

Improved Fisher Vector for Large Scale Image Classification

XRCE's participation for ILSVRC

Jorge Sánchez, Florent Perronnin and Thomas Mensink

Xerox Research Centre Europe (XRCE)

Overview

- Fisher Vector
- Improved FV + results on VOC 07
- Compression
- Classification
- Results on VOC2010 & ILSVRC2010

Fisher Vector

- Exploiting Generative Models in discriminative classifiers [Jaakkola & Haussler 1999]
- Feature vector is derivative wrt probabilistic model
- Measure Similarity using the Fisher Kernel

$$K(X, Y) = G_{\lambda}^{X'} F_{\lambda}^{-1} G_{\lambda}^Y$$

- Fisher Information Matrix

$$F_{\lambda} = E_{x \sim u_{\lambda}} [\nabla_{\lambda} \log u_{\lambda}(x) \nabla_{\lambda} \log u_{\lambda}(x)']$$

- Learning a classifier on Fisher Kernel equals learning a linear classifier on $\mathcal{G}_{\lambda}^X = L_{\lambda} G_{\lambda}^X$ with $F_{\lambda} = L_{\lambda}' L_{\lambda}$

Fisher Vector (2)

- Fisher Kernels on visual vocabularies for image categorization
[Perronnin & Dance 2007]
- $X = \{x_t, t = 1 \dots T\}$ D-dimensional local features from an image

- GMM:
$$u_\lambda(x) = \sum_{i=1}^K w_i u_i(x)$$

- Gradient:
$$\mathcal{G}_{\mu,i}^X = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^T \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right),$$
$$\mathcal{G}_{\sigma,i}^X = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^T \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right],$$

Fisher Vector (3)

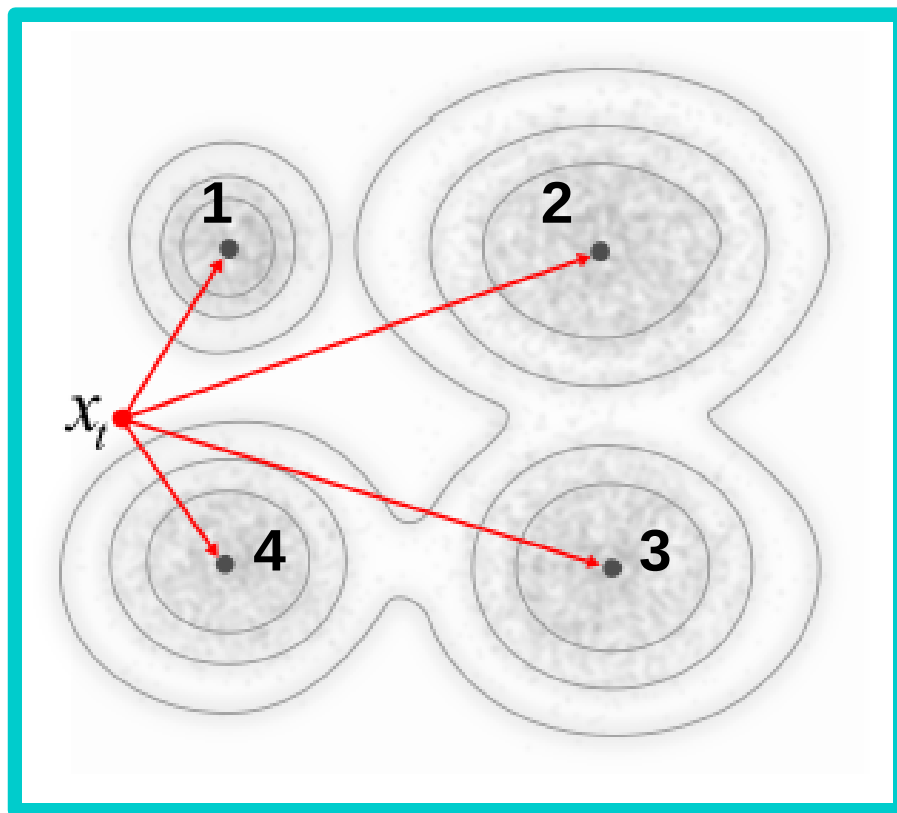
BOV

Hard Assignment

[0 0 0 1]

Soft Assignment

[.3 .1 .1 .5]



Fisher Vector

Gradient wrt w

[.15 -.2 -.35 .2]

Gradient wrt mean

[.8 -1.5 -3.7 -1.3 -3.8 1.2 -.9 1.4]

Gradient wrt var

[-1.2 -.9 1.4 -.8 1.5 -3.7 1.3 -3.8]

BOV Histogram has size: K

Fisher Vector (wrt to mean and var): $2 * D * K$

Improving the Fisher Vector

- L2 Normalization
- Power Normalization
- Spatial Pyramid Matching

Improving the Fisher Kernel for Large-Scale Image Classification

Florent Perronnin, Jorge Sánchez and Thomas Mensink
Xerox Research Centre Europe (XRC-E)
0000-0001-1111-1111, 1111-1111-1111-1111, 1111-1111-1111-1111



Summary

- The Fisher kernel (FK) combines the benefits of generative and discriminative approaches and extends the popular bag-of-visual-words (BOV) by going beyond count statistics.
- However in image classification BOV still outperforms FK.
- We improve the FK, by L2 normalization, power normalization and spatial pyramids.
- On PASCAL VOC 2007 we increase the Average Precision from 47.9% to 50.3%, using these improvements. Using only SIFT descriptors and linear classifiers.
- Large scale experiments: we learn classifiers from large datasets obtained from ImageNet and Flickr groups.
- Although not intended for that purpose, Flickr groups are great for training classifiers.
- Combining all these resources, using only SIFT features and linear classifiers, we obtain 51.0% Average Precision on PASCAL VOC 2007, which equals the current-state-of-the-art.

L2 Normalisation

- We can write the IV $\{u_i\}_{i=1}^K$ as:

$$C_i^T = V_i \int p(x) \log u_i(x) dx \quad (1)$$
- Decompose p into two parts: a background $\{u_b\}$ with λ estimated to maximize $\int u_b \log u_b(x) dx$ and an image specific part $\{p\}$:

$$C_i^T = \lambda V_i \int u_b \log u_b(x) dx + (1 - \lambda) V_i \int p \log u_i(x) dx \quad (2)$$
- The Fisher vector focuses on image-specific content, but, depends on the proportion of image-specific information in S_0 , two images containing the same object at different scales will have different signatures.
- To remove the dependence on λ , we can L2-normalize C_i^T or equivalently u_i^T .

Experimental Setup

- Densely sampled local SIFT and color features, with PCA reduced to 64D.
- Train GMM with $K = 256$ using EM algorithm.
- Learn linear SVMs in primal using Stochastic Gradient Descent.
- Take (pre-) feature of SIFT and color features.

PASCAL VOC 2007

- Around 10K images of 20 classes, evaluated using mean Average Precision (mAP).
- Best results up to date in 51.0% mAP obtained by combined Classification and Localization approach [Lacah et al., 2009].
- The FK obtains 50.3% mAP, an improvement of over 10% compared to FK. FK is close to the best SIFT-only results (50.3%) of [Wang et al., 2010].
- Similarly, we demonstrate state-of-the-art accuracy on CalTech 256.

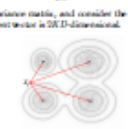
Table 1. Impact of the proposed modifications to the FK on PASCAL VOC 2007.

Mod.	L2	SP	Power	Cal	256
-	-	-	-	47.9	47.9
+	-	-	-	48.2	48.2
+	+	-	-	50.3	50.3
+	+	+	-	50.3	50.3
+	+	+	+	51.0	51.0

Fisher Kernel Framework

- Model a sample X of T i.i.d. local descriptors u_i by its deviation from a Gaussian mixture model $u_i(x) = \sum_{j=1}^K w_j u_j(x)$:

$$C_i^T = \frac{1}{T} \sum_{t=1}^T V_i \log u_i(x_t) \quad (1)$$
- Assume diagonal covariance matrix, and consider the gradient w.r.t. mean and covariance, then gradient vector is $2K \cdot D$ -dimensional.



- Measure the similarity using the Fisher Kernel:

$$K(X, Y) = C_i^T F_2^{-1} C_j^T$$
- with $F_2 = \text{Eve}_{u_i} [V_i \log u_i(x) V_i \log u_i(x)]$.
- This equals a dot-product on normalized Fisher Vectors (FV):

$$C_i^T = L_2 C_i^T \quad \text{with} \quad F_2^{-1} = L_2^T L_2 \quad (2)$$
- Learning a classifier on the Fisher Kernel is equivalent to learning a linear classifier on the Fisher Vectors C_i^T .

Power Normalization

- As the number of Gaussians increases, Fisher vector becomes sparse.
- The dot-product (L2 distance) is a poor measure of similarity on sparse vectors.
- Replace kernel with e.g. Laplacian kernel, or:
- 2."compress" the representation, advantage linear classification.
- Power Normalization to uniquely representation:

$$\tilde{f}(x) = \text{sign}(x) |x|^\alpha \quad (3)$$
- Parameter $\alpha: 0 < \alpha \leq 1$, optimal trade-off with the number K of Gaussians. But $\alpha = 0.5$ is a good value for $10 \leq K \leq 256$.
- When combined with L2 normalization, we first apply power normalization and then L2 normalization.



Figure 3. Effect of $K = \{10, 64, 256\}$ on sparsity and effect of power normalization.

Large-Scale Experiments

- ImageNet: 18 groups, up to 25K per class, total of 25K images.
- Flickr Groups: 18 groups, up to 25K per class, total of 25K images.

Table 2. Learning from different training resources using SIFT only. V+L+T is combination of the classifiers, evaluated on VOC 2007 "dev" set.

Resources	class	size	train	test	train	test	train	test	train	test
I	1	82.8	43.5	74.8	43.1	31.7	98.8	74.4	68.8	68.8
F	1	31.8	47.7	56.3	49.8	75.4	28.1	46.0	82.1	82.1
V	1	56.3	43.7	74.3	44.4	62.7	24.3	56.7	56.4	74.7
V+L+T	1	74.7	56.3	74.7	56.3	74.7	56.3	74.7	56.3	74.7
I	11	74.7	56.3	74.7	56.3	74.7	56.3	74.7	56.3	74.7
F	11	74.7	56.3	74.7	56.3	74.7	56.3	74.7	56.3	74.7
V	11	74.7	56.3	74.7	56.3	74.7	56.3	74.7	56.3	74.7
V+L+T	11	74.7	56.3	74.7	56.3	74.7	56.3	74.7	56.3	74.7

Spatial Pyramids

- Introduced by Leifert et al., 2006, to take into account the rough geometry.
- Follow the splitting of the training pyramid of INCAAL VOC 2006.
- Power normalisation even more important since FV becomes even sparser.

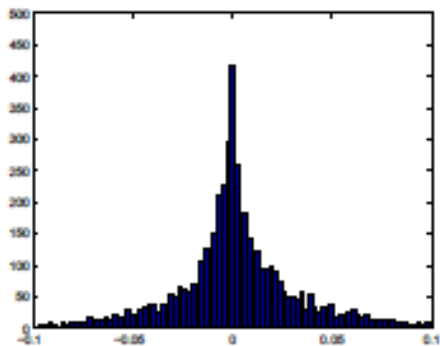


L2 Normalization

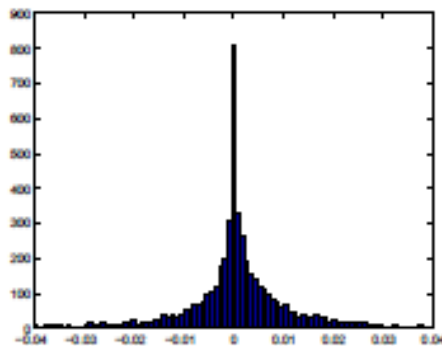
- By construction the Fisher Vector discards descriptors which are likely to occur in any image
- The FV **focus on image specific features**
- However, the FV depends on the **amount** of image specific information / background information
 - 2 images with same object on a different scale will have a different feature vector
- L2 Normalization to remove this dependence

Power Normalization

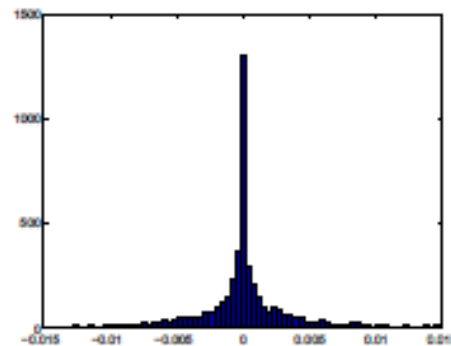
- As the number of Gaussians increase, the FV becomes sparser
 - Replace dot-product with other kernel
 - Unsparsify the representation
- Power normalization to unsparsify: $f(z) = \text{sign}(z)|z|^\alpha$



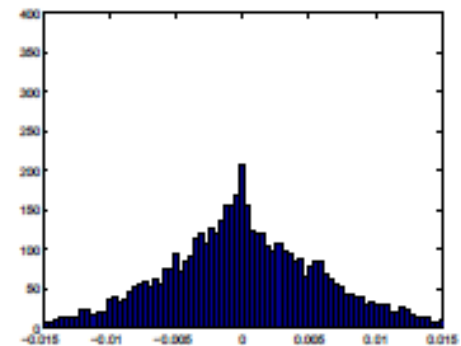
K = 16



K = 64



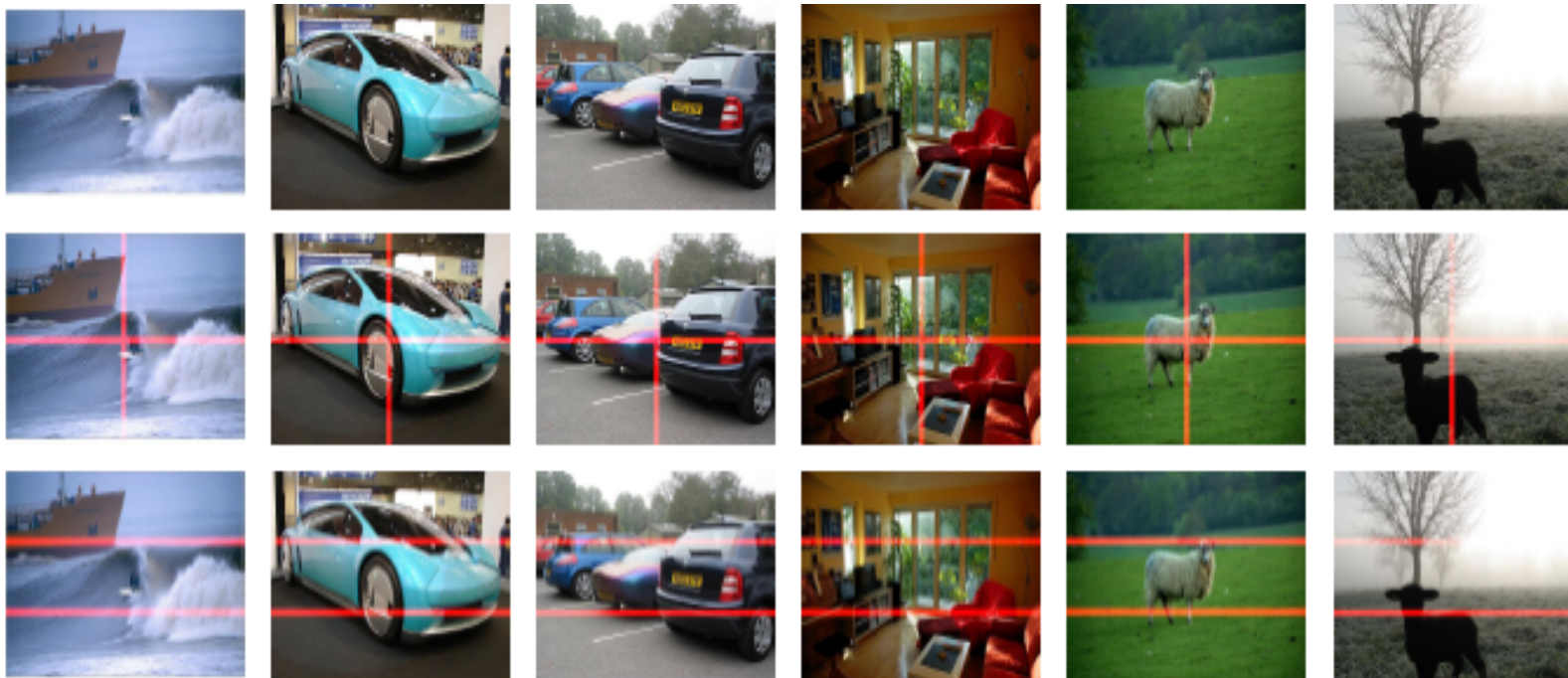
K = 256



K = 256
Power Normalized

Spatial Pyramids

- Take rough geometry into account [Lazebnik 2006]



- Power normalization becomes even more important (FV is sparser)

Experiments VOC 2007

- Improved Fisher Vector [ECCV 2010]
- Dense multiscale sampling, PCA, K=256
- Linear SVM

PN	L2	SP	SIFT	Col	S+C
-	-	-	47.9	34.2	45.9
✓	-	-	54.2	45.9	57.6
-	✓	-	51.8	40.6	53.9
-	-	✓	50.3	37.5	49.0
✓	✓	✓	58.3	50.9	60.3

Experiments VOC 2007 (2)

- Improved Fisher Vector [ECCV 2010]
- Larger Scale
 - Flickr Group Images
 - Up to 25k per class / 350k in total
 - Late fusion with VOC07 trainset
 - 63.3% (SIFT only)
- Best results 63.5% Localization and Classification [Harzallah et al. 2009]
- Flickr Groups are a great resource for labelled images
 - No additional labelling used!

11 More training data improves performance

So far...

- FV is a rich representation, extends BOV.
- High dimensional (2 D K S) but allows for linear SVM
- Performance is compatible to state of the art

However...

- $2 * 64 * 256 * 8 = 262,144$ dimensions
- Almost dense feature
- $\sim 1\text{MB}$ per image / per modality
- ImageNet Train/Test/Val $\rightarrow 1.4\text{ TB}$ (per modality)

Compression [unpublished]

Two options

A) Dimension Reduction

- PCA / Dense Random projections
 - is costly in high dimensional dense space
- Hash Kernels
 - Observation: performance decreases rapidly (already by factor 4)
- Can improve learning speed (not necessary)

B) Data Compression

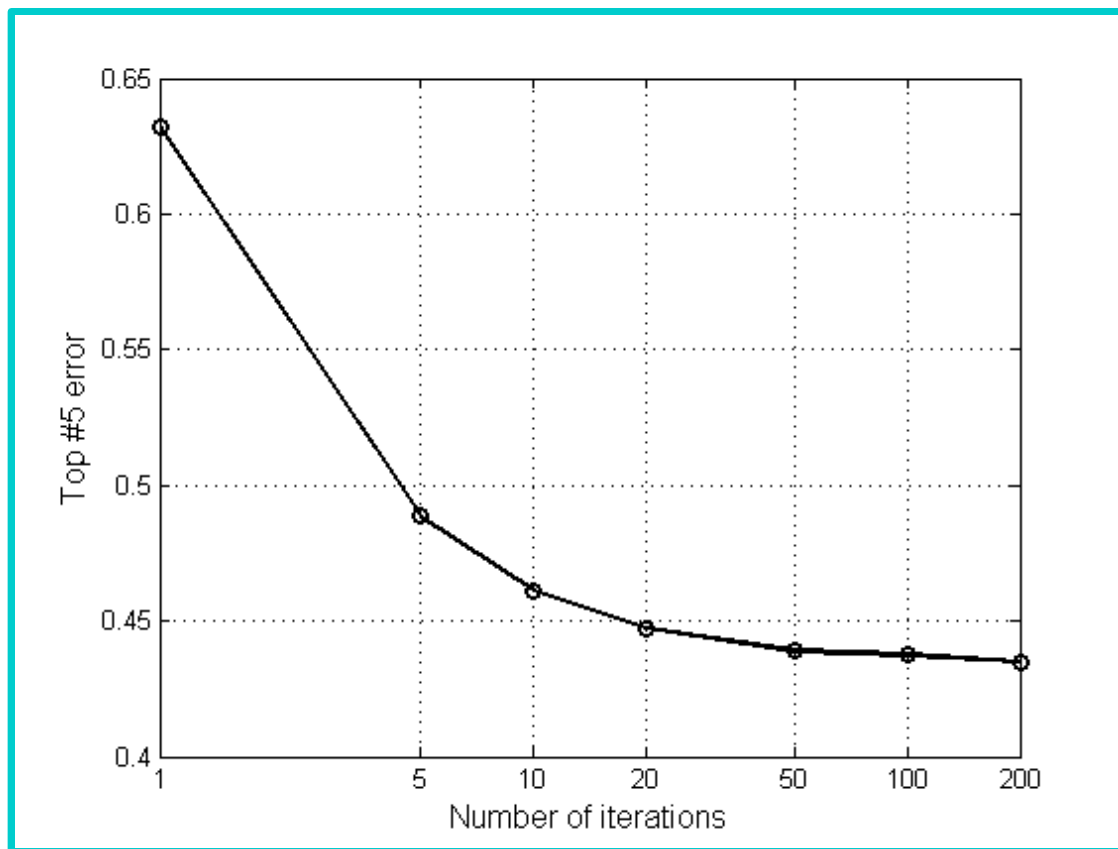
- Use same dimensionality
- But lossy compression up to factor 64 possible.
- 1.4TB → 20GB (per modality)
- Not able to learn in compressed space

Stochastic Gradient Descent (SGD)

- Learn Linear SVM in the primal, PEGASOS
[Shalev-Shwartz et al. 2007]
- SGD inspired on Pegasos by L. Bottou
[<http://leon.bottou.org>]
- Online algorithm, using one sample at the time
- Our approach is:
 1. Load compressed vector
 2. Decompress vector
 3. SGD Update

Stochastic Gradient Descent (2)

- Performance vs number of passes through data



Categorization Pipeline

- Extract dense sampled features (SIFT, Colour)
- Project (with PCA) to 64D
- Learn codebook with K (256) Gaussians on 1M features
- Learn Compressor (on small set of FV)
- Compute and compress FV
- Learn Linear Classifiers using SGD
- Classify test images

Categorization Pipeline (2)

- Computation time for Learning ImageNET

Intel Xeon double quadcore (16 proc) @ 2.53GHz, 32GB RAM

	CPU	Wall-Clock
Extract SIFT+Projection	36h	18h
GMM	minutes	
Learn Compressor	48h	3h
Extract FV + Compression	96h	6h
500 SGD Iterations*	960h	66h
Total (SIFT)	1140h	93h

1.2M train images, training ~ 4 CPU sec per image / modality

* Without significant loss of performance 500 → 50 iterations

Categorization Pipeline (3)

- Computation time for Testing on ImageNET

	CPU
Feature Extraction + Projection	2.5h
FV Extraction	30m
Classifiers	12h

- Total SIFT + Col = 30h
- 150K images / 1000 classes

Classification << 1ms per image/class/modality

Results

- Pascal VOC 2010 and ImageNet Challenge
- Same approach and settings
- $K = 256$
- FV + L2 & Power Norm + pyramids
- Compression
 - Except for VOC train/val set
- Linear SVM in primal
 - Number of SGD iterations are different

Pascal VOC 2010

- 10K test images / 20 classes / multi-label
- Challenge 1: Only provided train/val data

Rank		MAP
1	NUSPSL	73.8
2	NLPR	71.2
3	NEC	70.9
9	XRCE Improved FV	61.2

Pascal VOC 2010 (2)

- 10K test images / 20 classes / multi-label
- Challenge 2: Any data except test data
- 1M Flickr Group images of 18 classes
 - Tv/monitor and sofa are missing
 - No additional labelling, just the group labels

Rank		MAP	# Classes
4	BIT	26.9	20
3	UCI	51.7	9
2	XRCE Flickr 1M	65.5	18
1	XRCE Optimal Fuse F & V	68.3	20

21 More data helps: 61.2 → 68.3 MAP

ImageNet Challenge

- 150k test images / 1000 classes / single labelled.
 - Flat cost: is the correct label in the top 5?
 - Hierarchical cost: distance to lowest common ancestor

	flat cost	hie cost
NEC-UIUC	0.28191	2.1144
XRCE	0.33649	2.5553
ISIL	0.44558	3.6536
UCI	0.46624	3.6288

Conclusions

- Improved Fisher Vector for Image Classification
- Linear Classification → scales to **larger** scale
- More data (Flickr Groups) helps
 - Pascal VOC 2010: 61.2 → 68.3 MAP
- Using compression → scales to **LARGE** scale
- **Very very fast**: training (8s/i) & classifying (2s/i)

Questions?

L2 Normalization (2)

- Fisher Vector $G_{\lambda}^X = \frac{1}{T} \sum_{t=1}^T \nabla_{\lambda} \log u_{\lambda}(x_t)$ 1

$$G_{\lambda}^X \approx \nabla_{\lambda} E_{x \sim p} \log u_{\lambda}(x) = \nabla_{\lambda} \int_x p(x) \log u_{\lambda}(x) dx. \quad 2$$

$$G_{\lambda}^X \approx \omega \nabla_{\lambda} \int_x q(x) \log u_{\lambda}(x) dx + (1 - \omega) \nabla_{\lambda} \int_x u_{\lambda}(x) \log u_{\lambda}(x) dx \quad 3$$

- GMM Trained with Maximum Likelihood, ie maximize $E_{x \sim u_{\lambda}} \log u_{\lambda}(x)$

$$\nabla_{\lambda} \int_x u_{\lambda}(x) \log u_{\lambda}(x) dx = \nabla_{\lambda} E_{x \sim u_{\lambda}} \log u_{\lambda}(x) \approx 0$$

- Fisher Vectors automatically **focus on image specific features** and discard image independent/background features
- L2 Normalization to remove dependence on ω