IMAGE REPRESENTATION

Xinyi Fan
COS598c  Spring2014
APPROACHES

• Bag of Words
• Spatial Pyramid Matching
• Descriptor Encoding
APPROACHES

• Bag of Words

• Spatial Pyramid Matching

• Descriptor Encoding
Object
Object  ➔  Bag of Feature Words

Adapted from slides by Fei-Fei Li
BAG OF WORDS

• Definition
  • Independent, orderless features
BAG OF WORDS

- Definition

- Independent, orderless features
BAG OF WORDS

• Definition
  • Independent, orderless features
  • Histogram representation

Adapted from slides by Fei-Fei Li

Monday, April 7, 14
BAG OF WORDS

• Definition
  • Independent, orderless features
  • Histogram representation

Adapted from slides by Fei-Fei Li
Feature detection & representation

Codewords dictionary formation

Image representation

Category models/classifiers

Category decision

Adapted from slides by Fei-Fei Li
Feature detection & representation

Codewords dictionary formation

Image representation

Representation

Category models/classifiers

Category decision

Adapted from slides by Fei-Fei Li
Feature detection & representation

Codewords dictionary formation

Image representation

Learning

Category models/classifiers

Category decision

Adapted from slides by Fei-Fei Li
Feature detection & representation
Codewords dictionary formation
Image representation

Learning

Category models/classifiers

Recognition

Category decision

Adapted from slides by Fei-Fei Li
Feature Detection & Representation

dense regular grids

points of interest

random
Feature Detection & Representation

[Image of three figures with circular markers on them]
Codewords Dictionary Formation
Codewords Dictionary Formation

Vector quantization
Codewords Dictionary Formation

Vector quantization
Codewords Dictionary Formation
Image Representation

![Bar chart showing feature frequency against codewords. The chart has multiple bars with varying heights.]
Using BoW Representation

Use BoW as feature vector for standard classifier
- Naive Bayesian
- SVM

Cluster BoW vectors over image collection
- Category classification (supervised)
- Object discovery (unsupervised)

Use BoW to build hierarchical models
- Decompose scene/object
BoW Summary

Issues

• **Sampling strategy**
  dense uniform, interest points, random...

• **Codebook learning**
  supervised/unsupervised, size...

• **Similarity measurement**
  SVM, Pyramid Matching

• **Spatial information**

• **Scalability**
BoW Summary

Issues

• Sampling strategy
dense uniform, interest points, random...

• Codebook learning
  supervised/unsupervised, size...

• Similarity measurement
  SVM, Pyramid Matching

• Spatial information

• Scalability
BoW Summary

Issues

• Sampling strategy
dense uniform, interest points, random...

• Codebook learning
supervised/unsupervised, size...

• Similarity measurement
SVM, Pyramid Matching

• Spatial information

• Scalability
Pyramid Matching
Pyramid Matching
Pyramid Matching

optimal partial matching between sets of features
Pyramid Matching

\[ Y = \{ \tilde{y}_1, \ldots, \tilde{y}_n \} \]

\[ X = \{ \tilde{x}_1, \ldots, \tilde{x}_m \} \]

\[ \tilde{x}_i \in \mathbb{R}^d \]

\[ \tilde{y}_i \in \mathbb{R}^d \]

optimal partial matching between sets of features

Adapted from slides by Grauman and Darrell
Feature Extraction

\[ X = \{ \vec{x}_1, \ldots, \vec{x}_m \} \quad \vec{x}_i \in \mathbb{R}^d \]

Histogram pyramid:
level \( \ell \) has bins of size \( 2^\ell \)

\[ \Psi(X) = [H_0(X), \ldots, H_L(X)] \]
Histogram intersection:

\[
\mathcal{I}(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j)
\]

Adapted from slides by Grauman and Darrell
**Counting New Matches**

Histogram intersection:
\[ \mathcal{I}(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j) \]

- matches at current level
- matches at previous level

\[ N_\ell = \mathcal{I}(H_\ell(X), H_\ell(Y)) - \mathcal{I}(H_{\ell-1}(X), H_{\ell-1}(Y)) \]

Difference in histogram intersections across levels counts **number of new pairs matched**

Adapted from slides by Grauman and Darrell
Pyramid Match Kernel

The Pyramid Match Kernel is given by:

\[ K_\Delta(\Psi(X), \Psi(Y)) = \sum_{\ell=0}^{L} \frac{1}{2^\ell} (\mathcal{I}(H_\ell(X), H_\ell(Y)) - \mathcal{I}(H_{\ell-1}(X), H_{\ell-1}(Y))) \]

- **histogram pyramids**
- **measure of difficulty of a match at level** \( \ell \)
- **number of newly matched pairs at level** \( \ell \)

Adapted from slides by Grauman and Darrell
Approximation of Optimal Partial Matching

Approximation of the optimal partial matching

100 sets with 2D points, cardinalities vary between 5 and 100

Adapted from slides by Grauman and Darrell
Building a Classifier

- Train SVM by computing kernel values between all labeled training examples

- Classify novel examples by computing kernel values against support vectors

- One-versus-all for multi-class classification
BoW Summary

Issues

• Sampling strategy
dense uniform, interest points, random...

• Codebook learning
  supervised/unsupervised, size...

• Similarity measurement
  SVM, Pyramid Matching

• Spatial information

• Scalability
BoW Summary

Spatial information

BoW removes spatial layout

Increases the invariance to scale/translation/deformation

Sacrifices discriminative power
APPROACHES

• Bag of Words

• Spatial Pyramid Matching

• Descriptor Encoding
Spatial Pyramid Matching

Image credit: Lazebnik et al
Spatial Pyramid Matching
Spatial Pyramid Matching
Feature detection & representation

Codewords dictionary formation

Image representation

Category models/classifiers

Category decision
Feature detection & representation

Codewords dictionary formation

Image representation

Spatial Pyramid Matching

Category models/classifiers

Category decision
Spatial Pyramid Matching

• Quantize feature vectors into $M$ discrete types

• Perform pyramid matching in 2D image space for each channel $m = 1, \ldots, M$

• Assume features of the same type $m$ can be matched to each other
Spatial Pyramid Matching

\[ \sum_{m=1}^{M} \kappa^L(X_m, Y_m) \]

Image credit: Lazebnik et al. 2006
Pyramid Match Kernel

$$K_\Delta(\Psi(X), \Psi(Y)) =$$

$$\sum_{\ell=0}^{L} \frac{1}{2^\ell} (I(H_\ell(X), H_\ell(Y)) - I(H_{\ell-1}(X), H_{\ell-1}(Y)))$$

histogram pyramids

number of newly matched pairs at level $\ell$

measure of difficulty of a match at level $\ell$

Adapted from slides by Grauman and Darrell
Pyramid Match Kernel

\[ \kappa^L(X, Y) = K_\Delta(\Psi(X), \Psi(Y)) = \]

\[ \sum_{\ell=0}^{L} \frac{1}{2^\ell} (I(H_\ell(X), H_\ell(Y)) - I(H_{\ell-1}(X), H_{\ell-1}(Y))) \]

The final kernel is sum of the separate channel kernels:

\[ \sum_{m=1}^{M} \kappa^L(X_m, Y_m) \]
APPROACHES

• Bag of Words

• Spatial Pyramid Matching

• Descriptor Encoding
  - Linear SPM using Sparse Coding
  - Locality-constrained Linear Coding
  - Fisher Vector
Feature detection & representation

Codewords dictionary formation

Image representation

(Spatial Pyramid Matching)

Category models/classifiers

Category decision
Feature detection & representation

Codewords dictionary formation

Image representation

For better performance

Other ways of encoding local feature descriptors

(Spatial Pyramid Matching)

Category models/classifiers

Category decision
APPROACHES

• Bag of Words

• Spatial Pyramid Matching

• Descriptor Encoding
  - Linear SPM using Sparse Coding
  - Locality-constrained Linear Coding
  - Fisher Vector
Linear SPM using Sparse Coding

Image credit: Yang et al 2009
Linear SPM using Sparse Coding

(a) Nonlinear SPM
(b) Linear ScSPM

Image credit: Yang et al. 2009
Encoding SIFT: From VQ to SC

$$\min_{\mathbf{V}} \sum_{m=1}^{M} \min_{k=1, \ldots, K} \left\| \mathbf{x}_m - \mathbf{v}_k \right\|^2$$

codebook: \( \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K]^\top \)
Encoding SIFT: From VQ to SC

\[
\min_{\mathbf{V}} \sum_{m=1}^{M} \min_{k=1, \ldots, K} \| \mathbf{x}_m - \mathbf{v}_k \|^2
\]

\[
\min_{\mathbf{V}, \mathbf{U}} \sum_{m=1}^{M} \| \mathbf{x}_m - \mathbf{u}_m \mathbf{V} \|^2
\]

s.t. \( \text{Card}(\mathbf{u}_m) = 1, |\mathbf{u}_m| = 1, \mathbf{u}_m \geq 0, \forall m \)

\[
\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]^T
\]
Encoding SIFT: From VQ to SC

\[
\min_{V} \sum_{m=1}^{M} \min_{k=1,\ldots,K} \|x_m - v_k\|^2
\]

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2
\]

s.t. \(\text{Card}(u_m) = 1, |u_m| = 1, u_m \geq 0, \forall m\)

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda |u_m|
\]

s.t. \(\|v_k\| \leq 1, \forall k\)
Why L1 encourages sparsity

\[
\min \| x - uV \|_2^2 + \lambda \| u \|_1 \\
\min \| x - uV \|_2^2 + \lambda \| u \|_2
\]
Encoding SIFT: From VQ to SC

\[
\min_{V} \sum_{m=1}^{M} \min_{k=1,\ldots,K} \|x_m - v_k\|^2
\]

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2
\]

s.t. \( \text{Card}(u_m) = 1, |u_m| = 1, u_m \geq 0, \forall m \)

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda |u_m|
\]

s.t. \( \|v_k\| \leq 1, \forall k \)
Encoding SIFT: From VQ to SC

\[
\begin{align*}
\min_{\mathbf{V}} & \sum_{m=1}^{M} \min_{k=1, \ldots, K} \| \mathbf{x}_m - \mathbf{v}_k \|^2 \\
\min_{\mathbf{V}, \mathbf{U}} & \sum_{m=1}^{M} \| \mathbf{x}_m - \mathbf{u}_m \mathbf{V} \|^2 \\
\quad \text{s.t.} & \quad \text{Card}(\mathbf{u}_m) = 1, |\mathbf{u}_m| = 1, \mathbf{u}_m \geq 0, \forall m
\end{align*}
\]

\[
\begin{align*}
\min_{\mathbf{V}, \mathbf{U}} & \sum_{m=1}^{M} \| \mathbf{x}_m - \mathbf{u}_m \mathbf{V} \|^2 + \lambda |\mathbf{u}_m| \\
\quad \text{s.t.} & \quad \| \mathbf{v}_k \| \leq 1, \quad \forall k
\end{align*}
\]

Implementation:
feature-sign search algorithm [Lee et al 2006]
http://ai.stanford.edu/~hllee/softwares/nips06-sparsecoding.htm
Algorithm Architecture

Image credit: Yang et al. 2009
Linear SPM

\[ U = [u_1, \ldots, u_M]^\top \]

\[ z = \mathcal{F}(U) \]

define \( \mathcal{F} \) as:

\[ z_j = \max \{|u_{1j}|, |u_{2j}|, \ldots, |u_{Mj}|\} \]

\[ \kappa(z_i, z_j) = z_i^\top z_j = \sum_{l=0}^{2^l} \sum_{s=1}^{2^l} \sum_{t=1}^{2^l} \langle z_i^l(s, t), z_j^l(s, t) \rangle \]
APPROACHES

• Bag of Words
• Spatial Pyramid Matching
• Descriptor Encoding
  - Linear SPM using Sparse Coding
  - Locality-constrained Linear Coding
  - Fisher Vector
Locality-constrained Linear Coding

**Step 3:**
c_i is an Mx1 vector with K non-zero elements whose values are the corresponding c^* of step 2

$$c_i = \text{argmin}_{c_i} \| x_i - c_i^T b_i \|^2$$
subject to $\sum_{j} c_j = 1$

**Step 2:**
Reconstruct $x_i$ using $B_i$

**Step 1:**
Find K-Nearest Neighbors of $x_i$, denoted as $B_i$
Locality-constrained Linear Coding

\[ \text{SC} \]

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda |u_m| \\
\text{s.t.} \quad \|v_k\| \leq 1, \quad \forall k
\]
Locality-constrained Linear Coding

**SC**

\[
\min_{V, U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda |u_m|
\]

s.t. \(\|v_k\| \leq 1, \quad \forall k\)

**LLC**

\[
\min_{V, U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda \|d_m \odot u_m\|^2
\]

s.t. \(1^T u_m = 1, \quad \forall m\)
Locality-constrained Linear Coding

**SC**

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda |u_m| \\
\text{s.t. } \|v_k\| \leq 1, \quad \forall k
\]

**LLC**

\[
\min_{V,U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda \|d_m \odot u_m\|^2 \\
\text{s.t. } 1^\top u_m = 1, \quad \forall m
\]

locality adaptor: \(d_m = \exp\left(\frac{\text{dist}(x_m, V)}{\sigma}\right)\)

where

\[
\text{dist}(x_m, V) = [\text{dist}(x_m, v_1), \ldots, \text{dist}(x_m, v_K)]^\top
\]

\(\text{dist}(x_m, v_k)\) is the Euclidean distance between \(x_m\) and \(v_k\)

\(\sigma\) is for adjusting the decay speed
Properties of LLC

• Better reconstruction

• Local smooth sparsity

• Analytical solution

Image credit: Wang et al 2010
Properties of LLC

- Better reconstruction
- Local smooth sparsity
- Analytical solution

\[ \hat{u}_m = \left( (V - 1x_m^T)(V - 1x_m^T)^T + \lambda \text{diag}(d) \right) \backslash 1 \]
\[ u_m = \hat{u}_m / 1^T \hat{u}_m \]
Approximated LLC for Fast Encoding

\[
\min_{V, U} \sum_{m=1}^{M} \|x_m - u_m V\|^2 + \lambda \|d_m \odot u_m\|^2
\]

s.t. \(1^\top u_m = 1, \quad \forall m\)
Approximated LLC for Fast Encoding

\[
\min_{V, U} \sum_{m=1}^{M} \| x_m - u_m V \|^2 + \lambda \| d_m \odot u_m \|^2 \\
\text{s.t.} \quad 1^T u_m = 1, \quad \forall m
\]

Select \textit{local bases} of each descriptor to form a \textit{local coordinate system}

\[
\min_{\tilde{U}} \sum_{m=1}^{M} \| x_m - \tilde{u}_m V_m \|^2 \\
\text{s.t.} \quad 1^T \tilde{u}_m = 1, \quad \forall m
\]
Approximated LLC for Fast Encoding

\[
\min_{\hat{V}, \hat{U}} \sum_{m=1}^{M} \|x_m - \hat{u}_m \hat{V}\|^2 + \lambda \|d_m \odot \hat{u}_m\|^2
\]

s.t. \(1^\top \hat{u}_m = 1, \quad \forall m\)

Select **local bases** of each descriptor to form a **local coordinate system**

the \(K\) nearest neighbors of \(x_m\) forms the local basis \(V_m\)

\[
\min_{\tilde{U}} \sum_{m=1}^{M} \|x_m - \tilde{u}_m V_m\|^2
\]

s.t. \(1^\top \tilde{u}_m = 1, \quad \forall m\)
APPROACHES

- Bag of Words
- Spatial Pyramid Matching
- Descriptor Encoding
  - Linear SPM using Sparse Coding
  - Locality-constrained Linear Coding
  - Fisher Vector
Fisher Kernel

\begin{align*}
X = \{x_1, \ldots, x_T\}: \text{a sample of } T \text{ observations } x_t \in \mathcal{X} \\
u_\lambda: \text{the pdf} \\
\lambda = [\lambda_1, \ldots, \lambda_M]^{\top} \in \mathbb{R}^M
\end{align*}

core function:

\begin{align*}
G_\lambda^X = \nabla_\lambda \log u_\lambda(X)
\end{align*}

similarity measurement: [Jaakkola and Haussler, 1998]

\begin{align*}
K_{FK}(X, Y) = G_\lambda^X \top F_\lambda^{-1} G_\lambda^Y
\end{align*}
Fisher Kernel

similarity measurement:

\[ K_{FK}(X, Y) = G^X_\lambda F^{-1}_\lambda G^Y_\lambda \]

Fisher Information Matrix:

\[ F_\lambda = E_{x \sim u_\lambda} \left( G^X_\lambda G^X_\lambda^\top \right) \quad F^{-1}_\lambda = L^\top_\lambda L_\lambda \]
**Fisher Kernel**

**similarity measurement:**

\[ K_{FK}(X, Y) = G_X^\top F_\lambda^{-1} G_Y \]

**Fisher Information Matrix:**

\[ F_\lambda = E_{x \sim u_\lambda} (G_X^\lambda G_X^{\lambda \top}) \quad F_\lambda^{-1} = L_\lambda^\top L_\lambda \]

**Fisher Kernel re-written as:**

\[ K_{FK}(X, Y) = G_X^{\lambda \top} G_Y^\lambda \]

where \( G_X^\lambda = L_\lambda \nabla_\lambda \log u_\lambda(X) \) **Fisher Vector**
Fisher Vector on Images

Fisher Vector:

\[
G^X_\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t)
\]

GMM:

\[
u_\lambda(x) = \sum_{k=1}^{K} w_k u_k(x)
\]

where \( u_k(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) \right\} \)
Fisher Vector on Images

Fisher Vector:

\[ G_X^\lambda = \sum_{t=1}^T L_\lambda \nabla_\lambda \log u_\lambda(x_t) \]

GMM:

\[ u_\lambda(x) = \sum_{k=1}^K w_k u_k(x) \]

where \[ u_k(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) \right\} \]

EM algorithm to estimate the parameters:

\[ \lambda = \{ w_k, \mu_k, \Sigma_k, k = 1, \ldots, K \} \]
Soft Assignment

Fisher Vector:

\[ G^X_\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t) \]

Gradients:

\[ \nabla_{\alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k \]

\[ \nabla_{\mu_k} \log u_\lambda(x_t) = \gamma_t(k) \left( \frac{x_t - \mu_t}{\sigma_k^2} \right) \]

\[ \nabla_{\sigma_k} \log u_\lambda(x_t) = \gamma_t(k) \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^4} - \frac{1}{\sigma_k} \right] \]

Monday, April 7, 14
Soft Assignment

Fisher Vector:

\[ G_X^\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t) \]

Gradients:

\[ \nabla_{\alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k \]
\[ \nabla_{\mu_k} \log u_\lambda(x_t) = \gamma_t(k) \left( \frac{x_t - \mu_t}{\sigma_k^2} \right) \]
\[ \nabla_{\sigma_k} \log u_\lambda(x_t) = \gamma_t(k) \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^4} - \frac{1}{\sigma_k} \right] \]

Posterior Probability:

\[ \gamma_t(k) = \frac{w_k u_k(x_k)}{\sum_{j=1}^{K} w_j u_j(x_t)} \]
Soft Assignment

Fisher Vector:

\[ G_{\lambda}^{x} = \sum_{t=1}^{T} L_{\lambda} \nabla_{\lambda} \log u_{\lambda}(x_{t}) \]

Gradients:

\[ \nabla_{\alpha_{k}} \log u_{\lambda}(x_{t}) = \gamma_{t}(k) - w_{k} \]

\[ \nabla_{\mu_{k}} \log u_{\lambda}(x_{t}) = \gamma_{t}(k) \left( \frac{x_{t} - \mu_{t}}{\sigma_{k}^{2}} \right) \]

\[ \nabla_{\sigma_{k}} \log u_{\lambda}(x_{t}) = \gamma_{t}(k) \left[ \frac{(x_{t} - \mu_{k})^{2}}{\sigma_{k}^{4}} - \frac{1}{\sigma_{k}} \right] \]

\[ w_{k} = \frac{\exp(\alpha_{k})}{\sum_{j=1}^{K} \exp(\alpha_{j})} \]
Soft Assignment

Fisher Vector:

\[
G^X_\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t)
\]

Gradients:

\[
\nabla_{\alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k
\]

\[
\nabla_{\mu_k} \log u_\lambda(x_t) = \gamma_t(k) \left( \frac{x_t - \mu_t}{\sigma_k^2} \right)
\]

\[
\nabla_{\sigma_k} \log u_\lambda(x_t) = \gamma_t(k) \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^4} - \frac{1}{\sigma_k} \right]
\]
Soft Assignment

Fisher Vector:

\[ G^X_\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t) \]

Fisher Information Matrix:

\[ F_\lambda = E_{x \sim u_\lambda} (G^X_\lambda G^X_\lambda^\top) \quad F^{-1}_\lambda = L_\lambda^\top L_\lambda \]
Soft Assignment

**Fisher Vector:**


g^X_\lambda = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t)

**Fisher Information Matrix:**

\[ F_\lambda = E_{x \sim u_\lambda} (G^X_\lambda G^X_\lambda^\top) \quad F^{-1}_\lambda = L_\lambda^\top L_\lambda \]

Assume almost hard assignment 

FIM diagonal 

coordinate-wise normalization on gradient vectors
Fisher Vector

Assume almost hard assignment

FIM diagonal coordinate-wise normalization on gradient vectors

Normalized Gradients:

\[ G^X_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \]

\[ G^X_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{x_t - \mu_k}{\sigma_k} \right) \]

\[ G^X_{\sigma_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1 \right] \]

Concatenate the gradient vectors:
Dimension = \((2D+1)K\)
Fisher Vector

Assume almost hard assignment

coordinate-wise normalization on gradient vectors

Normalized Gradients:

\[ G^X_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \]

\[ G^X_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{x_t - \mu_k}{\sigma_k} \right) \]

\[ G^X_{\sigma_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1 \right] \]

Concatenate the gradient vectors:

Dimension = \((2D+1)K\)

\[ G^X_{\lambda} \leftarrow \frac{1}{T} G^X_{\lambda} \]

\(T\) : patch size
Algorithm 1 Compute Fisher vector from local descriptors

Input:
- Local image descriptors $X = \{x_t \in \mathbb{R}^D, t = 1, \ldots, T\}$,
- Gaussian mixture model parameters $\lambda = \{w_k, \mu_k, \sigma_k, k = 1, \ldots, K\}$

Output:
- normalized Fisher Vector representation $\mathcal{F}^X_{\lambda} \in \mathbb{R}^{K(2D+1)}$

1. Compute statistics
   - For $k = 1, \ldots, K$ initialize accumulators
     - $S_k^0 \leftarrow 0$, $S_k^1 \leftarrow 0$, $S_k^2 \leftarrow 0$
   - For $t = 1, \ldots, T$
     - Compute $\gamma_t(k)$ using equation (15)
     - For $k = 1, \ldots, K$:
       - $S_k^0 \leftarrow S_k^0 + \gamma_t(k)$,
       - $S_k^1 \leftarrow S_k^1 + \gamma_t(k)x_t^i$,
       - $S_k^2 \leftarrow S_k^2 + \gamma_t(k)x_t^2$

2. Compute the Fisher vector signature
   - For $k = 1, \ldots, K$:
     - $\mathcal{F}_{\alpha_k}^X = \left( S_k^0 - T w_k \right) / \sqrt{w_k}$
     - $\mathcal{F}_{\mu_k}^X = \left( S_k^2 - \mu_k S_k^0 \right) / \left( \sqrt{w_k \sigma_k} \right)$
     - $\mathcal{F}_{\sigma_k}^X = \left( S_k^2 - 2 \mu_k S_k^0 + (\mu_k^2 - \sigma_k^2) S_k^0 \right) / \left( \sqrt{2 w_k \sigma_k^2} \right)$
   - Concatenate all Fisher vector components into one vector
     $\mathcal{F}^X_{\lambda} = \left( \mathcal{F}_{\alpha_1}^X, \ldots, \mathcal{F}_{\alpha_K}^X, \mathcal{F}_{\mu_1}^X, \ldots, \mathcal{F}_{\mu_K}^X, \mathcal{F}_{\sigma_1}^X, \ldots, \mathcal{F}_{\sigma_K}^X \right)$

3. Apply normalizations
   - For $i = 1, \ldots, K (2D + 1)$ apply power normalization
     - $\left[ \mathcal{F}^X_{\lambda} \right]_i \leftarrow \text{sign} \left[ \left[ \mathcal{F}^X_{\lambda} \right]_i \right] \sqrt{\left| \left[ \mathcal{F}^X_{\lambda} \right]_i \right|}$
   - Apply $\ell_2$-normalization:
     $\mathcal{F}^X_{\lambda} = \mathcal{F}^X_{\lambda} / \sqrt{\mathcal{F}^X_{\lambda}' \mathcal{F}^X_{\lambda}}$

$\gamma_t(k) = \frac{w_k u_k(x_t)}{\sum_{j=1}^{K} w_j u_j(x_t)}$

To make it work with linear classifier

Image credit: Sanchez et al. 2013
Extension on FV

- Spatial Pyramid
  
  [Sanchez et al 2013]

- Deep Fisher Networks
  
  [Simonyan et al 2013]

- Other methods account scene geometry in FV framework
  
Extension on FV

• Spatial Pyramid
  [Sanchez et al, 2013]

• Deep Fisher Networks
  [Simonyan et al, 2013]

• Other methods account scene geometry in FV framework
Extension on FV

• Spatial Pyramid
  [Sanchez et al 2013]

• Deep Fisher Networks
  [Simonyan et al 2013]

• Other methods account scene geometry in FV framework
Deep Fisher Networks

One vs. rest linear SVMs

SSR & $L_2$ norm.
FV encoder

$L_2$ norm. & PCA
Spatial stacking
FV encoder

Dense feature extraction
SIFT, raw patches, ...

input image

classifier layer

1-st Fisher layer
(with optional global pooling branched out)

2-nd Fisher layer
(global pooling)

SSR & $L_2$ norm.
FV encoder

One vs rest linear SVMs

Image credit: Simonyan et al 2013

Monday, April 7, 14
Single Fisher Layer

Compressed semi-local Fisher vector encoding

Spatial stacking (2×2)

L₂ norm. & PCA

projection Uᵢ from 4hᵢ to dᵢ₊₁

projection Wᵢ from 2Kᵢdᵢ to hᵢ

mixture of Kᵢ Gaussians

Image credit: Simonyan et al 2013
Single Fisher Layer

more details: Deep Fisher Networks for Large-Scale Image Classification

Image credit: Simonyan et al 2013
# Evaluations - on Pascal VOC 2007

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) FK</td>
<td>Lin ss3</td>
<td>256</td>
<td>61.69</td>
<td>78.97</td>
<td>67.43</td>
<td>51.94</td>
<td>70.92</td>
<td>30.79</td>
</tr>
<tr>
<td>(b) SV</td>
<td>Lin ss3</td>
<td>1024</td>
<td>58.16</td>
<td>74.32</td>
<td>63.79</td>
<td>47.02</td>
<td>69.44</td>
<td>29.06</td>
</tr>
<tr>
<td>(c) LLC</td>
<td>Lin ss2</td>
<td>25k</td>
<td>57.60</td>
<td>71.05</td>
<td>62.85</td>
<td>47.40</td>
<td>67.67</td>
<td>25.21</td>
</tr>
<tr>
<td>(d) LLC-F</td>
<td>Lin ss2</td>
<td>25k</td>
<td>59.32</td>
<td>74.10</td>
<td>64.92</td>
<td>51.48</td>
<td>68.33</td>
<td>27.18</td>
</tr>
<tr>
<td>(e) VQ</td>
<td>Chi ss2</td>
<td>25k</td>
<td>56.07</td>
<td>70.00</td>
<td>58.90</td>
<td>42.86</td>
<td>66.75</td>
<td>26.59</td>
</tr>
<tr>
<td>(f) LLC</td>
<td>Lin ss3</td>
<td>14k</td>
<td>57.27</td>
<td>71.35</td>
<td>62.65</td>
<td>46.12</td>
<td>68.98</td>
<td>26.04</td>
</tr>
<tr>
<td>(g) LLC</td>
<td>Sqr ss3</td>
<td>25k</td>
<td>56.71</td>
<td>71.24</td>
<td>61.75</td>
<td>42.73</td>
<td>68.21</td>
<td>25.85</td>
</tr>
<tr>
<td>(h) LLC</td>
<td>Chi ss3</td>
<td>14k</td>
<td>57.66</td>
<td>72.41</td>
<td>62.19</td>
<td>47.30</td>
<td>68.91</td>
<td>25.78</td>
</tr>
<tr>
<td>(i) LLC-F</td>
<td>Lin ss5</td>
<td>25k</td>
<td>59.74</td>
<td>74.17</td>
<td>65.39</td>
<td>51.15</td>
<td>69.69</td>
<td>28.67</td>
</tr>
<tr>
<td>(j) VQ</td>
<td>Chi ss3</td>
<td>14k</td>
<td>55.30</td>
<td>70.10</td>
<td>59.24</td>
<td>44.14</td>
<td>66.34</td>
<td>26.79</td>
</tr>
<tr>
<td>(k) KCB</td>
<td>Chi ss3</td>
<td>14k</td>
<td>56.26</td>
<td>70.83</td>
<td>60.60</td>
<td>44.50</td>
<td>66.52</td>
<td>27.02</td>
</tr>
<tr>
<td>(l) LLC</td>
<td>Lin ss5</td>
<td>25k</td>
<td>56.96</td>
<td>69.82</td>
<td>61.63</td>
<td>46.71</td>
<td>68.27</td>
<td>25.66</td>
</tr>
<tr>
<td>(m) LLC-F</td>
<td>Lin ss5</td>
<td>25k</td>
<td>58.70</td>
<td>73.44</td>
<td>62.90</td>
<td>50.22</td>
<td>67.90</td>
<td>27.85</td>
</tr>
<tr>
<td>(n) VQ</td>
<td>Chi ss3</td>
<td>25k</td>
<td>53.87</td>
<td>68.74</td>
<td>57.14</td>
<td>41.24</td>
<td>64.54</td>
<td>25.20</td>
</tr>
<tr>
<td>(o) LLC</td>
<td>Lin ss3</td>
<td>14k</td>
<td>56.18</td>
<td>70.71</td>
<td>59.67</td>
<td>44.81</td>
<td>67.20</td>
<td>26.03</td>
</tr>
<tr>
<td>(p) VQ</td>
<td>Chi ss3</td>
<td>14k</td>
<td>54.82</td>
<td>69.09</td>
<td>58.61</td>
<td>41.27</td>
<td>66.30</td>
<td>26.49</td>
</tr>
<tr>
<td>(q) LLC</td>
<td>Lin ss3</td>
<td>10k</td>
<td>56.01</td>
<td>69.66</td>
<td>60.44</td>
<td>44.21</td>
<td>67.78</td>
<td>24.66</td>
</tr>
<tr>
<td>(r) VQ</td>
<td>Chi ss3</td>
<td>10k</td>
<td>54.98</td>
<td>69.56</td>
<td>57.97</td>
<td>42.86</td>
<td>65.84</td>
<td>23.52</td>
</tr>
<tr>
<td>(s) LLC</td>
<td>Lin ss3</td>
<td>4k</td>
<td>53.79</td>
<td>69.83</td>
<td>57.63</td>
<td>42.04</td>
<td>66.46</td>
<td>22.44</td>
</tr>
<tr>
<td>(t) LLC</td>
<td>Sqr ss3</td>
<td>4k</td>
<td>52.07</td>
<td>68.52</td>
<td>54.62</td>
<td>40.14</td>
<td>65.34</td>
<td>21.53</td>
</tr>
<tr>
<td>(u) LLC</td>
<td>Chi ss3</td>
<td>4k</td>
<td>53.47</td>
<td>70.17</td>
<td>56.20</td>
<td>42.73</td>
<td>65.27</td>
<td>22.23</td>
</tr>
<tr>
<td>(v) LLC-F</td>
<td>Lin ss3</td>
<td>4k</td>
<td>36.06</td>
<td>53.39</td>
<td>43.20</td>
<td>22.47</td>
<td>46.32</td>
<td>11.40</td>
</tr>
<tr>
<td>(w) LLC-F</td>
<td>Lin ss3</td>
<td>4k</td>
<td>55.87</td>
<td>72.27</td>
<td>61.41</td>
<td>44.08</td>
<td>67.85</td>
<td>24.97</td>
</tr>
<tr>
<td>(x) VQ</td>
<td>Sqr ss3</td>
<td>4k</td>
<td>51.97</td>
<td>67.29</td>
<td>55.22</td>
<td>36.58</td>
<td>64.42</td>
<td>21.89</td>
</tr>
<tr>
<td>(y) VQ</td>
<td>Chi ss3</td>
<td>4k</td>
<td>53.42</td>
<td>68.65</td>
<td>57.04</td>
<td>39.86</td>
<td>64.59</td>
<td>21.96</td>
</tr>
<tr>
<td>(z) VQ</td>
<td>Lin ss3</td>
<td>4k</td>
<td>46.54</td>
<td>60.63</td>
<td>48.80</td>
<td>32.76</td>
<td>58.54</td>
<td>16.26</td>
</tr>
<tr>
<td>(α) KCB</td>
<td>Chi ss3</td>
<td>4k</td>
<td>54.60</td>
<td>69.82</td>
<td>59.20</td>
<td>41.97</td>
<td>64.85</td>
<td>23.90</td>
</tr>
</tbody>
</table>

Table 1: Image classification results using Pascal VOC 2007 dataset (continued on next page)

VQ – baseline method; FK – Fisher kernel; SV – super vector coding; KCB – kernel codebook; LLC – locally-constrained linear coding; LLC-F – LLC encoding with original-left-right flipped training images; LLC-1 – L1-normalized LLC encoding; Lin/Sqr/Chi – linear/hellinger/$\chi^2$ kernel map; third column: SIFT sampling density; fourth column: visual vocabulary size
# Evaluations - on Pascal VOC 2007

<table>
<thead>
<tr>
<th>Category</th>
<th>FK</th>
<th>SV</th>
<th>LLC</th>
<th>LLC-F</th>
<th>VQ</th>
<th>KCB</th>
<th>LLC</th>
<th>LLC-F</th>
<th>VQ</th>
<th>LLC</th>
<th>LLC-F</th>
<th>VQ</th>
<th>KCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>55.98</td>
<td>50.19</td>
<td>54.24</td>
<td>54.41</td>
<td>53.80</td>
<td>53.96</td>
<td>53.91</td>
<td>54.25</td>
<td>55.31</td>
<td>53.12</td>
<td>54.31</td>
<td>53.17</td>
<td>54.12</td>
</tr>
<tr>
<td>Cow</td>
<td>49.61</td>
<td>46.46</td>
<td>45.27</td>
<td>47.24</td>
<td>41.35</td>
<td>46.34</td>
<td>45.63</td>
<td>45.97</td>
<td>49.52</td>
<td>40.40</td>
<td>41.64</td>
<td>44.89</td>
<td>47.33</td>
</tr>
<tr>
<td>Sheep</td>
<td>58.40</td>
<td>51.86</td>
<td>51.56</td>
<td>52.75</td>
<td>51.55</td>
<td>52.10</td>
<td>49.61</td>
<td>51.11</td>
<td>53.03</td>
<td>51.27</td>
<td>51.91</td>
<td>50.70</td>
<td>52.31</td>
</tr>
<tr>
<td>Dog</td>
<td>44.77</td>
<td>44.07</td>
<td>44.24</td>
<td>44.55</td>
<td>42.57</td>
<td>42.39</td>
<td>42.91</td>
<td>43.22</td>
<td>44.60</td>
<td>39.16</td>
<td>41.05</td>
<td>40.70</td>
<td>42.90</td>
</tr>
<tr>
<td>Horse</td>
<td>78.84</td>
<td>77.85</td>
<td>77.52</td>
<td>78.10</td>
<td>76.33</td>
<td>77.17</td>
<td>75.50</td>
<td>76.74</td>
<td>78.52</td>
<td>76.87</td>
<td>76.40</td>
<td>76.88</td>
<td>77.69</td>
</tr>
<tr>
<td>Motorcycle</td>
<td>70.81</td>
<td>67.12</td>
<td>67.05</td>
<td>68.48</td>
<td>65.11</td>
<td>67.15</td>
<td>65.93</td>
<td>67.07</td>
<td>68.85</td>
<td>63.09</td>
<td>66.06</td>
<td>66.84</td>
<td>68.39</td>
</tr>
<tr>
<td>Plant</td>
<td>84.96</td>
<td>83.07</td>
<td>83.29</td>
<td>83.73</td>
<td>82.98</td>
<td>83.36</td>
<td>83.03</td>
<td>83.49</td>
<td>84.24</td>
<td>82.16</td>
<td>82.97</td>
<td>82.87</td>
<td>83.28</td>
</tr>
<tr>
<td>Bear</td>
<td>31.72</td>
<td>27.56</td>
<td>27.57</td>
<td>29.90</td>
<td>26.70</td>
<td>23.11</td>
<td>27.10</td>
<td>27.73</td>
<td>28.44</td>
<td>20.96</td>
<td>23.86</td>
<td>26.19</td>
<td>27.43</td>
</tr>
<tr>
<td>Cat</td>
<td>51.00</td>
<td>48.50</td>
<td>45.73</td>
<td>50.95</td>
<td>43.47</td>
<td>44.45</td>
<td>43.82</td>
<td>44.77</td>
<td>50.47</td>
<td>42.96</td>
<td>43.66</td>
<td>45.46</td>
<td>50.06</td>
</tr>
<tr>
<td>Bottle</td>
<td>56.41</td>
<td>51.10</td>
<td>53.62</td>
<td>55.48</td>
<td>52.18</td>
<td>52.12</td>
<td>52.28</td>
<td>52.77</td>
<td>55.18</td>
<td>51.40</td>
<td>51.32</td>
<td>51.27</td>
<td>54.41</td>
</tr>
<tr>
<td>Bus</td>
<td>80.24</td>
<td>75.50</td>
<td>76.01</td>
<td>78.55</td>
<td>74.27</td>
<td>75.36</td>
<td>74.71</td>
<td>76.00</td>
<td>77.56</td>
<td>75.13</td>
<td>75.10</td>
<td>75.11</td>
<td>77.36</td>
</tr>
<tr>
<td>Truck</td>
<td>57.46</td>
<td></td>
<td>52.32</td>
<td>53.62</td>
<td>50.87</td>
<td></td>
<td>51.86</td>
<td>54.27</td>
<td>54.17</td>
<td>50.94</td>
<td>51.40</td>
<td>51.40</td>
<td>52.03</td>
</tr>
</tbody>
</table>

Results from: Chatfield et al 2011
## Evaluations - on SUN 397

<table>
<thead>
<tr>
<th>Method</th>
<th>ntrain=5</th>
<th>ntrain=10</th>
<th>ntrain=20</th>
<th>ntrain=50</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Xiao et al. (2010)</strong></td>
<td>14.5</td>
<td>20.9</td>
<td>28.1</td>
<td>38.0</td>
</tr>
<tr>
<td>FV (SIFT)</td>
<td>19.2 (0.4)</td>
<td>26.6 (0.4)</td>
<td>34.2 (0.3)</td>
<td>43.3 (0.2)</td>
</tr>
<tr>
<td>FV (SIFT + LCS)</td>
<td>21.1 (0.3)</td>
<td>29.1 (0.3)</td>
<td>37.4 (0.3)</td>
<td>47.2 (0.2)</td>
</tr>
</tbody>
</table>

*Table 5* Comparison of the FV with the state-of-the-art on SUN 397

Image credit: Sanchez et al. 2013
BoW Summary

Issues

• Sampling strategy
dense uniform, interest points, random...

• Codebook learning
supervised/unsupervised, size...

• Similarity measurement
SVM, Pyramid Matching

• Spatial information

• Scalability
From Vectors to Codes

Given global image representation, want to learn compact binary codes for image retrieval task on large dataset
From Vectors to Codes

Motivation

• Tractable memory usage
• Constant lookup time
• Similarity preserved by hamming distance
From Vectors to Codes

Problem definition:

Given a database of images \( \{x_i\} \) and a distance function \( D(i, j) \) seek a binary feature vector \( y_i = f(x_i) \) that preserves the nearest neighbor relationships using a Hamming distance.
What makes a good code

- Easily computed for a novel input
- Requires a small number of bits to code the full dataset
- Maps similar items to similar binary codewords
From Vectors to Codes

Approaches

• Locality Sensitive Hashing (LSH) [Andoni and Indyk]

• Boosting [Torralba et al 2008]

• Restricted Boltzmann Machines (RBM) [Torralba et al 2008]

• Spectral Hashing [Weiss et al 2008]

• Multidimensional Spectral Hashing [Weiss et al 2012]
Locality Sensitive Hashing

- Take random projections of data
- Quantize each projection with few bits

Adapted from slides by Weiss et al.

No learning involved
Boosting

- **Positive** examples are pairs of similar images
- **Negative** examples are pairs of unrelated images

Learn threshold & dimension for each bit (weak classifier)

Adapted from slides by Weiss et al.
Restricted Boltzmann machines

Single RBM layer

Hidden units

Symmetric weights $W$

Units are binary & stochastic

Visible units

Adapted from slides by Weiss et al
Restricted Boltzmann machines

Input Gist vector (512 dimensions)

Layer 1

Layer 2

Layer 3

Output binary code (N dimensional)

Linear units at first layer

Adapted from slides by Weiss et al.
Retrieval Algorithm: Semantic Hashing

Query Image

Semantic Hash Function

Binary code

Address Space

Images in database

Query address

Quite different to a (conventional) randomizing hash

Semantically similar images

Adapted from slides by Weiss et al
Spectral Hashing

Query Image

Non-linear dimensionality reduction

Real-valued vectors

Semantically similar images

Spectral Hash

Binary code

Address Space

Images in database

Query address

Quite different to a (conventional) randomizing hash

Adapted from slides by Weiss et al

Monday, April 7, 14
The Algorithm

Input: Data \{x_i\} of dimensionality d; desired # bits, k
The Algorithm

- Fit multi-dimensional rectangle

- Run PCA to align axes

- Bound uniform distribution

Adapted from slides by Weiss et al
The Algorithm

• Fit multi-dimensional rectangle

• Run PCA to align axes

• Bound uniform distribution

Adapted from slides by Weiss et al
The Algorithm

• Fit multi-dimensional rectangle

• Run PCA to align axes

• Bound uniform distribution

Adapted from slides by Weiss et al.

Monday, April 7, 14
The Algorithm

- Calculate Eigenfunctions

Adapted from slides by Weiss et al
The Algorithm

• Calculate Eigenfunctions

Adapted from slides by Weiss et al
The Algorithm

• Calculate Eigenfunctions
The Algorithm

• Calculate Eigenfunctions
The Algorithm

- Calculate Eigenfunctions

Adapted from slides by Weiss et al
The Algorithm

• Pick the $k$ smallest Eigenfunctions

$e.g. \ k=3$

Adapted from slides by Weiss et al
The Algorithm

- Threshold the chosen Eigenfunctions
The Algorithm

- Threshold the chosen Eigenfunctions

more details: Spectral Hashing, Yair Weiss, Antonio Torralba, Rob Fergus
Summary

Image representation

• Bag of Words

• Spatial Pyramid Matching

• Descriptor Encoding
  - Sparse Coding
  - Locality-constrained Linear Coding
  - Fisher Vector

• Binary Code
Thank you