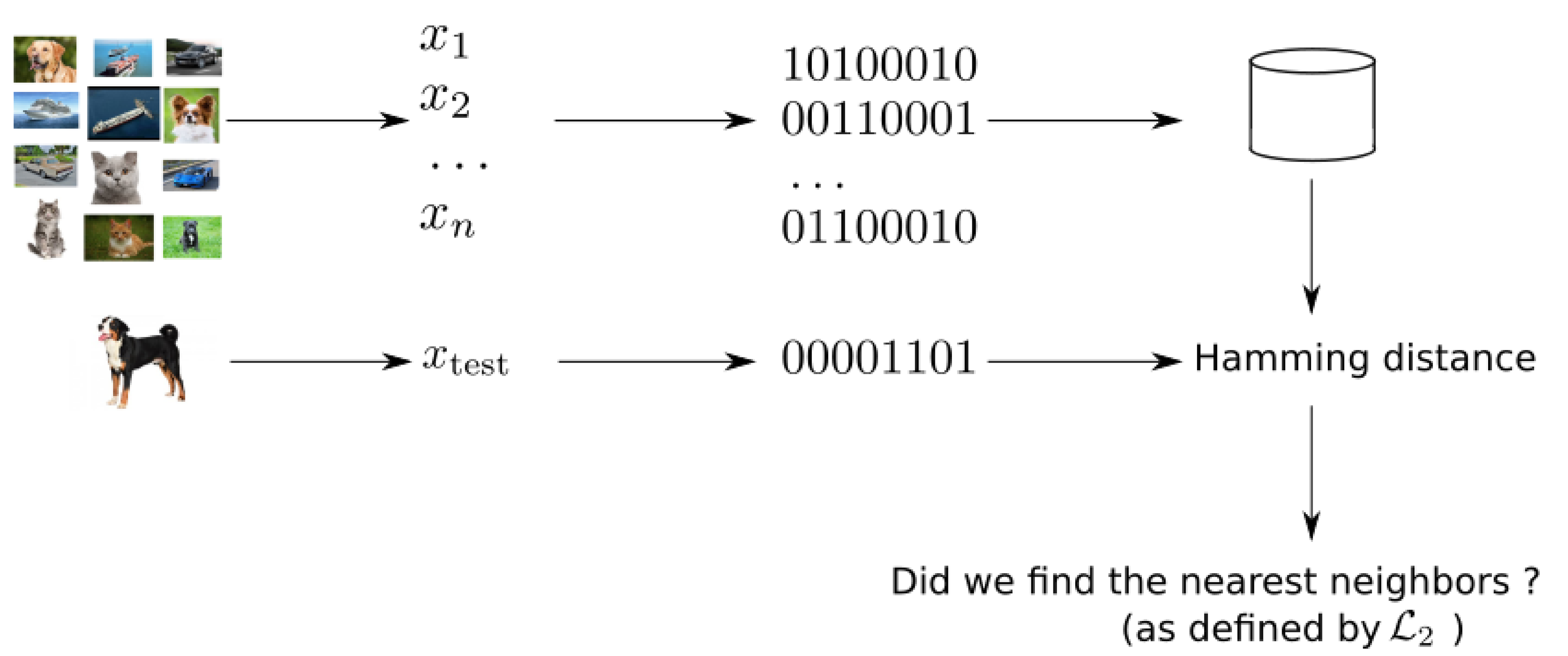
How should we evaluate supervised hashing?

Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, Hervé Jégou
Facebook AI Research

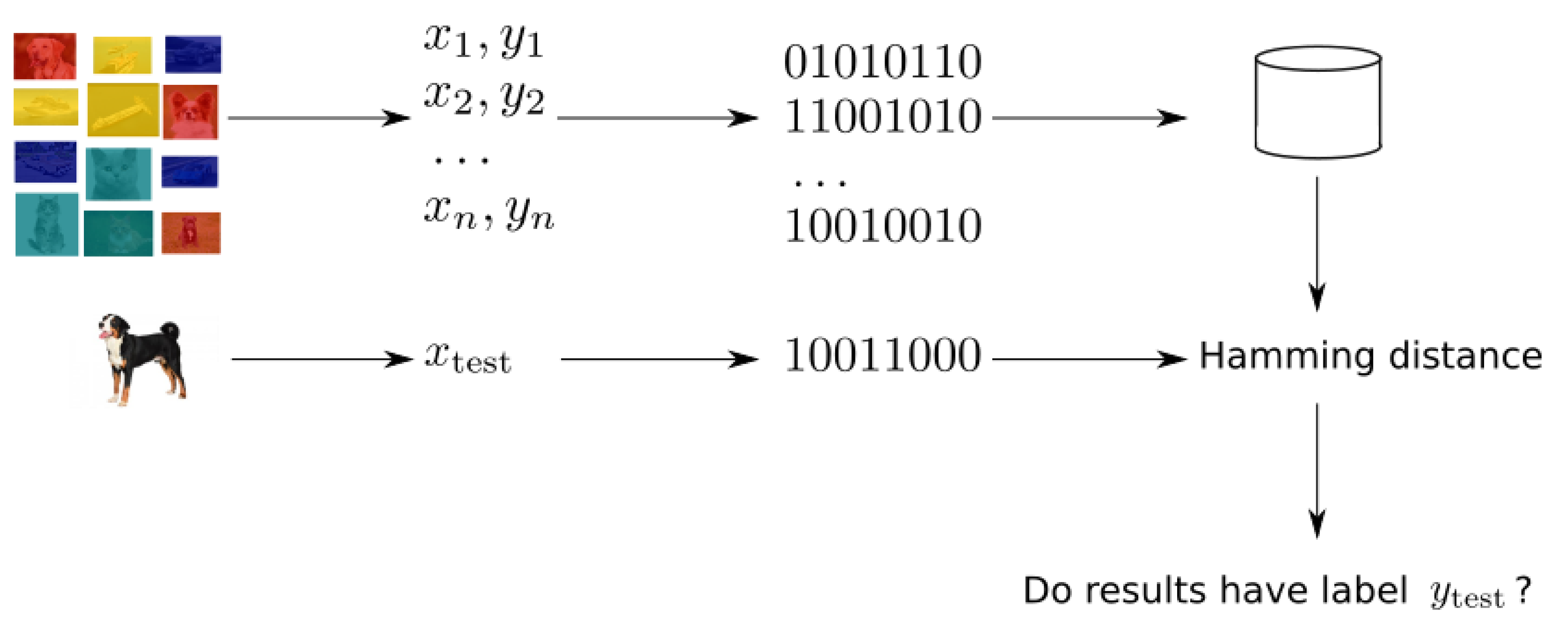
Hashing for nearest neighbor search

Compress a set of vectors $(x_i)_{i=1}^n$, $x_i \in \mathbb{R}^d$



Supervised hashing

Compress a set of vectors **and their labels** $((x_i, y_i))_{i=1}^n$, $x_i \in \mathbb{R}^d$, $y_i \in \{1, \dots, L\}$



- Supervised hashing [1, 2]: labels y known for all x in the reference set
- Semi-supervised hashing [3, 4]: labels y known for only n_{label} samples

References

[1] X. Wang, T. Zhang, G. Qi, J. Tang, and J. Wang, "Supervised quantization for similarity search," *CVPR*, 2016
[2] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," *CVPR*, 2016
[3] F. Shen, C. Shen, W. Liu, and H. Shen, "Supervised discrete hashing," *CVPR*, 2015
[4] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," *CVPR*, 2012
[5] H. Jégou, M. Douze and C. Schmid, "Product quantization for nearest neighbor search," *IEEE TPAMI*, 2011

Why not just encode the class id?

Supervised hashing with classification baseline

- Trivial binary encoding of the class id, *e.g.* $y = 9 \rightarrow 1001$

$$\begin{array}{lcl} \cancel{x_1}, y_1 & & 0011 \\ \cancel{x_2}, y_2 & \longrightarrow & 1011 \\ \dots & & \dots \\ \cancel{x_n}, y_n & & 0111 \end{array}$$

- Train classifier on pairs $(x_1, y_1), \dots, (x_n, y_n)$ and predict $\widehat{y}_{\text{test}}$ with the classifier
- Guaranteed performance: mAP $\geq$ classifier accuracy

Extension to semi-supervised hashing

$$\begin{aligned} \mathbb{P}(x \text{ correct for } q) &= \sum_{j \text{ label}} \mathbb{P}(j|x)\mathbb{P}(j|q) \\ &= \langle \mathbb{P}(\cdot|x), \mathbb{P}(\cdot|q) \rangle \\ &\approx \langle \underbrace{\widehat{\mathbb{P}}(\cdot|x)}_{\text{classifier}}, \widehat{\mathbb{P}}(\cdot|q) \rangle \end{aligned}$$

- Train $\widehat{\mathbb{P}}$ on labelled images
- Compute $\widehat{\mathbb{P}}(\cdot|x) \in [0,1]^L$ for x unlabelled
- Compress $\widehat{\mathbb{P}}(\cdot|x)$ with one-hot / LSH

Results

Features	n_{label}	n_{anchors}	Method	bits	mAP
GIST	59,000	1,000	SQ [1]	64	0.704
			SQ [1]	128	0.712
			One-hot	4	**0.762**
GIST	5,000	1,000	SDH[3]	64	0.402
			One-hot	4	0.377
			LSH	64	**0.430**
			Topline	-	*0.578*
GIST	1,000	300	KSH[4]	12	0.232
			KSH[4]	48	0.284
			One-hot	4	0.270
			LSH	48	**0.309**
			Topline	-	*0.350*
Deep	50,000	-	DSH[2]	12	0.616
			DSH [2]	48	0.621
AlexNet			One-hot	4	**0.870**

Results on CIFAR-10

Simple classification baseline
= 10x more compact
with similar performance

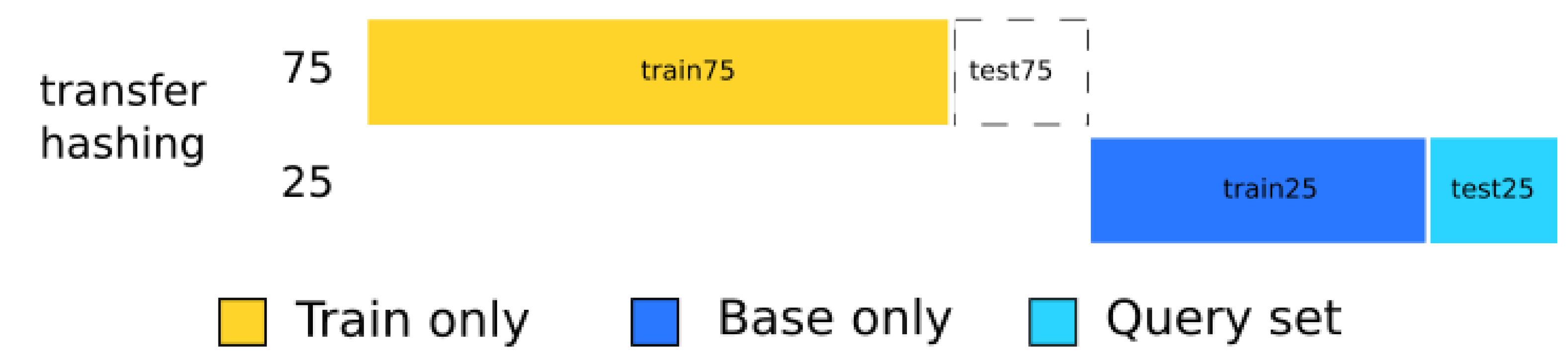
Features	Method	bits	mAP@1500
VGG	SQ [1]	128	0.620
VGG	one-hot	**10**	**0.664**

Results on ImageNet

Encoding schemes

- One-hot: $(0.15, 0.07, 0.08, \mathbf{0.7}) \rightarrow 3 \rightarrow 0011$
- LSH: $v = (0.15, 0.07, 0.08, 0.7) \rightarrow (\text{sign}(w_i^T \cdot v + b_i))_{i=1}^{n_{\text{bits}}}$
- Topline: no compression

How can we avoid this biased protocol?



- Test on classes **never seen at train time**
- Split classes in 4 folds, each with 25% of classes

Both setups

- Train hash functions on train75
- Encode train25 with hash functions

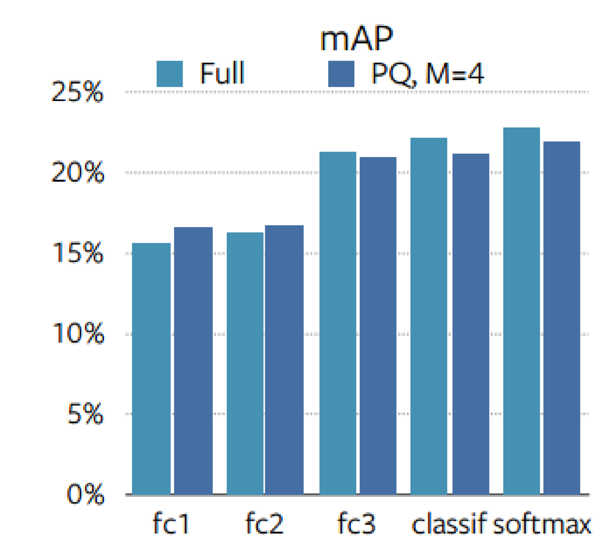
Setup 1: Retrieval with hash codes

- Use train25 as reference set
- Use test25 as queries

Setup 2: Classification on hash codes

- Train classifier using train25 labels
- Evalute accuracy on test25

Unsupervised baseline for proposed protocol



Experimental setting

- Experiments with CIFAR-10 using AlexNet
- Unsupervised PQ codes [5] with 4 bytes

Setup 1: Retrieval with hash codes

- PQ codes with asymmetric comparison
- Higher layers are better
- 4 bytes enough for most of performance
- Inner product on softmax gets the best result

Setup 2: Classification on hash codes

- Drop in accuracy due to encoding
- Lower layers more generic → better accuracy
- Lower layers high dimensional → larger gap between PQ and full vector
- → Trade-off encoding/accuracy