Deep image hashing based on twin-bottleneck hashing with variational autoencoders

Maxim Verwilst, Nina Žižakić, Lingchen Gu and Aleksandra Pižurica
Content-based image retrieval
Content-based image retrieval – contd.

- query image
- hash function
- hash code: 100110
- look up similar hash codes in the database
- hash database:
  - 100100
  - 100110
  - 100100
  - 100111
- retrieve images
Image hashing

Hash function → Hash code

1 0 0 1 1 0

Hand-crafted (data-independent) approach

Deep learning (data-dependent) approach – supervised and unsupervised

- autoencoder networks
- adversarial networks
- graph-based networks
- ...

Twin-bottleneck hashing (Shen et al, 2020)

TBH: Shen et al, Auto-Encoding Twin-Bottleneck Hashing, CVPR 2020
Proposed deep hashing method

Encoder

Variational bottleneck

GECO optimisation

Graph Convolutional Network

Decoder

\[ U(0,1)^L \]
Evaluation (precision-recall curves)

CIFAR-10:
60k 32x32 images, 10 classes

MS-COCO:
>200k labeled images, 80 object classes

Reference method TBH: Shen et al, Auto-Encoding Twin-Bottleneck Hashing, CVPR 2020
Evaluation (mAP and retrieval examples)

<table>
<thead>
<tr>
<th>Method</th>
<th>16 bits</th>
<th>32 bits</th>
<th>64 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSH [11]</td>
<td>0.106</td>
<td>0.102</td>
<td>0.105</td>
</tr>
<tr>
<td>SPH [9]</td>
<td>0.272</td>
<td>0.285</td>
<td>0.300</td>
</tr>
<tr>
<td>AGH [43]</td>
<td>0.333</td>
<td>0.357</td>
<td>0.358</td>
</tr>
<tr>
<td>SPHERH [44]</td>
<td>0.254</td>
<td>0.291</td>
<td>0.333</td>
</tr>
<tr>
<td>KMH [45]</td>
<td>0.279</td>
<td>0.296</td>
<td>0.334</td>
</tr>
<tr>
<td>ITQ [23]</td>
<td>0.305</td>
<td>0.325</td>
<td>0.349</td>
</tr>
<tr>
<td>DGH [46]</td>
<td>0.335</td>
<td>0.353</td>
<td>0.361</td>
</tr>
<tr>
<td>DeepBit [47]</td>
<td>0.194</td>
<td>0.249</td>
<td>0.277</td>
</tr>
<tr>
<td>SGH [6]</td>
<td>0.435</td>
<td>0.437</td>
<td>0.433</td>
</tr>
<tr>
<td>BGAN [8]</td>
<td>0.525</td>
<td>0.531</td>
<td>0.562</td>
</tr>
<tr>
<td>BINGAN [7]</td>
<td>0.476</td>
<td>0.5122</td>
<td>0.520</td>
</tr>
<tr>
<td>GreedyHash [48]</td>
<td>0.448</td>
<td>0.473</td>
<td>0.501</td>
</tr>
<tr>
<td>HashGAN [31]</td>
<td>0.447</td>
<td>0.463</td>
<td>0.481</td>
</tr>
<tr>
<td>DVB [49]</td>
<td>0.403</td>
<td>0.422</td>
<td>0.446</td>
</tr>
<tr>
<td>DistillHash [50]</td>
<td>0.284</td>
<td>0.285</td>
<td>0.288</td>
</tr>
<tr>
<td>PSEUDOLABEL [37]</td>
<td>0.517</td>
<td>0.572</td>
<td>0.596</td>
</tr>
<tr>
<td>TBH [4]</td>
<td>0.532</td>
<td>0.573</td>
<td>0.578</td>
</tr>
<tr>
<td><strong>PROPOSED METHOD</strong></td>
<td><strong>0.556</strong></td>
<td><strong>0.6021</strong></td>
<td><strong>0.6057</strong></td>
</tr>
</tbody>
</table>

Mean average precision (mAP) of other state-of-the-art unsupervised hashing methods and our method.
Summary

• Improved the **binary bottleneck** of TBH by using VAE with disentangled variables

• Improved the **continuous bottleneck** of TBH to use a VAE trained with a constrained optimisation setup

• Outperformed state-of-the-art unsupervised hashing methods
Thank you!

Questions?
References

