Robust and Compact Deep Learning Features for Image Matching and Retrieval

Shih-Fu Chang
With Xu Zhang and Svebor Karaman

ECCV 2018 Invited Talk
Feature Extraction

• A function (DNN) mapping an image (patch) to a feature vector

• Desirable properties
  • Concentrated: Samples from the same class to be close
  • Spread-out: Samples from different classes spread out in space
  • Compact: Easy to store and process
Good Feature and Bad Feature

“Good” Feature

Concentrated and Spread-out

Non-Concentrated Feature

Spread-out not Concentrated

Non-Spread-out Feature

Concentrated not Spread-out
Applications

• Image Retrieval

Query → Feature Extraction Network → Feature → Database → Feature

• Image Classification

Query → Feature Extraction Network

Volkswagen

? Fiats

Volvo
Familiar Approaches

• Contrastive loss (pairwise) [Chopra CVPR15]

1) \( \ell(x_i, x_i^+) = \max(0, \| f(x_i) - f(x_i^+) \|_2 - \epsilon^+) \)

2) \( \ell(x_i, x_i^-) = \max(0, \epsilon^- - \| f(x_i) - f(x_i^-) \|_2) \)

• Triplet loss [Hoffer IWSBPR15]

\[
\ell_{tri}(x_i, x_i^+, x_i^-) = \\
\max(0, \epsilon - (\| f(x_i) - f(x_i^-) \|_2 - \| f(x_i) - f(x_i^+) \|_2))
\]
Limitation

• Consider local relations only

• Complexity
  • Large number of pairs or triplets: $O(n^2)$
  • Sensitive to sampling methods

• Idea: consider global properties in feature space
Learning Spread-Out Local Feature Descriptor

Xu Zhang*, Felix Xinnan Yu†, Sanjiv Kumar†, Shih-Fu Chang*

ICCV 2017
Motivation

• to spread out non-matching points on unit sphere $S^d$ (Considering the whole space)
• Spread-out helps utilize the full space
• Learn distribution of the non-matching descriptor to be close to uniform distribution on unit sphere

Not Spread-Out (Bad)  Spread-Out (Good)
Properties of Uniform Distribution on Sphere

\[ E(p_1^T p_2) = E(p_1^T) E(p_2) = 0 \]

Mean

\[ E((p_1^T p_2)^2) = \frac{1}{d} \]

2nd Order Moment

Probability density of \( p_1^T p_2 \), with different dimensionalities

With high dimensionality, \( p_1, p_2 \) has high probability to be close to orthogonal!!
Global Orthogonal Regularization

\[ \ell_{GOR} = \left( \frac{1}{N} \sum_{i=1}^{N} f(x_i)^T f(x_i^-) \right)^2 \]

Mean of Non-Matching Pairs

\[ \text{E}(p_1^T p_2) = 0 \]

2nd Order Moment of Non-Matching Pairs

\[ \text{E}\left( \left( p_1^T p_2 \right)^2 \right) = \frac{1}{d} \]

Final Loss

\[ \ell_{*\_GOR} = \ell_* + \alpha \ell_{GOR} \]

Trade-off parameter
Training: add spread-out constraint to existing networks

- Positive: $x_0^+$, $x_1^+$
- Anchor: $x_0$, $x_1$
- Negative: $x_0^-$, $x_1^-$

Shared Weights

Shared Weights

Triplet Loss

Final Loss

Global Orthogonal Regularization

$\mathbf{f}(x_0)^T \mathbf{f}(x_0^-)$

$\mathbf{f}(x_1)^T \mathbf{f}(x_1^-)$
Experiment

• Training
  • Local descriptor benchmark, UBC patch dataset [Brown PAMI11]
    • three sets: Yosemite, Notre Dame, and Liberty.
    • Each set has more than 450k local image patches
    • Sampled the output of Difference of Gaussians (DoG) detector
    • Each patch has a size of 64*64.
  • Randomly sample 1M triplets from training set for training.
    Run 10 epochs.

• Testing
  • 100k testing pairs

• Metric
  • FPR95: False Positive Rate at 95% True Positive Rate.
## UBC patch dataset

### FPR95(%) Table

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training</th>
<th>Not</th>
<th>Lib</th>
<th>Not</th>
<th>Yos</th>
<th>Yos</th>
<th>Lib</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pairwise</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIFT [Lowe IJCV04]</td>
<td></td>
<td>27.29</td>
<td>29.84</td>
<td>22.53</td>
<td>26.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MatchNet [Han CVPR15]</td>
<td>512</td>
<td>11</td>
<td>13.58</td>
<td>8.84</td>
<td>13.02</td>
<td>7.7</td>
<td>4.75</td>
<td>9.82</td>
</tr>
<tr>
<td><strong>Triplet</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triplet Loss [Balntas, BMVC16]</td>
<td>256</td>
<td>7.95</td>
<td>8.10</td>
<td>7.64</td>
<td>9.88</td>
<td>3.83</td>
<td>3.39</td>
<td>6.79</td>
</tr>
<tr>
<td>Triplet Loss + AS + GOR</td>
<td></td>
<td>5.15</td>
<td>5.40</td>
<td>4.80</td>
<td>6.45</td>
<td>2.38</td>
<td>1.95</td>
<td>4.35</td>
</tr>
<tr>
<td><strong>Hard Mining</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HardNet [Anastasiya, NIPS 2017]</td>
<td>128</td>
<td>2.82</td>
<td>3.00</td>
<td>2.17</td>
<td>3.86</td>
<td>1.28</td>
<td>0.78</td>
<td>2.32</td>
</tr>
<tr>
<td>HardNet + GOR</td>
<td>128</td>
<td>2.53</td>
<td>2.85</td>
<td>2.01</td>
<td>3.52</td>
<td>1.21</td>
<td>0.76</td>
<td>2.15</td>
</tr>
</tbody>
</table>
Are features more spread-out?

• Train: Yosemite
  Test: Liberty

Baseline

Baseline + GOR

Ideally $E(p_1^T p_2) = 0$
Summary #1 – spread-out constraint

• A simple Global Orthogonal Regularization constraint makes features more spread out over the entire space
• Can be easily added to other metrics like pairwise, triplet and hard mining
Making Classifier Features
Spread-out and Concentrated

“Heated-Up Softmax Embedding”
Classifier Based Descriptor

• The second to the last layer (bottleneck) of the DNN classifier has been widely used.

• Showed strong performance in recent literatures [Movshovitz ICCV2017]
Classifier Feature

• A popular structure of DNN classifier trained with softmax function and cross entropy loss

\[ p(m|x) = \frac{\exp(z_m)}{\sum_{j=1}^{M} \exp(z_j)} \]

\[ W = [w_1, w_2, ..., w_M]^T \]
Benchmark

• Dataset
  • Stanford Car Dataset (Car196): fine-grained car category dataset, which contains 16,185 images of 196 car models. First 98 categories of 8,054 images for training, while the other 98 categories of 8,131 images are used for test.
Classifier Based Descriptor

- Showed strong performance in recent literature
  [Razavian CVPRW2014, Movshovitz ICCV2017]
Distribution of Existing Classifier Features Still not Desirable

- Feature trained on MNIST, each color for one digit
- Observation: current classifier features are still not concentrated, even after l2 normalization.
Deeper Analysis of Training Pipeline

• Applying $\ell_2$ normalization to both feature and weight shows strong performance in face verification and image retrieval [Wang MM 2017, Liu CVPR 2017]

• Use temperature parameter $\alpha$ for gradient tuning

$$p(m|x) = \frac{\exp(\alpha z_m)}{\sum_{j=1}^{M} \exp(\alpha z_j)}$$

Classifier weights:
$$W = [w_1, w_2, \ldots, w_M]^T$$
Temperature parameter $\alpha$ in Softmax Function

- $\alpha (1/T)$ affects the distribution of the final features
- Idea of adjusting temperature during training
  (Hinton et al, 2015 NIPS-W)
Classifier weights

Feature trained with normalization in MNIST dataset

$\alpha = 0.25$

$\alpha = 4$

$\alpha = 16$

$\alpha = 64$
Understand Effect of Temperature parameter $\alpha$

$$p(m|x, \alpha) = \frac{\exp(\alpha z_m)}{\sum_{j=1}^{M} \exp(\alpha z_j)}$$

• For all samples

$$\lim_{\alpha \to +\infty} p(m|x, \alpha) = \begin{cases} 
  1/K & z_m = \max(z_1, \ldots, z_M) \\
  0 & \text{otherwise,} 
\end{cases}$$

K is the number of the maximum logits

• For Correct samples (samples correctly classified by the classifier)

$$\lim_{\alpha \to +\infty} p(m|x, \alpha) = \begin{cases} 
  1 & m = y \\
  0 & \text{otherwise,} 
\end{cases}$$
Gradient in Classifier Training (Correct Samples)

- gradient with respect to descriptor

\[ \| \frac{\partial l}{\partial \hat{f}} \| = \| \sum_{m=1}^{M} \alpha (p(m|x, \alpha) - q(m|x)) \hat{w}_m \| \]

For Correct Sample

Increasing then decreasing with increasing \( \alpha \)

\[ \lim_{\alpha \to +\infty} \| \frac{\partial l}{\partial \hat{f}} \| = 0 \]
Gradient in Classifier Training (Incorrect Samples)

• gradient with respect to descriptor

\[ \| \frac{\partial l}{\partial \hat{f}} \| = \| \sum_{m=1}^{M} \alpha (p(m|x, \alpha) - q(m|x)) \hat{w}_m \| \]

For Incorrect Sample

\[ \lim_{\alpha \to +\infty} \| \frac{\partial l}{\partial \hat{f}} \| = +\infty \]
Initially train with Intermediate $\alpha$

- For Correct Centroid Sample
  - Start with intermediate $\alpha$ to assign large gradient to incorrect sample and medium gradient to boundary sample.

- For Correct Boundary Sample

- For Incorrect Sample

\[ \| \frac{\partial f}{\partial a} \|_2 \]

\[ \alpha \]

Incorrect Sample

Correct Centroid Sample

Correct Boundary Sample

Gradient for each sample

Area for “7”

Area for “1”
Then use heated-up $\alpha$

- For Correct Centroid Sample
  - Then use small $\alpha$ to assign large gradient to all the samples to further compress distribution.

- For Incorrect Sample
Heating up the Feature Training Process

• Training Strategy
  • use intermediate $\alpha$ value to start the training process

• Heated-up Feature
  • When most of the training samples are correctly classified, using small $\alpha$ value (higher temperature) to get a more concentrated descriptor.

• Batch Normalization instead of l2 normalization
  • Batch normalization adds more variance and help spread out features

“Heated-Up Softmax Embedding”
Experiment

• Dataset
  • **Stanford Car Dataset (Car196):** fine-grained car category dataset, which contains 16,185 images of 196 car models. First 98 categories of 8,054 images for training, while the other 98 categories of 8,131 images are used for test.
## Car196

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim</th>
<th>NMI</th>
<th>R @ 1</th>
<th>R @ 2</th>
<th>R @ 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Schroff, CVPR15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triplet S.H.</td>
<td>128</td>
<td>53.35</td>
<td>51.54</td>
<td>63.78</td>
<td>73.52</td>
</tr>
<tr>
<td>LIFTED</td>
<td>128</td>
<td>56.88</td>
<td>52.98</td>
<td>66.70</td>
<td>76.01</td>
</tr>
<tr>
<td>[Sohn, CVPR16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPAIRS</td>
<td>64</td>
<td>57.79</td>
<td>53.90</td>
<td>66.76</td>
<td>77.75</td>
</tr>
<tr>
<td>[Sohn, NIPS16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUCT C.</td>
<td>64</td>
<td>54.44</td>
<td>58.11</td>
<td>70.64</td>
<td>80.27</td>
</tr>
<tr>
<td>[Sohn, CVPR17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.C.</td>
<td>98</td>
<td>61.12</td>
<td>67.54</td>
<td>77.77</td>
<td>85.74</td>
</tr>
<tr>
<td>[Law, ICML17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FANNG</td>
<td>64</td>
<td>59.50</td>
<td>64.65</td>
<td>76.20</td>
<td>84.23</td>
</tr>
<tr>
<td>[Harwood, ICCV17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROXY NCA</td>
<td>64</td>
<td>64.90</td>
<td>73.22</td>
<td>82.42</td>
<td>86.36</td>
</tr>
<tr>
<td>[Movshovitz, ICCV17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTMAX</td>
<td>64</td>
<td>59.52</td>
<td>60.76</td>
<td>73.58</td>
<td>82.50</td>
</tr>
<tr>
<td>L2 normalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.M.+LN</td>
<td>64</td>
<td>62.40</td>
<td>68.59</td>
<td>78.55</td>
<td>86.18</td>
</tr>
<tr>
<td>Batch norm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.M.+BN</td>
<td>64</td>
<td>65.81</td>
<td>71.12</td>
<td>80.62</td>
<td>87.82</td>
</tr>
<tr>
<td>L2 normalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.M.+HLN</td>
<td>64</td>
<td>66.87</td>
<td>71.93</td>
<td>81.68</td>
<td>88.34</td>
</tr>
<tr>
<td>Heated-up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.M.+HBN</td>
<td>64</td>
<td>68.10</td>
<td>74.70</td>
<td>83.90</td>
<td>89.77</td>
</tr>
</tbody>
</table>
Qualitative Result

Query

Top 3 returned images
Summary #2 – temperature control

• Proper heat-up procedure (intermediate alpha $\rightarrow$ small alpha) makes features more spread out and concentrated
Compact features for large-scale indexing

Svebor Karaman, Xudong Lin, and Shih-Fu Chang
Need of Logarithmic Search

• Billion scale datasets are now common, exhaustive search unacceptable
• Standard ANN method (e.g. tree based) achieve limited speed/accuracy
• New graph based indexing dominates recent benchmarks

1.19M Glove wordvectors

1M SIFT image features

https://github.com/erikbern/ann-benchmarks
Navigable Small World: Overview

Navigable Small World Graph [Malkov14]
• Any vertex reachable from any other vertex in a few hops
• Typical distance between two random nodes: \( L \propto \log(N) \)

NN “greedy search”
• Given a query, select a random entry point
• Search over neighbors of current point
• Go to closest ‘neighbor’ and iterate
• Multiple searches or backtracking for K-NN

Navigable Small World: Construction Process

Sequential Insertion (illustration for 3-NN graph)
Navigable Small World: Construction Process

Edges in the graph serve two purposes:

• **short-range links**: created in later stages, connections to a sample’s nearest neighbors
• **long-range links**: created in earlier stages, responsible for the small world navigation property of the graph

• Empirically, hop distance between two random nodes: \( L \propto \log(N) \)
• Any vertex reachable from any other vertex in a few hops
Hierarchical Navigable Small World

Hierarchical NSW (HNSW) [Malkov16]

• Exponentially decaying probability of sample insertion in higher layers
• Separate long-range and short-range links in different layers
• Node degree bounded in each layer
• Edges diversification with occlusion rule
• hop distance between two random nodes: $L \propto \log(N)$

Hierarchical Navigable Small World: Search

Top-down process:
• From top layer, in each layer, greedy search to find best NN
• Next (lower) layer, search starts from previous layer’s best NN
• Bottom layer, push neighborhood of current best unvisited sample to a search buffer kept sorted, until all buffer samples are visited
• Node degree bounded in each layer $\rightarrow \log(N)$ search complexity for 1-NN

IJBC Face Search Performance Evaluation (up to 1M samples)

- IJBC: challenging unconstrained face recognition dataset
- **HNSW fully preserves accuracy of exhaustive search**
IJBC Face Search Speed Evaluation

- Logarithmic search time and number of distances computed
Graph based methods storage requirement problem

- Graph index: store **links** information and all **features** (no compression)
- Dataset of 1B samples:
  - Graph with 32 neighbors per samples: **119 GB** to store the links
  - 128 dimensional real-valued features (SIFT): **477 GB**
  - 960 dimensional real-valued features (GIST): **3.5 TB**
  - 256 bits hash codes: **30 GB**
- Idea: compress feature vector by hashing
- Hashing useful for search from mobile
Compressing HNSW

• Can we reduce the required storage? (original 128 or 960-dim floating point)

• Compress the samples’ representation
  • Product quantization
  • Hashing

Compressed features, \( c(f(x_i)) \), \(~100\ bits\)
Compressing HNSW (1): Product Quantization

- Split space of d dimensions into 2 subspaces
- Train K-means in each subspace
- PQ code is the combination of the closest cluster id \(c_i\) in each subspace:
  \[\text{PQ}(x_i) = \{c_k^{(1)}, c_j^{(2)}\}\]

**Link & Code [L&C]: two-stage strategy**

1. Product Quantized codes for coarse search
2. Re-ranking with refined representation, considering sample’s neighbors

---

Link & Code Experimental Results

- Memory budget 64 bytes (24 bytes for links, 40 bytes for features)
- Refined representation helps in some cases (M=8)
- Achieve 62.5% accuracy with 9.6X compression

<table>
<thead>
<tr>
<th>codec</th>
<th>vector quantization error ($\times 10^3$)</th>
<th>exhaustive</th>
<th>R@1 T=1024</th>
<th>R@1 T=16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100M</td>
<td>0.608 0.601</td>
<td>0.427 0.434</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1B</td>
<td>0.611 0.600</td>
<td>0.429 0.435</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6&amp;OPQ40</td>
<td>24.3 24.3</td>
<td>0.608 0.607</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6&amp;OPQ40 M=0</td>
<td>22.7 22.5</td>
<td>0.611 0.600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6&amp;OPQ36 M=4</td>
<td>21.9 21.5</td>
<td>0.608 0.607</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6&amp;OPQ32 M=8</td>
<td>20.0 19.8</td>
<td>0.625 0.612</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deep 1B (or subsampled to 100M)

Compressing HNSW (2): Hashing

Hashing: binary representation $H(x_i) = [010...110]$

Linear hashing, e.g., LSH

Hypersphere based hashing [SpH]

Compressing HNSW (2): Hashing Objective

Preserve HNSW performance with features:

• Top layers (e.g. $B_1$ and $B_2$): hashing shall preserve rank 1 result in batch in order to preserve search path

• Bottom layer: hashing shall preserve ranks in search buffer to preserve search results
Compressing HNSW (2): Hashing Training

- Top batch: first sample \( q \), current sample + neighbors sorted \( \{x_1, \ldots, x_{\text{maxM}}\} \)
- **Rank 1 loss:**
  \[
  \sum_{i=2}^{\text{maxM}+1} 1 \left[ d(h(x_1), h(q)) > d(h(x_i), h(q)) \right]
  \]
- Bottom batch: first sample \( q \), and sorted samples \( \{x_1, \ldots, x_{N-1}\} \) in search buffer
- **Full ranking loss:**
  \[
  \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} 1 \left[ d(h(x_i), h(q)) \geq d(h(x_j), h(q)) \right]
  \]
- Minimize rank loss + regularization with SGD

Diagram:
- Hash Layer
- FC + \text{tanh}
- Hidden Layer
- FC + ReLU
- Hidden Layer
- FC + ReLU
- \( f(x_i) \)
Hashing-compressed HNSW search performance on IJBC

Hashing for HNSW graph (work in progress)

- discriminative face features from UMD
- Compress features with 512 hash bits
- Allows 8X compression with small performance loss at 1M samples
Summary

• Objective: spread-out, concentrated, compact features

• A simple Global Orthogonal Regularization constraint makes features more spread out over the entire space

• Proper heat-up procedure makes features more spread out and concentrated

• Compact hashing for graph-based logarithmic search