Object Recognition via Convolution

An Example of Global Optimization

Gang Pan
gpan@zju.edu.cn

Department of Computer Science
Zhejiang University

Slide credit: Stanford CS 131

Recall convolutions...

12	3	19
25	10	1
9	7	17

*

1	2
3	4

=

?	?
?	?

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions…

$$\begin{bmatrix} 12 & 3 & 19 \\ 25 & 10 & 1 \\ 9 & 7 & 17 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 133 & ? \\ ? & ? \end{bmatrix}$$

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions...

12	3	19
25	10	1
9	7	17

*

1	2
3	4

=

133	75
?	?

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions...

12	3	19
25	10	1
9	7	17

$*$

1	2
3	4

$=$

133	75
100	?

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions...

12	3	19
25	10	1
9	7	17

*

1	2
3	4

=

133	75
100	101

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions...

$$\begin{bmatrix} 12 & 3 & 19 \\ 25 & 10 & 1 \\ 9 & 7 & 17 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 133 & 75 \\ 100 & 101 \end{bmatrix}$$

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Recall convolutions…

$$\begin{array}{|c|c|}\hline 12 & 21 \\ \hline 18 & 31 \\ \hline \end{array} * \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} = \boxed{?}$$

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

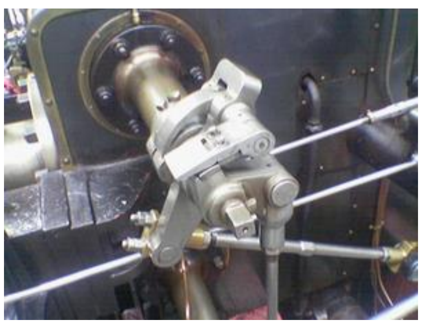
Recall convolutions...

$$\begin{bmatrix} 12 & 21 \\ 18 & 31 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 232 \end{bmatrix}$$

$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$

Why they are useful

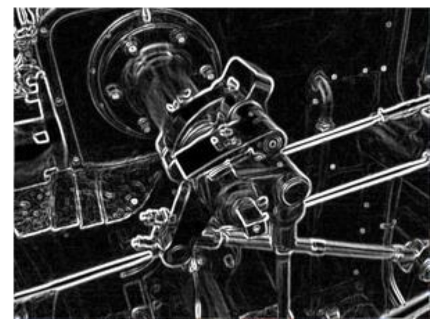
Allow us to find **interesting insights/features** from images!



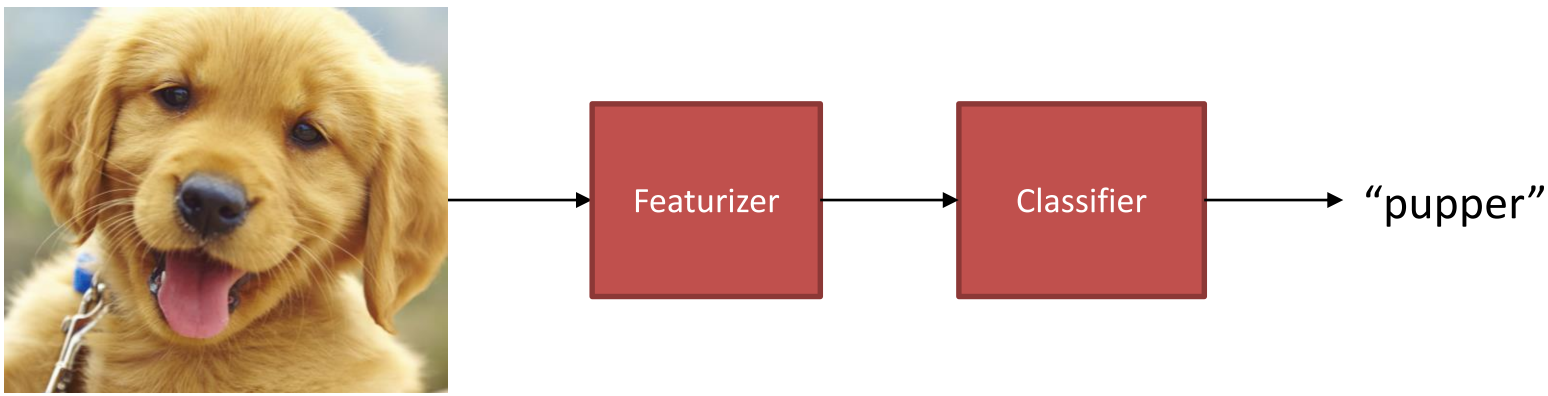
*

0	-½	0
0	0	0
0	½	0

=



Recall Image Classification…



Allow us to use features to put **images in categories**!

Wait a Minute...

Convolution = Image -> Features

Classification Algorithm = Features -> Category

Wait a Minute…

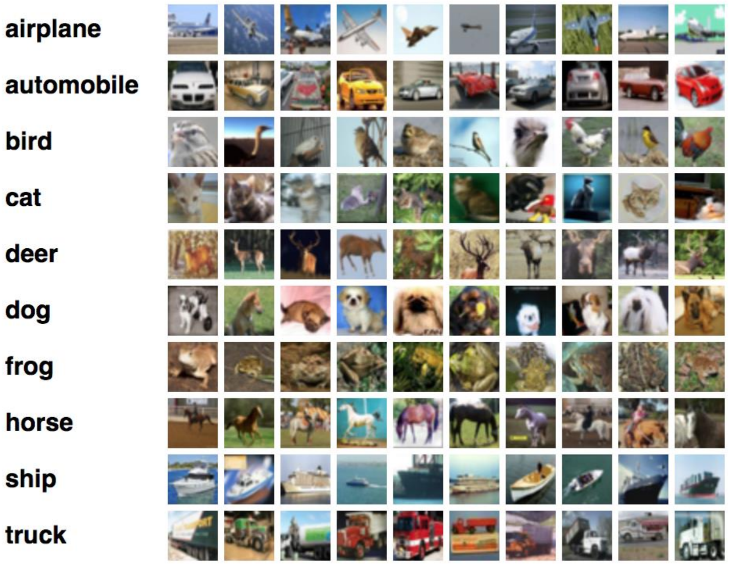
Convolution = Image -> Features

Classification Algorithm = Features -> Category

Let’s put ‘em together!

In Specific…

Let's build a **convolution-based** classification algorithm for the CIFAR-10 dataset (10 classes, 32x32 images):



Feature Extractor

*

32x32 “Airplane Filter”

=

Feature Extractor

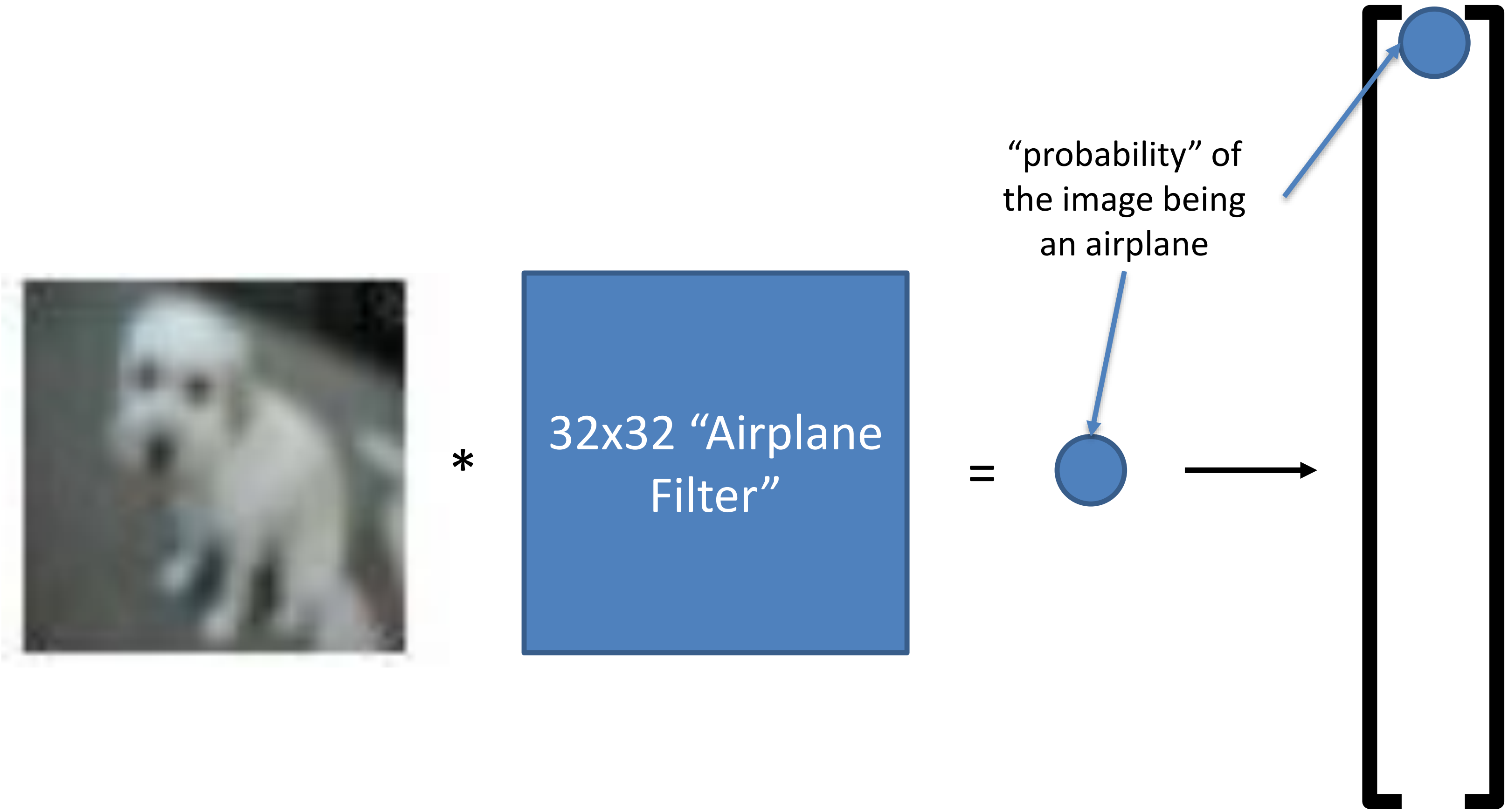
“probability” of
the image being
an airplane

*

32x32 “Airplane
Filter”

=

Feature Extractor



Feature Extractor

This is not really a probability but a score, because it can be less than 0 and greater than 1

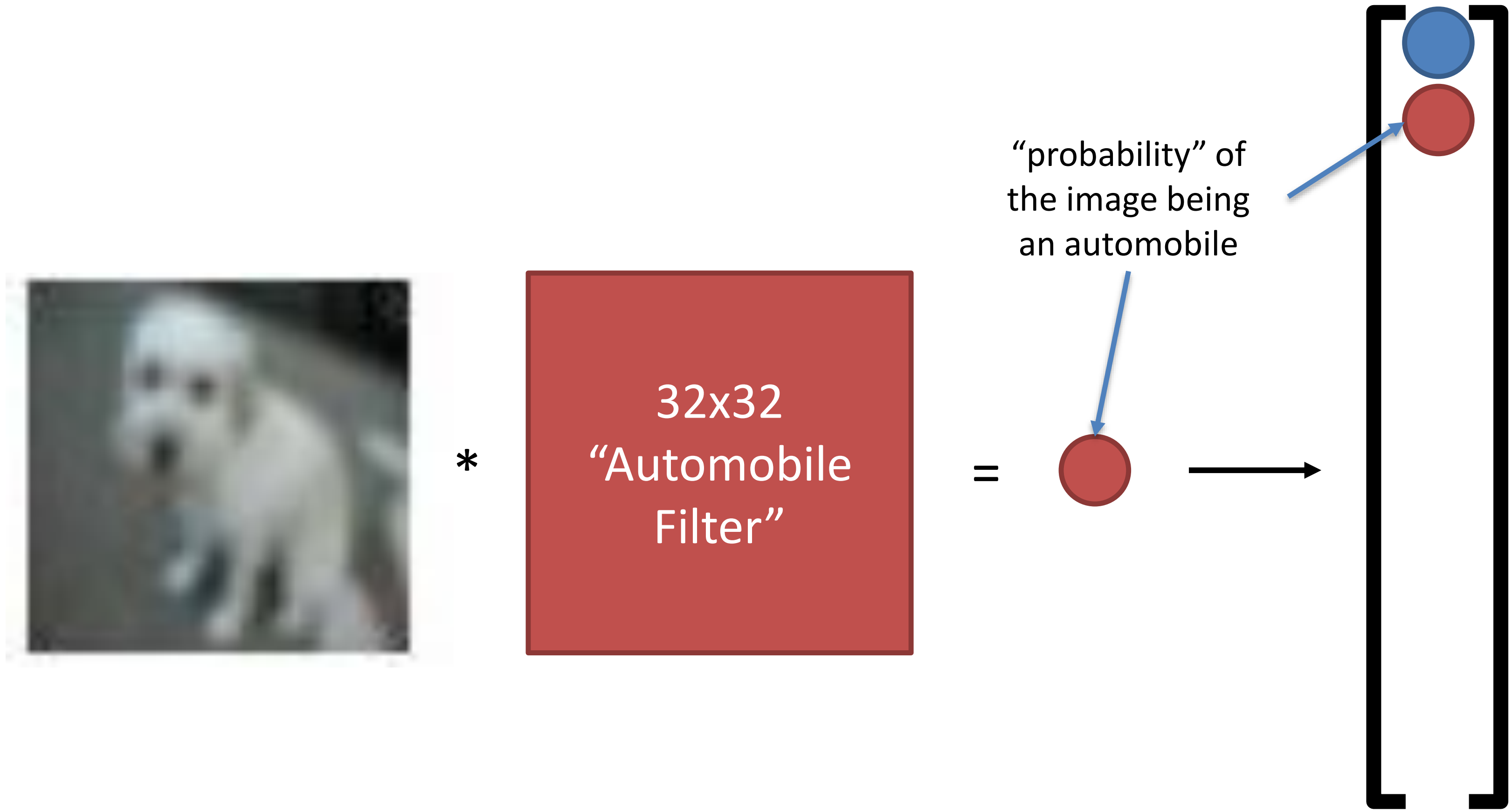
"probability" of the image being an airplane

*

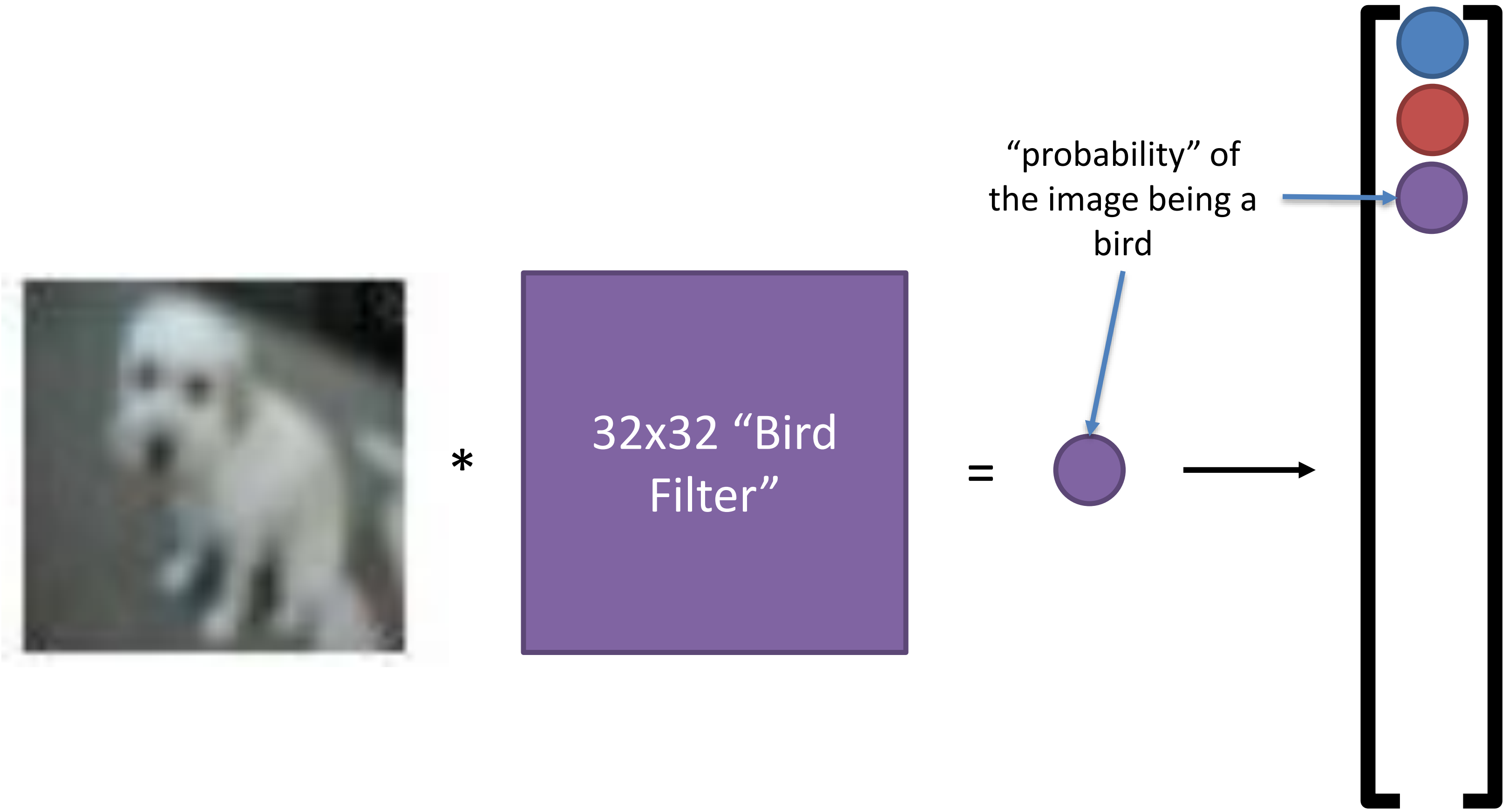
32x32 "Airplane Filter"

=

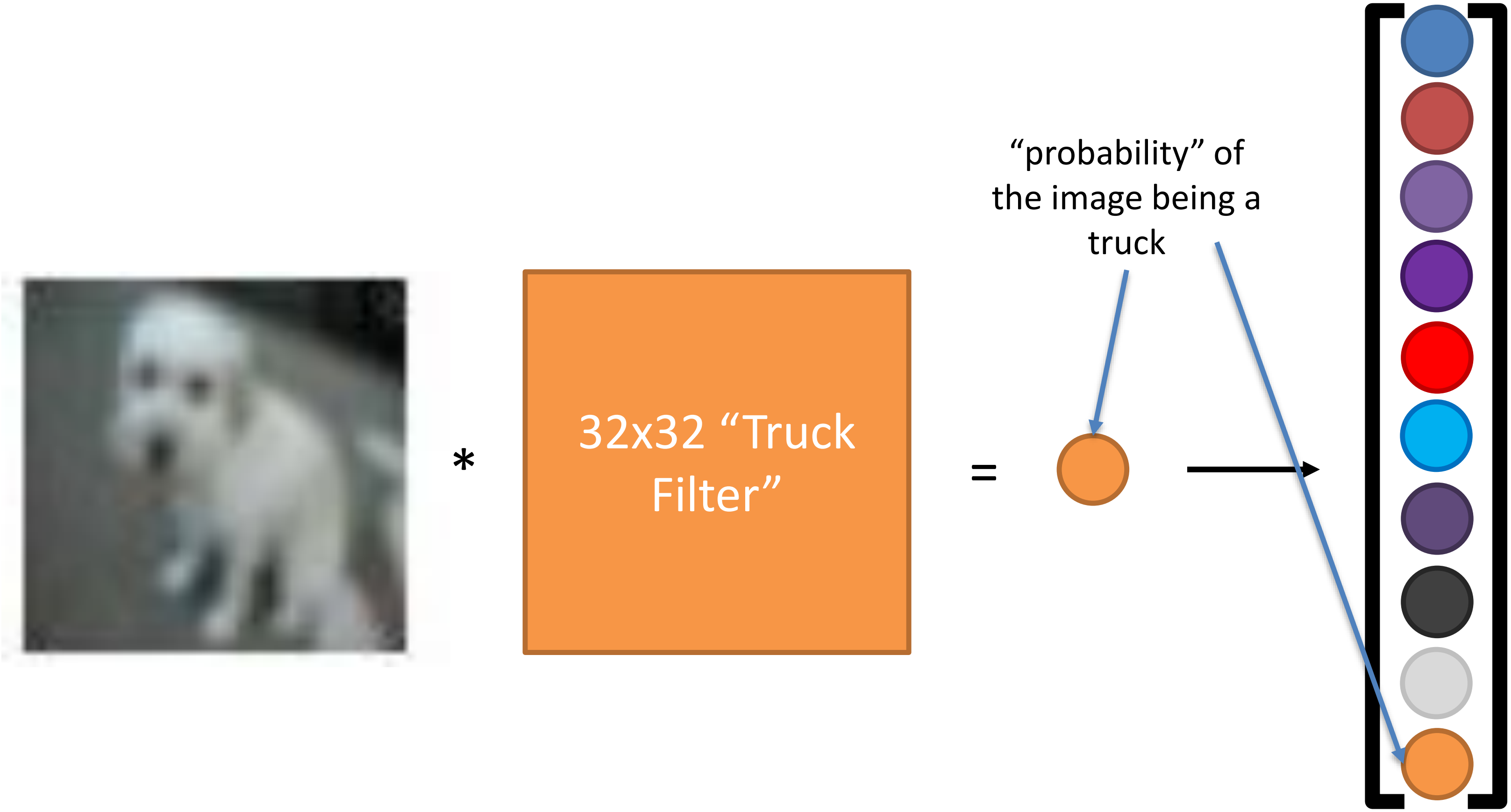
Feature Extractor



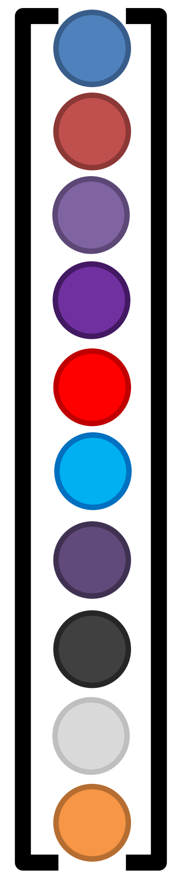
Feature Extractor



Feature Extractor

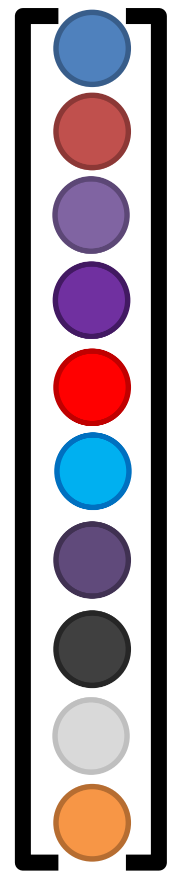


Classifier

$c_{pred} = \arg\max($  $)$

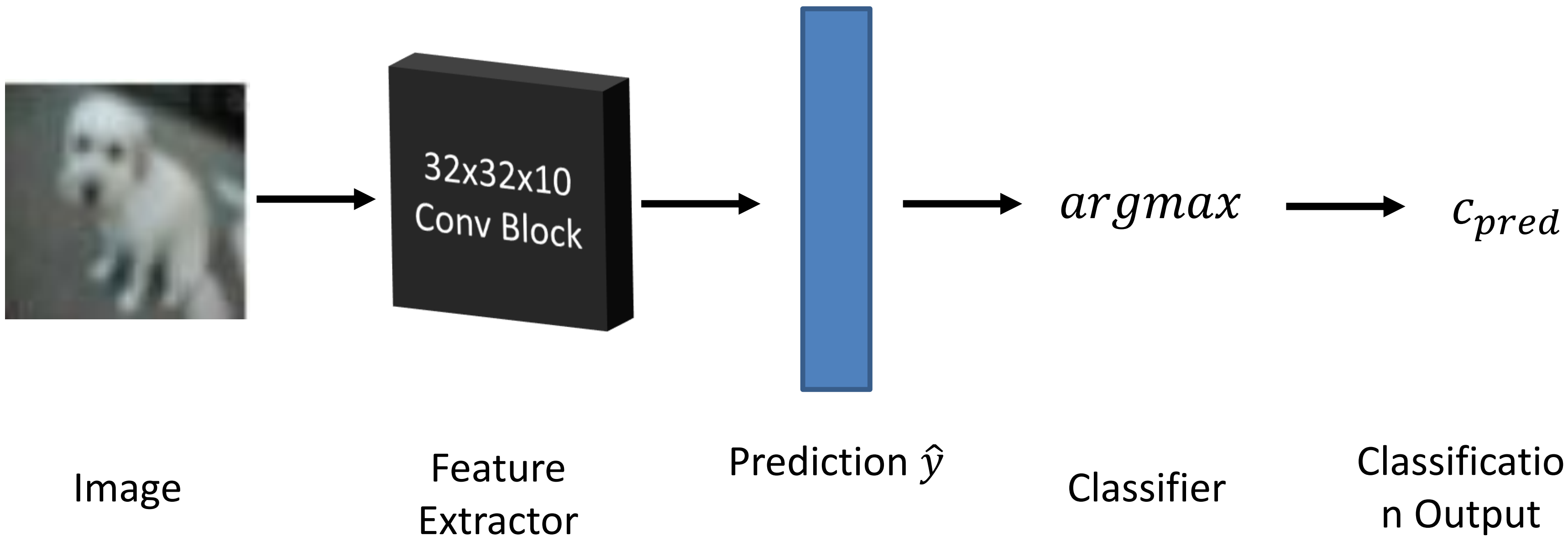
Classifier

$$c_{pred} = \arg\max(\quad)$$

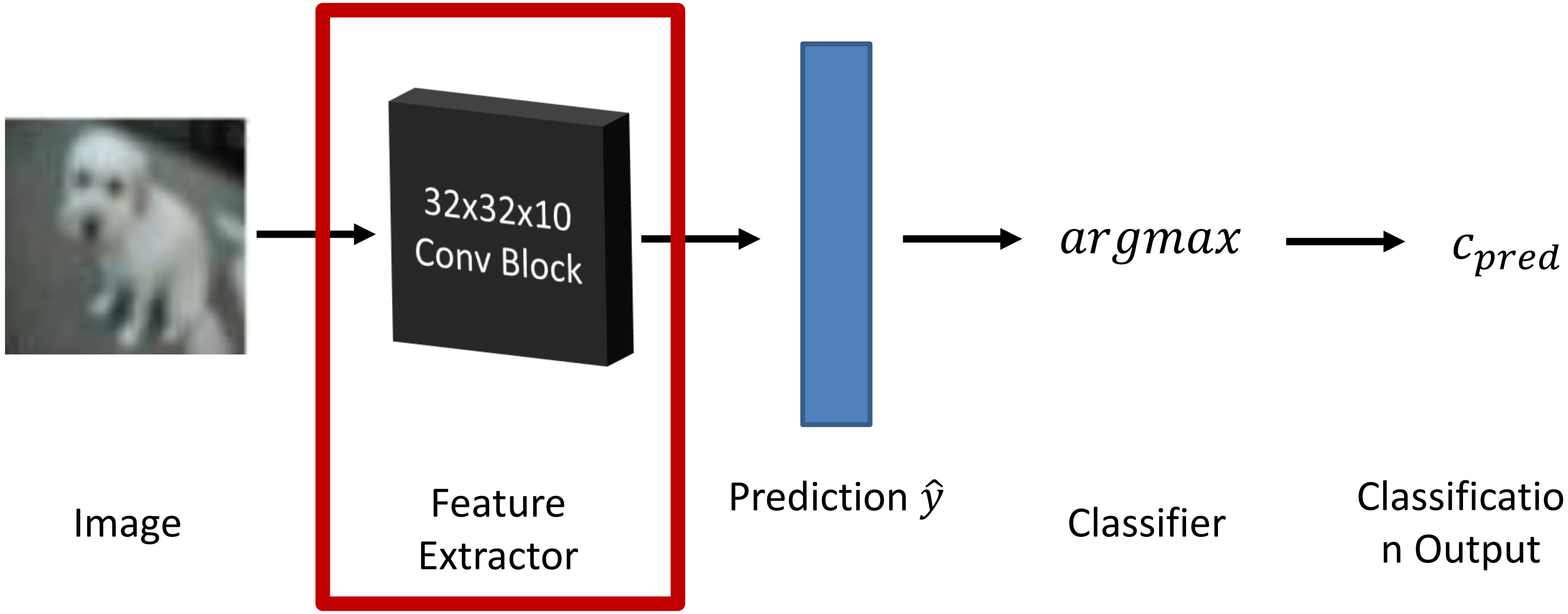


We predict the class that
has the highest probability!

The Whole Shebang



The Whole Shebang



The Whole Shebang

Image

Extractor

c_{pred}

Classificatio
n Output

Reframing convolution

12	21
18	31

*

1	2
3	4

Reframing convolution

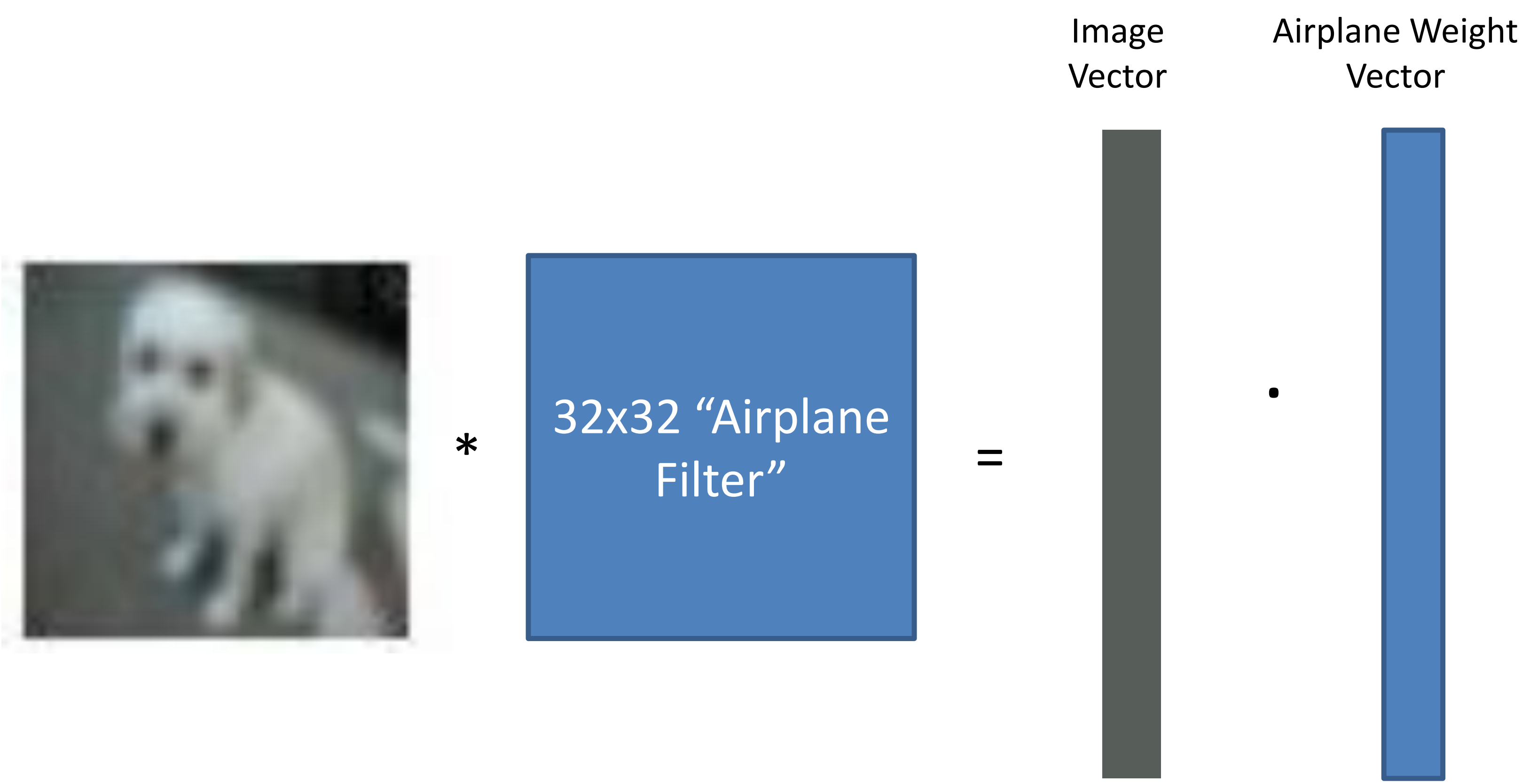
$$\begin{array}{|c|c|}\hline 12 & 21 \\ \hline 18 & 31 \\ \hline \end{array} * \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} = \begin{bmatrix} 12 \\ 21 \\ 18 \\ 31 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Reframed Feature Extractor

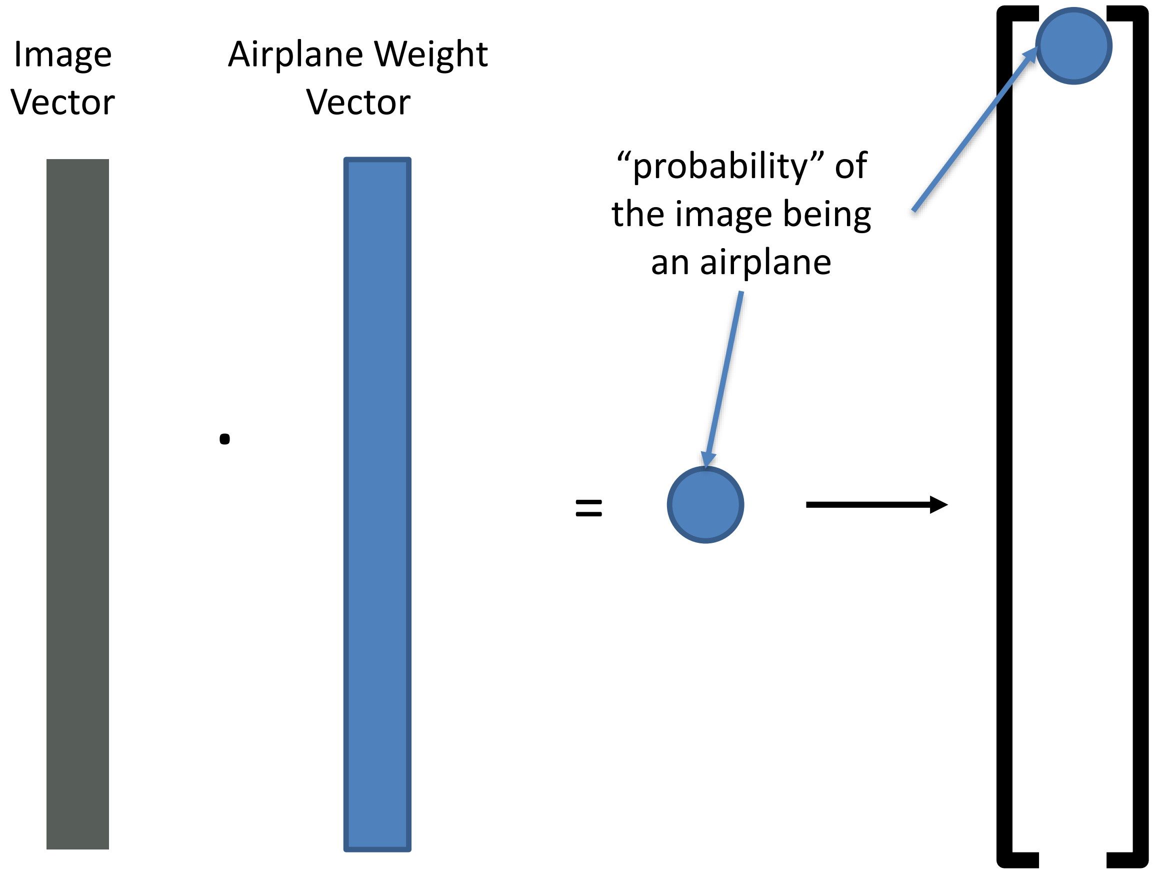
*

32x32 “Airplane Filter”

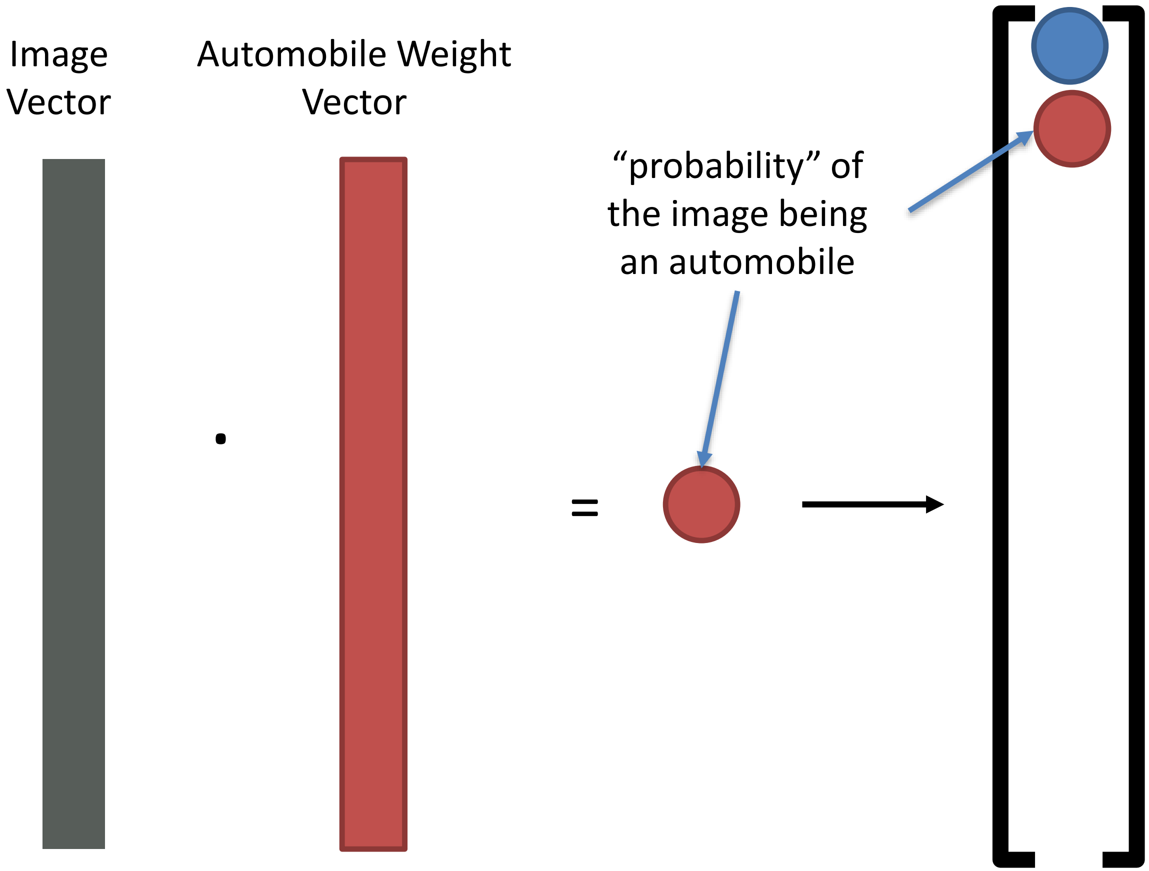
Reframed Feature Extractor



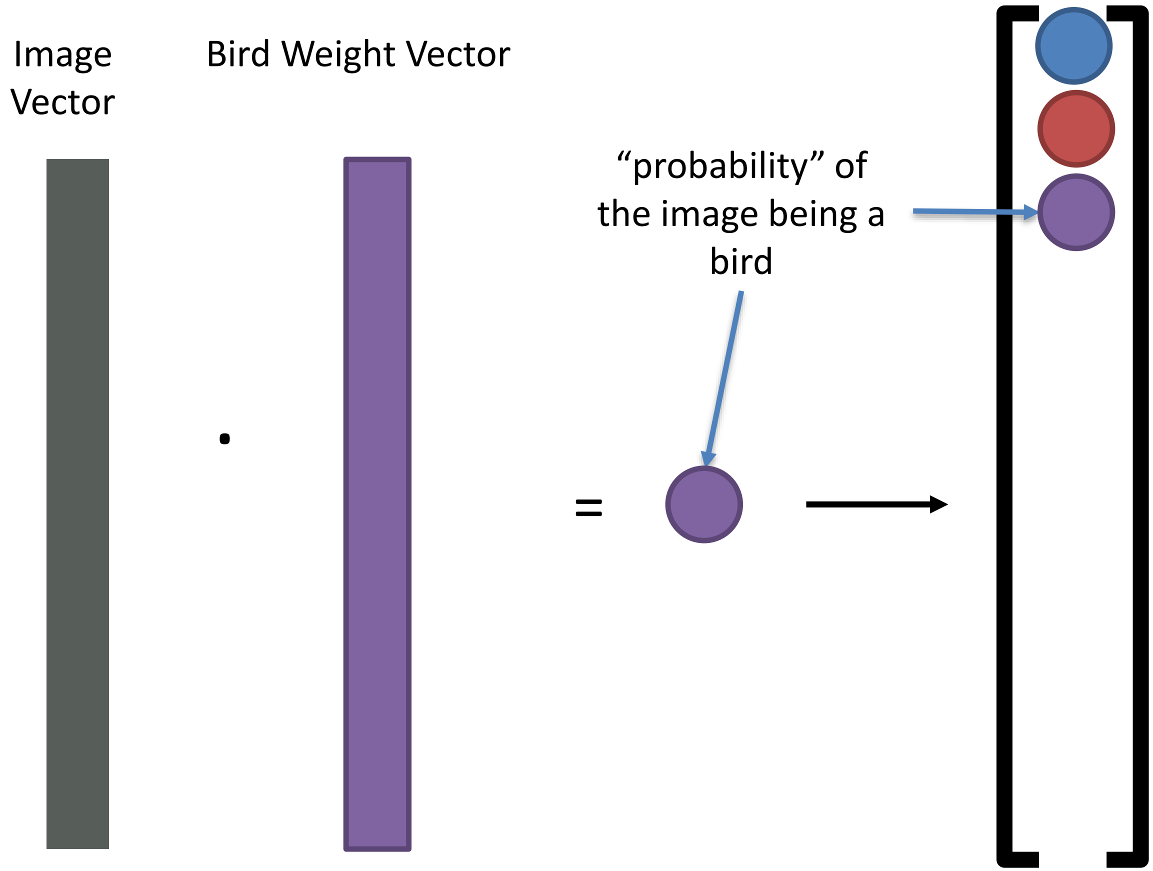
New Feature Extractor



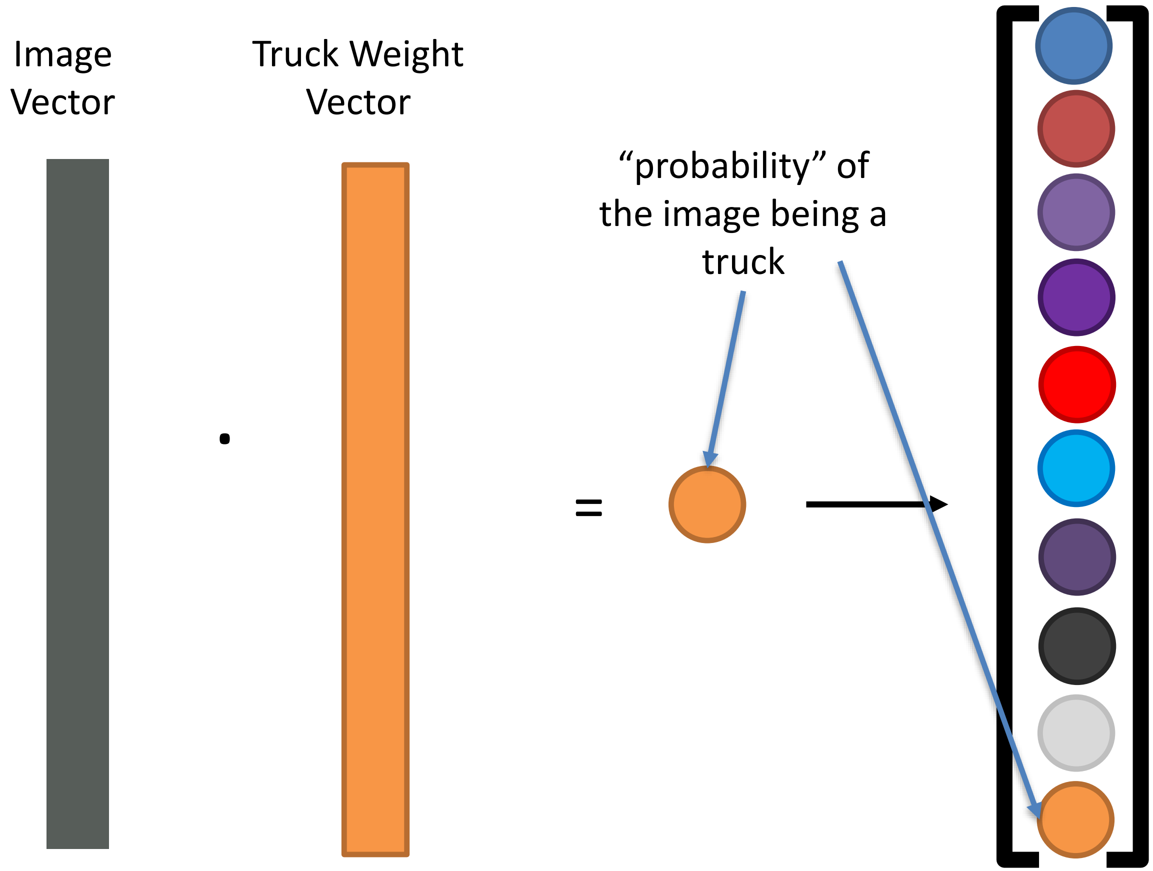
New Feature Extractor



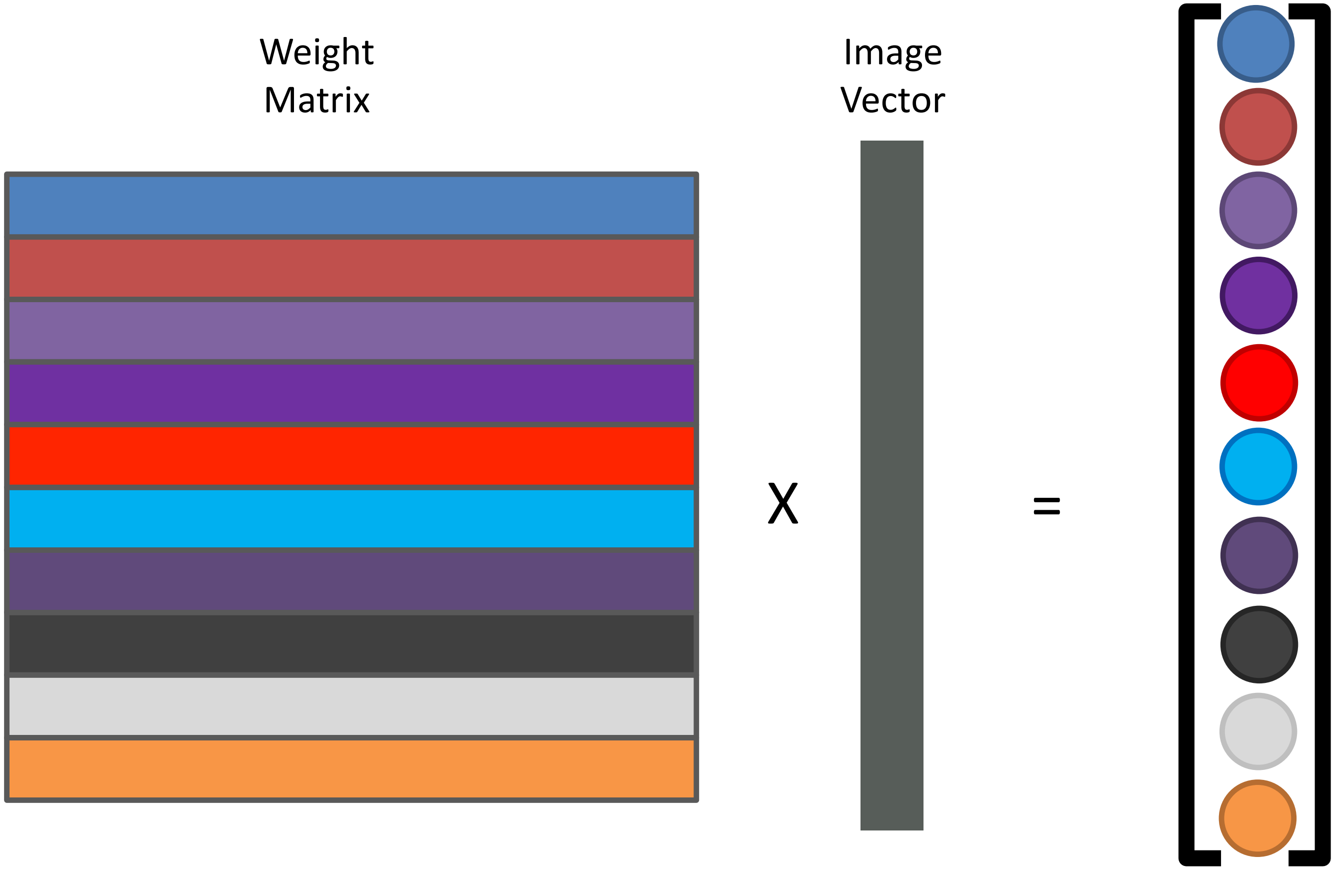
New Feature Extractor



New Feature Extractor



New Feature Extractor



New Feature Extractor

$$Wx = \hat{y}$$

W: the (10x1024) matrix of weight vectors

x: the (1024x1) image vector

$\hat{y}$: the (10x1) vector of class "probabilities"

New Feature Extractor

This simple computation is called a *fully-connected layer*!

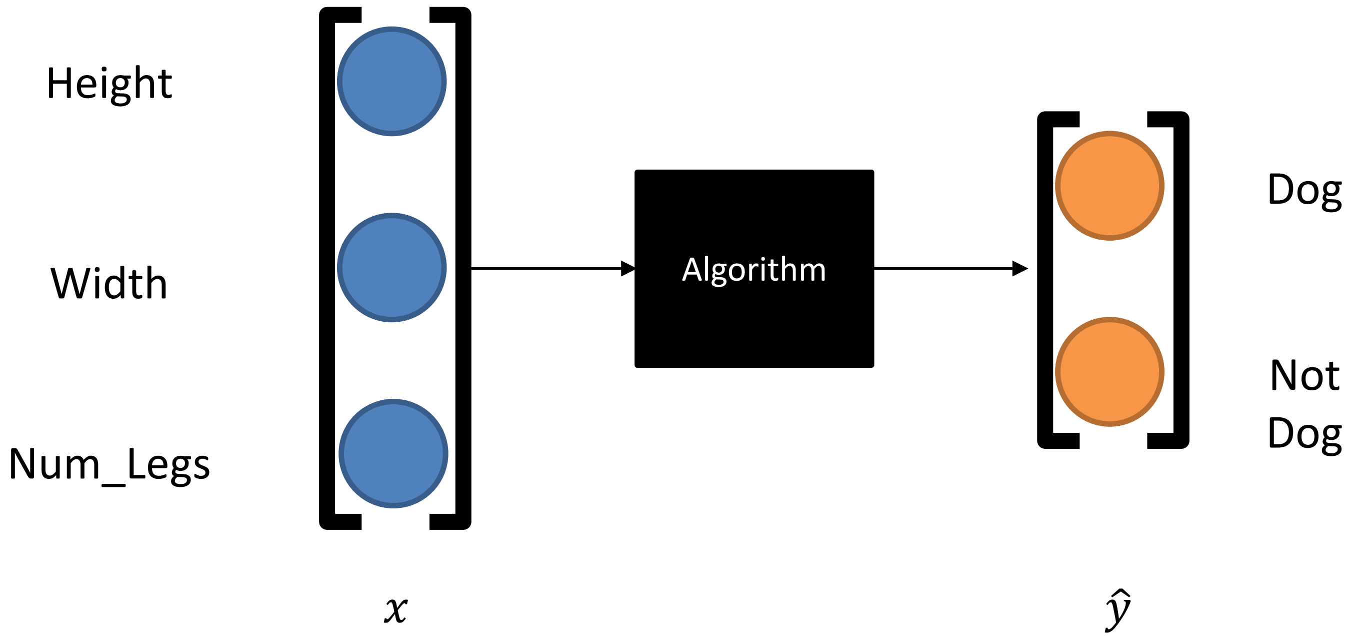
$$\boxed{Wx} = \hat{y}$$

W: the (10x1024) matrix of weight vectors

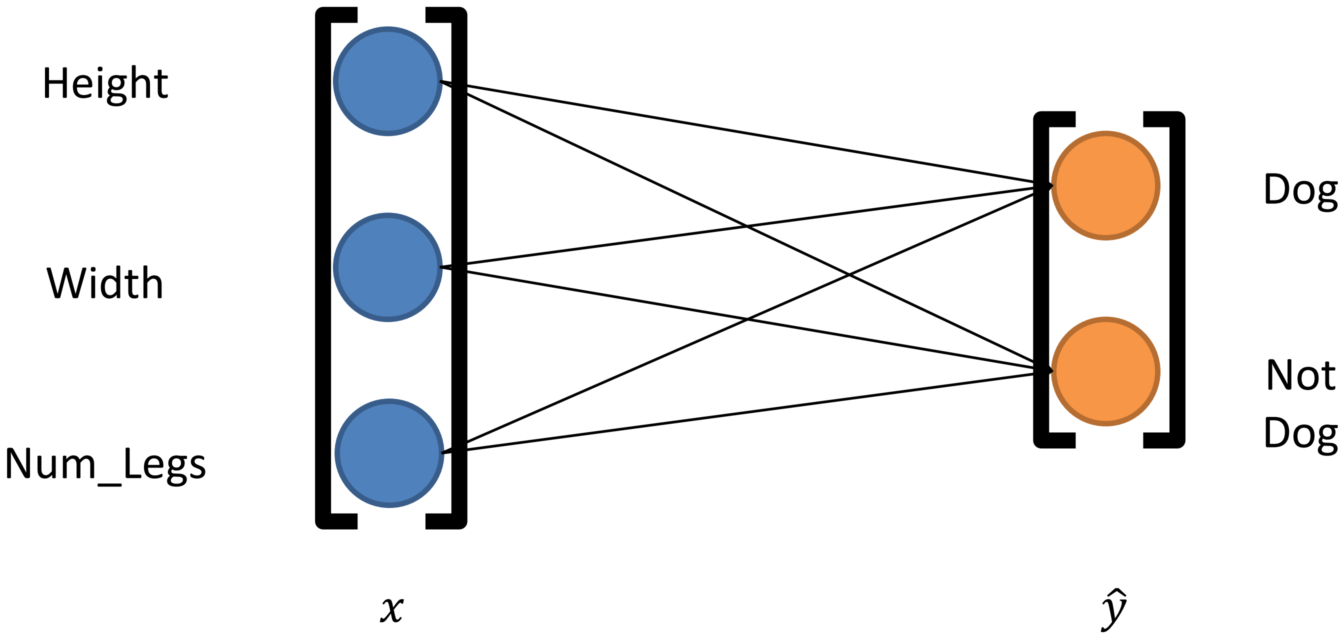
x: the (1024x1) image vector

$\hat{y}$: the (10x1) vector of class “probabilities”

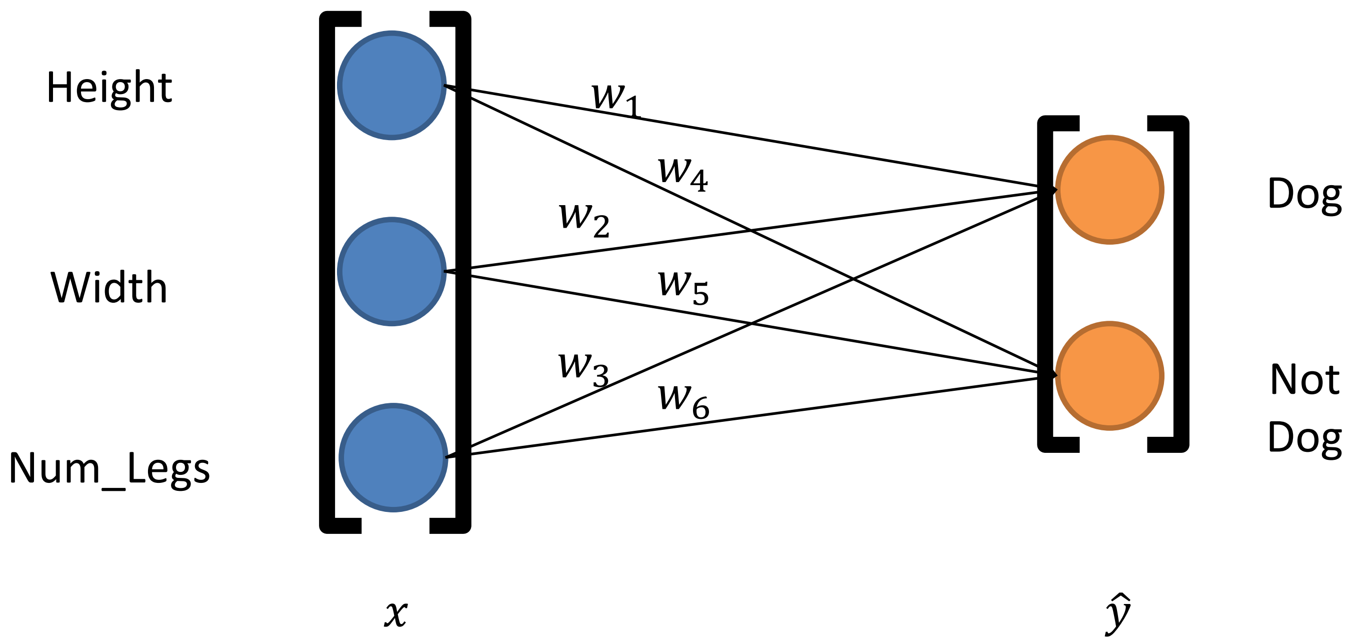
Aside: Fully-Connected Neural Networks



Aside: Fully-Connected Neural Networks

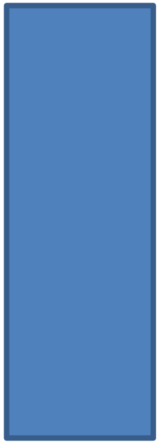


Aside: Fully-Connected Neural Networks

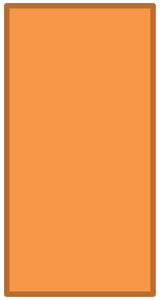


Aside: Fully-Connected Neural Networks

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \end{bmatrix} \cdot$$



$$=$$

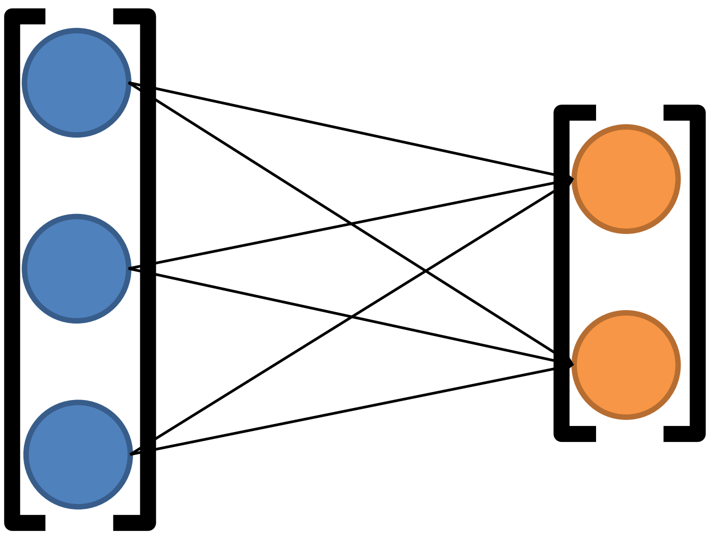


W $\quad$ x $\quad$ $\hat{y}$

$$Wx = \hat{y}$$

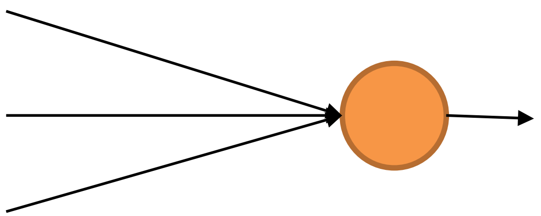
Aside: Fully-Connected Neural Networks

"Fully-Connected"



Every node is connected
to every other node

"Neural Network"



Kinda looks like a
neuron!

New Feature Extractor

$$Wx = \hat{y}$$

W: the (10x1024) matrix of weight vectors

x: the (1024x1) image vector

$\hat{y}$: the (10x1) vector of class "probabilities"

New Feature Extractor

$$Wx = \hat{y}$$

W: the (10x1024) matrix of weight vectors

x: the (1024x1) image vector

$\hat{y}$: the (10x1) vector of class “probabilities”?

Class Probability Vector

- Must have values between 0 and 1
- Must sum to 1
- There’s no guarantee either requirement is satisfied!

$$\hat{y} = Wx$$

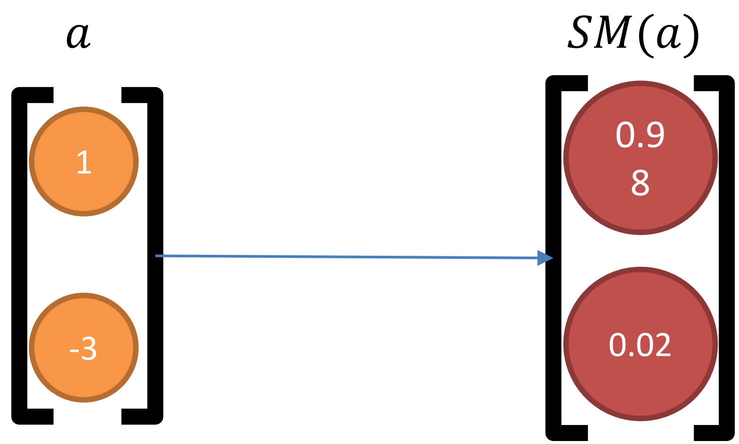
Softmax Function

a

$$\begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

Softmax: $a(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Softmax Function



Softmax: $a(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Class Probability Vector

- Must have values between 0 and 1
- Must sum to 1

$$\hat{y} = Wx$$

Class Probability Vector

- Must have values between 0 and 1
- Must sum to 1

$$\hat{y} = SM(Wx)$$

System so far...

- Feature extractor:
- Classifier:

$$\hat{\mathrm{y}} = SM(Wx)$$

$$c_{pred} = \arg\max(\hat{y})$$

System so far…

- Feature extractor:
- Classifier:

$$\hat{y} = SM(\boxed{W}x)$$

$$c_{pred} = \arg\max(\hat{y})$$

System so far…

- Feature extractor:

- Classifier:

Using the label

Let's compare our prediction with the real answer!
For each image, we have the label y which tells us the true class:

X

y

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dog class index

Key Insight:

We want:

$$\arg\max(\hat{y}) = \arg\max(y)$$

Key Insight:

We want:

$$\arg\max(\hat{y}) = \arg\max(y)$$

Which we can accomplish by:

$$W^* = \arg\min_{W}\left(-\sum_{x,y}\log(p_c)\right)$$

Key Insight:

We want:

$$\arg\max(\hat{y}) = \arg\max(y)$$

Which we can accomplish by:

$$W^* = \arg\min_{W}\left(-\sum_{x,y}\log(p_c)\right)$$

Where p_c is the probability of the true class in $\hat{y}$

Cross-Entropy Loss

Our loss function represents *how bad we are currently doing*:

$$L = -\log(p_c)$$

Cross-Entropy Loss

Our loss function represents *how bad we are currently doing*:

$$L = -\log(p_c)$$

Examples:

$$p_c = 0 \rightarrow L = -\log(0) = \infty$$

$$p_c = 0.1 \rightarrow L = -\log(0.1) = 2.3$$

$$p_c = 0.9 \rightarrow L = -\log(0.9) = 0.1$$

$$p_c = 1 \rightarrow L = -\log(1) = 0$$

Cross-Entropy Loss

Our loss function represents *how bad we are currently doing*:

$$L = -\log(p_c)$$

Examples:

$$p_c = 0 \rightarrow L = -\log(0) = \infty$$

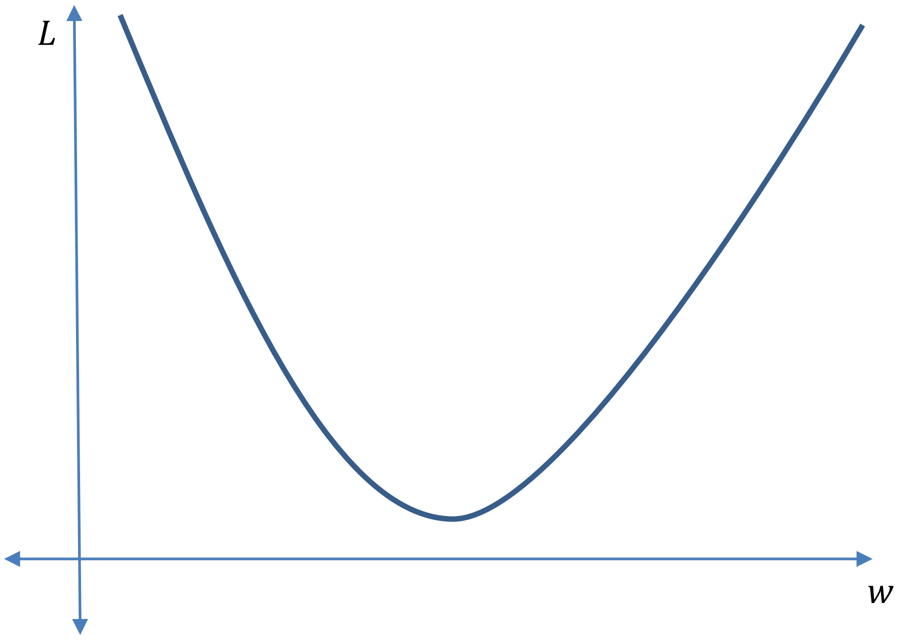
$$p_c = 0.1 \rightarrow L = -\log(0.1) = 2.3$$

$$p_c = 0.9 \rightarrow L = -\log(0.9) = 0.1$$

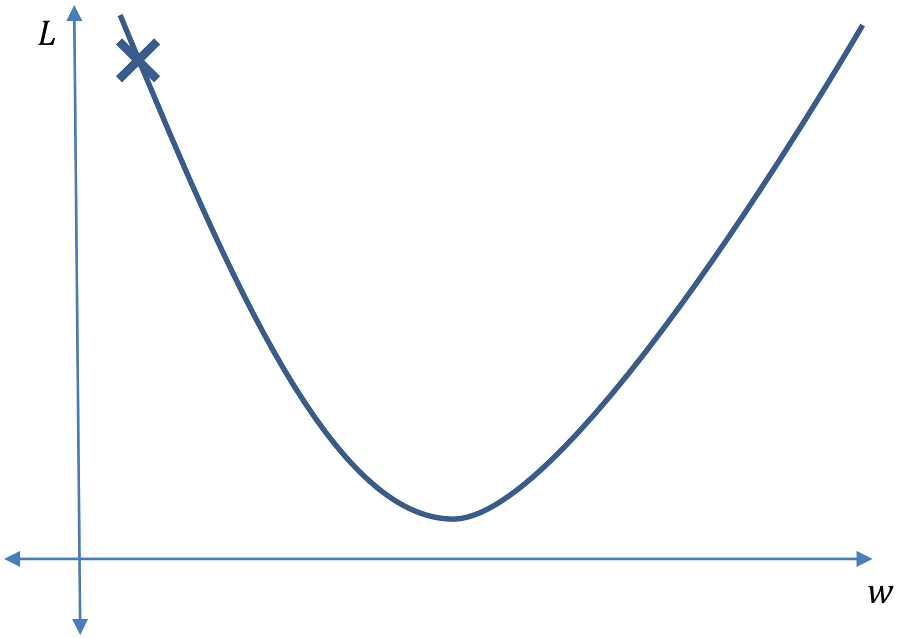
$$p_c = 1 \rightarrow L = -\log(1) = 0$$

The larger the loss, the worse our prediction.
We want to minimize L!

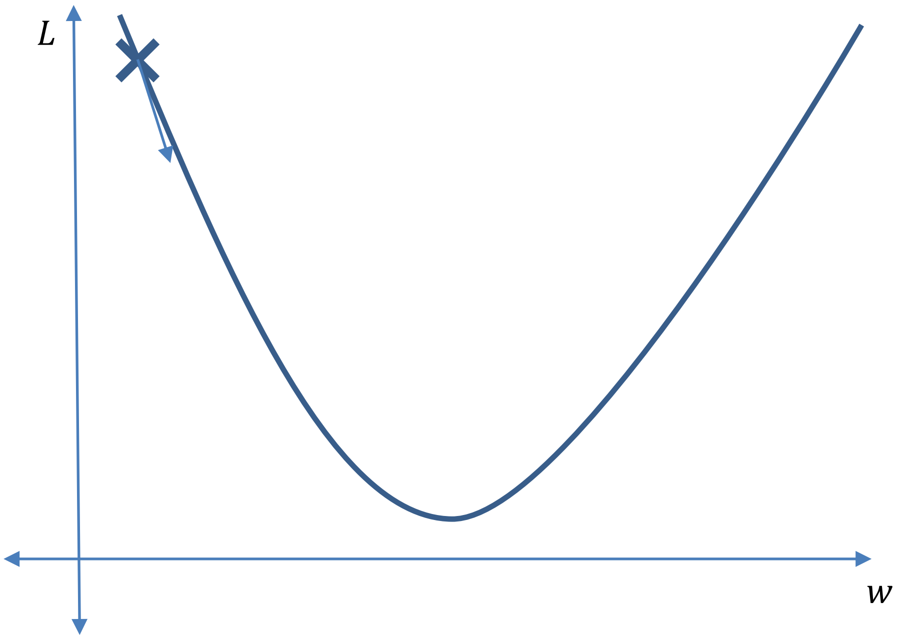
Minimizing Loss



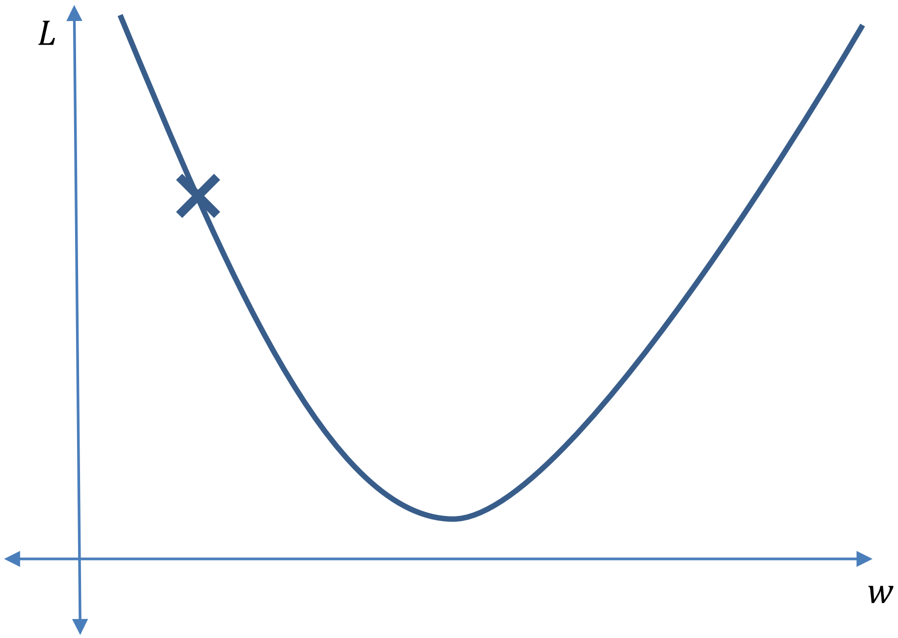
Minimizing Loss



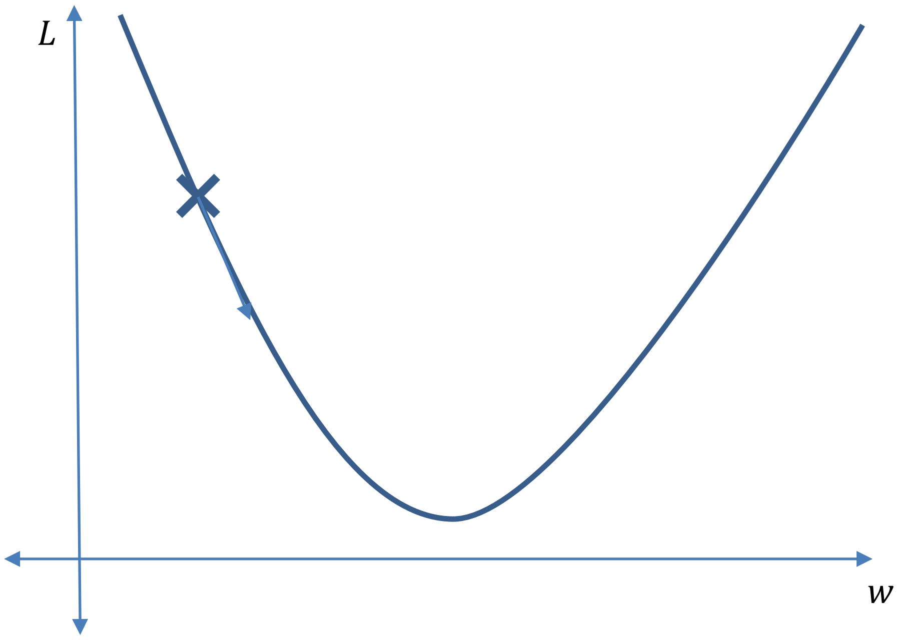
Minimizing Loss



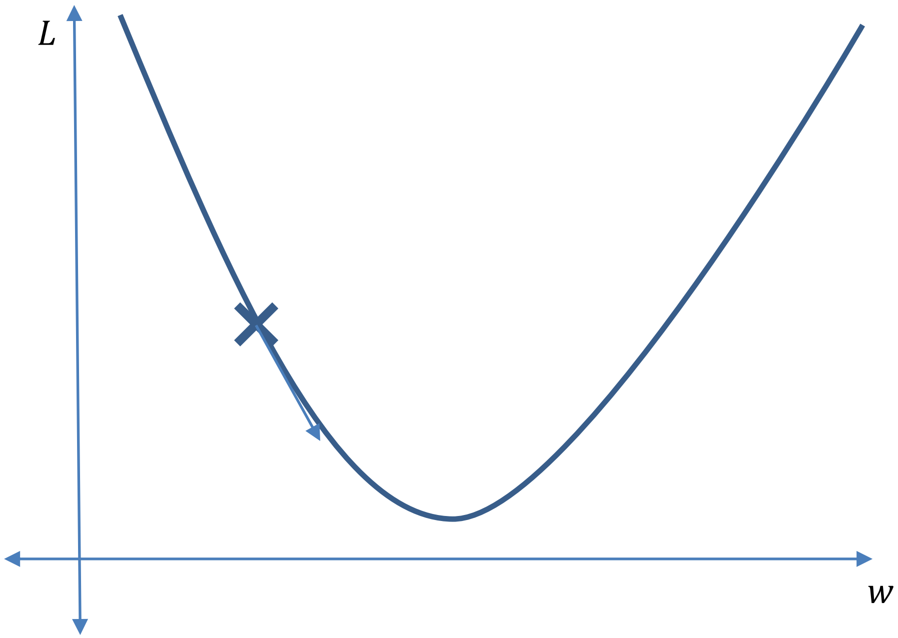
Minimizing Loss



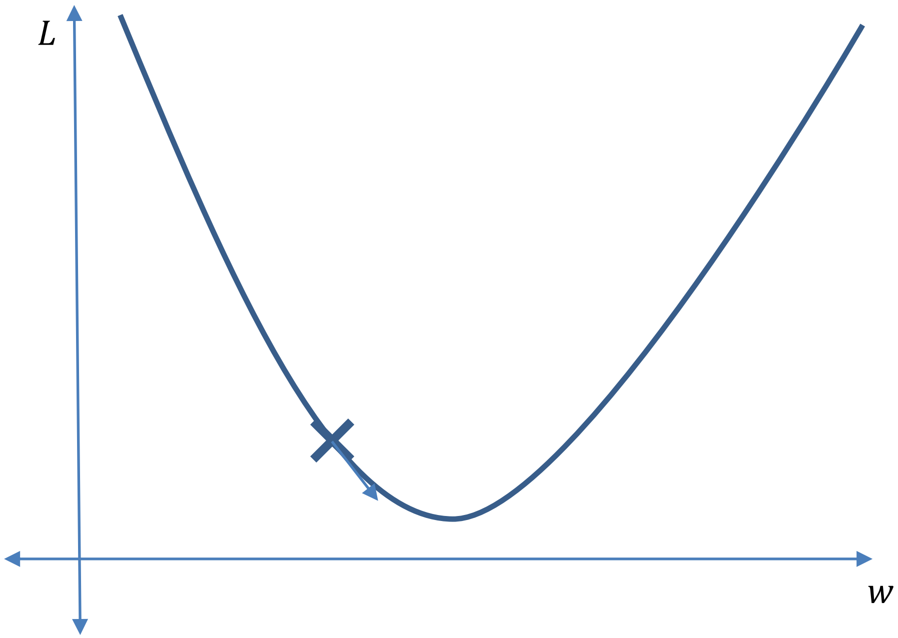
Minimizing Loss



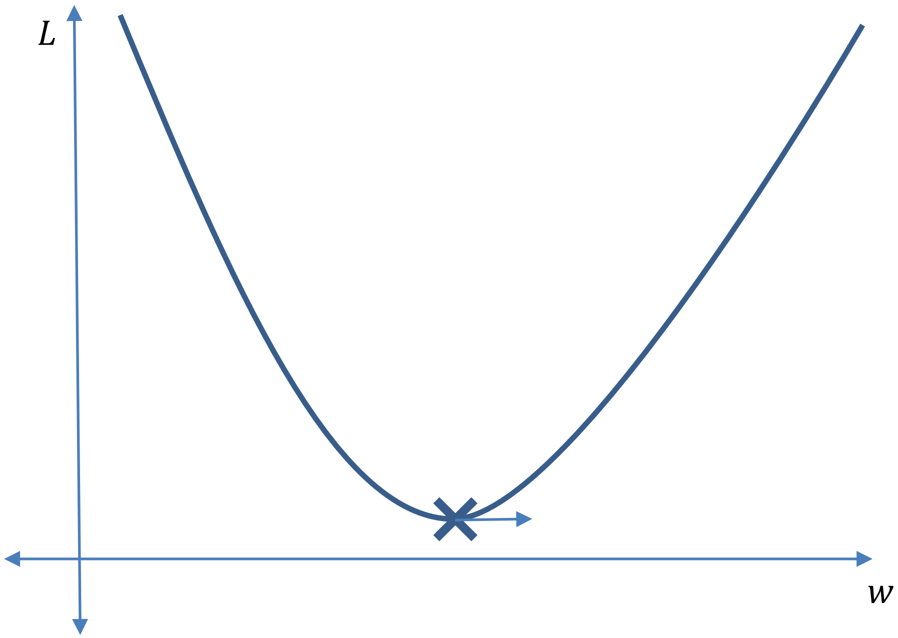
Minimizing Loss



Minimizing Loss



Minimizing Loss



Minimizing Loss

Our Classification System

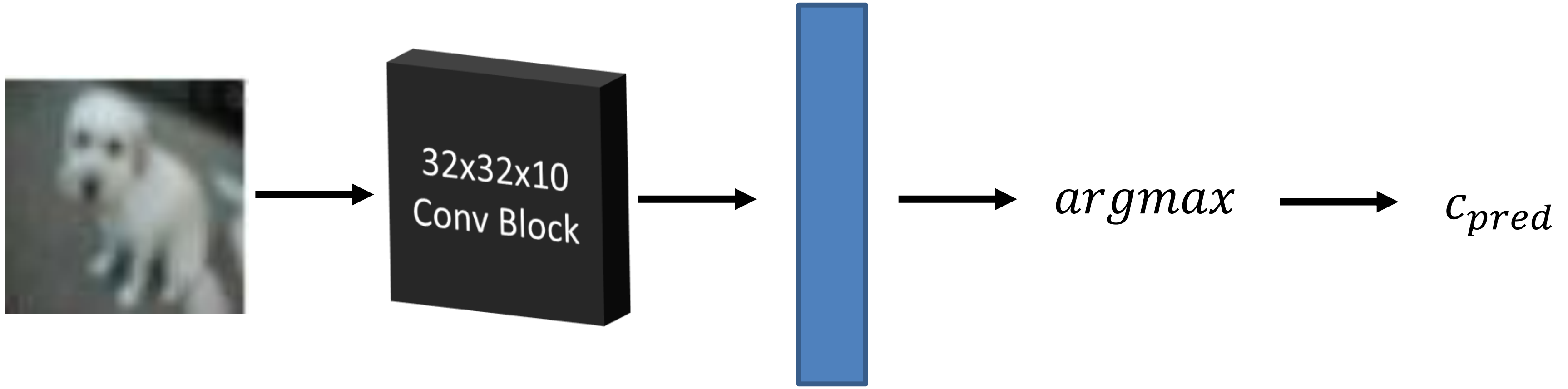
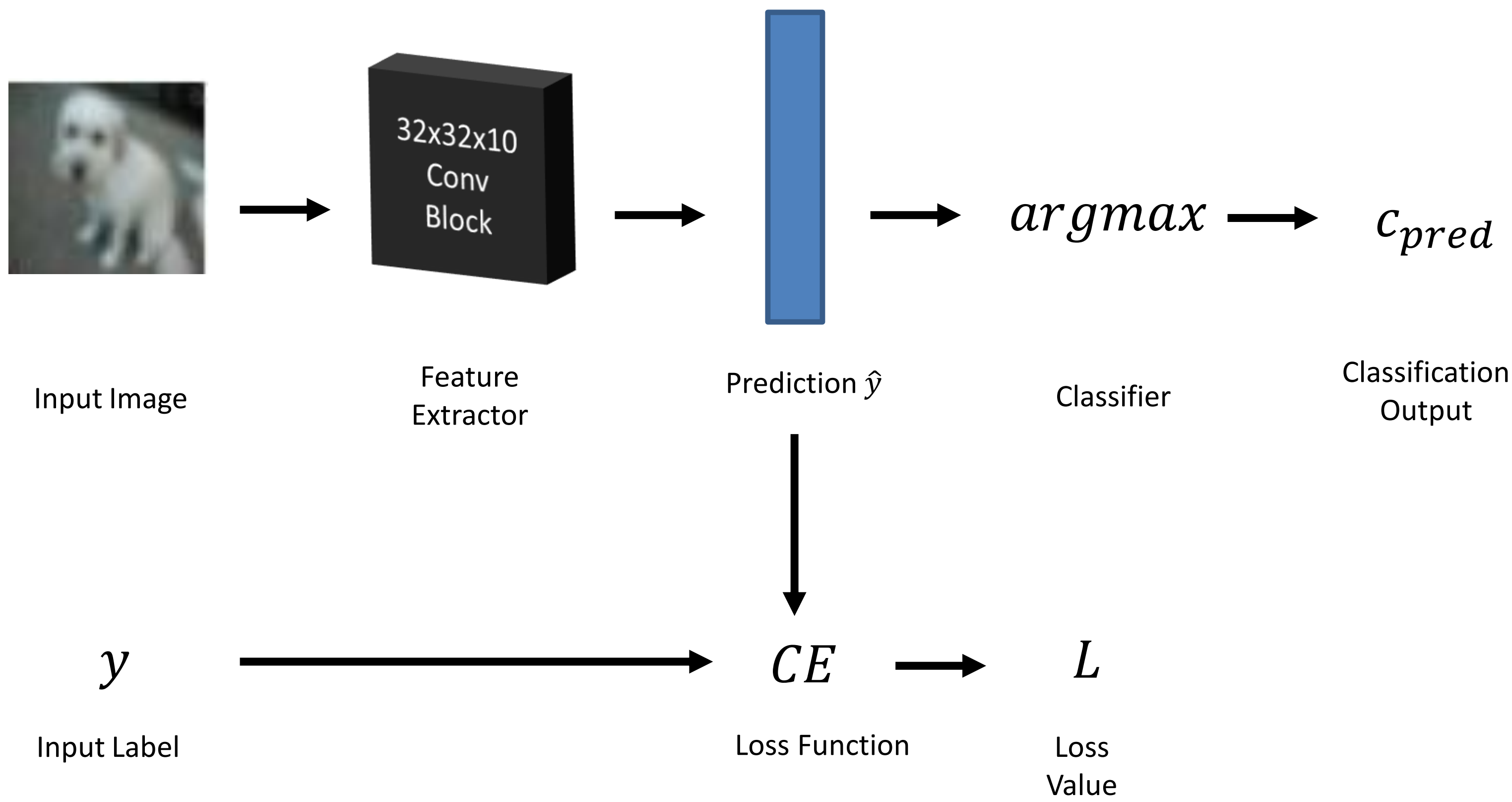
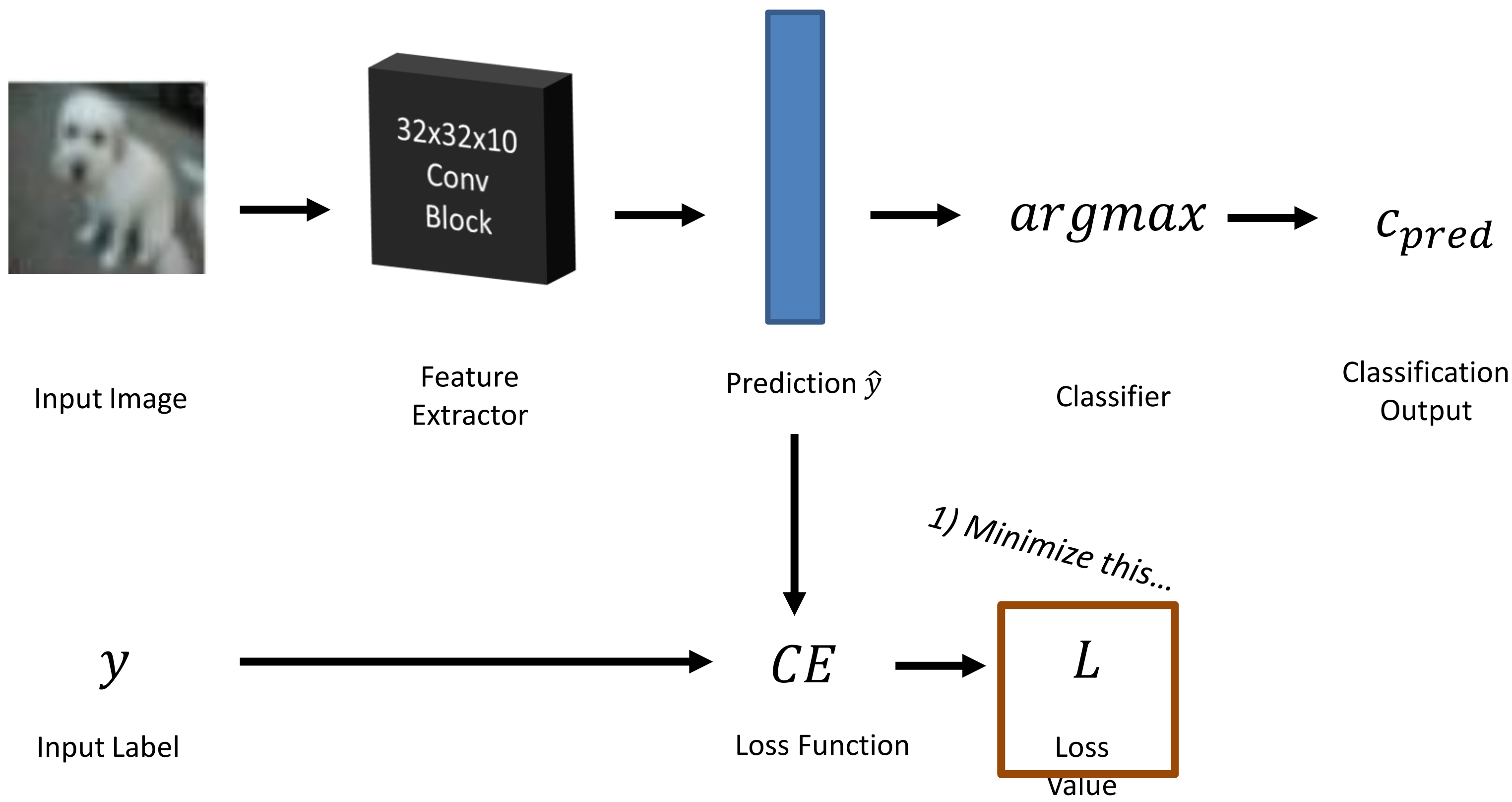


Image | Feature Extractor | Prediction $\hat{y}$ | Classifier | Classification Output

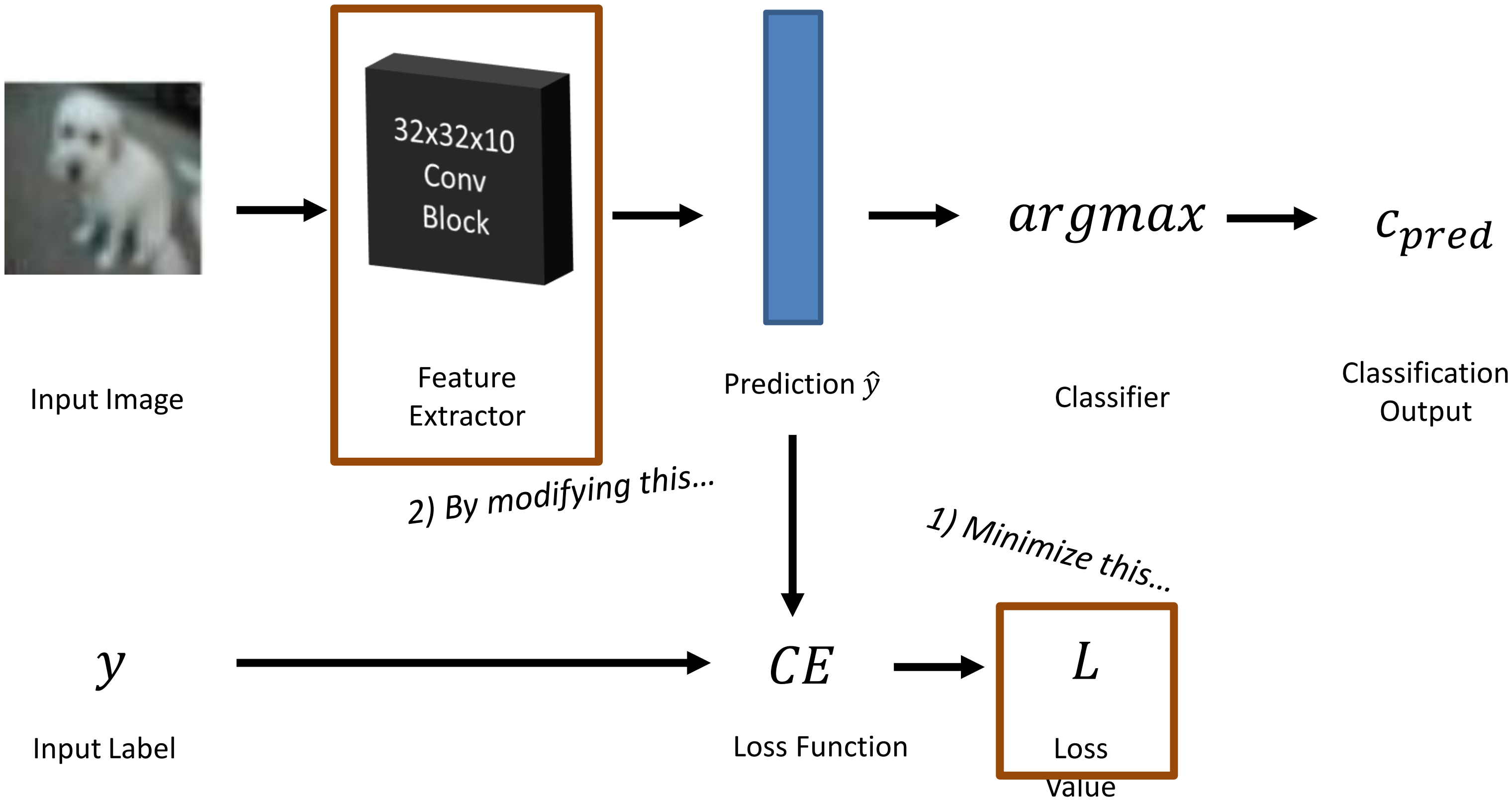
Our Classification System (modified)



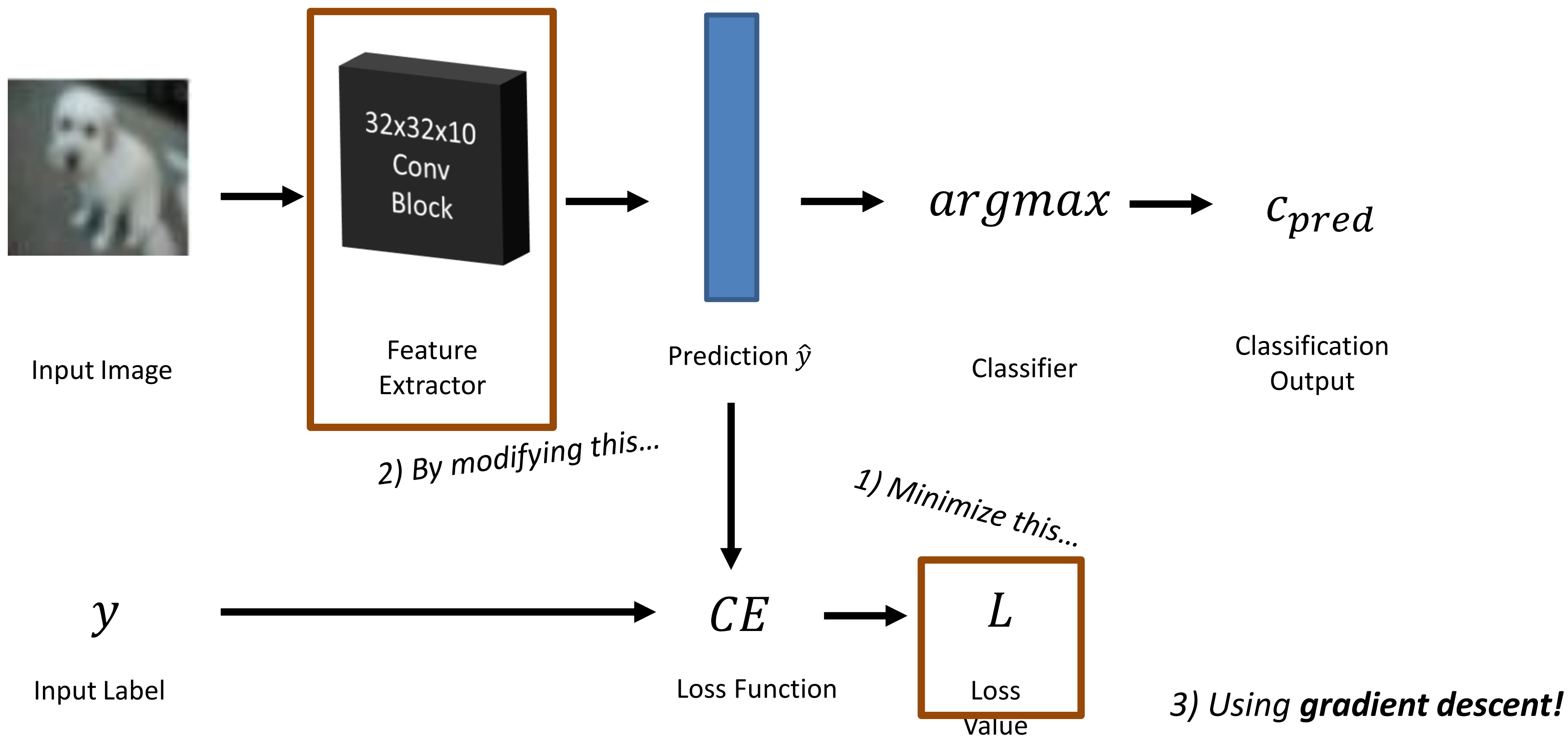
Our Classification System (modified)



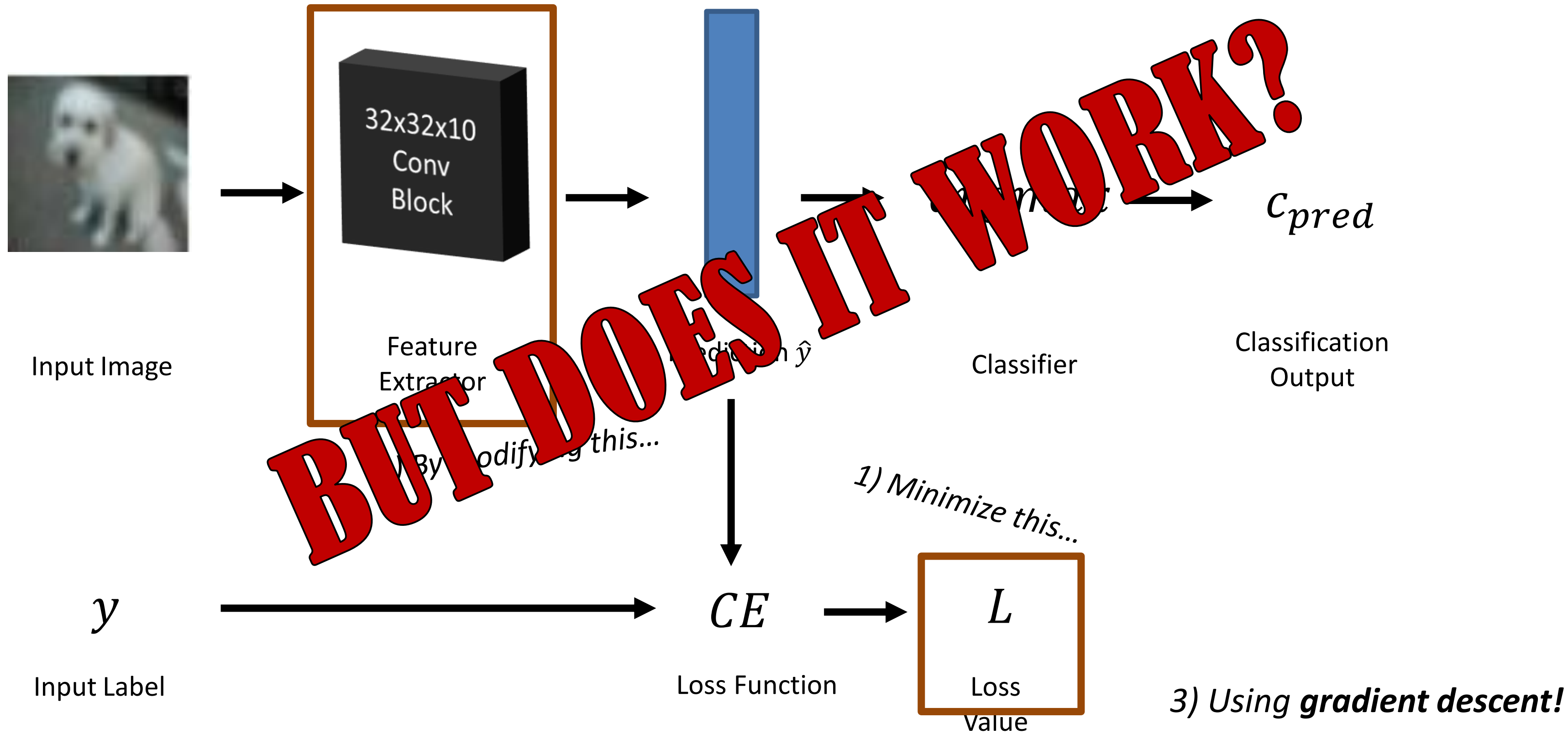
Our Classification System (modified)



Our Classification System (modified)



Our Classification System (modified)



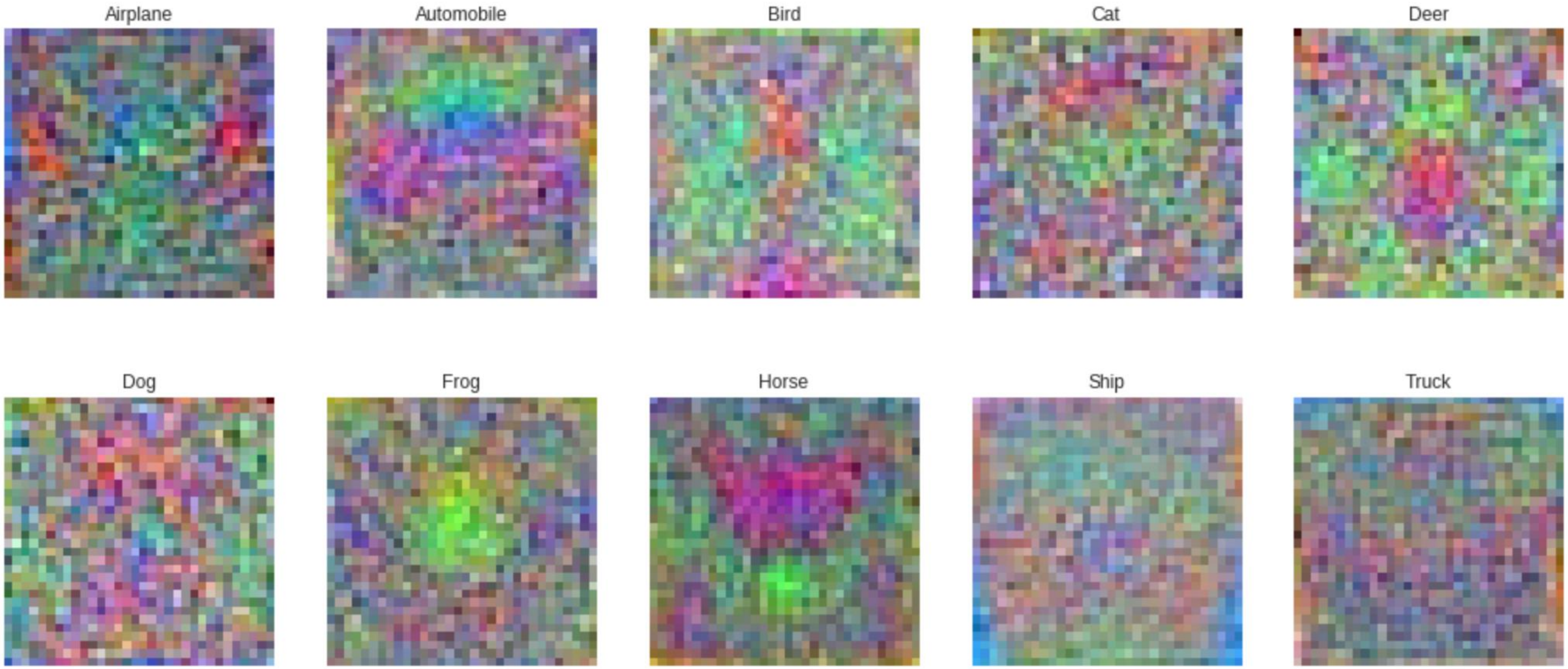
Our System’s Performance

- ~40% accuracy on CIFAR-10 test
  - Best class: Truck (~60%)
  - Worst class: Horse (~16%)

- Check out the model at: **https://tinyurl.com/cifar10**

- What about the filters? What do they look like?

Visualizing the Filters



Next Time...

Building a stronger convolution-based feature extractor

History of deep learning + computer vision (Convolutional Neural Nets!)

Applications of CNNs