Recognition

Learning

- Key idea: teach computer visual concepts by *providing examples*

$$\mathcal{X} \text{ :Images}$$

$$\mathcal{Y} \text{ :Labels}$$

$$\mathcal{D} \text{ :Distribution over } \mathcal{X} \times \mathcal{Y}$$

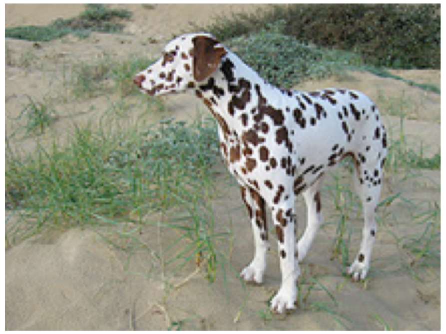
$$P(x, y) \qquad\qquad P(y|x)$$

Training Set

$$S = \{(x_i, y_i) \sim \mathcal{D}, i = 1, \ldots, n\}$$

Example

- Binary classifier "Dog" or "not Dog"
- Labels: {0, 1}
- Training set

{( , 1), ( , 1), ( , 0) , … }

Choosing a model class

- Will try and find P(y = 1 | x)
- P(y=0 | x) = 1 - P(y=1 | x)
- Need to find $h : \mathcal{X} \rightarrow [0, 1]$
- But: *enormous number of possible mappings*

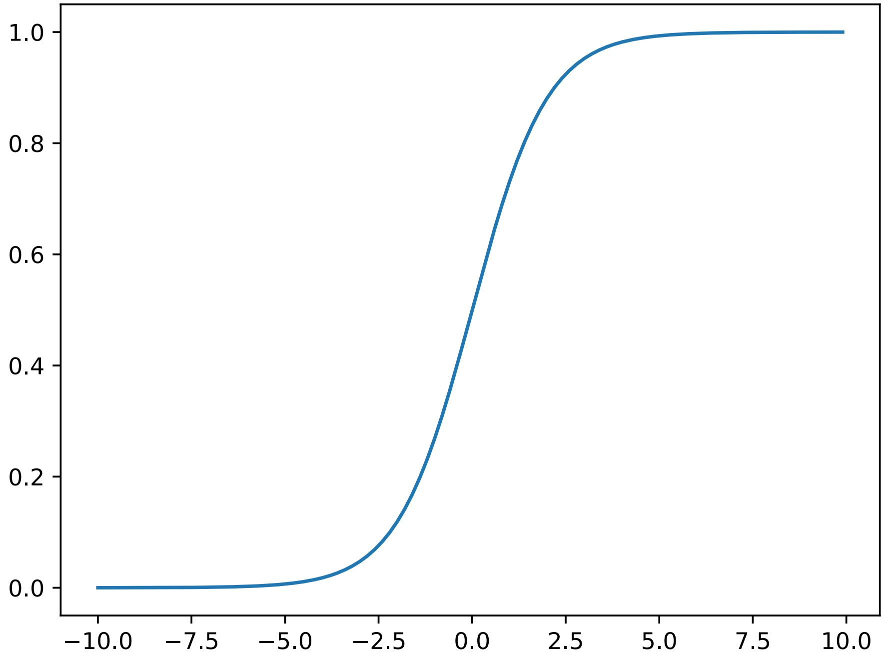
Choosing a model class

$$h : \mathcal{X} \to [0, 1]$$

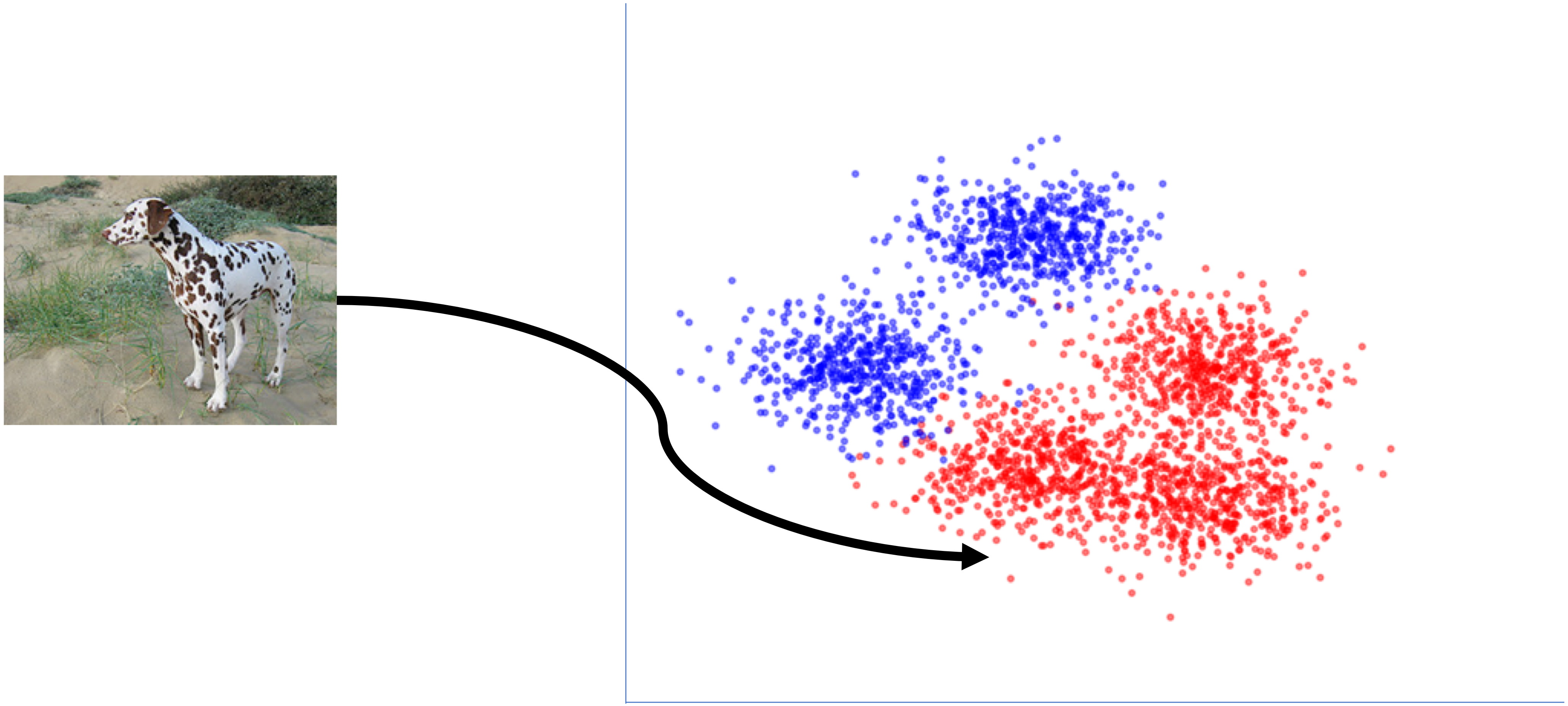
- E.g: Assume h is a linear classifier in feature space

$$h_{\mathbf{w}}(x) = \sigma(\mathbf{w}^T \phi(x))$$

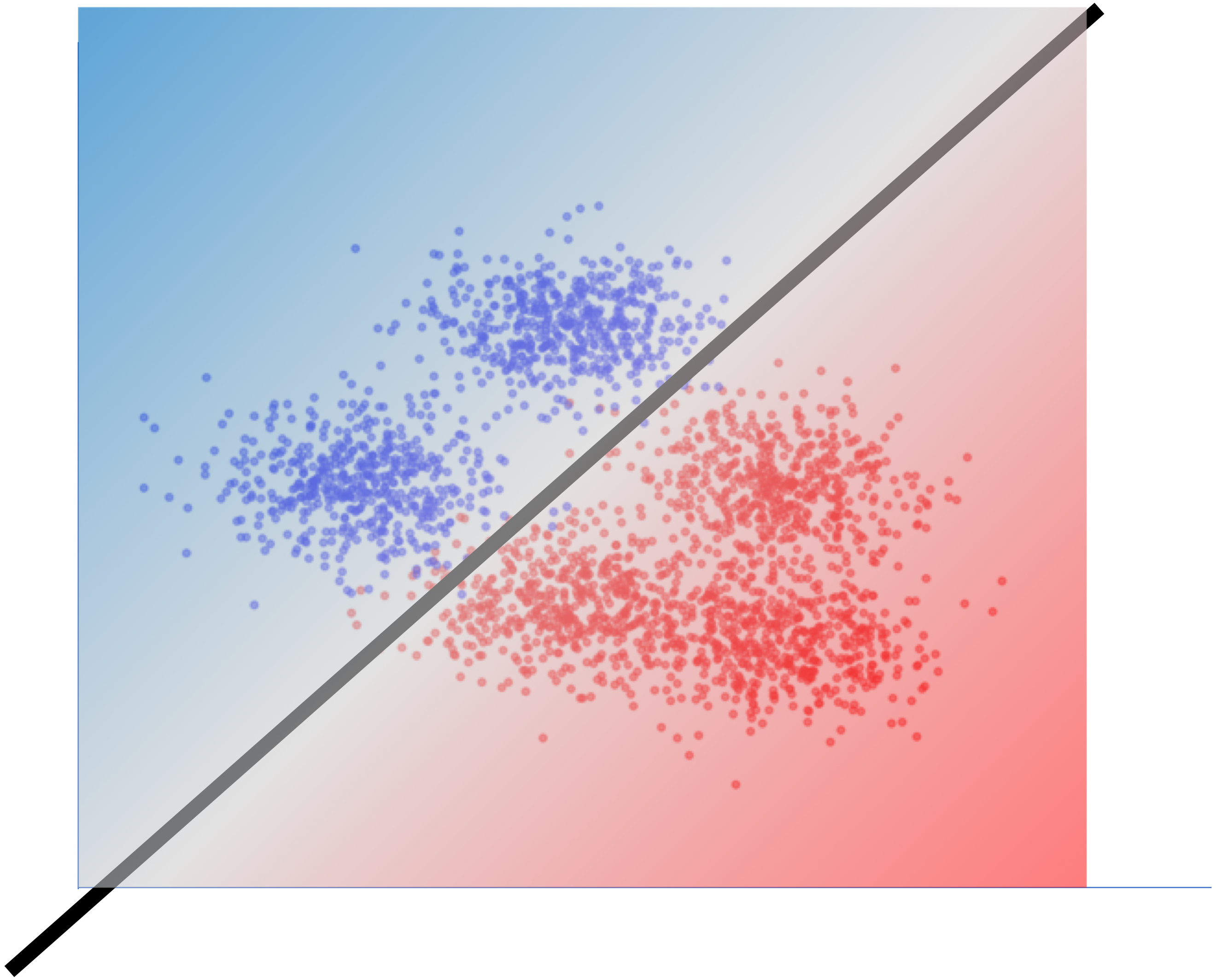
$$\sigma(s) = \frac{1}{1 + e^{-s}}$$



Feature extractor



Linear classifiers



Linear classifiers in feature space

$$h_{\mathbf{w}}(x) = \sigma(\mathbf{w}^T \phi(x))$$

- *Family* of functions
- Each function is called a *hypothesis*
- Family is called a *hypothesis class*
- Hypotheses indexed by **w**
- Need to find the best hypothesis = need to find best **w**

Training: Choosing the best hypothesis

- Use training set to find *best-fitting* hypothesis
- Question: how do we define fit?
- *Loss function*
    - $h_{\mathbf{w}}(x)$ is estimated probability label is 1
    - *Log likelihood : negative log of estimated probability of the true label*

Training: Choosing the best hypothesis

$$li(h_{\mathbf{w}}(x), y) = \begin{cases} h_{\mathbf{w}}(x) & \text{if } y = 1 \\ 1 - h_{\mathbf{w}}(x) & \text{ow} \end{cases}$$

$$li(h_{\mathbf{w}}(x), y) = h_{\mathbf{w}}(x)^y (1 - h_{\mathbf{w}}(x))^{1-y}$$

Training: Choosing the best hypothesis

$$li(h_{\mathbf{w}}(x), y) = h_{\mathbf{w}}(x)^y (1 - h_{\mathbf{w}}(x))^{1-y}$$

- Likelihood of a single data point
- Fit = *total likelihood of entire training dataset*

$$S = \{(x_i, y_i) \sim \mathcal{D}, i = 1, \ldots, n\}$$

$$\prod_{i=1}^{n} h_{\mathbf{w}}(x_i)_i^y (1 - h_{\mathbf{w}}(x_i))^{1-y_i}$$

Training: Choosing the best hypothesis

- Use negative log likelihood

$$-\sum_{i=1}^{n} y_i \log h_{\mathbf{w}}(x_i) + (1 - y_i) \log(1 - h_{\mathbf{w}}(x_i))$$

Training: Choosing the best hypothesis

- Maximizing log likelihood = *Minimizing negative log likelihood*

$$\max_{\mathbf{w}} \sum_{i=1}^{n} y_i \log h_{\mathbf{w}}(x_i) + (1 - y_i) \log(1 - h_{\mathbf{w}}(x_i))$$

$$\min_{\mathbf{w}} (-\sum_{i=1}^{n} y_i \log h_{\mathbf{w}}(x_i) + (1 - y_i) \log(1 - h_{\mathbf{w}}(x_i)))$$

Training = Optimization

- Need to minimize an objective
- Simple solution: *gradient descent*

$$\min_{\mathbf{w}} f(\mathbf{w})$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla_{\mathbf{w}} f(\mathbf{w}^{(t)})$$

Stochastic gradient descent

$$f(\mathbf{w}) = \frac{1}{n}\sum_i L(h_{\mathbf{w}}(x_i), y_i)$$

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{n}\sum_i \nabla_{\mathbf{w}} L(h_{\mathbf{w}}(x_i), y_i)$$

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = < \nabla_{\mathbf{w}} L(h_{\mathbf{w}}(x_i), y_i) >$$

$$g_i(\mathbf{w}) = L(h_{\mathbf{w}}(x_i), y_i)$$

Stochastic gradient descent

- Randomly sample small subset of examples
- Compute gradient on small subset
  - *Unbiased estimate of true gradient*
- Take step along estimated gradient

$$h_{\mathbf{w}}(x) = \sigma(\mathbf{w}^T \phi(x)) \qquad \sigma(s) = \frac{1}{1+e^{-s}}$$

$$L(h_{\mathbf{w}}(x), y) = (-y \log h_{\mathbf{w}}(x) + (1-y)\log(1-h_{\mathbf{w}}(x)))$$

$$\min_{\mathbf{w}} \sum_{i=1}^{n} L(h_{\mathbf{w}}(x_i), y_i)$$

Logistic Regression!

General recipe

- Fix hypothesis class

$$h_{\mathbf{w}}(x) = \sigma(\mathbf{w}^T \phi(x))$$

- Define loss function

$$L(h_{\mathbf{w}}(x), y) = (-y \log h_{\mathbf{w}}(x) + (1 - y) \log(1 - h_{\mathbf{w}}(x)))$$

- Minimize total loss on the training set

$$\min_{\mathbf{w}} \sum_{i=1}^{n} L(h_{\mathbf{w}}(x_i), y_i)$$

- *Why should this work?*

Risk

- Given:
  - Distribution $\mathcal{D}$
  - A hypothesis $h \in H$
  - Loss function L
- We are interested in Expected Risk:

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y)$$

- Given training set S, and a particular hypothesis h, Empirical Risk:

$$\hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

Risk

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y) \qquad \hat{R}(S,h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

- By central limit theorem,

$$\mathbb{E}_{S\sim\mathcal{D}^n} \hat{R}(S,h) = R(h)$$

- Variance proportional to 1/n
- For randomly chosen h, empirical risk is an *unbiased estimator* of expected risk

Risk

- Empirical risk unbiased estimate of expected risk
- Want to minimize expected risk
- Idea: Minimize *empirical risk* instead
- This is the **Empirical Risk Minimization Principle**

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y) \qquad \hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

$$\boxed{h^* = \arg\min_{h\in H} \hat{R}(S, h)}$$

Generalization

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y) \qquad \hat{R}(S,h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

$$R(h) = \underbrace{\hat{R}(S,h)}_{\text{Training error}} + \underbrace{(R(h) - \hat{R}(S,h))}_{\text{Generalization error}}$$

Overfitting

- We are minimizing training error
- Empirical risk of chosen hypothesis *no longer* unbiased estimate:
  - We chose hypothesis based on S
  - Might have chosen h for which S is a special case
- Overfitting:
  - Minimize training error, but generalization error *increases*

Controlling generalization error

- Variance of empirical risk inversely proportional to size of S
  - Choose very large S!

- *Larger* the hypothesis class H, *Higher* the chance of hitting bad hypotheses with low training error and high generalization error
  - Choose small H!

- For many models, can *bound* generalization error using some property of parameters
  - Regularize during optimization!
  - Eg. L2 regularization

Controlling generalization error

- How do we know we are overfitting?
  - Use a *held-out* “validation set”
  - To be an unbiased sample, must be completely *unseen*

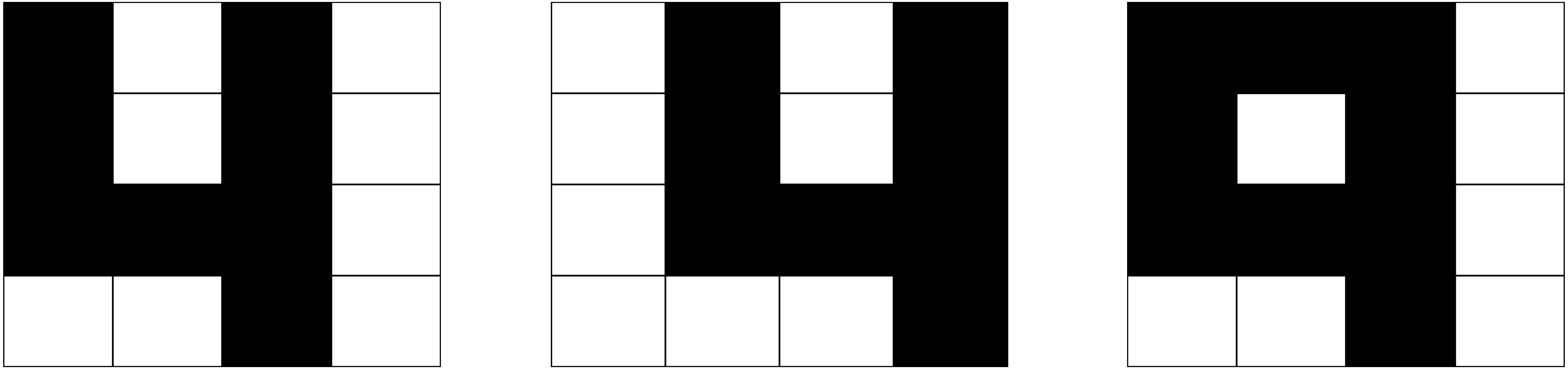
Putting it all together

- Collect training set and validation set
- Pick hypothesis class
- Pick loss function
- Minimize empirical risk (+ regularization)
- Measure performance on held-out validation set
- Profit!

Loss functions and hypothesis classes

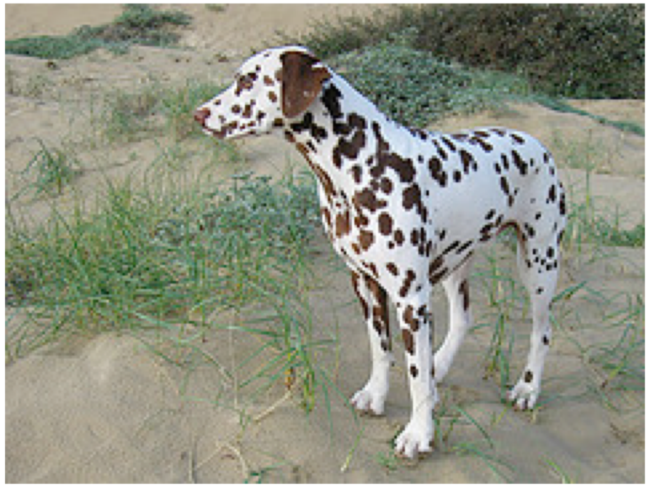
Loss function	Problem	Range of h	$\mathcal{Y}$	Formula
Log loss	Binary Classification	$\mathbb{R}$	$\{0,1\}$	$\log(1+e^{-yh(x)})$
Negative log likelihood	Multiclass classification	$[0,1]^k$	$\{1,\ldots,k\}$	$-\log h_y(x)$
Hinge loss	Binary Classification	$\mathbb{R}$	$\{0,1\}$	$\max(0,1-yh(x))$
MSE	Regression	$\mathbb{R}$	$\mathbb{R}$	$(y-h(x))^2$

Linear classifiers on pixels are bad



- Better feature vectors
- Non-linear classifiers

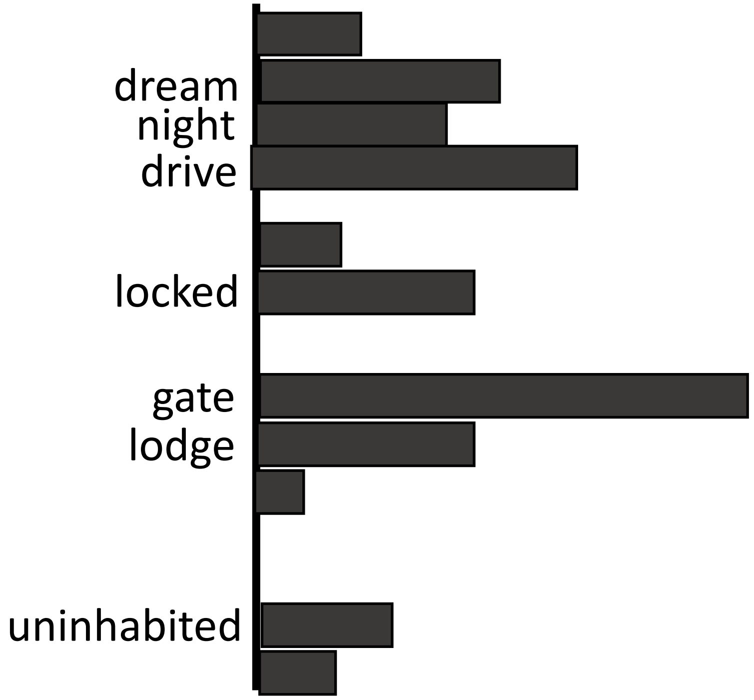
Invariance to large deformations



Important aspect of a class: presence / absence of parts?

Bags of words

Last night I dreamt I went to Manderley again. It seemed to me I stood by the iron gate leading to the drive, and for a while I could not enter, for the way was barred to me. There was a padlock and a chain upon the gate. I called in my dream to the lodge-keeper, and had no answer, and peering closer through the rusted spokes of the gate I saw that the lodge was uninhabited….



Bags of visual words

What should be visual words?

- A visual word is a cluster of image patches that mean the same thing



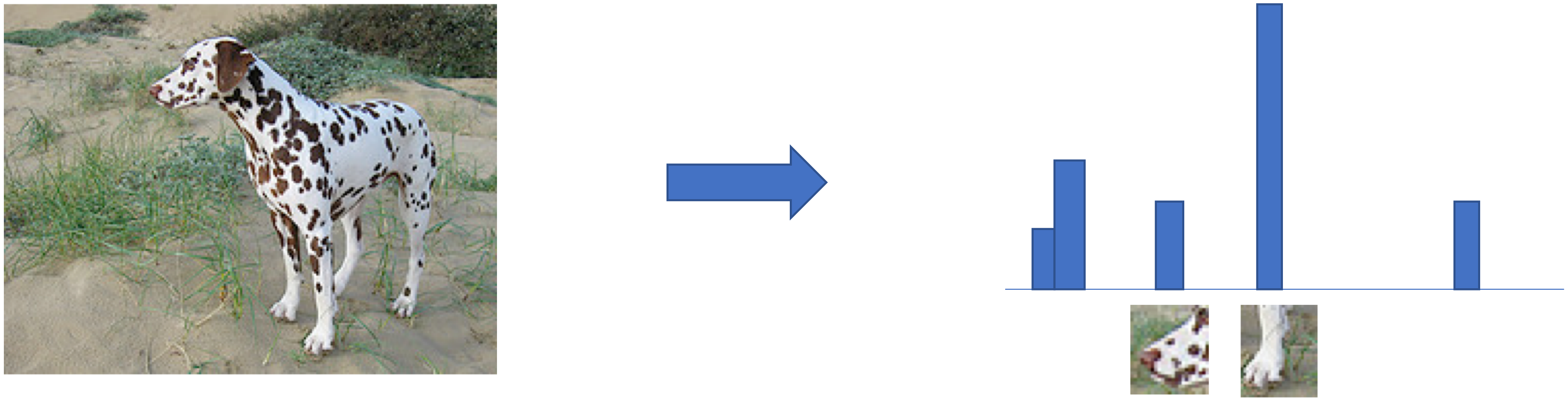

  - Object parts
  - Texture patterns
- Idea: collect patches from many different images
- Cluster using k-means - cluster centers are words

Detecting visual words

Nearest neighbor

Encoding images as bag of words

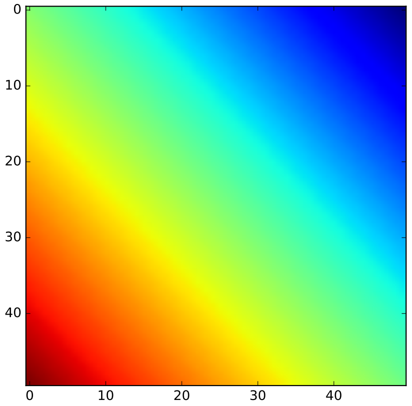
- Densely extract image patches from image
- Assign each patch to a visual word
- Compute histogram of occurrence

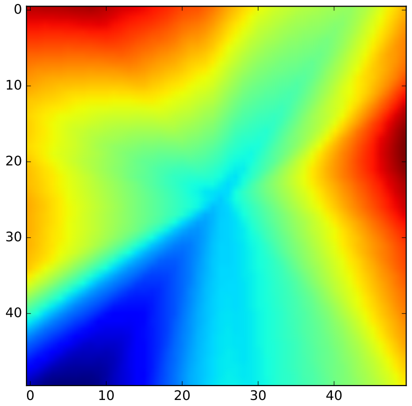


Multilayer perceptrons

- Key idea: build complex functions by composing simple functions

f(x) = Wx → g(x) = max(x,0) → f(x) = Wx → g(x) = max(x,0) → f(x) = Wx

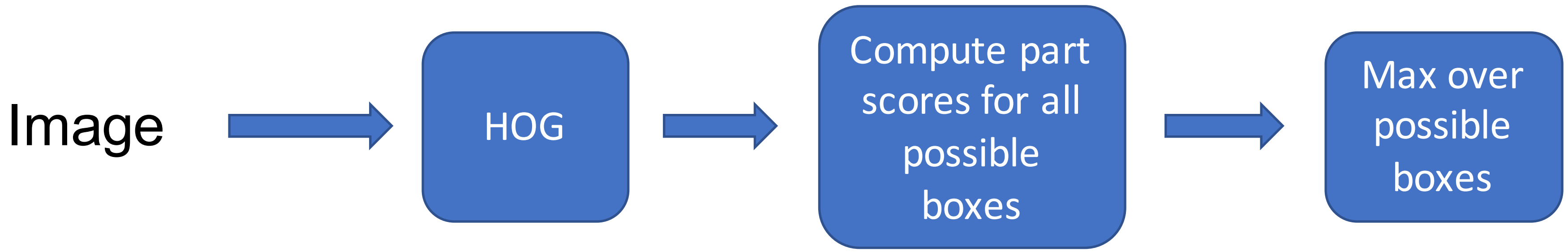




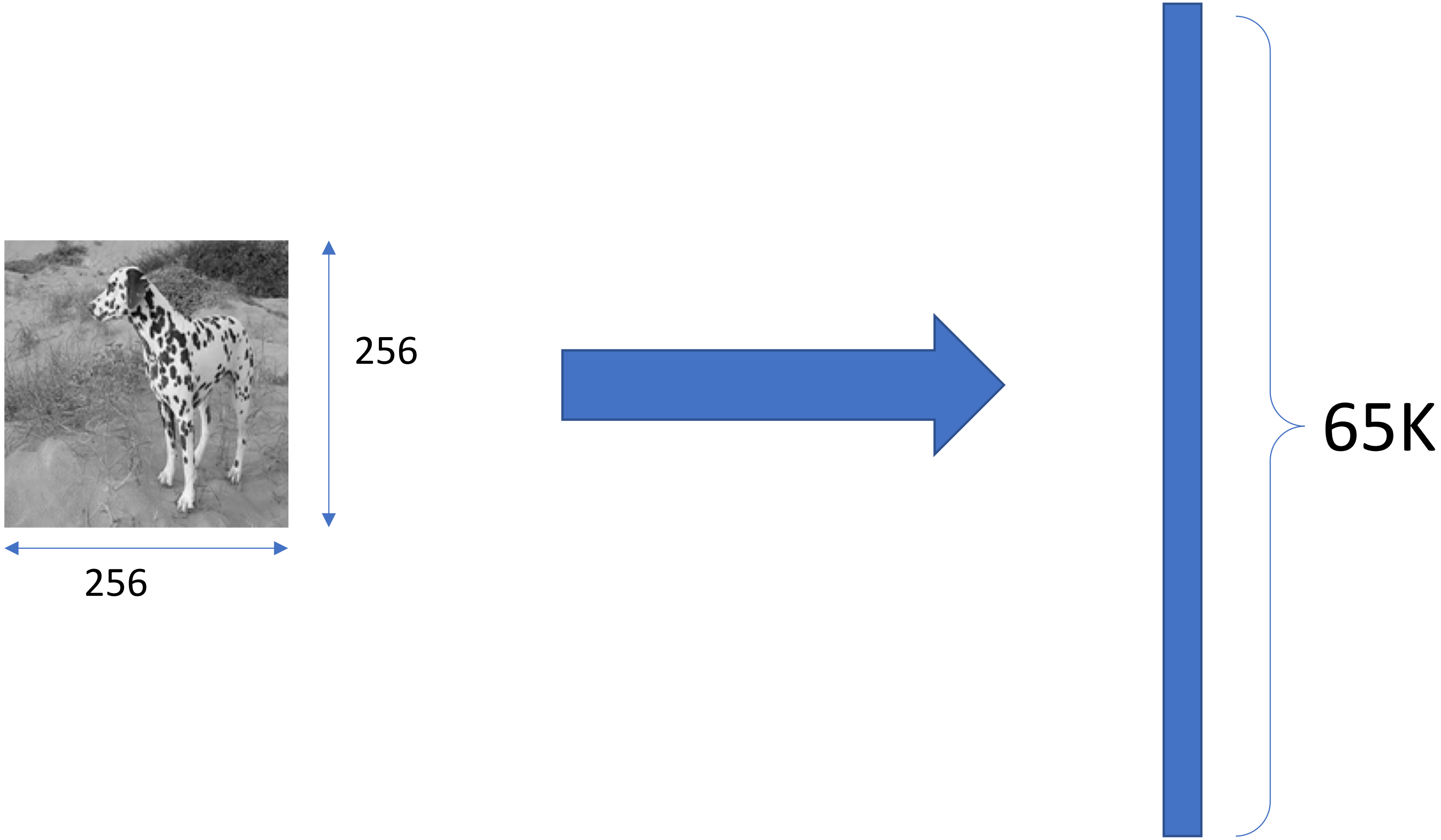
Multilayer perceptrons

- Key idea: build complex functions by composing simple functions
- Caveat: simple functions must include non-linearities
- $W(U(Vx)) = (WUV)x$

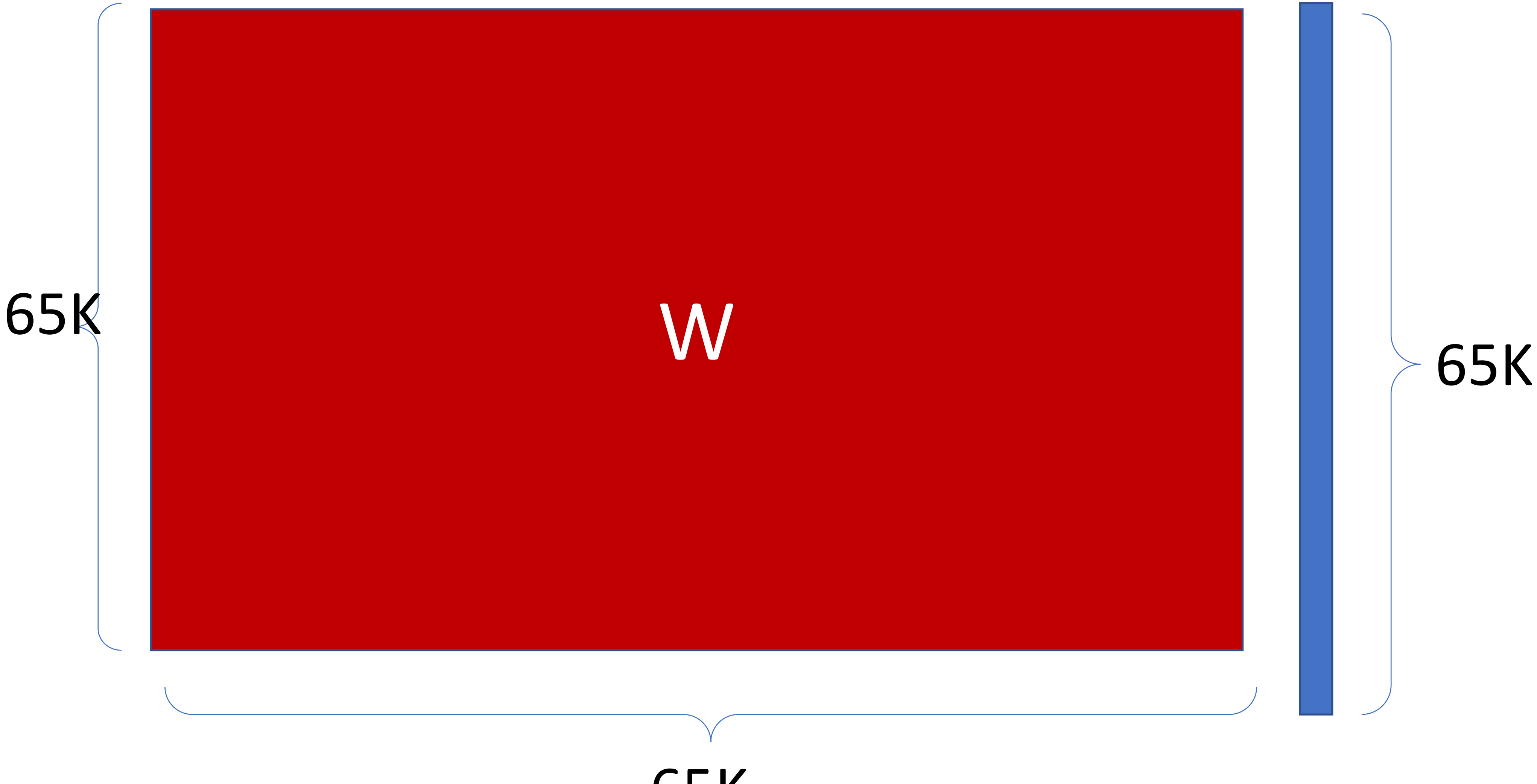
Multilayer perceptrons



Reducing capacity

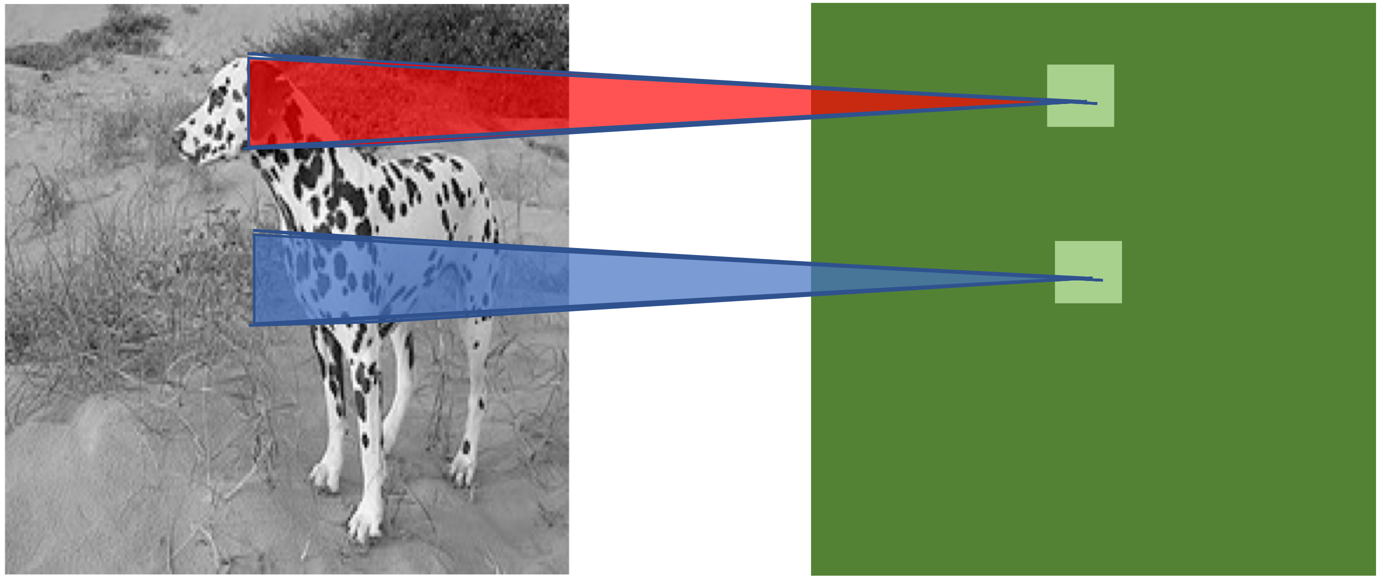


Reducing capacity



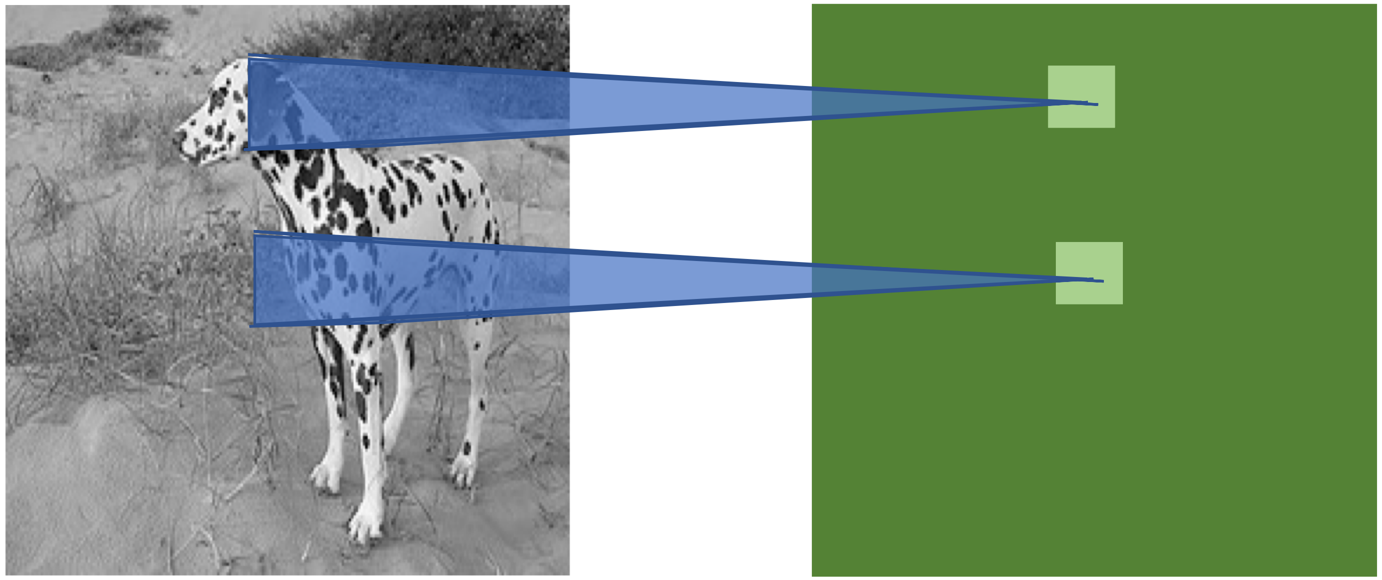
Idea 1: local connectivity

- Pixels only related to nearby pixels



Idea 2: Translation invariance

- Pixels only related to nearby pixels

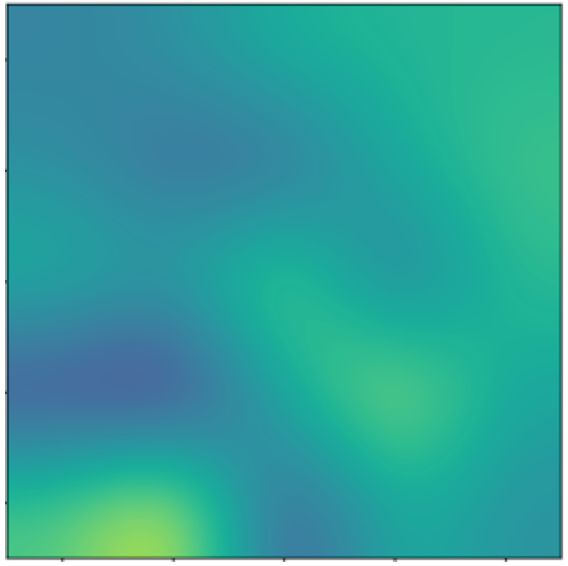


Local connectivity + translation invariance = *convolution*

5.4	0.1	3.6
1.8	2.3	4.5
1.1	3.4	7.2

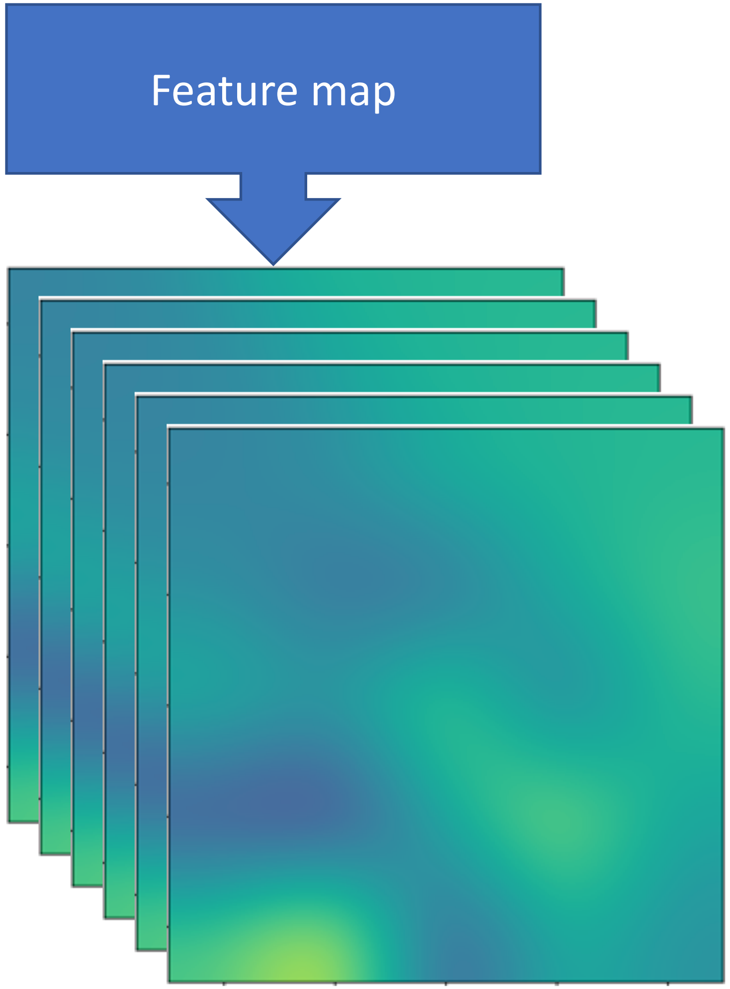
Local connectivity + translation invariance = *convolution*

5.4	0.1	3.6
1.8	2.3	4.5
1.1	3.4	7.2

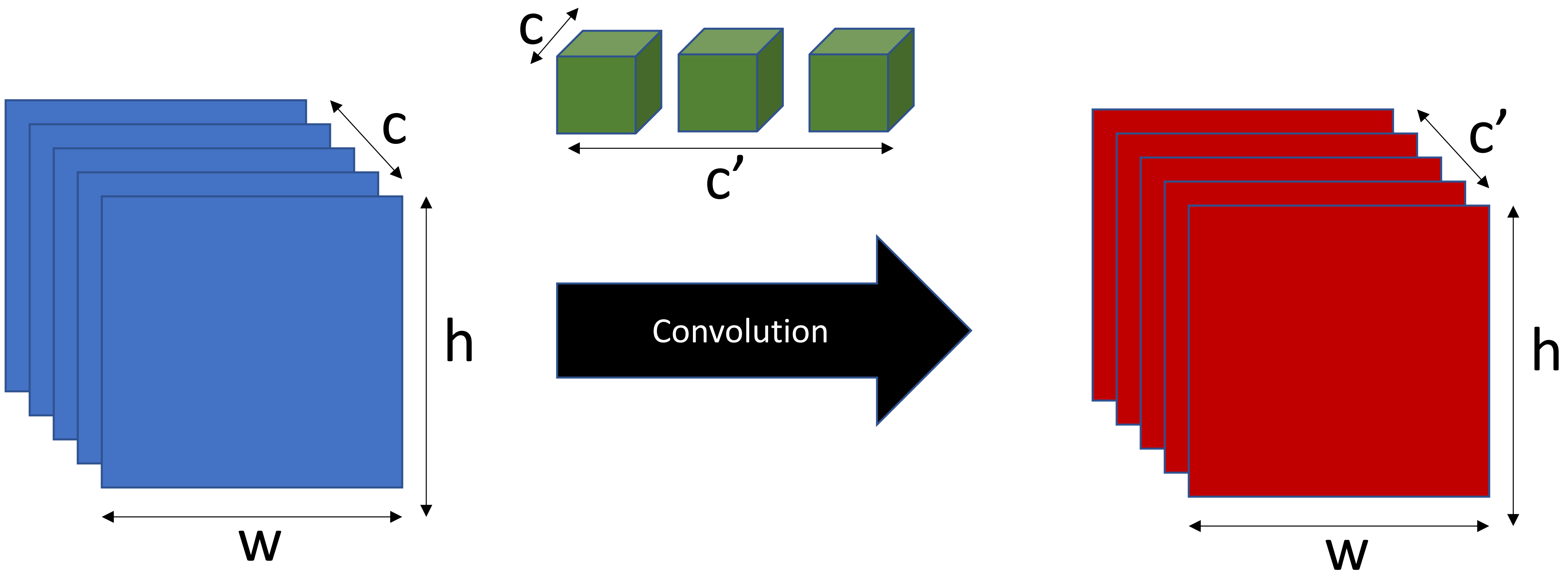


Local connectivity + translation invariance = *convolution*

5.4	0.1	3.6
1.8	2.3	4.5
1.1	3.4	7.2

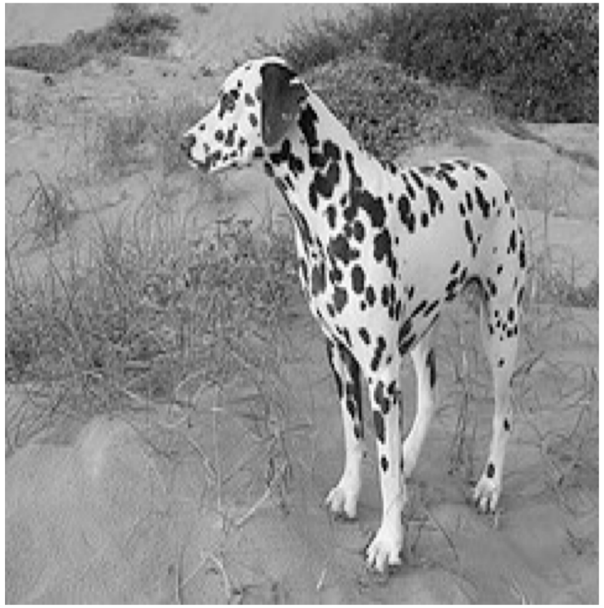


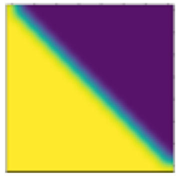
Convolution as a primitive

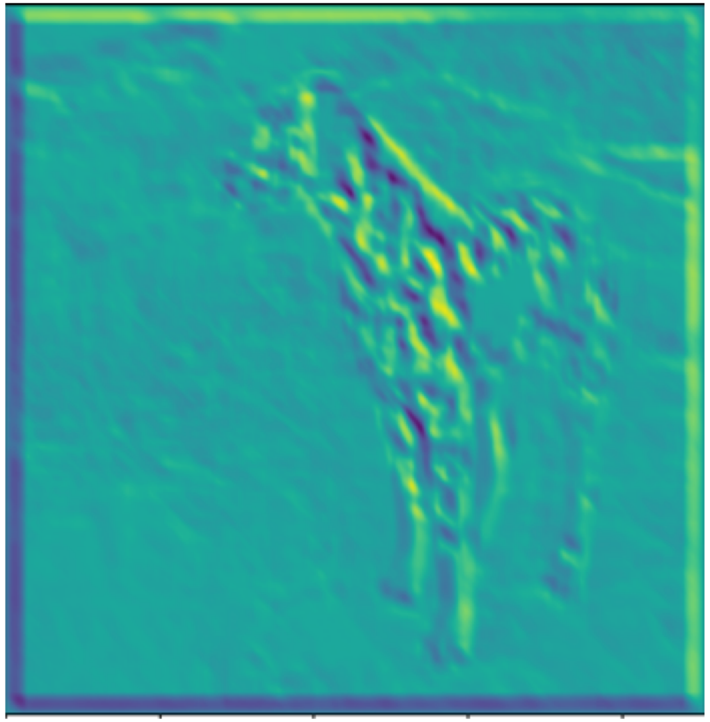


Convolution as a feature detector

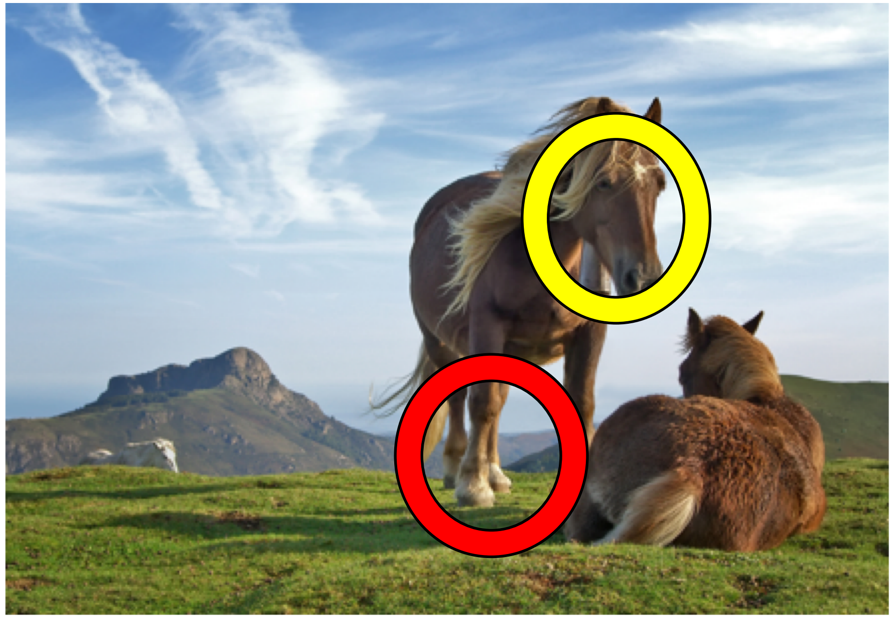
- score at (x,y) = dot product (filter, image patch at (x,y))
- Response represents similarity between filter and image patch



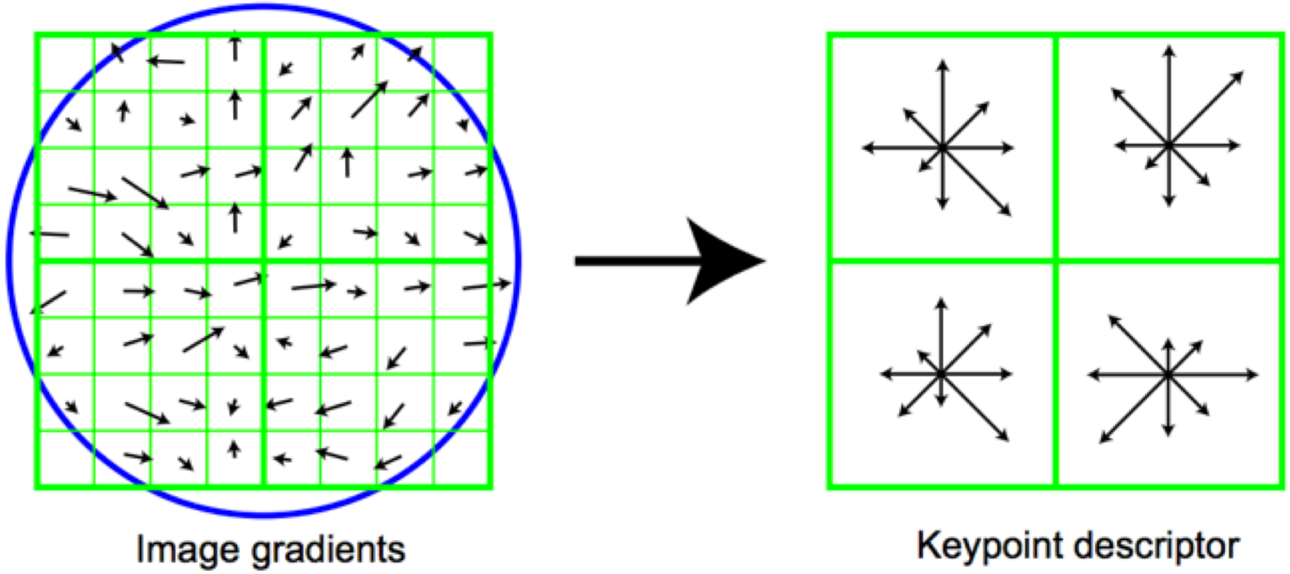




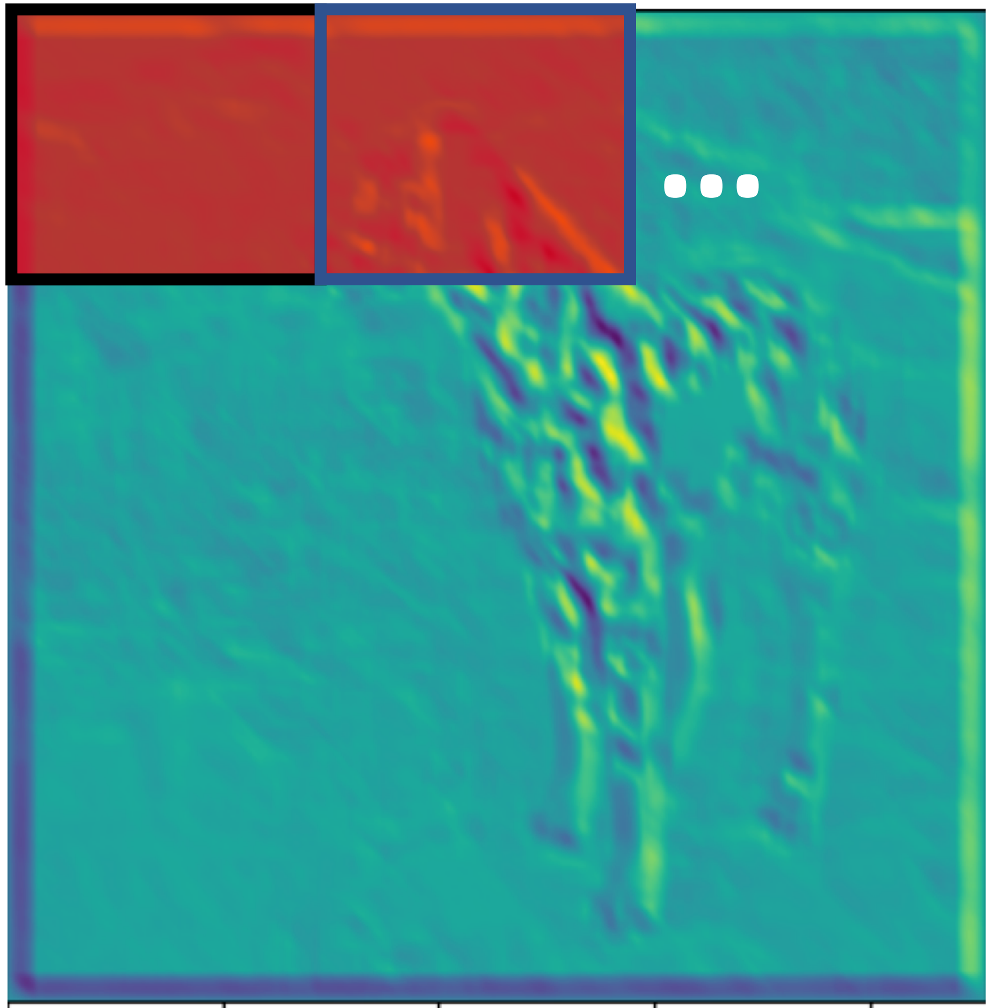
Invariance to distortions



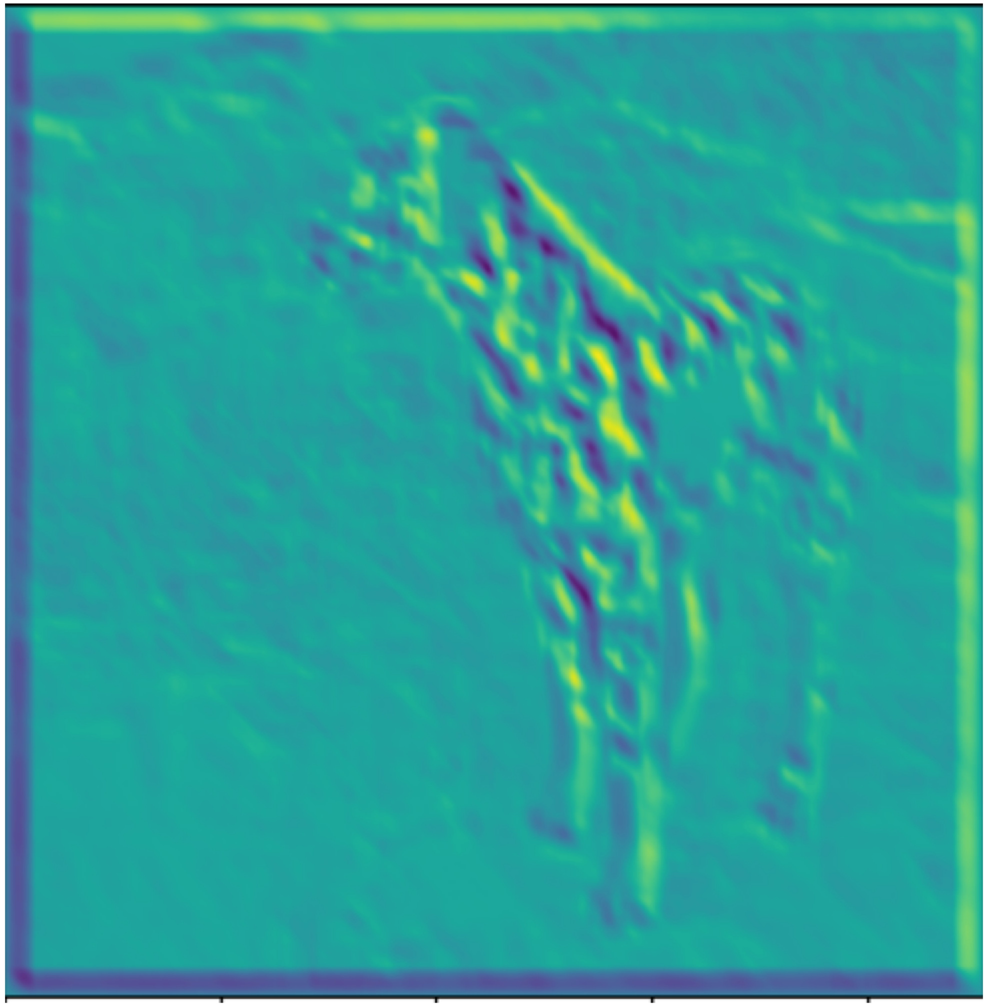
Invariance to distortions

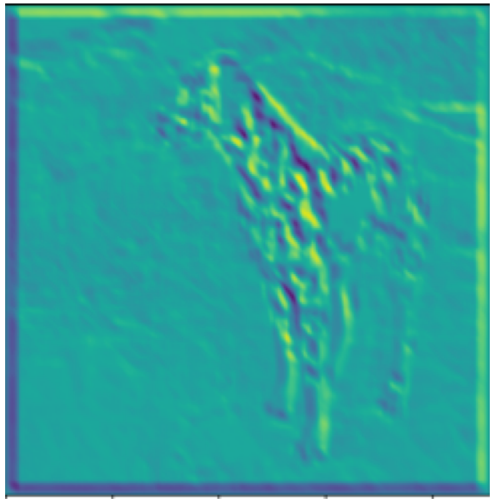


Invariance to distortions: Pooling

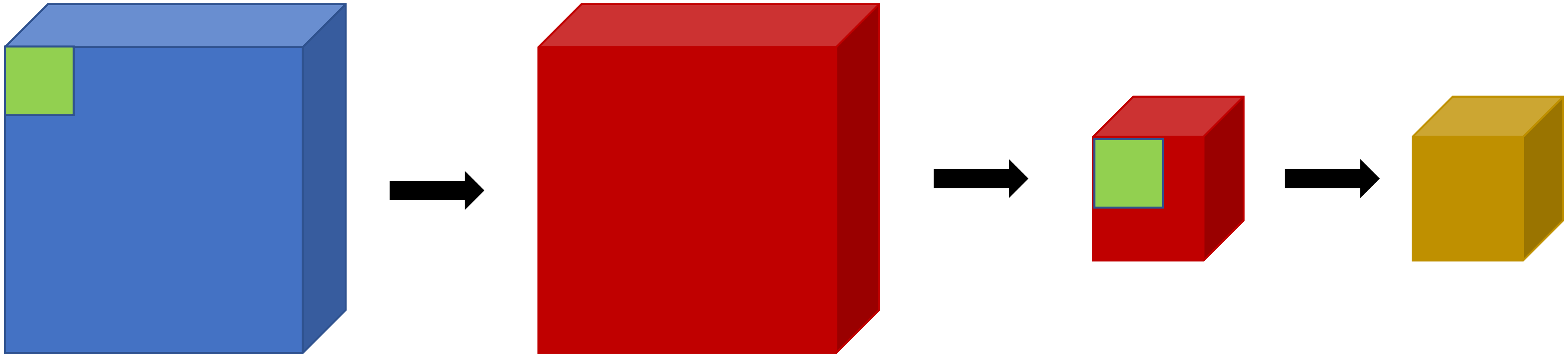


Invariance to distortions: Subsampling





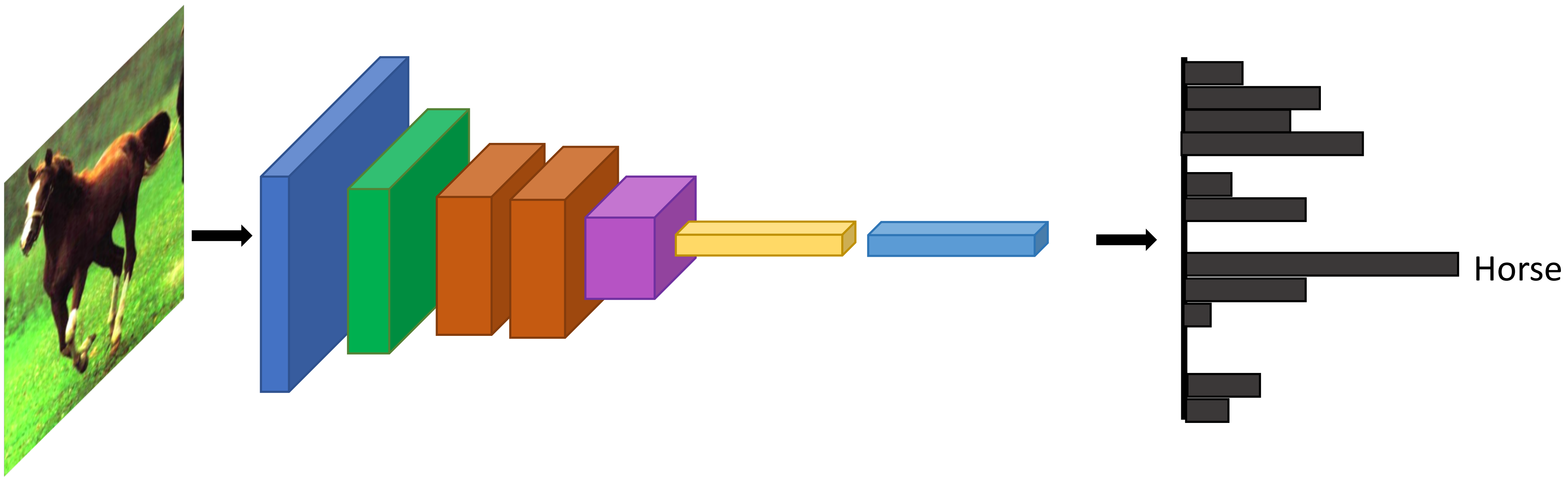
Convolution subsampling convolution



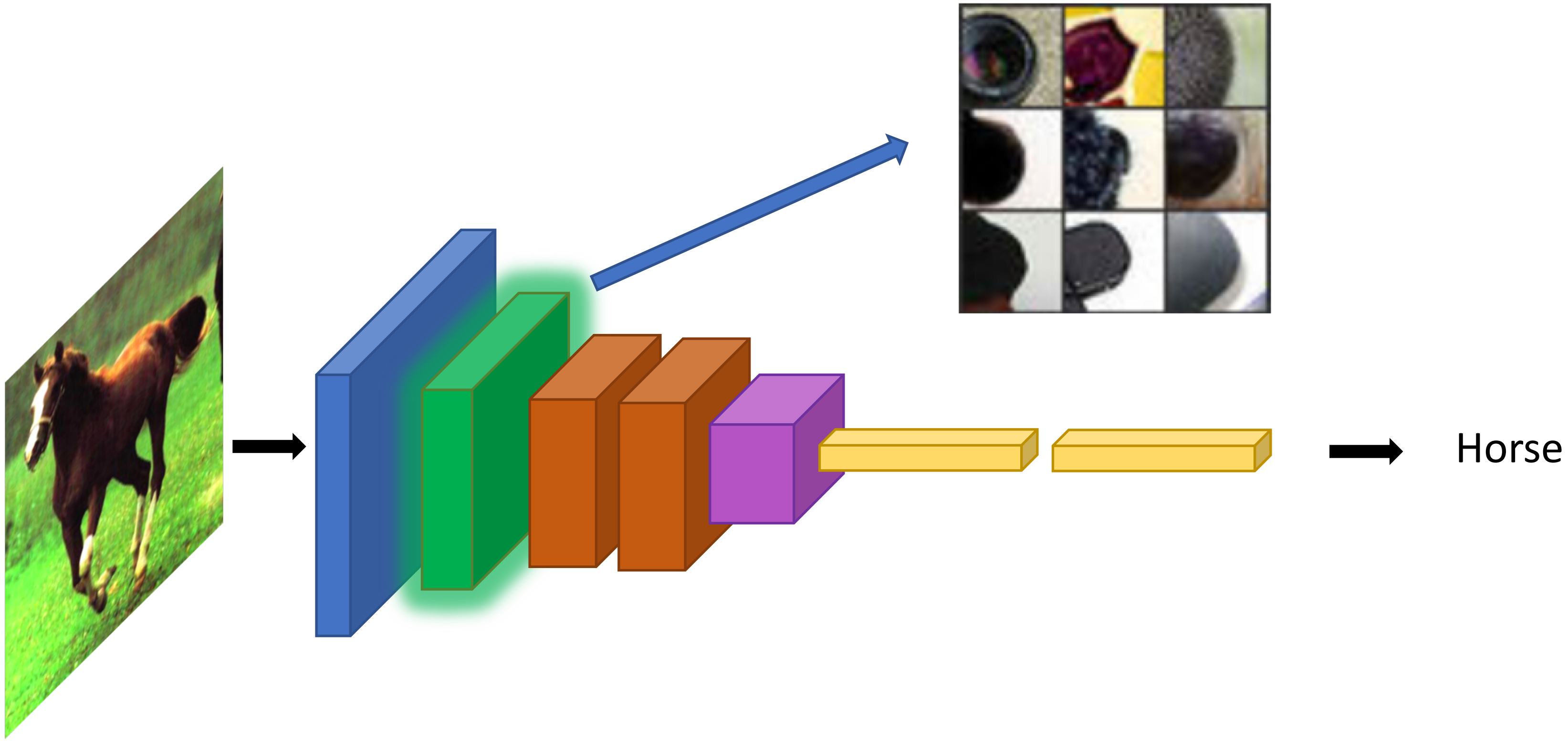
Convolution subsampling convolution

- Convolution in earlier steps detects *more local* patterns *less resilient* to distortion
- Convolution in later steps detects *more global* patterns *more resilient* to distortion
- Subsampling allows capture of *larger, more invariant* patterns

Convolutional networks

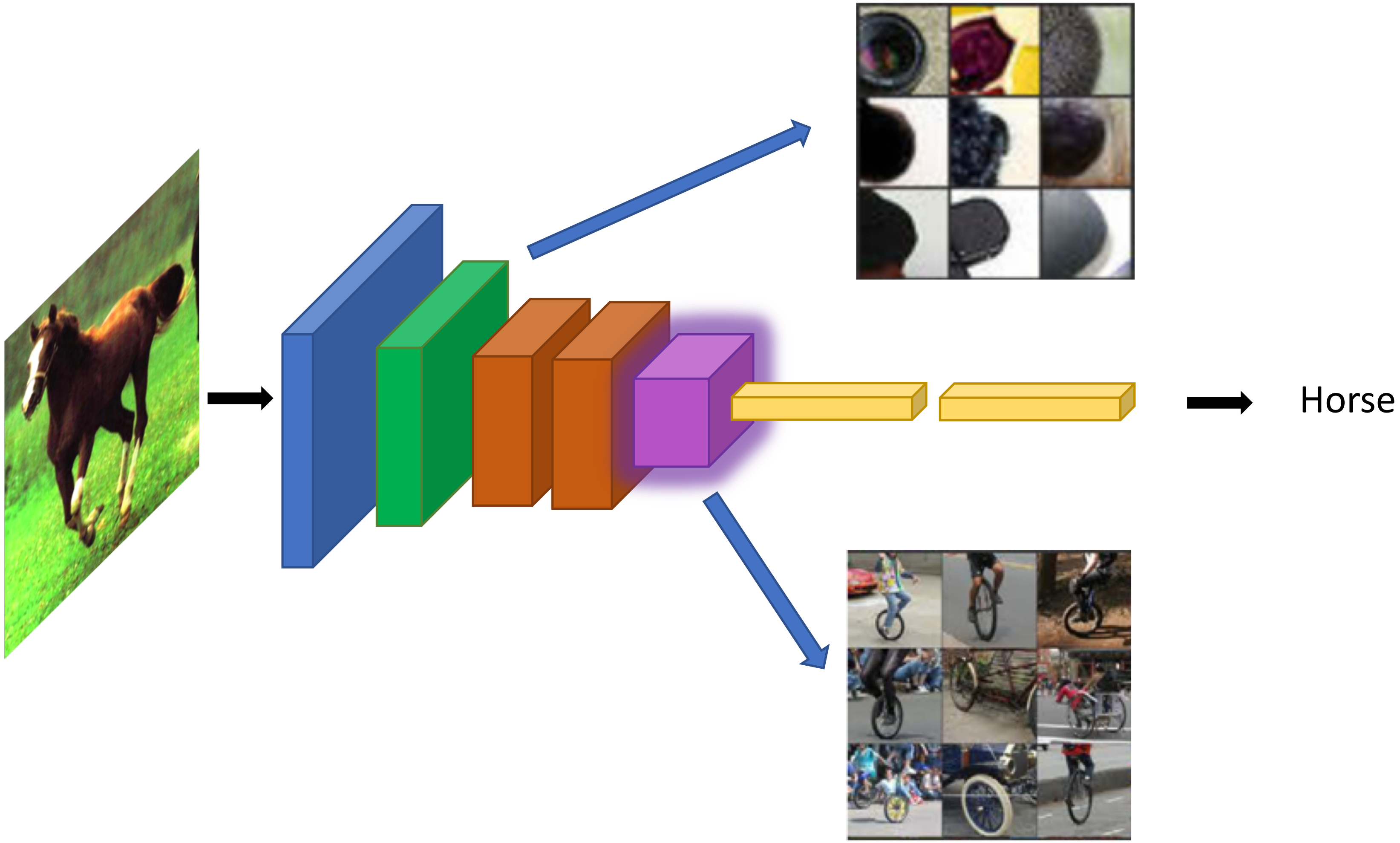


Convolutional networks



Visualizations from : M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In ECCV 2014.

Convolutional networks



Visualizations from : M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV* 2014.