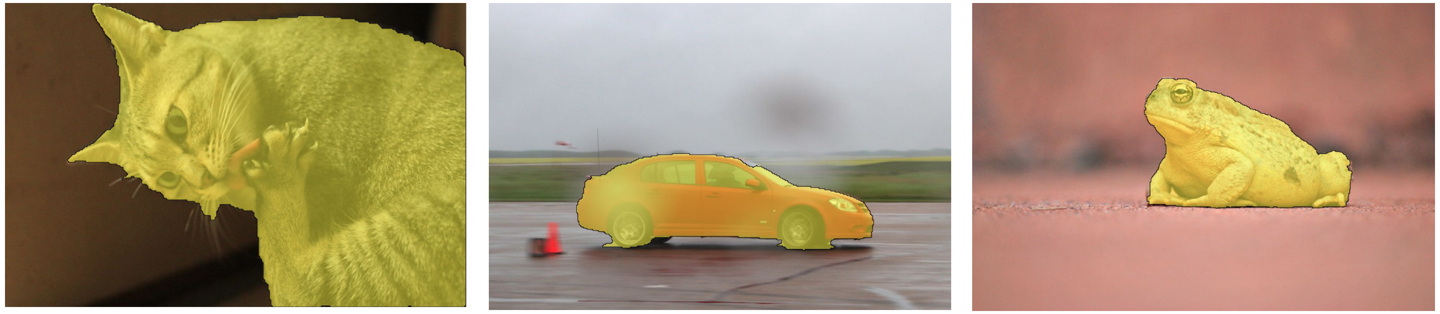
Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} Loss_i(f(x_i, W), y_i)$$

where ***N*** is the total number of images (i.e., 3), ***i*** is a unique index for each image, ***x_i*** is the image itself, ***y_i*** is the image label, ***Loss_i*** is the loss for that image, and **W** is the weights being tested.

f(image, W) = scores

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1



Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

Multiclass SVM Loss

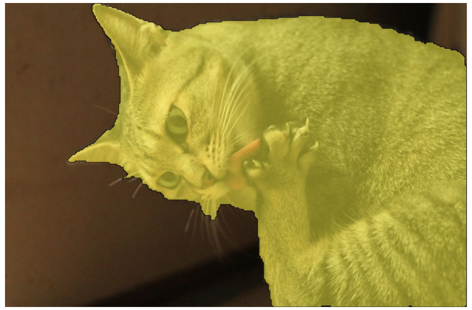
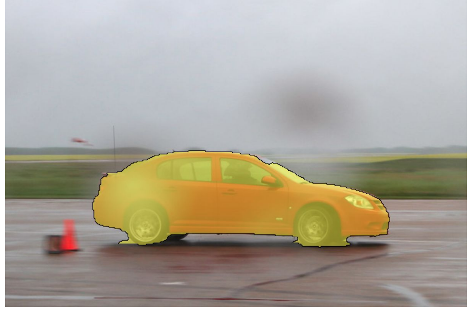
J is the total number of classes, represented by index *j*. In the current example, *j=1* would be “Cat”, *j=2* would be “Car”, etc.

$$\sum_{j \neq y_i}^{J} max(0, s_j - s_{y_i} + \varepsilon)$$

s is the score for a given category. For the first image (the Cat), s_1 would be 3.2, s_2 would be 5.1, and s_3 would be -1.7.

Epsilon (**ε**) is a tolerance term, essentially defining how sure the algorithm needs to be about a class before we call it right.

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

Multiclass SVM Loss

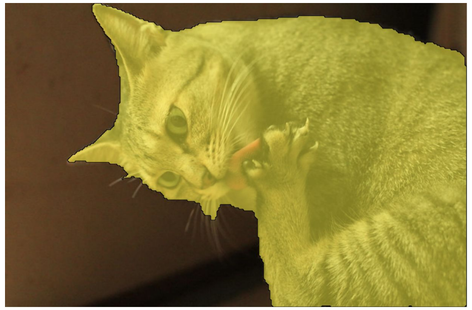
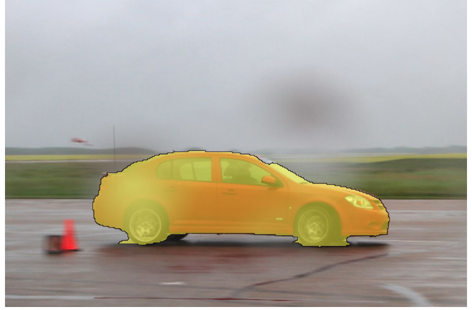
J is the total number of classes, represented by index *j*. In the current example, *j=1* would be "Cat", *j=2* would be "Car", etc.

s is the score for a given category. For the first image (the Cat), s_1 would be 3.2, s_2 would be 5.1, and s_3 would be -1.7.

Epsilon (**ε**) is a tolerance term, essentially defining how sure the algorithm needs to be about a class before we call it right.

$$\boxed{\sum_{j \neq y_i}^{J}} max(0, s_j - s_{y_i} + \varepsilon)$$

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

Multiclass SVM Loss

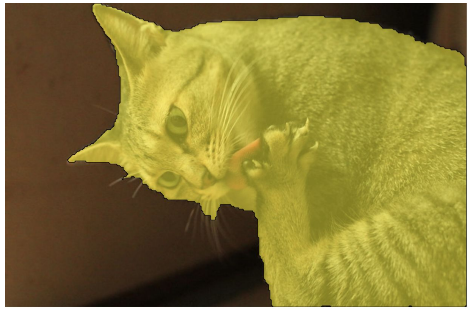
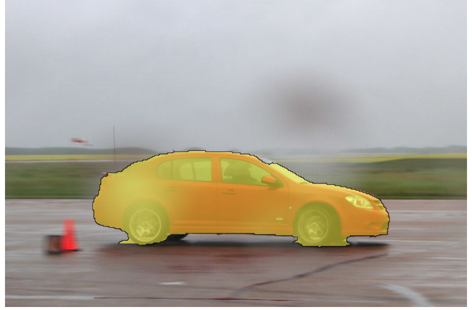
J is the total number of classes, represented by index *j*. In the current example, *j=1* would be "Cat", *j=2* would be "Car", etc.

s is the score for a given category. For the first image (the Cat), s_1 would be 3.2, s_2 would be 5.1, and s_3 would be -1.7.

Epsilon (**ε**) is a tolerance term, essentially defining how sure the algorithm needs to be about a class before we call it right.

$$\sum_{j \neq y_i}^{J} \boxed{max(0, s_j - s_{y_i} + \varepsilon)}$$

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

J is the total number of classes, represented by index *j*. In the current example, *j=1* would be "Cat", *j=2* would be "Car", etc.

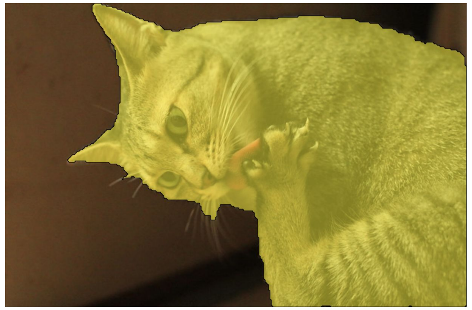
s is the score for a given category. For the first image (the Cat), s_1 would be 3.2, s_2 would be 5.1, and s_3 would be -1.7.

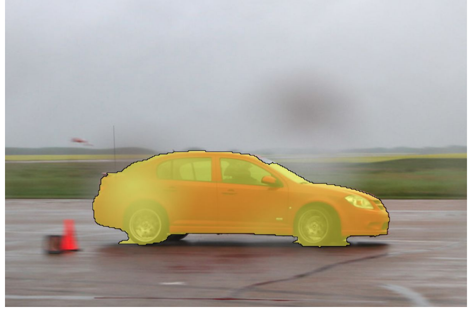
Epsilon (**ε**) is a tolerance term, essentially defining how sure the algorithm needs to be about a class before we call it right.

Multiclass SVM Loss

$$\sum_{j \neq y_i}^{J} max(0, s_j - s_{y_i} + \varepsilon)$$

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1



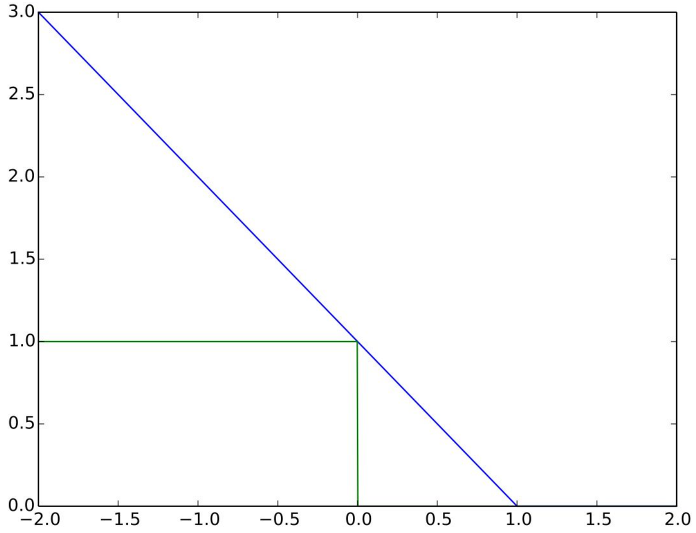


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

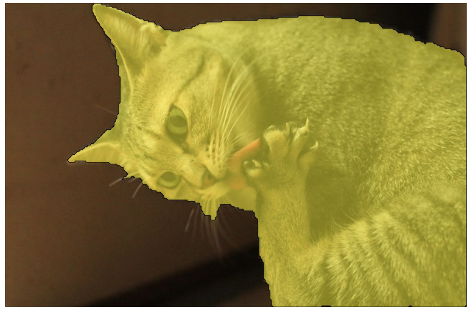
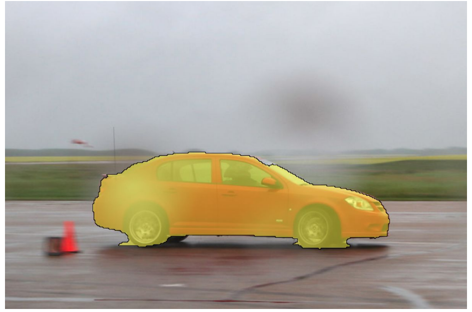
Multiclass SVM Loss

Epsilon (**ε**) is a tolerance term, essentially defining how sure the algorithm needs to be about a class before we call it right.

$$\sum_{j \neq y_i}^{J} max(0, s_j - s_{y_i} + \boxed{\varepsilon})$$



Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

If we set Epsilon = 1

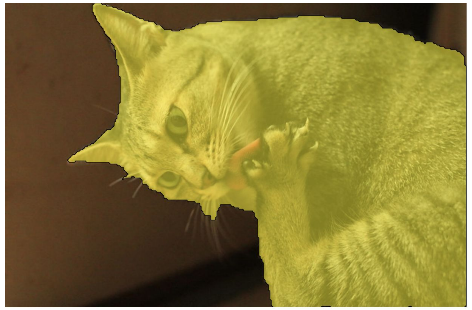
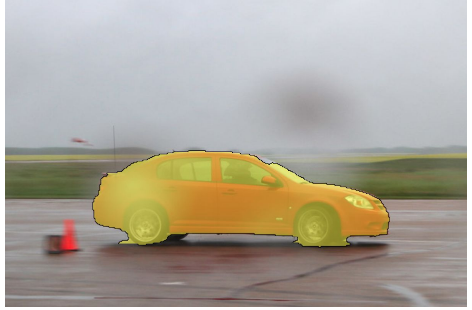
Multiclass SVM Loss

$$\sum_{j\neq y_i}^{J} max(0, \boxed{s_j} - \boxed{s_{y_i}} + \varepsilon)$$

Image X_1 (Cat) Loss:

max(0, 5.1 - 3.2 + 1) =
max(0, 2.9) =
2.9

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

If we set Epsilon = 1

Multiclass SVM Loss

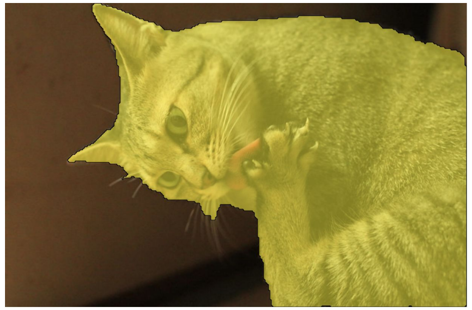
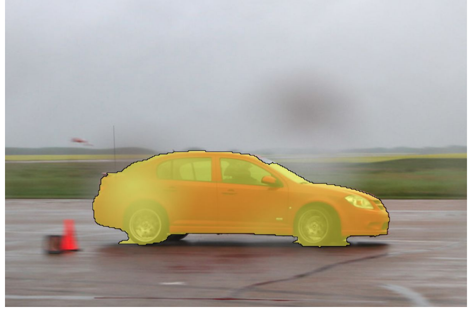
$$\sum_{j \neq y_i}^{J} max(0, \boxed{s_j} - \boxed{s_{y_i}} + \varepsilon)$$

Image X_1 (Cat) Loss:

<u>Car</u>
max(0, 5.1 - 3.2 + 1) =
max(0, 2.9) =
2.9

<u>Frog</u>
max(0, -1.7 - 3.2 + 1) =
max(0, -3.9) =
0

Cat	**3.2**	**1.3**	**2.2**
Car	**5.1**	**4.9**	**2.5**
Frog	**-1.7**	**2.0**	**-3.1**


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

If we set Epsilon = 1

$$\sum_{j \neq y_i}^{J} max(0, \boxed{s_j} - \boxed{s_{y_i}} + \varepsilon)$$

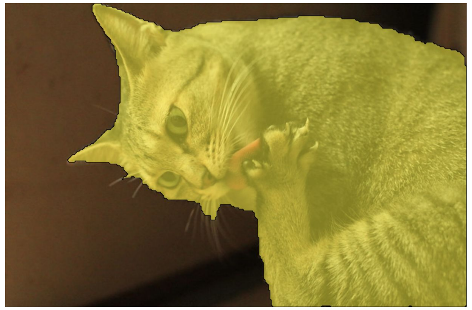
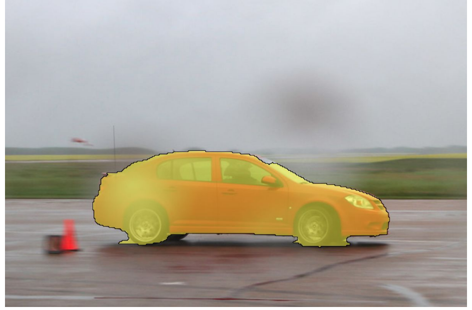
Multiclass SVM Loss

Image X_2 (Car) Loss:

Cat
max(0, 1.3 - 4.9 + 1) =
max(0, -2.6) =
0

Frog
max(0, 2.0 - 4.9 + 1) =
max(0, -1.9) =
0

Cat	**3.2**	**1.3**	**2.2**
Car	**5.1**	**4.9**	**2.5**
Frog	**-1.7**	**2.0**	**-3.1**


Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

If we set Epsilon = 1

Multiclass SVM Loss

$$\sum_{j \neq y_i}^{J} max(0, \boxed{s_j} - \boxed{s_{y_i}} + \varepsilon)$$

Image X_2 (Car) Loss:

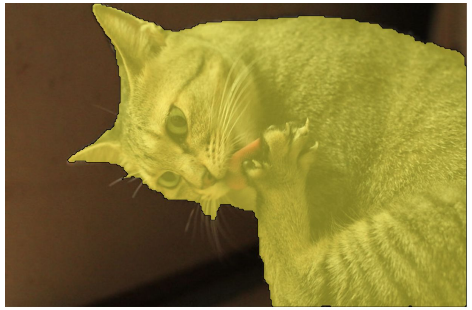
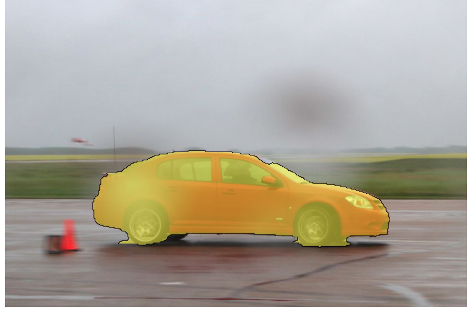
<u>Cat</u>

max(0, 2.2 - -3.1 + 1) =
max(0, 6.3) =
6.3

<u>Car</u>

max(0, 2.5 - -3.1 + 1) =
max(0, -6.6) =
6.6

Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

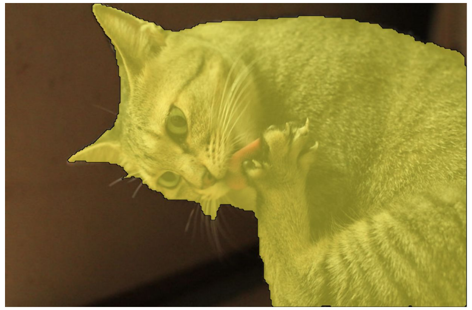



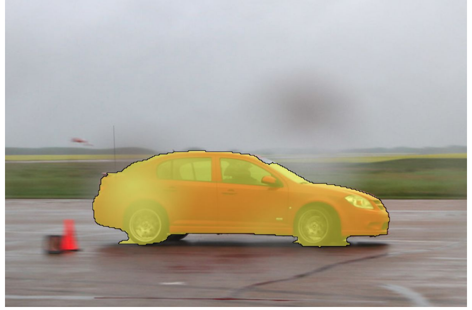

Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} \boxed{Loss_i(f(x_i, W), y_i)}$$

$$\boxed{\sum_{j \neq y_i}^{J} max(0, s_j - s_{y_i} + \varepsilon)}$$

Loss	2.9	0	12.9
Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1





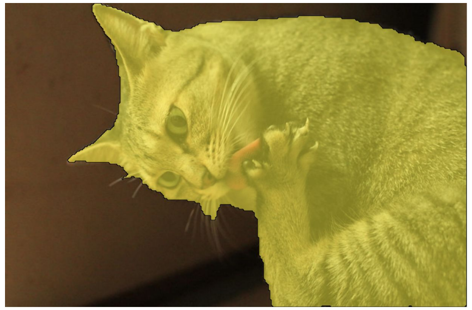
Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

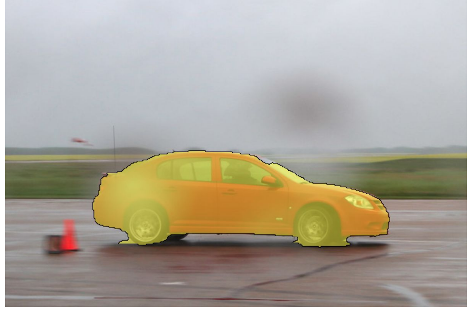
$$\text{Total Loss} = \frac{1}{N}\sum_{i}^{N} \boxed{Loss_i(f(x_i, W), y_i)}$$

$$\boxed{\sum_{j \neq y_i}^{J} max(0, s_j - s_{y_i} + \varepsilon)}$$

(2.9 + 0 + 12.9) / 3 = ~5.27

Loss	2.9	0	12.9
Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1





Images CC Attribution, Left to Right: Yutaka Fujiki, Grant C., Zion National Park

Wrap Up

- Parametric Models
- Linear Classifier
  - Solving
  - Visualizing
- Loss Functions
  - Multiclass SVM Loss