Optical flow for moving scenes

Optical flow for moving scenes

Optical flow constraint equation

- Image intensity *continuous* function of x, y, t
- In time dt, pixel (x,y,t) moves to (x + u dt, y + v dt, t + dt)

$$\min_{u,v}(I(x+u\Delta t, y+v\Delta t, t+\Delta t) - I(x,y,t))^2$$

$$\equiv \min_{u,v}(I(x,y,t) + I_x u\Delta t + I_y v\Delta t + I_t\Delta t - I(x,y,t))^2$$

$$\equiv \min_{u,v}(I_x u\Delta t + I_y v\Delta t + I_t\Delta t)^2$$

$$I_x u + I_y v + I_t = 0$$

- Optical flow constraint equation: One equation, two variables

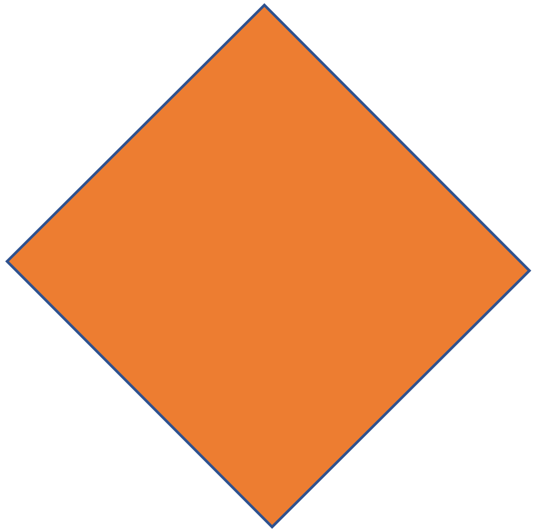
Lucas-Kanade

- Assume all pixels in patch have the same flow
- When will this produce a unique solution?

$$\begin{pmatrix} \nabla I(x_1, y_1)^T \\ \vdots \\ \nabla I(x_n, y_n)^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(x_1, y_1) \\ \vdots \\ I_t(x_n, y_n) \end{pmatrix}$$

Aperture problem

Aperture problem



Lucas-Kanade

$$\begin{pmatrix} \nabla I(x_1, y_1)^T \\ \vdots \\ \nabla I(x_n, y_n)^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(x_1, y_1) \\ \vdots \\ I_t(x_n, y_n) \end{pmatrix}$$

- Equation of the form Ax = b
- Solve using Normal equations: $x = (A^T A)^{-1} A^T b$
- Need $A^T A$ to be invertible - corners!

Lucas-Kanade

- What if we consider the whole image as one patch?
  - Constant optical flow for the entire image?
- Better: what if we consider flow as a *parametric function* of pixel location?
  - e.g. affine $\begin{bmatrix} u \\ v \end{bmatrix} = A\mathbf{x} + b$
  - More generally: $\begin{bmatrix} u \\ v \end{bmatrix} = f(\mathbf{x}, \theta)$
  - “Motion models”

Lucas-Kanade

$$\min_{\theta} \sum_{\mathbf{x}} (I(\mathbf{x} + f(\mathbf{x}, \theta)dt, t + dt) - I(\mathbf{x}, t))^2$$

- Solve by iterating on θ
- Newton iteration
- Can we remove the parametric assumption?

Baker, Simon, and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework." *International journal of computer vision* 56.3 (2004): 221-255.

Horn-Schunk

$$E(\mathbf{u},\mathbf{v}) = E_{data}(\mathbf{u},\mathbf{v}) + E_{smoothness}(\mathbf{u},\mathbf{v})$$

$$E(\mathbf{u},\mathbf{v}) = \int\int (I(x+u(x,y)\Delta t, y+v(x,y)\Delta t, t+\Delta t) - I(x,y,t))^2$$

Data

$$+\alpha(\|\nabla u\|^2 + \|\nabla v\|^2)dxdy$$

Smoothness

Horn-Schunk

$$E(\mathbf{u},\mathbf{v}) = E_{data}(\mathbf{u},\mathbf{v}) + E_{smoothness}(\mathbf{u},\mathbf{v})$$

$$E(\mathbf{u},\mathbf{v}) = \int\int (I_x u + I_y v + I_t)^2 + \alpha(\|\nabla u\|^2 + \|\nabla v\|^2)dxdy$$

Data

Smoothness

Variational minimization

- u and v are *functions*
- Euler-lagrange equations
  - Similar to “gradient=0”

$$\min_q \int L(t, q(t), \dot{q}(dt))dt$$

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0$$

Variational minimization

$$\min_q \int L(t, q(t), \dot{q}(dt))dt$$

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0$$

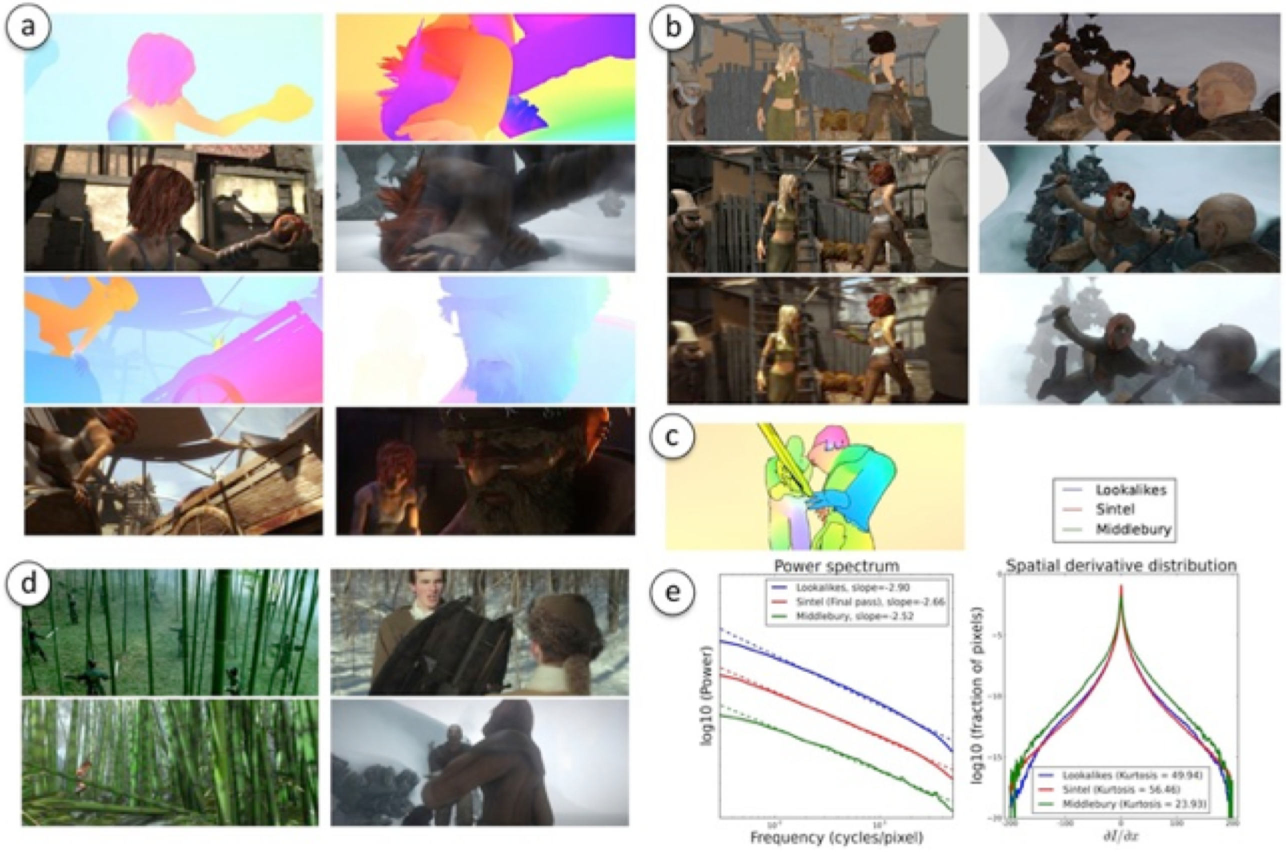
$$\min_{u,v} \int\int f(x, y, u, v, u_x, u_y, v_x, v_y)dxdy$$

$$\frac{\partial f}{\partial u} - \frac{d}{dx}\frac{\partial f}{\partial u_x} - \frac{d}{dy}\frac{\partial f}{\partial u_y} = 0$$

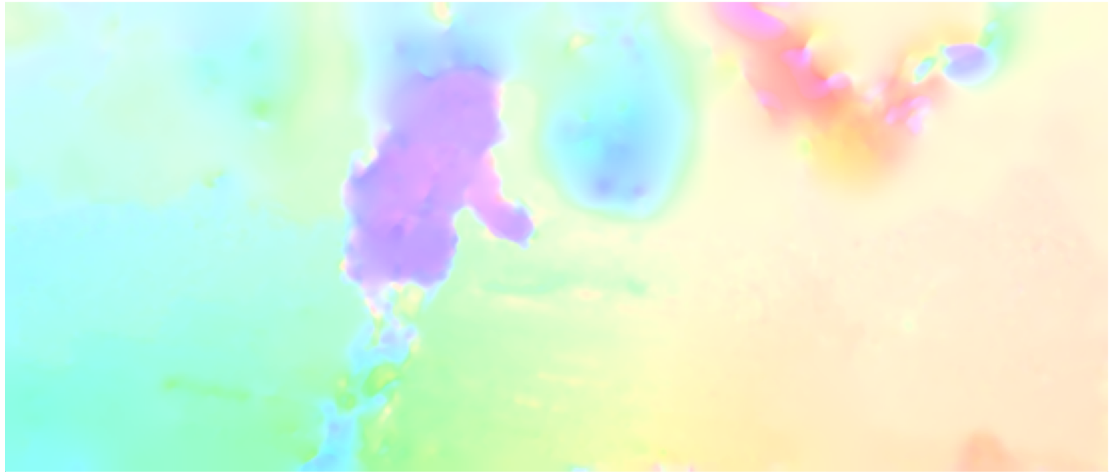
$$\frac{\partial f}{\partial v} - \frac{d}{dx}\frac{\partial f}{\partial v_x} - \frac{d}{dy}\frac{\partial f}{\partial v_y} = 0$$

MPI-Sintel

- Open-source animated movie “Sintel”
- “Naturalistic” video
- Ground truth optical flow
- Large motions
- Complex scenes



MPI-Sintel results

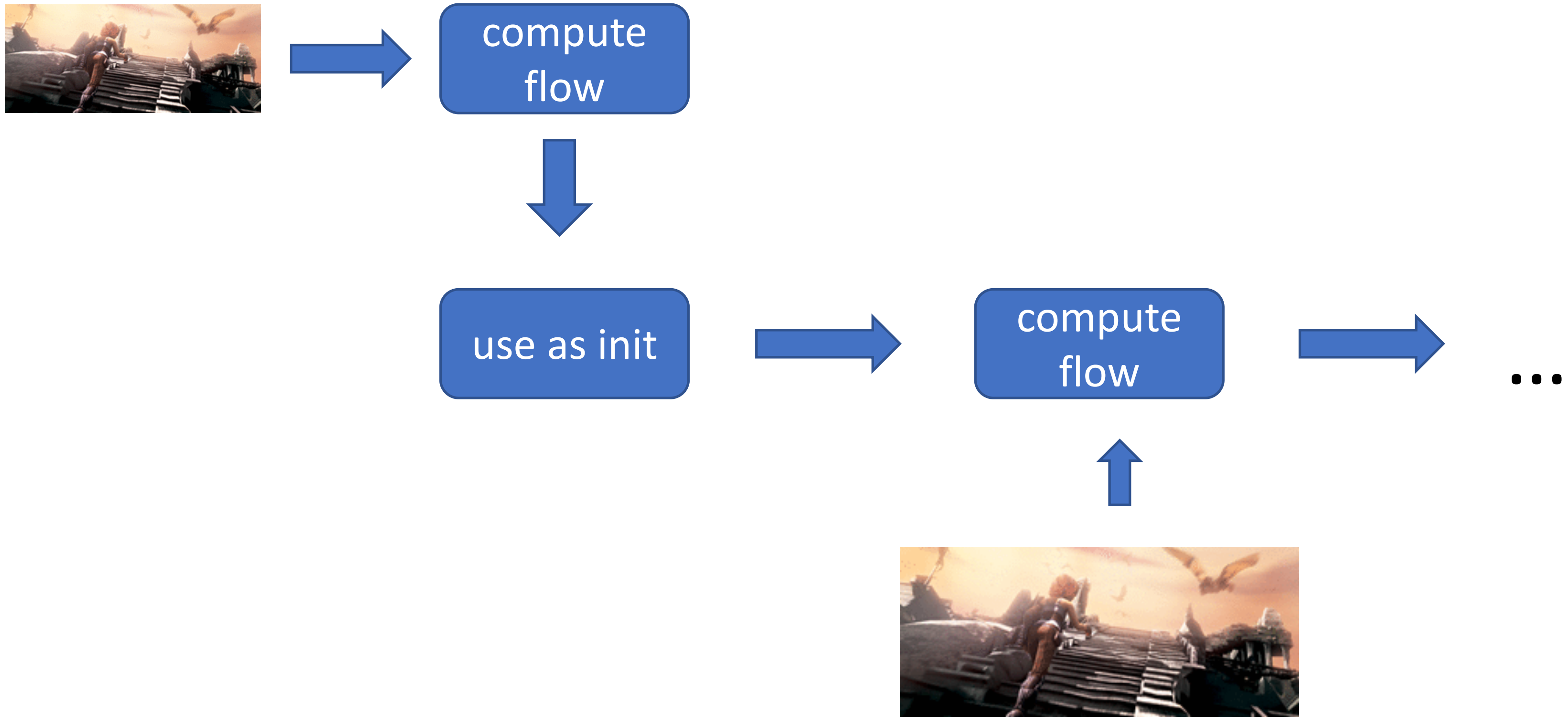


Optical flow with large displacements

- Optical flow constraint equation assumes differential optical flow
- “Large displacement”?
- Key idea: reducing resolution reduces displacement
- Reduce resolution, then upsample?
  - will lose fine details

Optical flow with large displacements

- Key idea 2: Use upsampled flow as *initialization*
- *Changes to initialization will be infinitesimal*



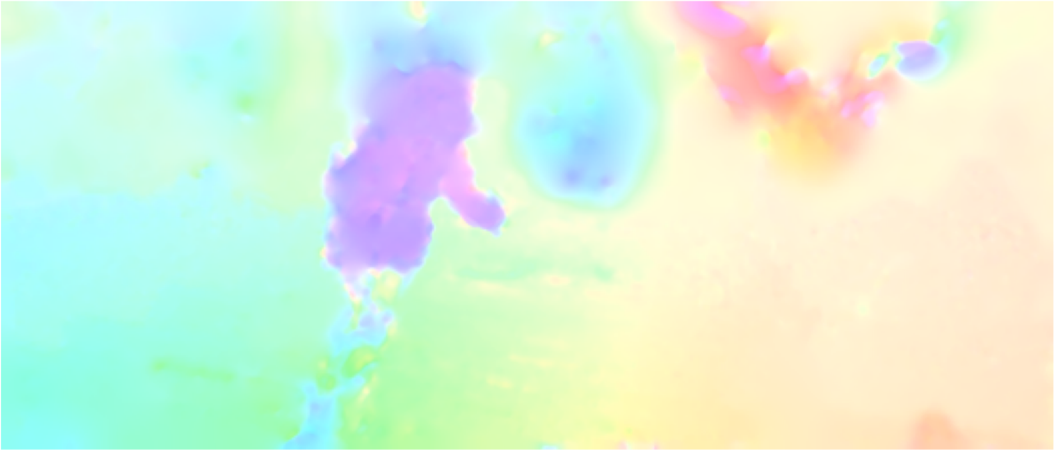
Brox, Thomas, et al. "High accuracy optical flow estimation based on a theory for warping." *Computer Vision-ECCV 2004* (2004)

Optical flow for large displacements

- Horn-schunk variants match using color - Bad!
- Use descriptor matching (e.g. SIFT) on sparse points
- Use smoothness to propagate

Brox, Thomas, Christoph Bregler, and Jitendra Malik. "Large displacement optical flow." *CVPR,* 2009.

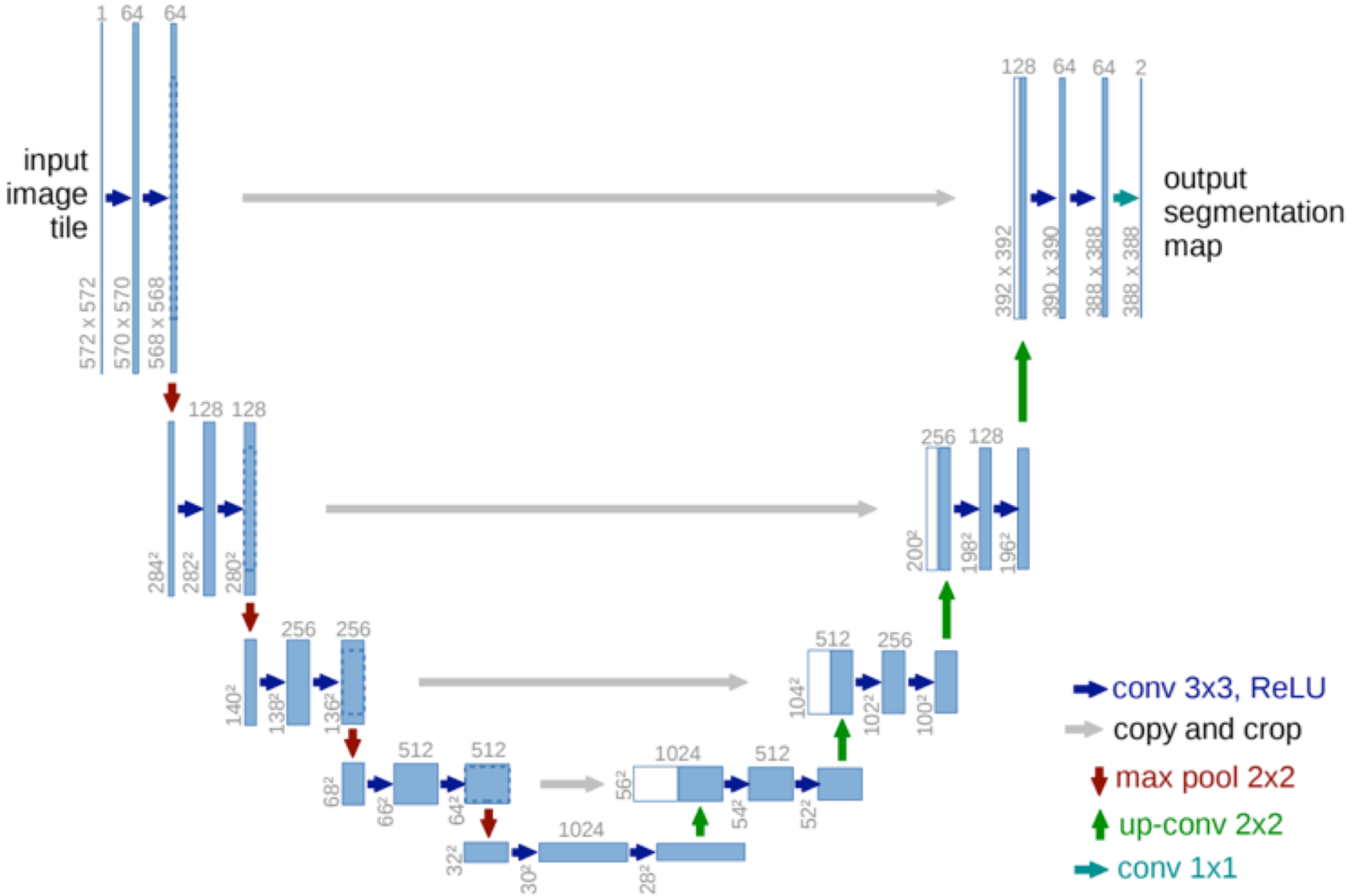
Large displacement optical flow (LDOF)

Coarse-to-fine processing

- A specific instance of a general idea
- Coarse scales:
  - Global / large structures
  - Long-range relationships
  - But: imprecise localization
- Fine scales:
  - Precise localization
  - But: aperture problem
- Idea: start from coarse scales, add fine scale detail

Coarse-to-fine processing



U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. In *MICCAI,* 2015.

Intro to ML

Image classification

- Given an image, produce a label
- Label can be:
  - 0/1 or yes/no: *Binary classification*
  - one-of-k: *Multiclass classification*
  - 0/1 for each of k concepts: *Multilabel classification*

Image classification

Is this a dog?

Yes

Image classification

Which of these is it:
dog, cat or zebra?
Dog

Image classification

Is this a dog? Yes

Is this furry? Yes

Is this sitting down? Yes

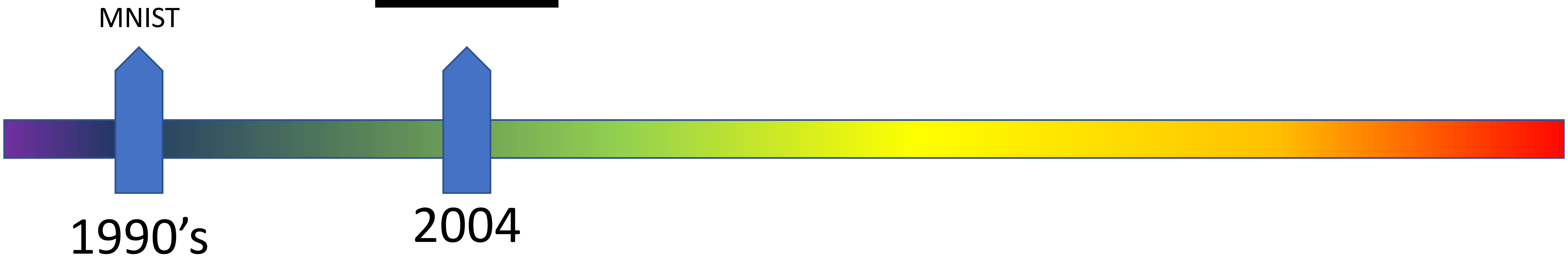
MNIST

- 2D
- 10 classes
- 6000 examples per class

1990’s

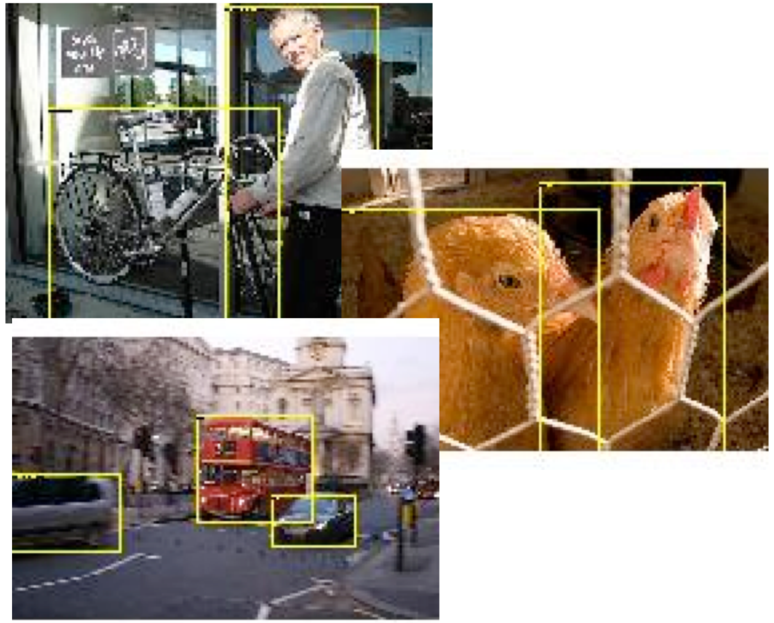
Caltech 101

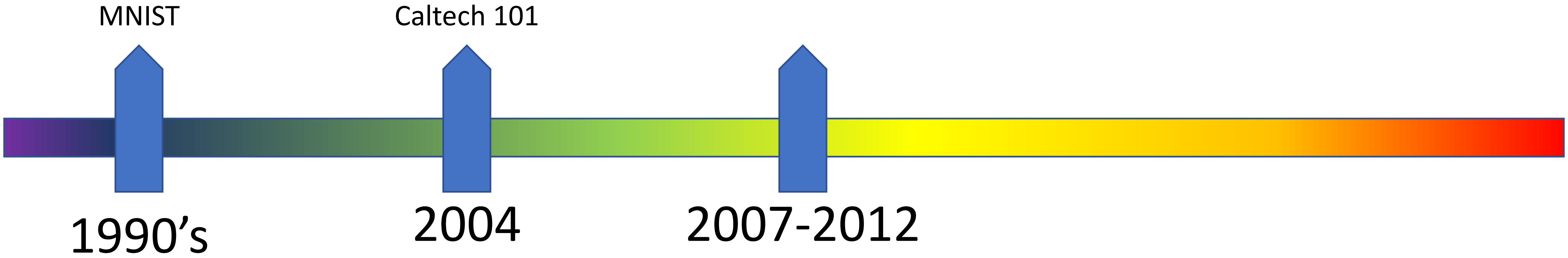
- 101 classes
- 10 classes
- 30 examples per class
- Strong category-specific biases
- Clean images



PASCAL VOC

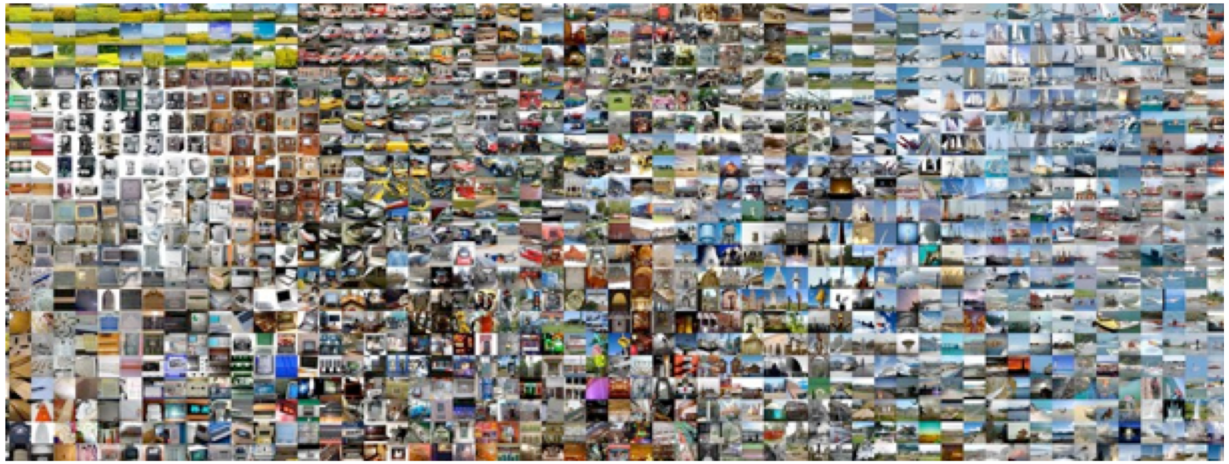
- 20 classes
- ~500 examples per class
- Clutter, occlusion, natural scenes

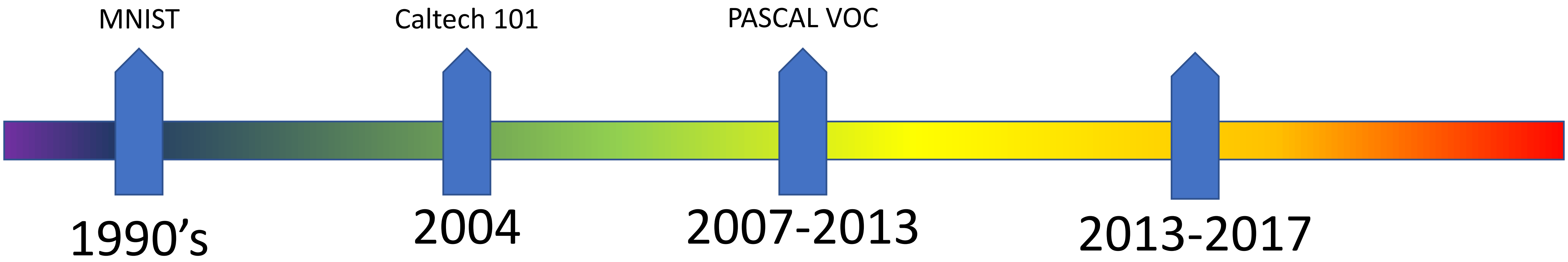




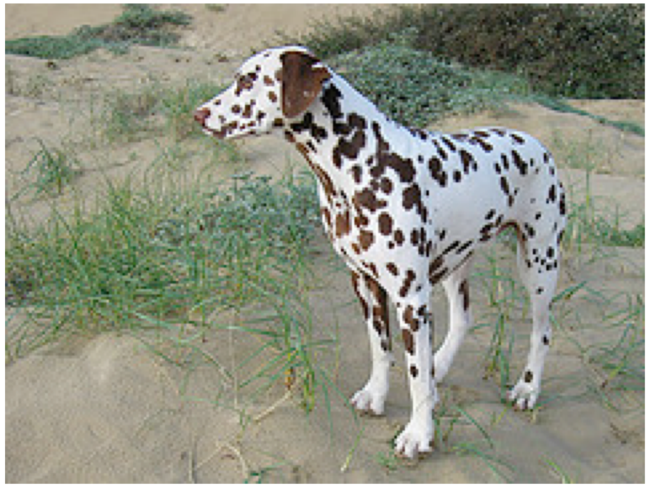
ImageNet

- 1000 classes
- ~1000 examples per class
- Mix of cluttered and clean images



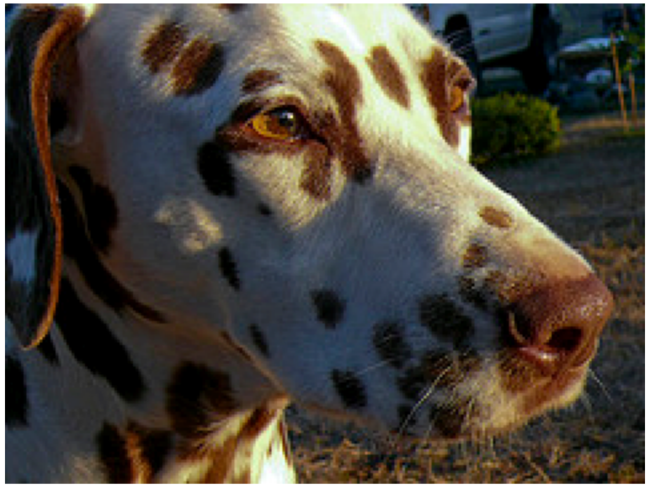


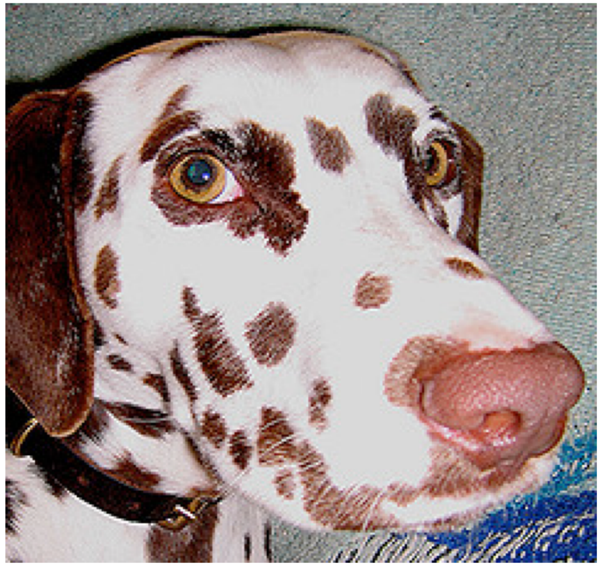
Why is recognition hard?



Pose variation

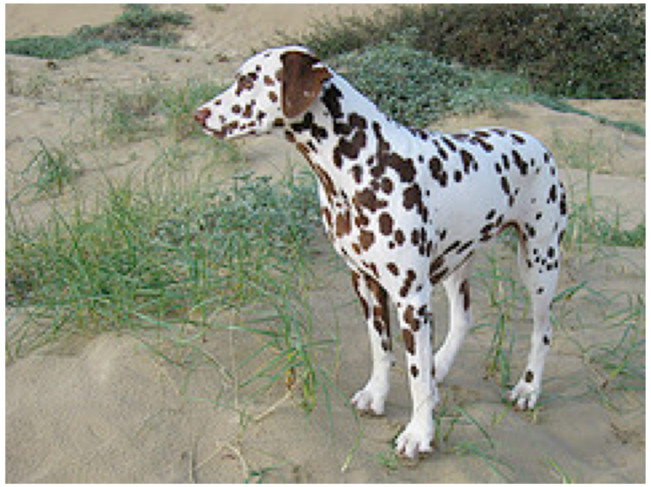
Why is recognition hard?





Lighting variation

Why is recognition hard?

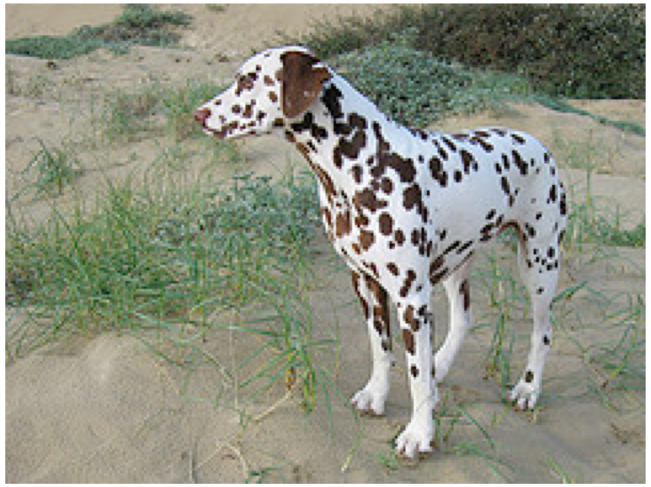


Scale variation

Why is recognition hard?

Clutter and occlusion

Why is recognition hard?



Intrinsic intra-class variation

Why is recognition hard?

Inter-class similarity

Discussion

Learning

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

$\mathcal{X}$:Images

$\mathcal{Y}$:Labels

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

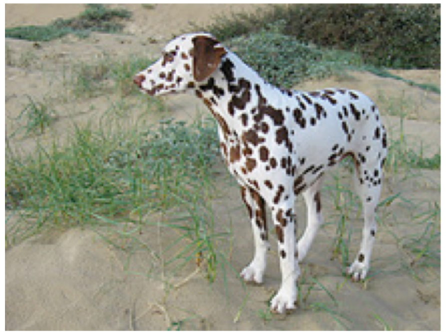
$P(x, y)$ $\qquad$ $P(y|x)$

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*

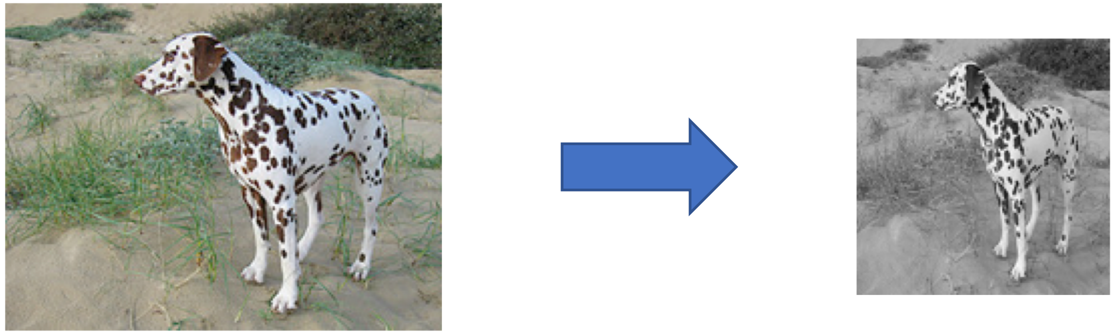
Choosing a model class

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

- Assume h is a linear classifier in feature space
- Feature space?
- Linear classifier?

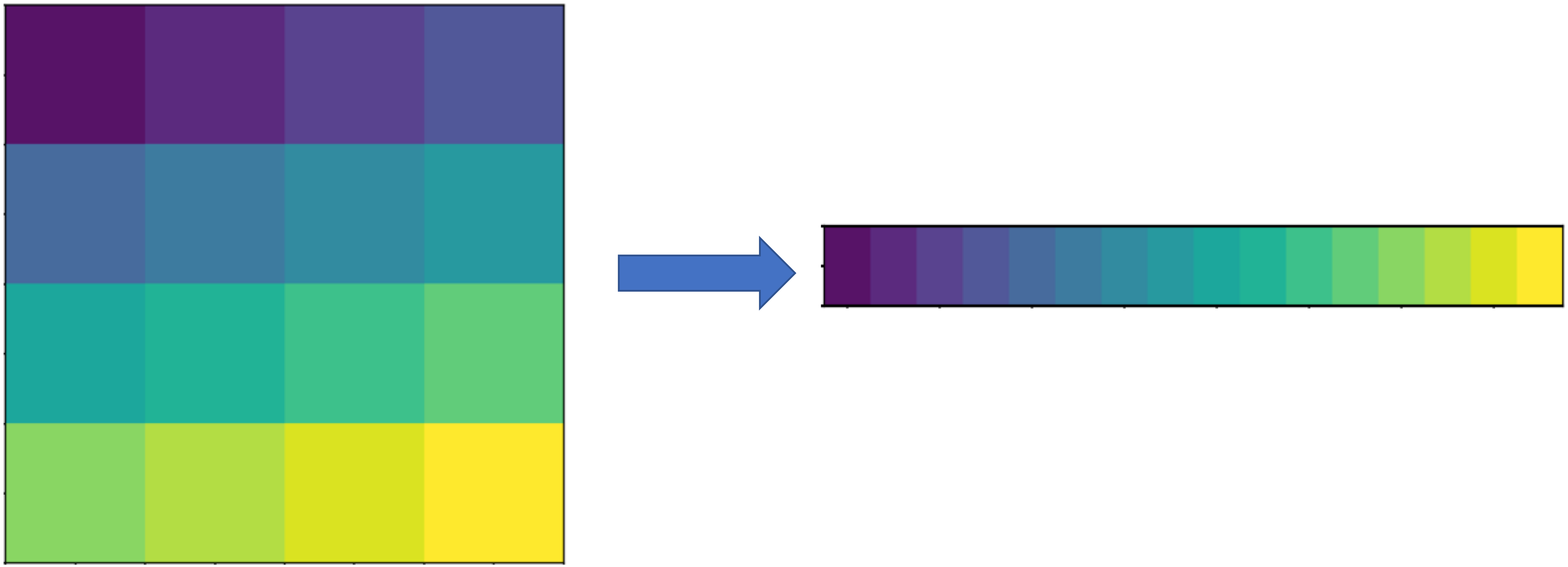
Feature space: representing images as vectors

- Find a way to project images onto $\mathbb{R}^d$



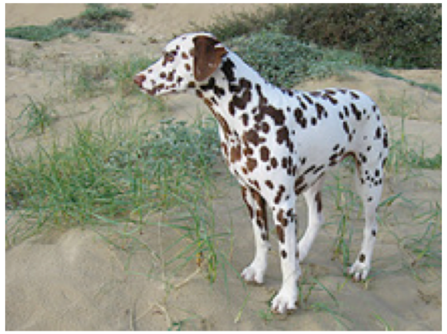
Feature space: representing images as vectors

- Find a way to project images onto $\mathbb{R}^d$

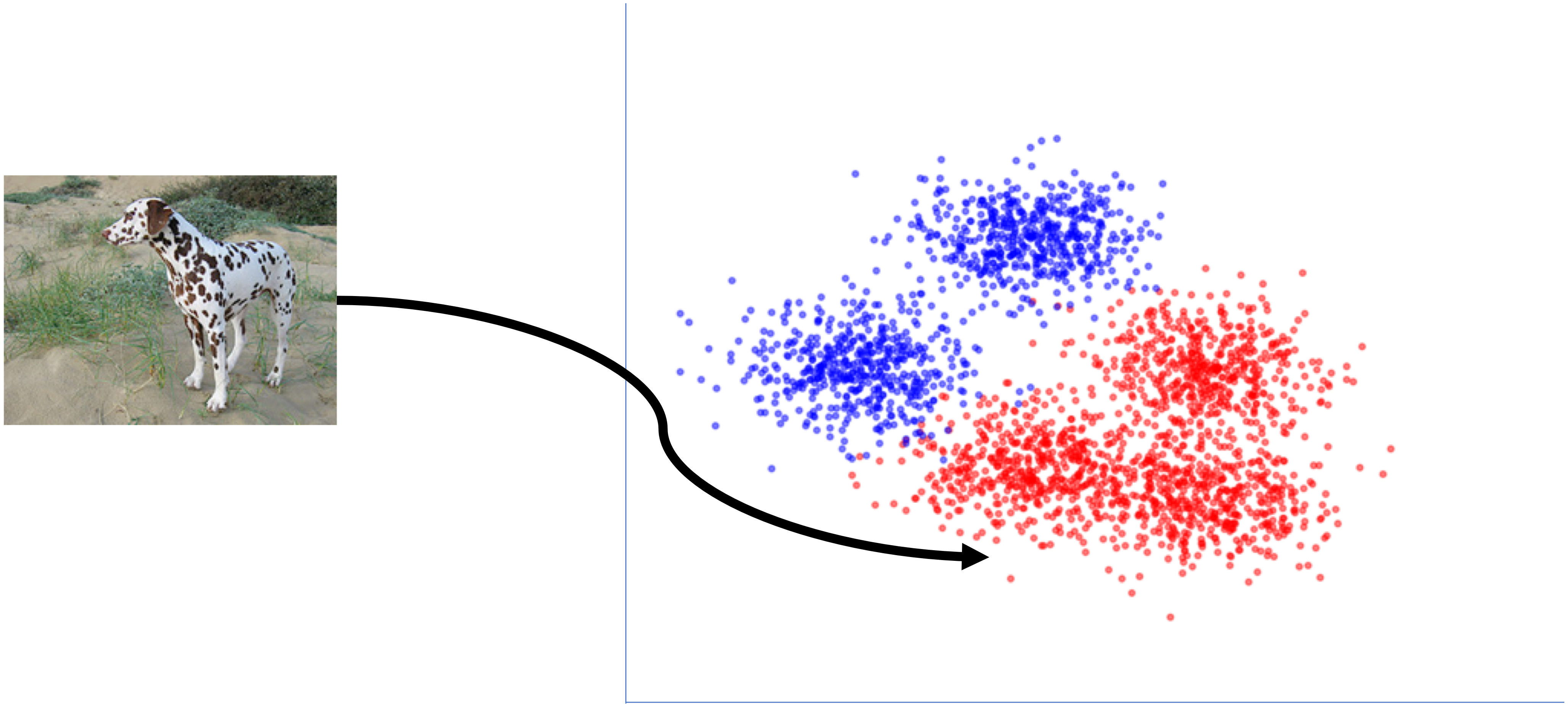


Feature space: representing images as vectors

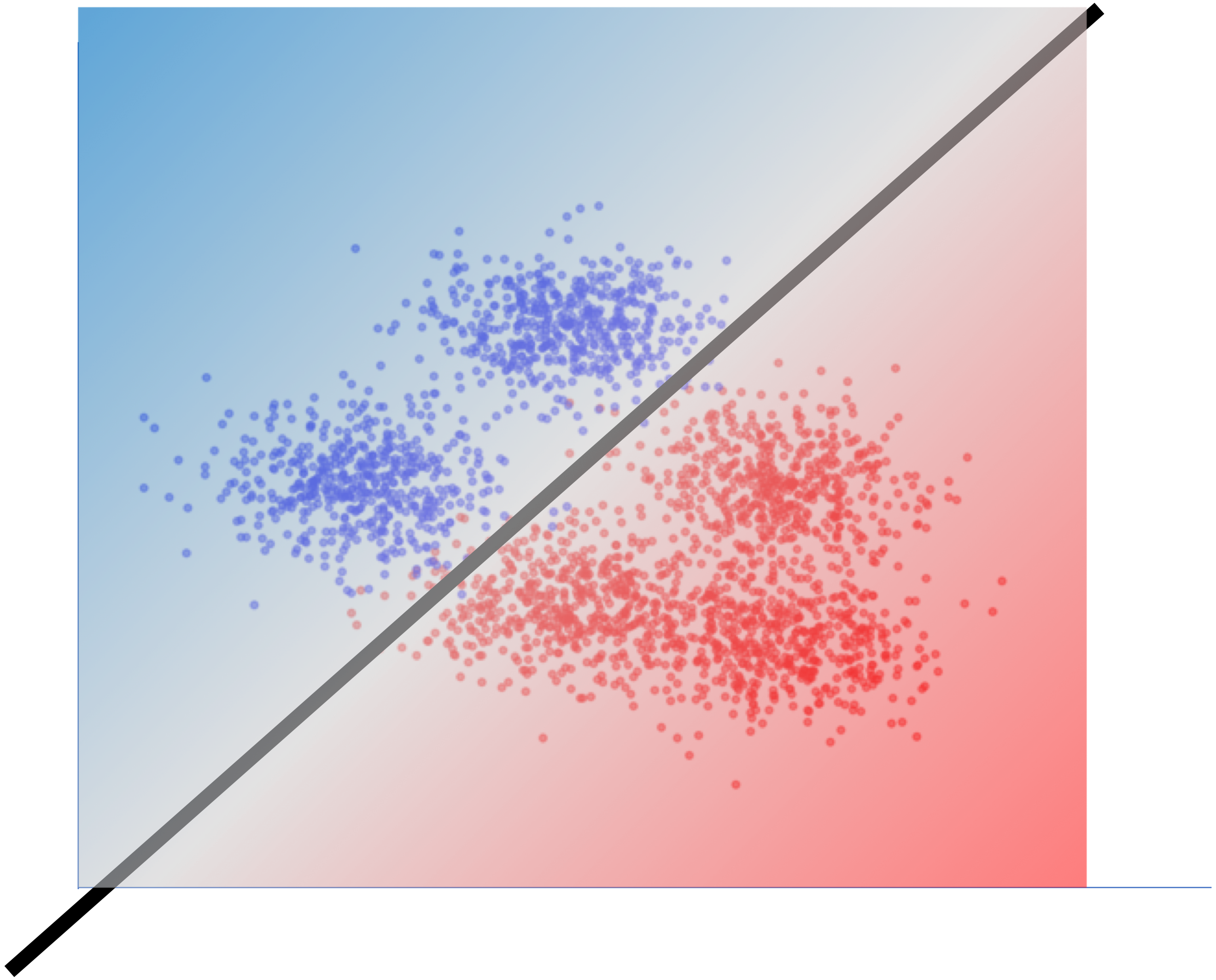
- Find a way to project images onto $\mathbb{R}^d$

$\phi($  $) =$ 

Linear classifiers



Linear classifiers

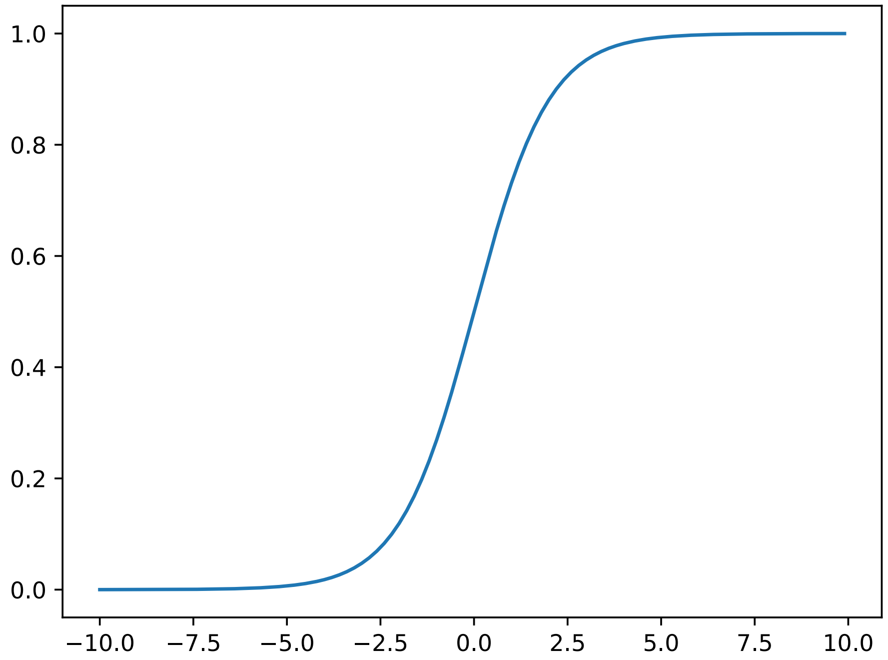


Linear classifiers in feature space

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

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

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



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?
- Given (x,y), and candidate hypothesis $h_{\mathbf{w}}$
  - $h_{\mathbf{w}}(x)$ is estimated probability label is 1
  - Idea: compute estimated probability for true label y
  - Want this probability to be high
  - *Likelihood*

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

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 log likelihood

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

- *Maximize* log likelihood

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

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: Choosing the best hypothesis

- Negative log likelihood is a *loss function*

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

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

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

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

$$R(h) = \hat{R}(S,h) + \left(R(h) - \hat{R}(S,h)\right)$$

Training error

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

- Want model with least expected risk = expected loss
- But expected risk hard to evaluate
- Empirical Risk Minimization: minimize empirical risk in training set
- Might end up picking special case: overfitting
- Avoid overfitting by:
  - Constructing large training sets
  - Reducing size of model class
  - Regularization

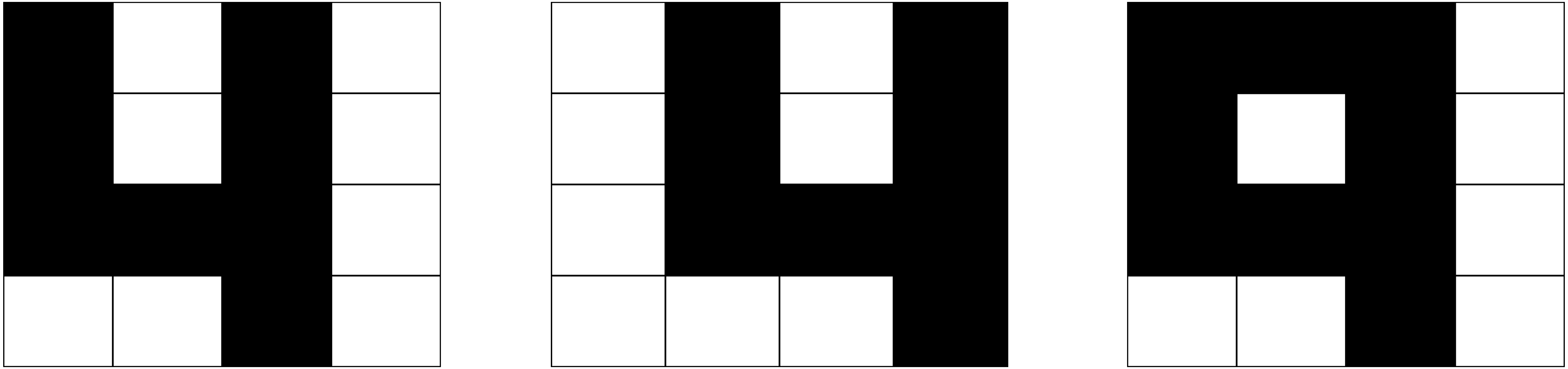
Putting it all together

- 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



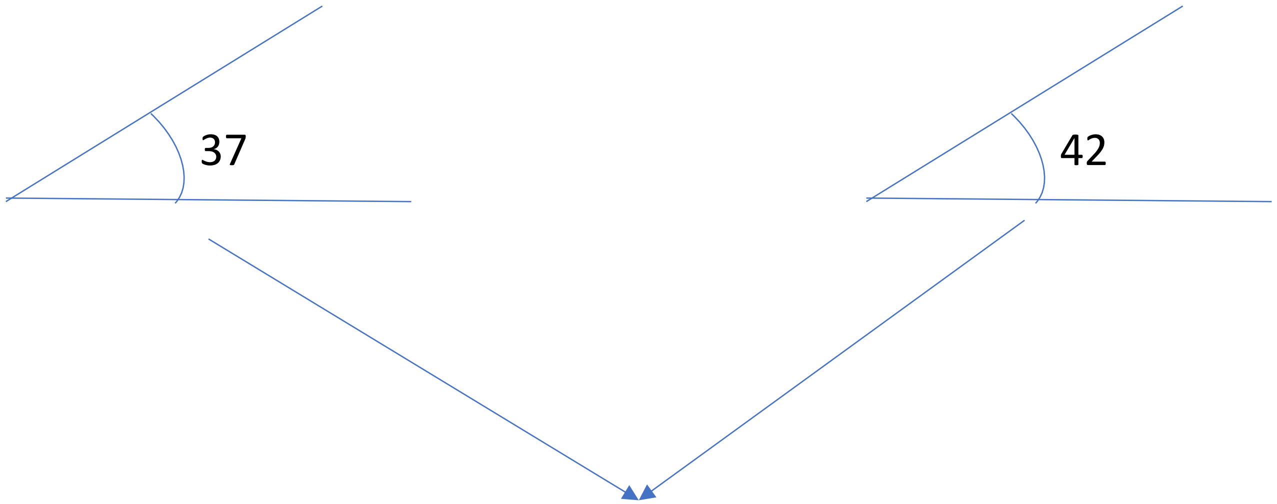
- Better feature vectors
- Non-linear classifiers

Recognition before convnets

Better feature vectors

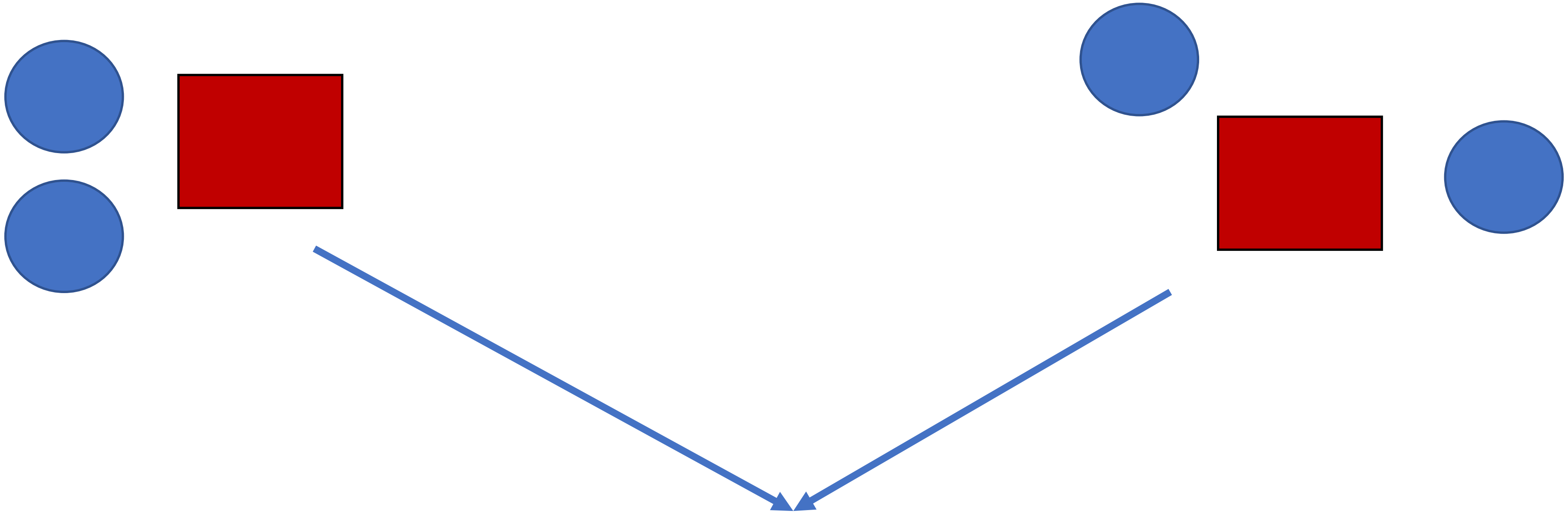
- Need to be invariant to:
  - *Small deformations*
  - *Small orientation changes*
  - *Color / lightness variation*

Rotational invariance by quantization



Between 30 and 45

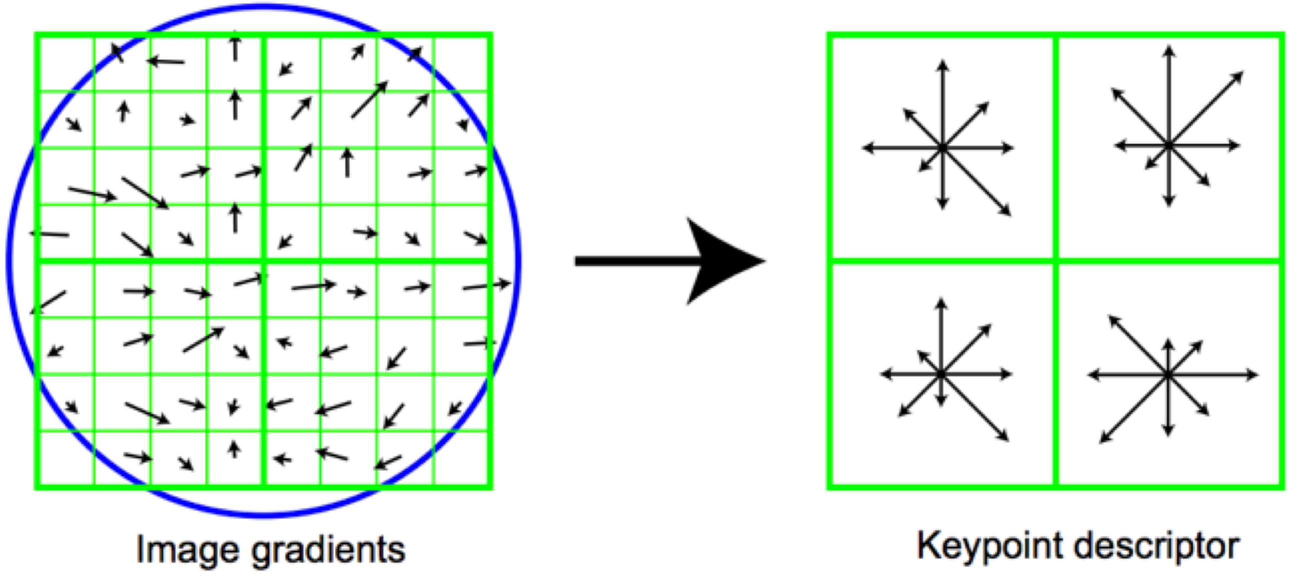
Spatial invariance by histogramming



2 blue balls, one red box

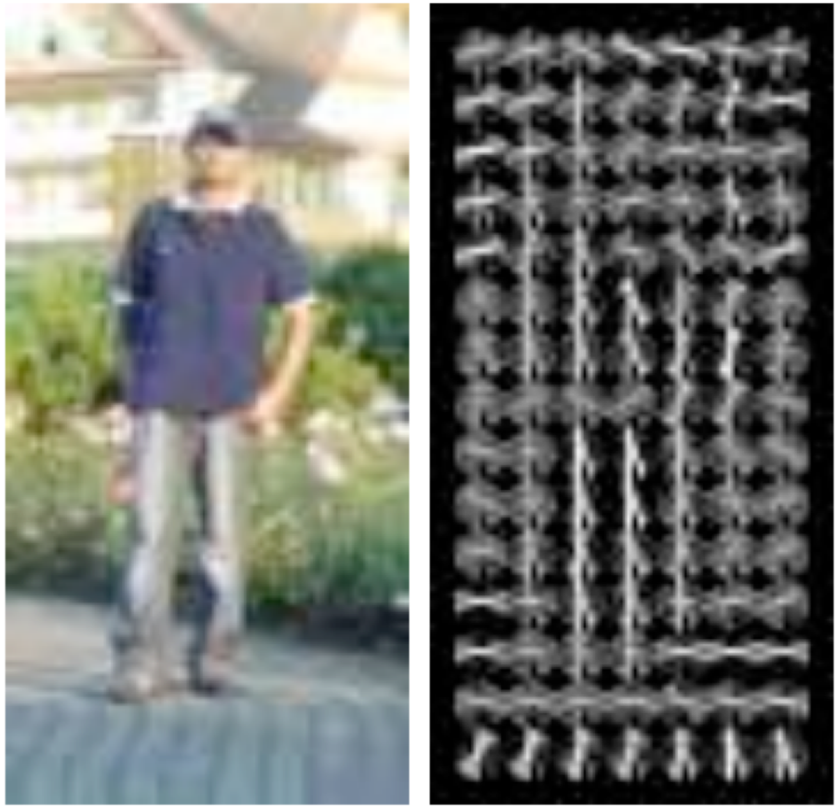
The SIFT descriptor

- Compute edge magnitudes + orientations
- Quantize orientations *(rotational invariance)*
- Divide into spatial cells
- Compute orientation histogram in each cell *(spatial invariance)*



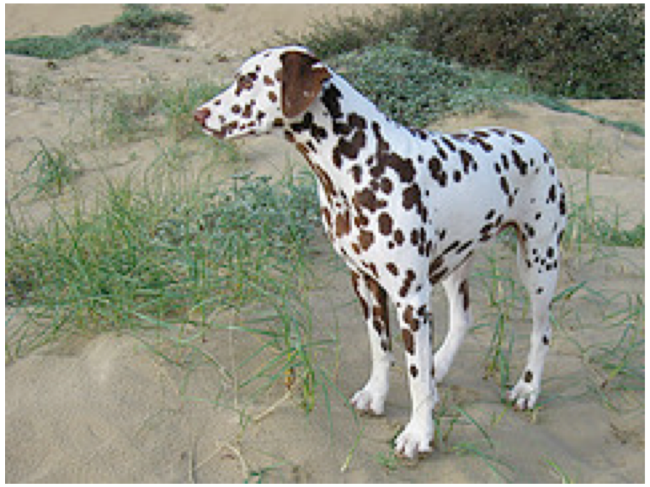
Distinctive Image Features from Scale-Invariant Keypoints. Lowe. In IJCV 2004

Same but different: HOG



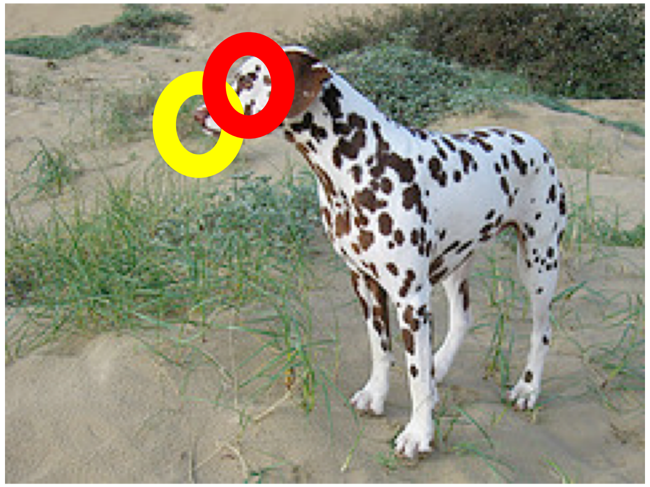
Histogram of oriented gradients

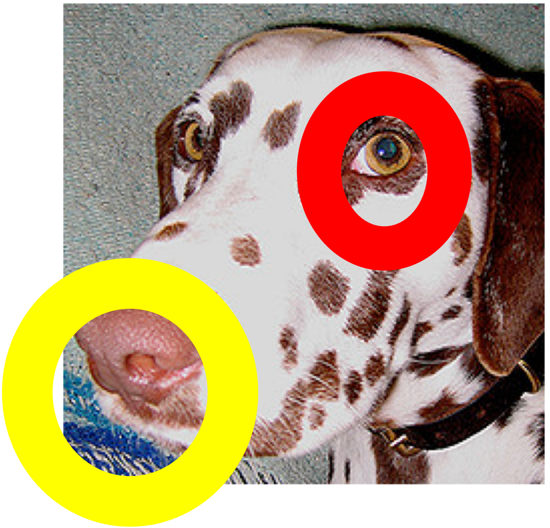
Invariance to large deformations



Invariance to large deformations

- Issue: object / object part may occur at any image location

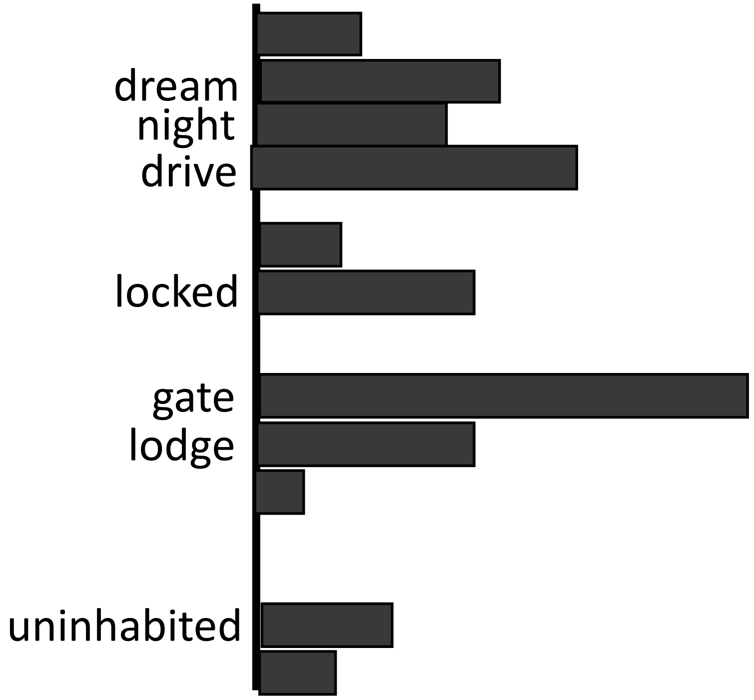




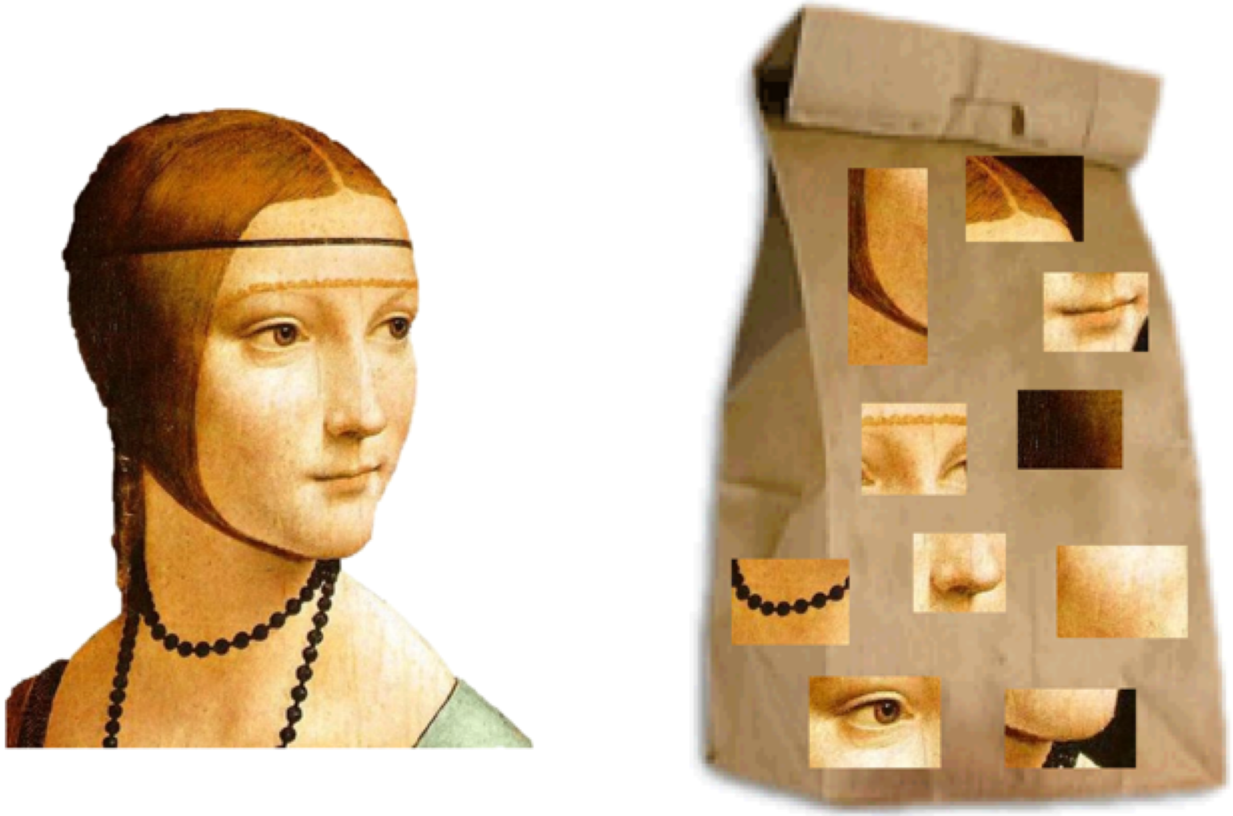
- Idea: want to represent the image as a "bag of object parts"

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….

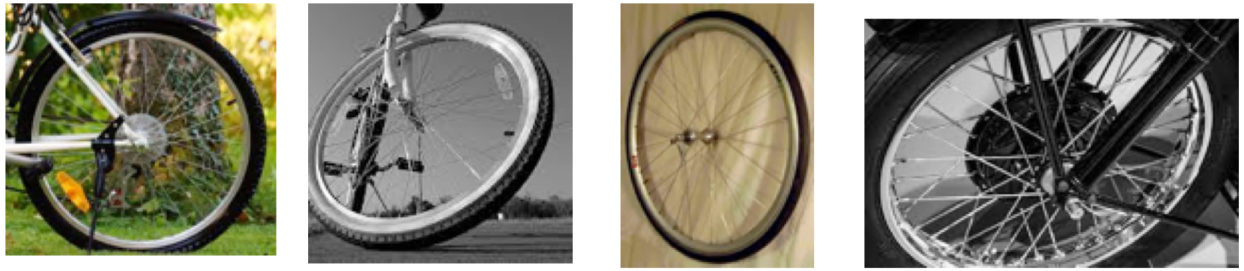


Bags of visual words



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

What should be visual words?

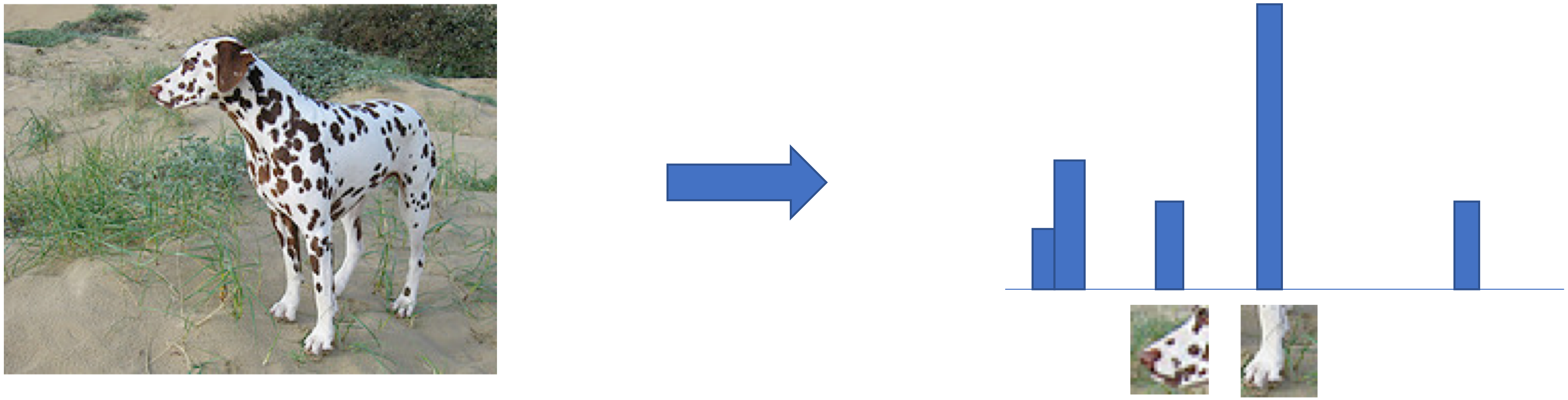
- Cluster in what space?

- Need invariance to color, small deformations, orientation changes
- Cluster in SIFT space!
- Each cluster = *visual word*

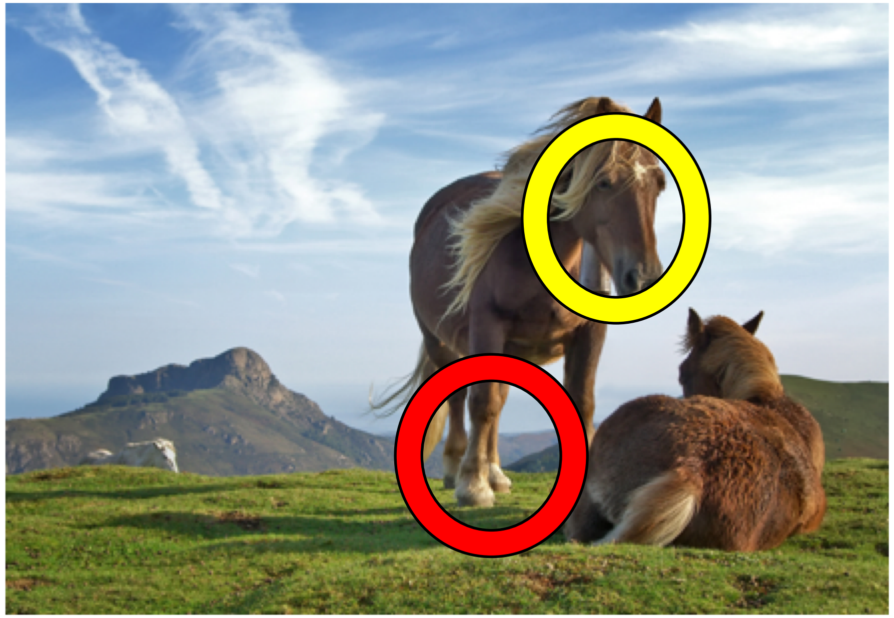
Encoding images as bag of words

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

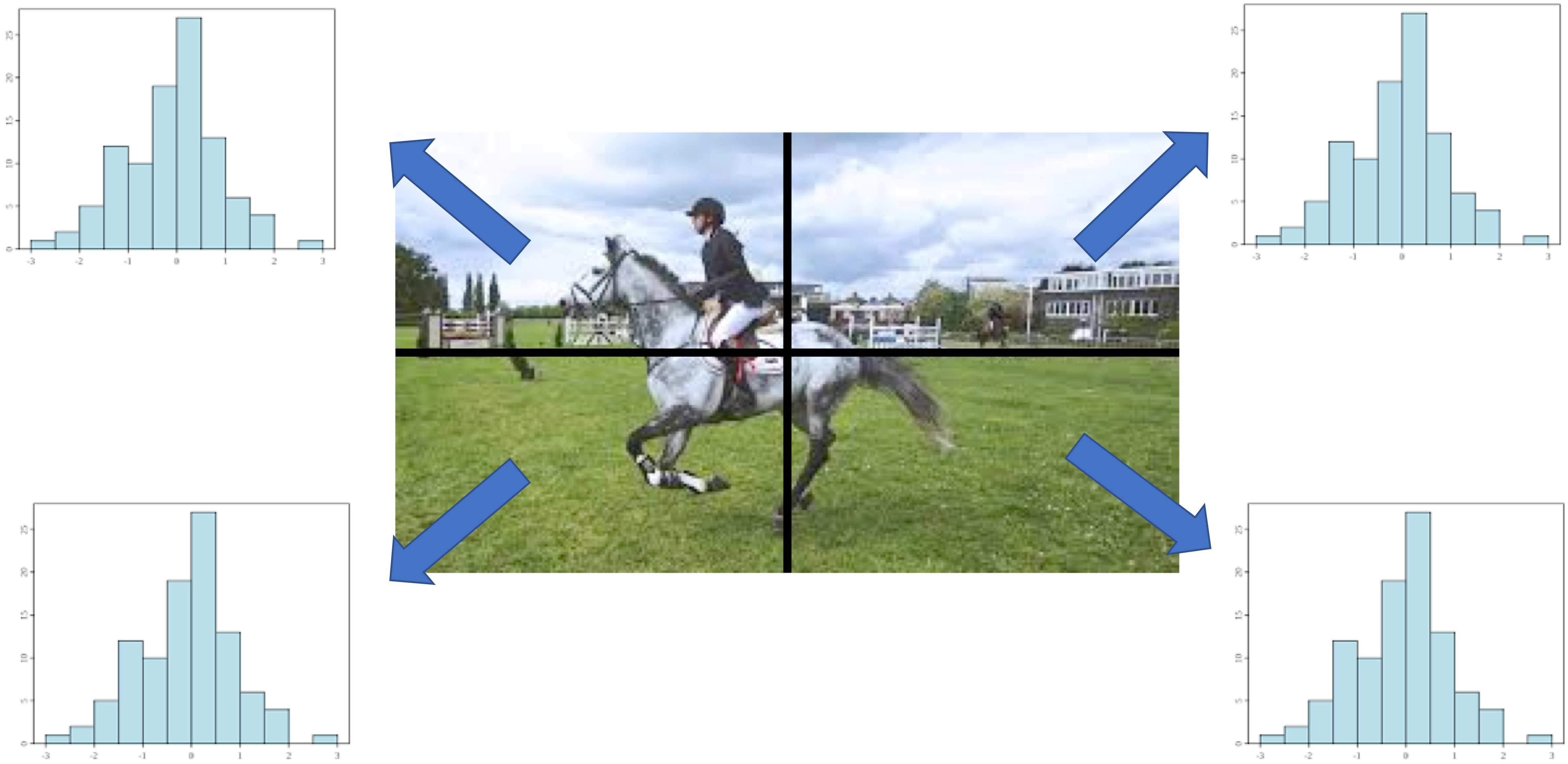


Too much invariance?

- Object parts appear in somewhat fixed relationships



Idea: Spatial pyramids



Discussion