Supplement for BIER

1. Introduction

In this document we provide further insights into Boosting Independent Embeddings Robustly (BIER). First, in Section 2 we describe our method for loss functions operating on triplets. Next, in Section 3 we show how our method behaves when we vary the embedding size and the number of groups. In Section 4 we summarize the effect of our boosting based training approach and our initialization approach. We provide an experiment evaluating the impact of end-to-end training in Section 5. Further, in Section 6 we demonstrate that our method is applicable to generic image classification problems. Finally, we show a qualitative comparison of the different embeddings in our ensemble in Section 7 and some qualitative results in Section 8.

2. BIER for Triplets

For loss functions operating on triplets of samples, we illustrate our training method in Algorithm 1. In contrast to our tuple based algorithm, we sample triplets $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$ and $\boldsymbol{x}^{(3)}$ which satisfy the constraint that the first pair ($\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$) is a positive pair (*i.e.* $y^{(1),(2)} = 1$) and the second pair ($\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(3)}$) is a negative pair (*i.e.* $y^{(1),(3)} = 0$). We accumulate the positive and negative similarity scores separately in the forward pass. In the backward pass we reweight the training set for each learner m according to the negative gradient ℓ' at the ensemble predictions of both image pairs up to stage $m-1$.

```
Let $\eta_m = \frac{2}{m+1}$, for $m = 1, 2, \ldots, M$,
$M$ = number of learners, $I$ = number of iterations
for $n = 1$ to $I$ do
    /* Forward pass */
    Sample triplet ($\boldsymbol{x}_n^{(1)}$, $\boldsymbol{x}_n^{(2)}$, $\boldsymbol{x}_n^{(3)}$),
      s.t. $y^{(1),(2)} = 1$ and $y^{(1),(3)} = 0$.
    $s_n^{0^+} := 0$
    $s_n^{0^-} := 0$
    for $m$ = $1$ to $M$ do
        $s_n^{m^+} := (1-\eta_m) s_n^{m-1^+} + \eta_m s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(2)}))$
        $s_n^{m^-} := (1-\eta_m) s_n^{m-1^-} + \eta_m s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(3)}))$
    end
    Predict $s_n^+$ = $s_n^{M^+}$
    Predict $s_n^-$ = $s_n^{M^-}$

    /* Backward pass */
    $w_n := 1$
    for $m = 1$ to $M$ do
        $s_m^{(1),(2)} := s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(2)})$
        $s_m^{(1),(3)} := s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(3)})$
        Backprop $w_n \ell(s_m^{(1),(2)}, s_m^{(1),(3)})$
        $w_n := -\ell'(s_n^{m^+}, s_n^{m^-})$
    end
end
```

Algorithm 1: Online gradient boosting algorithm for our CNN using triplet based loss functions.

3. Evaluation of Embedding and Group Sizes

To analyse the performance of BIER with different embedding and group sizes we run an experiment on the CUB-200-2011 dataset [9]. We train a model with an embedding size of 512 and 1024 and vary the number of groups (*i.e.* learners) in the ensemble. The group sizes of the individual models are shown in Table 1. We report the R@1 scores of the different models in Figure 1. The performance of our method gracefully degrades when the number of groups is too small or too large. Further, for larger embedding sizes a larger number of groups is beneficial. This is due to the tendency of larger embeddings to overfit. To address this problem, we train several embeddings which are smaller and therefore, less prone to overfitting.

Embedding	Group Size	Groups
512	2	170-342
512	3	96-160-256
512	4	52-102-152-204
512	5	34-68-102-138-170
1024	3	170-342-512
1024	4	102-204-308-410
1024	5	68-136-204-274-342
1024	6	50-96-148-196-242-292
1024	7	36-74-110-148-182-218-256

Table 1. Group sizes used in our experiments.

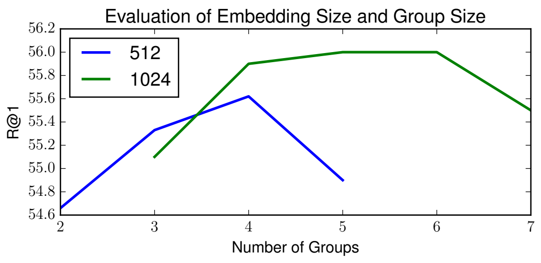

Figure 1. Evaluation of an embedding size of 512 and 1024 with different numbers of groups.

4. Impact of Matrix Initialization and Boosting

We summarize the impact of matrix initialization and the proposed boosting method on the CUB-200-2011 dataset [9] in Table 2. Both our initialization method and our boosting based training method improve the final R@1 score of the model.

Method	R@1
Baseline	51.76
Our initialization	53.73
Boosting with random initialization	54.41
Boosting with our initialization	55.33

Table 2. Summary of the impact of our initialization method and boosting on the CUB-200-2011 dataset.

5. Evaluation of End-to-End Training

To show the benefits of end-to-end training with our method we apply our online boosting approach to a fine-tuned network and fix all hidden layers in the network (denoted as *Stagewise training*). We compare the results against end-to-end training and summarize the results in Table 3. End-to-end training significantly improves final R@1 score, since weights of lower layers benefit from the increased diversity of the ensemble.

Method	R@1
Stagewise training	52.0
End-to-End training	55.3

Table 3. Influence of end-to-end training on the CUB-200-2011 dataset.

6. General Applicability

Ideally, our idea of boosting several independent classifiers with a shared feature representation should be applicable beyond the task of metric learning. To analyse the generalization capabilities of our method on regular image classification tasks, we run an experiment on the CIFAR-10 [4] dataset. CIFAR-10 consists of $60,000$ color images grouped into 10 categories. Images are of size 32×32 pixel. The dataset is divided into $10,000$ test images and $50,000$ training images. In our experiments we split the training set into $10,000$ validation images and $40,000$ training images. We select the number of groups for BIER based on the performance on the validation set.

The main objective of this experiment is not to show that we can achieve state-of-the-art accuracy on CIFAR-10 [4], but rather to demonstrate that it is generally possible to improve a CNN with our method. To this end, we run experiments on the CIFAR-10-Quick [2] and an enlarged version of the CIFAR-10-Quick architecture [1] (see Table 4). In the enlarged version, denoted as CIFAR-10-Quick-Wider, the number of convolution channels and the number of neurons in the fully connected layer is doubled. Further, an additional fully connected layer is inserted into the network. In both architectures, each convolution layer is followed by Rectified Linear Unit (ReLU) nonlinearity and a pooling layer of size 3×3 with stride 2. The last fully connected layer in both architectures has no nonlinearity.

To apply our method, we divide the last fully connected layer into 2 and 4 non-overlapping groups for the CIFAR-10-Quick and CIFAR-10-Quick-Wider architecture, respectively, and append a classifier to each group (see Table 4). As loss function we use crossentropy. Further, instead of pre-initializing the weights with our optimization method, we directly apply the optimization objective from Equation (3) in the main manuscript to the last hidden layer of the network during training time. This encourages the groups to be independent of each other. The main reason for adding the loss function during training time is that weights change too drastically in networks trained from scratch compared to fine-tuning a network from a pre-trained ImageNet model. Hence, for this type of problems it is more effective to additionally encourage diversity of the learners with a separate loss function.

We compare our method to dropout [8] applied to the last hidden layer of the network. As we see in Tables 5 and 6, BIER improves on the CIFAR-10-Quick architecture over a baseline with just weight decay by 2.68% and over dropout by 0.78%. On the larger network which is more prone to overfitting, BIER improves over the baseline by 2.42% and over dropout by 1.41%.

These preliminary results indicate that BIER generalizes well for other tasks beyond metric learning. Thus, we will further investigate the benefits of BIER for other computer vision tasks in our future work.

7. Qualitative Comparison of Embeddings

To illustrate the differences between the learned embeddings we show several qualitative examples in Figure 2.

CIFAR-10-Quick	CIFAR-10-Quick-Wider
conv $5 \times 5 \times 32$ max-pool $3 \times 3/2$	conv $5 \times 5 \times 64$ max-pool $3 \times 3/2$
conv $5 \times 5 \times 32$ avg-pool $3 \times 3/2$	conv $5 \times 5 \times 64$ avg-pool $3 \times 3/2$
conv $5 \times 5 \times 64$ avg-pool $3 \times 3/2$	conv $5 \times 5 \times 128$ avg-pool $3 \times 3/2$
fc 64	fc 128
clf 10×2	fc 128
	clf 10×4

Table 4. We use the CIFAR-10-Quick [2] and an enlarged version of CIFAR-10-Quick [1] architecture.

Method	Accuracy
Baseline	78.72
Dropout	80.62
BIER	**81.40**

Table 5. Results on CIFAR-10 [4] with the CIFAR-10-Quick architecture.

Method	Accuracy
Baseline	80.67
Dropout	81.69
BIER	**83.10**

Table 6. Results on CIFAR-10 [4] with the CIFAR-10-Quick-Wider architecture.

Successive learners typically perform better at harder examples compared to previous learners, which have a smaller embedding size.

8. Qualitative Results

To illustrate the effectiveness of BIER we show some qualitative examples in Figures 3, 4, 5, 6 and 7.

References

[1] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra. Reducing Overfitting in Deep Networks by Decorrelating Representations. In *Proc. ICLR*, 2016. 2, 3

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv*, abs/1408.5093, 2014. 2, 3

[3] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *Proc. ICCV Workshops*, 2013. 5

[4] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. 2, 3

[5] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles. In *Proc. CVPR*, 2016. 7

[6] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *Proc. CVPR*, 2016. 6

[7] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *Proc. CVPR*, 2016. 6

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 15:1929–1958, 2014. 2

[9] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1, 2, 4, 5

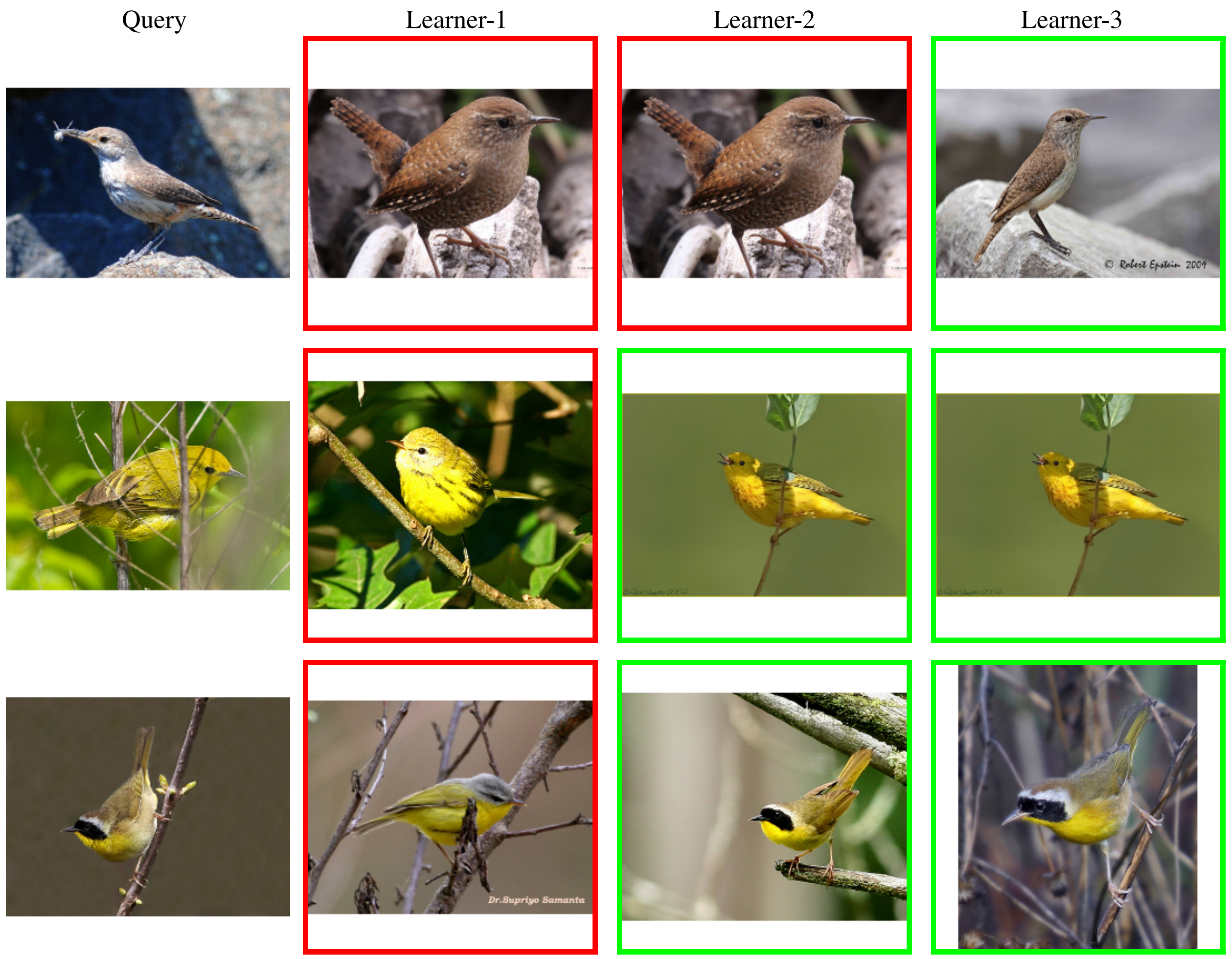

Figure 2. Qualitative results on the CUB-200-2011 [9] dataset of the different learners in our ensemble. We retrieve the most similar image to the query image for learner 1, 2 and 3, respectively. Correct results are highlighted green and incorrect results are highlighted red.

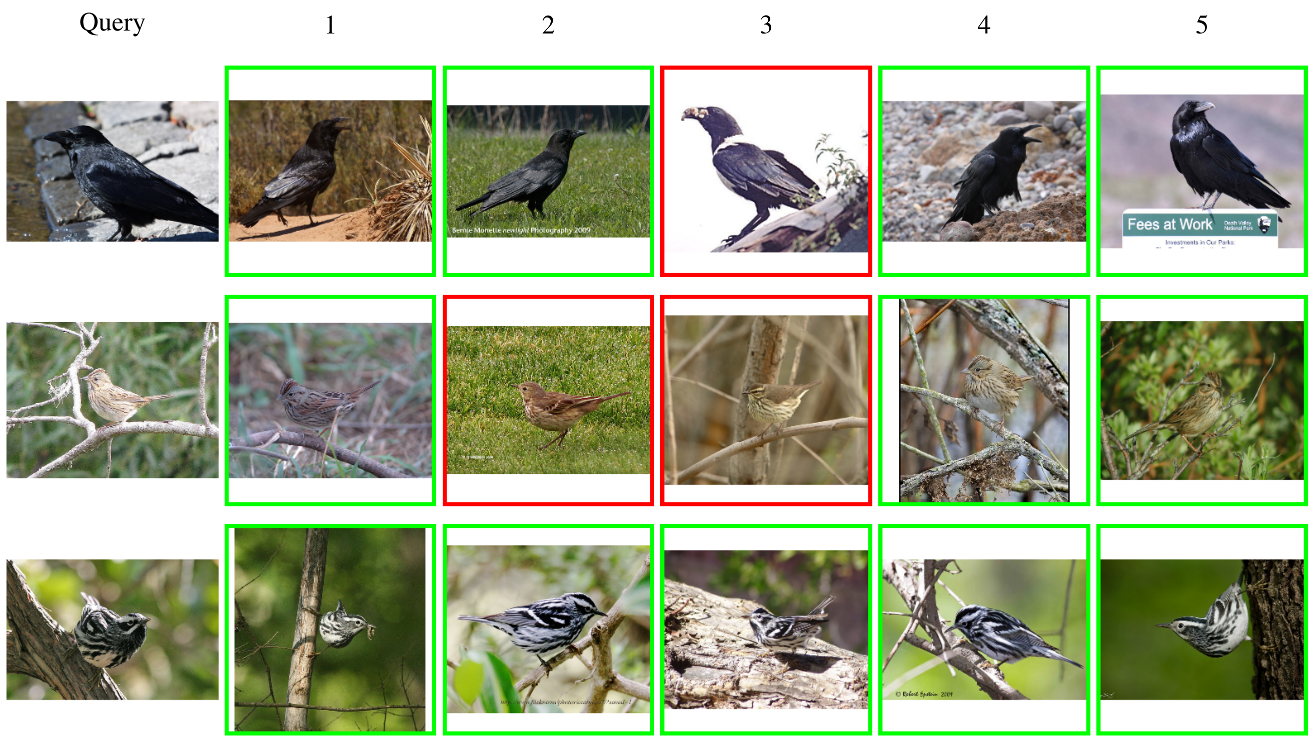

Figure 3. Qualitative results on the CUB-200-2011 [9] dataset. We retrieve the 5 most similar images to the query image. Correct results are highlighted green and incorrect results are highlighted red.

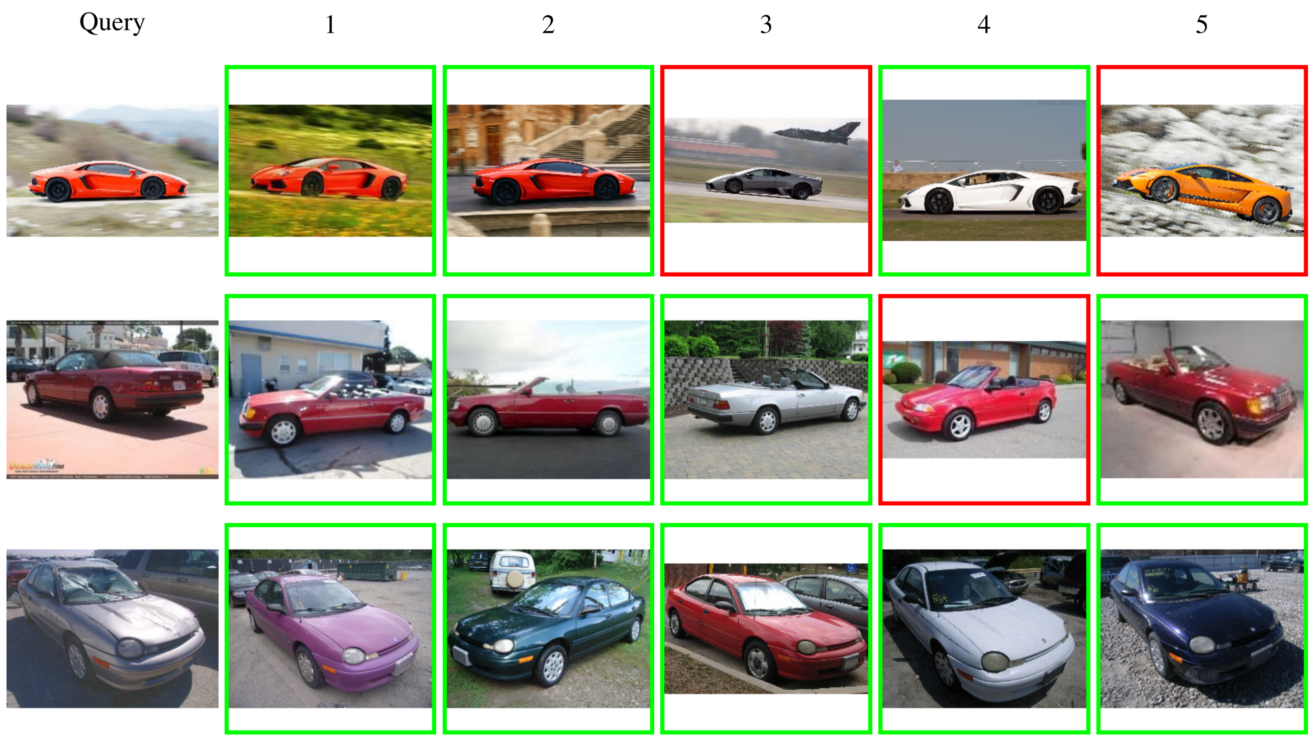

Figure 4. Qualitative results on the Cars-196 [3] dataset. We retrieve the 5 most similar images to the query image. Correct results are highlighted green and incorrect results are highlighted red.

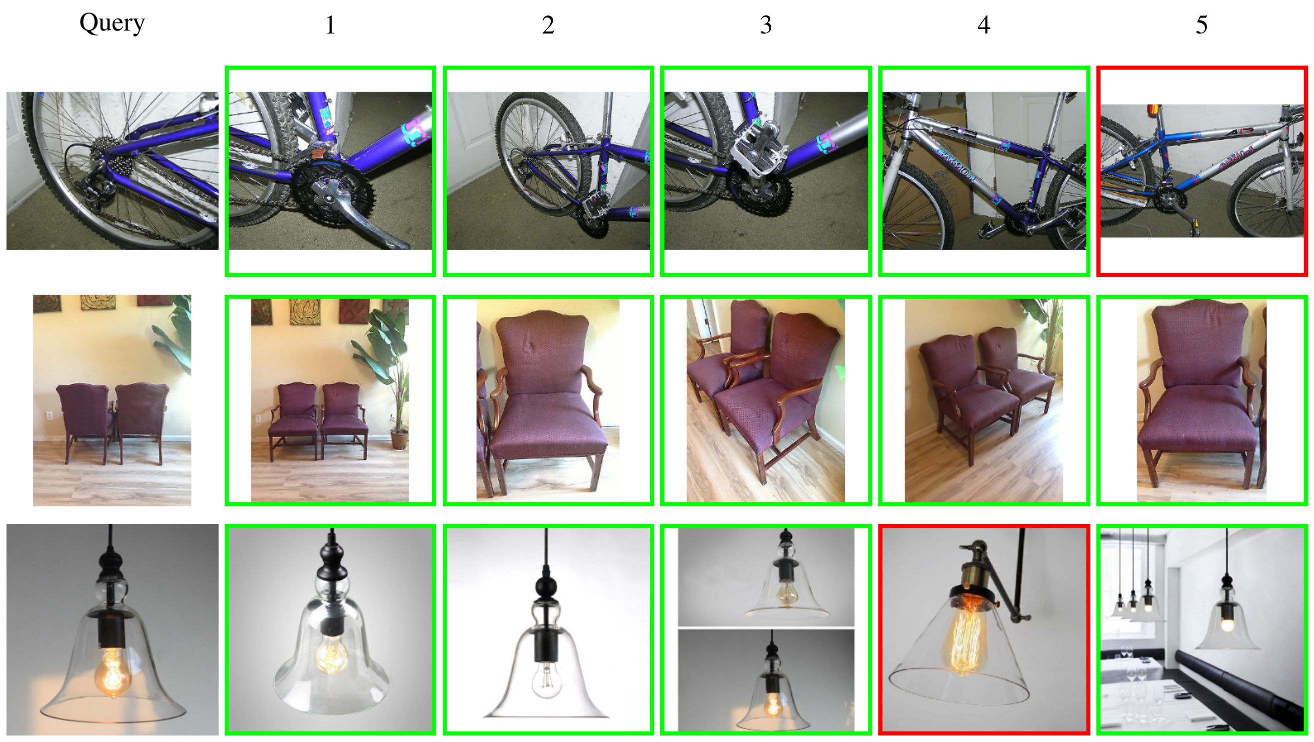

Figure 5. Qualitative results on the Stanford Online Products [7] dataset. We retrieve the 5 most similar images to the query image. Correct results are highlighted green and incorrect results are highlighted red.

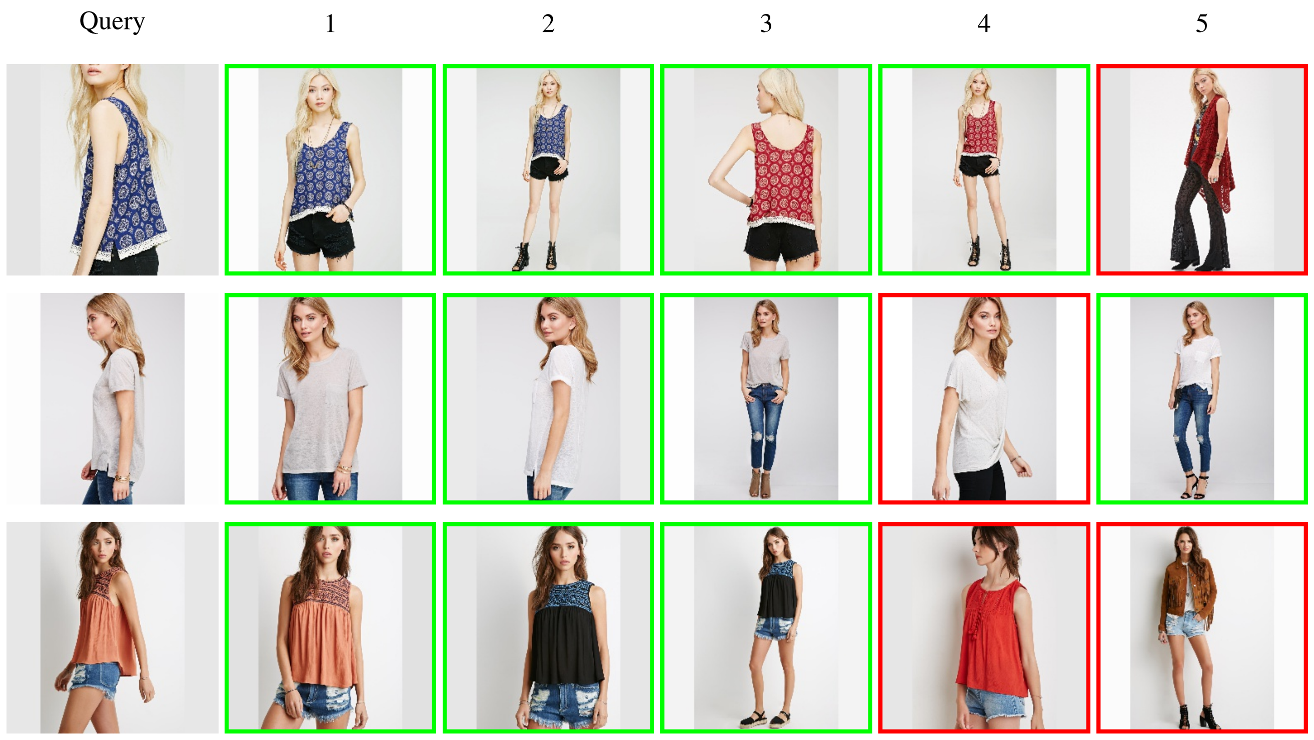

Figure 6. Qualitative results on the In-Shop Clothes Retrieval [6] dataset. We retrieve the 5 most similar images to the query image. Correct results are highlighted green and incorrect results are highlighted red.

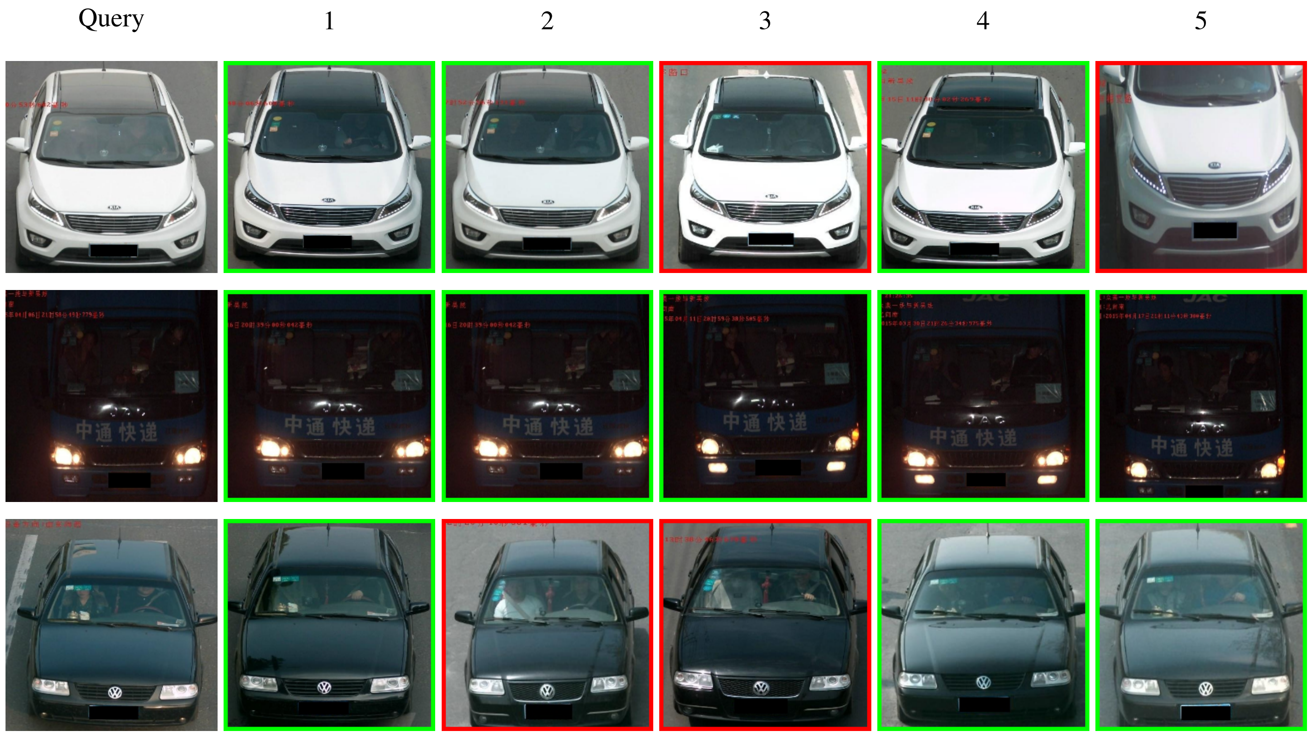

Figure 7. Qualitative results on the VehicleID [5] dataset. We retrieve the 5 most similar images to the query image. Correct results are highlighted green and incorrect results are highlighted red.