End-to-End Supervised Product Quantization for Image Search and Retrieval - Supplementary Material

1 Summary

In this supplement, we provide:

1. The hyper-parameters and training details for each experiment.
2. Ablation study of the hyperparameters.

2 Training details

The DPQ loss function is composed of several terms:

$$
\begin{aligned}
&-\alpha^{hard}\sum_{b=1}^{B} logp_{b,y_b}^{hard} - \alpha^{soft}\sum_{b=1}^{B} logp_{b,y_b}^{soft} \\
&+\frac{\beta^{hard}}{2B}\sum_{b=1}^{B}||hard_b - o_{y_b}||^2 + \frac{\beta^{soft}}{2B}\sum_{b=1}^{B}||soft_b - o_{y_b}||^2 \\
&+\frac{\mu}{2}\sum_{m=1}^{M}\sum_{k=1}^{K}\left(\frac{1}{B}\sum_{b=1}^{B} q_m^b(k)\right)^2 \\
&-\frac{\eta}{2B}\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{b=1}^{B}\left(q_m^b(k)\right)^2
\end{aligned}
$$

The two terms in the first line are the softmax loss for the hard and soft representations. Where B is the batch size, p_{b,y_b}^{hard}, and p_{b,y_b}^{soft} are the probability of the $b-th$ sample belonging to its correct class, y_b, according to the $hard$ and $soft$ representation respectively. We denote by α^{hard} and α^{soft} the weights of the hard softmax loss and the soft softmax loss respectively.

The two following terms in the second line are the central loss for the hard and soft representations. Where o_{y_b} is the center vector of the correct class, y_b, and $hard_b$ and $soft_b$ are the hard and the soft representations of the $b-th$ sample. We denote by β^{hard} and β^{soft} the weights of the $hard$ central loss and the $soft$ central loss respectively.

The term in the third line is the Gini Batch Diversity regularization. Where M is the number of partitions used by DPQ, and K is the number of centroids for each partition. The probability of the $m-th$ sub-vector of the $b-th$ sample being assigned to the $k-th$ centroid is $q_m^b(k)$. The influence of this regularization is controlled by the weight μ.

Finally, the term in the last line is the Gini Sample Sharpness. The influence of this regularization is controlled by the weight η.

We now provide the training details and the hyper-parameters of each experiment.

2.1 Single-domain category retrieval

2.1.1 CIFAR10 - First Protocol

We train DPQs with $M = 4$ partitions and $K = (8, 64, 512, 4096)$ centroids per partition, to match our experiments with the protocol. DPQ is learned on top of the embedding layer of the base network, that has $U = 500$ units. We start by adding a fully connected layer, F, on top of U, with $V = M \cdot K$ units. We then split $F \in \mathbb{R}^V$ into M equal parts: $F = (F_1, F_2, \dots, F_M)$ where $F_i \in \mathbb{R}^K$.

We then apply a softmax function that outputs a probability distribution, p_m, with K entries.

The centroid vectors, C_m, for each partition $\{1 \dots m\}$ are chosen to be in R^{30} such that the final soft and hard representations are in R^{120}.

The optimization is using SGD with a learning rate of 0.001, and a momentum of 0.9. A weight decay of 0.0015 is introduced to the DPQ loss function. The optimization is performed for $200K$ iterations.

The hyper-parameter values are described in the following table:

Hyper-parameter	Value
α^{hard}	1.0
α^{soft}	1.0
β^{hard}	0.5
β^{soft}	0.5
μ	0.777
η	0.06
B	200
K	$\{8, 64, 512, 4096\}$
M	4

Where K is chosen to match the $12, 24, 36$ and 48 bits in the experiment protocol.

2.1.2 CIFAR10 - Second Protocol

DPQ is learned on top of the embedding layer of the VGG-CNN-F base network described in [1], that has $U = 4096$ units. We start by adding a fully connected layer, F, on top of U, with $V = M \cdot K$ units. We then split $F \in \mathbb{R}^V$ into M equal parts: $F = (F_1, F_2, \dots, F_M)$ where $F_i \in \mathbb{R}^K$. We then apply a softmax function that outputs q_m, with $K = 16$ entries.

The centroid vectors, C_m, for each partition $\{1 \dots m\}$ are chosen to be in R^{32} such that the final soft and hard representations are in $R^{32 \cdot M}$.

The network is fine-tuned using the weights of [1].

The optimization is using SGD with a learning rate of 0.0005, and a momentum of 0.9. A weight decay of 0.004 is introduced to the DPQ loss function. The optimization is performed for $12K$ iterations.

The hyper-parameter values are described in the following table:

Hyper-parameter	Value
α^{hard}	1.0
α^{soft}	1.0
β^{hard}	0.1
β^{soft}	0.1
μ	0.9
η	0.12
B	200
K	16
M	$\{4, 6, 8, 12\}$

Where M is chosen to match the $16, 24, 32$ and 48 bits in the experiment protocol.

2.1.3 CIFAR10 - Third Protocol

The same configuration from the second protocol is used.

2.1.4 ImageNet-100

In this protocol, suggested by [2], DPQ is learned on top of the embedding layer of the ResNet V2 50 [3]. We start by adding a fully connected layer, F, on top of U, with $V = M \cdot K$ units. We then split $F \in \mathbb{R}^V$ into $M = 8$ equal parts: $F = (F_1, F_2, \ldots, F_M)$ where $F_i \in \mathbb{R}^K$. We then apply a softmax function that outputs q_m, with K entries. We use values of $K = \{4, 16, 256\}$ to follow the protocol of [2] which is using $\{16, 32, 64\}$ bits.

The centroid vectors, C_m, for each partition $\{1 \ldots m\}$ are chosen to be in R^{64} such that the final soft and hard representations are in R^{512}.

The network is fine-tuned using the weights of [3].

The optimization is using AdaGrad [4] with a learning rate of 0.1, and a weight decay of 0.0001. The optimization is performed for $30K$ iterations.

The hyper-parameter values are described in the following table:

Hyper-parameter	Value
α^{hard}	1.0
α^{soft}	1.0
β^{hard}	0.25
β^{soft}	0.25
μ	80
η	0.82
B	200
K	$\{4, 16, 256\}$
M	8

Where K is chosen to match the $16, 32$ and 64 bits in the experiment protocol.

2.2 Cross-domain category retrieval

The input of DPQ is the fixed representation computed by applying the VGG-128 [1] pre-trained network of [1] and extracting the embedding layer in the 2-layer experiment, and the layer before the embedding layer in the 3-layer experiment. A fully connected layer with $U = 2048$ units is learned on top of the input. A Batch Normalization [5] layer and a ReLU activation is then applied. The output is then split into $M = 8$ equal parts: $F = (F_1, F_2, \dots, F_M)$, where $F_i \in \mathbb{R}^{256}$. On each sub-vector, F_i, we apply a softmax function that outputs q_m, with $K = 256$. The centroid vectors, C_m, are chosen to be in $\mathbb{R}^{64}$. Therefore, both the final hard and soft representations are in $\mathbb{R}^{512}$.

The optimization is using AdaGrad [4] with a learning rate of 0.1. A weight decay is not being used. The optimization is performed for $30K$ iterations.

The hyper-parameter values are described in the following table:

Hyper-parameter	Value
α^{hard}	1.0
α^{soft}	1.0
β^{hard}	0.5
β^{soft}	0.5
μ	80.0
η	0.82
B	200
K	256
M	8

2.3 Image classification

The exact same network trained in 2.2 is used for the classification experiment.

3 Ablation Study

In addition to the study of the effect of the joint central loss (Fig. 2 in the paper), we show the effect of the regularization terms, GiniBatch and GiniSample, on the results of the cross-domain experiments.

3.1 GiniBatch

The table below shows the effect of the GiniBatch hyperparameter on the mAP metric in the cross-domain experiments. The hyperparameter of GiniSample was fixed at 0.8 and the hyperparameter of the join central loss was fixed at 0.25.

	0	20	40	60	80
Caltech	0.3467	0.3955	0.4095	0.4099	0.4086
Caltech+IntraNorm	0.3596	0.41601	0.4257	0.4246	0.4245
VOC	0.5407	0.51081	0.5254	0.53731	0.5353
VOC+IntraNorm	0.5475	0.558	0.5625	0.5645	0.5636
ImgNet	0.2647	0.3134	0.323	0.324	0.324
ImgNet+IntraNorm	0.25496	0.3149	0.3239	0.3253	0.3231

3.2 GiniSample

The table below shows the effect of the GiniSample hyperparameter on the mAP metric in the cross-domain experiments. The hyperparameter of GiniBatch was fixed at 80 and the hyperparameter of the joint central loss was fixed at 0.25.

	0	0.2	0.4	0.6	0.8
Caltech	0.365	0.3756	0.3952	0.411	0.4118
Caltech+IntraNorm	0.3949	0.4059	0.4226	0.4276	0.4287
VOC	0.516	0.521	0.519	0.5278	0.5372
VOC+IntraNorm	0.5291	0.5513	0.5609	0.5654	0.5643
ImgNet	0.24878	0.2861	0.317	0.3269	0.3272
ImgNet+IntraNorm	0.2491	0.2837	0.3182	0.3277	0.3276

References

[1] Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)

[2] Loncaric, M., Liu, B., Weber, R.: Learning hash codes via hamming distance targets. arXiv preprint arXiv:1810.01008 (2018)

[3] He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision, Springer (2016) 630–645

[4] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**(Jul) (2011) 2121–2159

[5] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. (2015) 448–456