Learning Fine-grained Image Similarity with Deep Ranking Supplemental Materials

Anonymous CVPR submission

Paper ID 709

1. Details of the Network Architecture

In this section, we will give the details of the network architecture of the proposed deep ranking model. The triplet-based network architecture for the ranking loss function is illustrated in Fig. 2 and Fig. 3 in the main paper. We will give detailed descriptions of the layers used in the architecture.

We have a *ranking layer* on the top of the network, which takes the embeddings of the three images in a triplet and computes the hinge ranking loss of the triplet:

$$l(p_i, p_i^+, p_i^-) = \max\{0, g + \|f(p_i) - f(p_i^+)\|_2^2 - \|f(p_i) - f(p_i^-)\|_2^2\} \tag{1}$$

where g is a gap parameter that regularizes the gap between the distance of two image pairs: (p_i, p_i^+) and (p_i, p_i^-). The hinge loss is a convex approximation to the 0-1 ranking error loss, which measures the model's violation of the ranking order specified in the triplet.

When the embeddings of the images are normalized to have unit l_2 norm, the hinge loss function (1) can be simplified to

$$l(p_i, p_i^+, p_i^-) = \max\{0, g - 2f(p_i)(p_i^+) + 2f(p_i)f(p_i^-)\} \tag{2}$$

The ranking layer does not have any parameter. During learning, it evaluates the the model's violation of the ranking order, and back-propagates the gradients to the lower layers so that the lower layers can adjust their parameters to minimize the ranking loss.

The *l2 normalization layer* normalize the features to have unit $l2$-norm:

$$\boldsymbol{x}^l = \frac{\boldsymbol{x}^{l-1}}{\|\boldsymbol{x}^{l-1}\|_2} \tag{3}$$

where $\boldsymbol{x}^l$ is the concatenation of all the output feature maps of the l-th layer. This layer does not have any parameter.

The network architecture of the ConvNet in [4] is shown in Fig. 2.

The *convolutional layer*, *subsampling layer* and *local normalization layer* all take overlapping blocks from the feature maps of the previous layer as input. The blocks are slid around its center with a given stride in the feature maps of the previous layer. For example, In convolution, the kernel is applied to the image by placing the kernel over the image to be convolved and sliding it around to center with a given stride size in the feature maps of the previous layer. At each placement the numbers (pixel values) from the feature maps of the previous layer are multiplied by the kernel number that is currently aligned above it. An example of convolution with kernel size 4×4 and stride 2×2 is shown in Fig. 1.

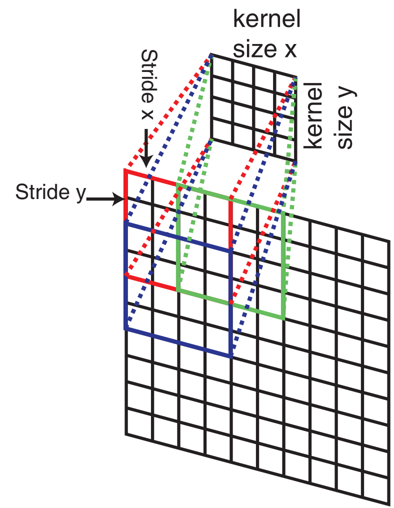

Figure 1. An illustration of convolution with kernel size 4×4 and stride 2×2. The red, blue and green rectangles are the three adjacent blocks for the convolution.

At a *convolutional layer*, the previous layer's feature maps are convolved with learnable kernels and put through the activation function to form the output feature map. Each output map may combine convolutions with multiple input maps. In general, we have

$$\boldsymbol{x}_j^l = a\left(\sum_{i \in M_j} \boldsymbol{x}_i^{l-1} \otimes \boldsymbol{k}_{ij}^l + b_j^l\right) \tag{4}$$

where $\boldsymbol{x}_j^l$ is the j-th feature map of the l-th layer, $\otimes$ is convolution, M_j represents a selection of input maps, and $a(.)$ is an activation function, such as sigmoid function and rectified linear function. In this paper, we use all the feature maps from the previous layers in a convolutional layer, and we employ rectified linear function as activation function. A convolutional layer can be viewed as a set of local feature detector. The parameters of this layer are the kernels $\boldsymbol{k}_{ij}^l$ and offsets b_j^l.

The *subsampling layer* produces downsampled versions of the input maps. If there are N input maps, then there will be exactly N output maps. More formally,

$$\boldsymbol{x}_j^l = down(\boldsymbol{x}_j^{l-1}) \tag{5}$$

where $down(.)$ represents a subsampling function. It takes the average or maximum over each overlapping block in the input feature map. The subsampling layer is called a *max pooling layer* if "maximum" is used in this layer. This layer does not have any parameter.

The *local normalization layer* locally normalizes the feature maps to have zero mean and unit norm, so that they are robust to contrast and illumination changes. The function is defined as:

$$\boldsymbol{x}_{j,i}^l = \frac{\boldsymbol{x}_{j,i}^{l-1} - \bar{\boldsymbol{x}}_{j,i}^{l-1}}{norm(\boldsymbol{x}_{j,i}^{l-1} - \bar{\boldsymbol{x}}_{j,i}^{l-1})} \tag{6}$$

where $\boldsymbol{x}_{j,i}^l$ is j-th feature map at location i for l-th layer. $\bar{\boldsymbol{x}}_{j,i}^{l-1}$ is the mean value of the j-th feature map values for l-th layer in the overlapping block around the location i. $norm(\boldsymbol{x}_{j,i}^{l-1})$ is the l_2-norm of the features map values of the j-th feature maps for l-th layer in the overlapping block around the location i. This layer does not have any parameter.

2. Details of the Optimization

The objective function of the proposed deep ranking neural network is:

$$\begin{aligned} &\min \sum_i \xi_i + \lambda \|\boldsymbol{W}\|_2^2 \\ &s.t. : \max\{0, g + D(f(p_i), f(p_i^+)) - D(f(p_i), f(p_i^-))\} \leq \xi_i \\ &\qquad \forall p_i, p_i^+, p_i^- \text{ for } r(p_i, p_i^+) > r(p_i, p_i^-) \end{aligned} \tag{7}$$

This objective can be converted to unconstrained optimization by replacing $\xi_i = \max\{0, g + D(f(p_i), f(p_i^+)) -$

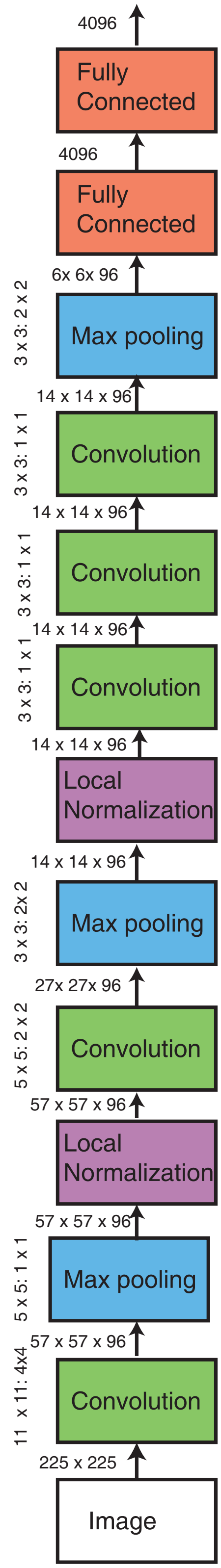

Figure 2. The ConvNet structure in [4]. The number shown next to the arrow is the size of the output image or feature. The number shown at the left side of the box is the size of the kernel and the size of the stride for the corresponding layer.

$D(f(p_i), f(p_i^-))\}$:

$$\min \sum_i \max\{0, g + D(f(p_i), f(p_i^+)) - D(f(p_i), f(p_i^-))\} + \lambda \|\boldsymbol{W}\|_2^2 \quad \forall p_i, p_i^+, p_i^- \text{ for } r(p_i, p_i^+) > r(p_i, p_i^-) \tag{8}$$

where $f(.)$ is the function of the image embedding deep neural network, which can be represented as a composition:

$$f(.) = g_n(g_{n-1}(g_{n-2}(\cdots g_1(.)\cdots))) \tag{9}$$

where $g_l(.)$ is the forward transfer function of the l-th layer. The parameters of layer function g_l is denoted as $\boldsymbol{w}_l$. Then the gradient $\frac{\partial f(.)}{\partial \boldsymbol{w}_l}$ can be represented as: $\frac{\partial f(.)}{\partial g_l} \times \frac{\partial g_l}{\partial \boldsymbol{w}_l}$, and $\frac{\partial f(.)}{\partial g_l}$ can be efficiently computed in an iterative way: $\frac{\partial f(.)}{\partial g_{l+1}} \times \frac{\partial g_{l+1}(.)}{\partial g_l}$. Thus, we only need to compute the gradients $\frac{\partial g_l}{\partial \boldsymbol{w}_l}$ and $\frac{\partial g_l}{\partial g_{l-1}}$ for the function $g_l(.)$

The *ranking layer* does not have any parameters. The gradients of the ranking layer loss (2) with respect to the image embedding inputs $f(p_i), f(p_i^+), f(p_i^-)$ are

$$\frac{\partial l}{\partial f(p_i)} = \begin{cases} 0 & l = 0 \\ -2\left(f(p_i^-) - f(p_i^+)\right) & l > 0 \end{cases} \tag{10}$$

$$\frac{\partial l}{\partial f(p_i^+)} = \begin{cases} 0 & l = 0 \\ -2f(p_i) & l > 0 \end{cases} \tag{11}$$

$$\frac{\partial l}{\partial f(p_i^-)} = \begin{cases} 0 & l = 0 \\ 2f(p_i) & l > 0 \end{cases} \tag{12}$$

The back-propagates algorithm adjusts $f(.)$ so that the distance of $f(p_i)$ and $f(p_i^+)$ is small, and the distane of $f(p_i)$ and $f(p_i^-)$ is large.

To compute the gradient *convolution layers*. Denote the gradient of the activation function $a(\boldsymbol{\delta}_j^l)$ as $\frac{\partial a(\boldsymbol{\delta}_j^l)}{\partial \boldsymbol{\delta}_j^l}$, and the u, v element of $\boldsymbol{x}_j^l$ and $\boldsymbol{\delta}_j^l$ as $E_{u,v}$ and $(\boldsymbol{\delta}_j^l)_{uv}$, respectively. The the gradients of each transfer function $E_{u,v}$ of the convolution layers with respect to the parameters $\boldsymbol{k}_{ij}^l, b_j^l$ are:

$$\frac{\partial E_{u,v}}{\partial b_j^l} = \frac{\partial a(\boldsymbol{\delta}_j^l)}{\partial (\boldsymbol{\delta}_j^l)_{uv}} \tag{13}$$

$$\frac{\partial E_{u,v}}{\partial \boldsymbol{k}_{ij}^l} = \frac{\partial a(\boldsymbol{\delta}_j^l)}{\partial (\boldsymbol{\delta}_j^l)_{uv}} . (\boldsymbol{p}_i^{l-1})_{uv} \tag{14}$$

where $(\boldsymbol{p}_i^{l-1})_{uv}$ is the *patch* that was multiplied elementwise by $\boldsymbol{k}_{ij}^l$ during the convolution in order to compute the element at (u, v) in the output convolution map $\boldsymbol{x}_j^l$.

The the gradient of each transfer function $E_{u,v}$ of the convolution layers with respect to the input $\boldsymbol{x}_{ij}^{l-1}$ are:

$$\frac{\partial E_{u,v}}{\partial \boldsymbol{x}_{ij}^{l-1}} = \frac{\partial a(\boldsymbol{\delta}_j^l)}{\partial (\boldsymbol{\delta}_j^l)_{uv}} . (k_{ij}^l)^* \tag{15}$$

where $(k_{ij}^l)^*$ are the kernel elements that are used to multiple $\boldsymbol{x}_{ij}^{l-1}$ to crate the element at (u, v) in the output convolution map $\boldsymbol{x}_j^l$.

The *subsampling layer* does not have any parameters. The gradients with respect to the input $\boldsymbol{x}_{ij}^{l-1}$ can be computed as:

$$\frac{\partial E_{uv}}{\partial (\boldsymbol{x}_{ij}^{l-1})_{u'v'}} = \begin{cases} \frac{\partial down(\boldsymbol{x}_{ij})}{\partial (\boldsymbol{x}_{ij}^{l-1})_{u'v'}} & \text{If } (\boldsymbol{x}_{ij}^{l-1})_{u'v'} \text{ is used to generate } (u, v) \text{ in output } \boldsymbol{x}_j^l \\ 0 & \text{Otherwise} \end{cases} \tag{16}$$

Since *local normalization layer* and *l2 normalization layer* do not have any parameters, we just need to unnormalize the gradients from the top layer back accordingly.

3. Details of Triplet Sampling

The details of the algorithm can be found in Alg. 1. Readers can refer to [2] for proof of the correctness of this algorithm.

4. Details of the Hand-Crafted Visual Features

The details of the hand-crafted features are listed below:

- Wavelet: The implementation follows [3], which computes weighted L_0 distance between Harr wavelet decompositions of image pairs.
- Color: The best color-based metric uses normalized L1 distance between color histograms in LAB space with a bin size of 4096 ($16 \times 16 \times 16$). We have also tried color histograms in HSV and RGB space and found them to be inferior.
- SIFT [6]-like features are Gabor wavelet texture features on local image patches. Such features are vector-quantized and accumulated across the image to produce histograms. Similarity between histograms are given by their normalized L1 distance. We have tried histograms of 128, 512 and 2048 bins. Performance peaks at 2048 bins.
- SIFT-like Fisher (SF): We followed [7] to compute Fisher vectors from the SIFT-like features as follows: First, we assume the SIFT-like features are generated i.i.d from a Gaussian Mixture Model (GMM) and estimate the parameters of the GMM; Then we compute

Given a set of images $\mathcal{P}$ with category label c_j, total relevance score r_j, and pairwise relevance score $r_{i,j}$.
Initialize the minimum key in all each buffer $M_l = 1$.
for *each image* $p_j \in \mathcal{P}$ **do**
    Compute the key $k_j = u_j^{(1/r_j)}$, where $u_i = \text{uniform}(0, 1)$.
    Find the buffer correspond to the category c_j.
    if *buffer* c_j *is not full* **then**
        Insert the image p_j into the buffer with key k_j.
        Update minimum key for buffer c_j.
    else
        if $k_j > M_{c_j}$ **then**
            Replace image that has minimum key in buffer c_j with p_j.
            Update minimum key for buffer c_j.
        end
    end
    while *no triplet is accepted and the number of tries is less than limit* **do**
        Uniformly sample a query image sample p_i from the buffer c_j.
        Uniformly sample a positive image sample p_i^+ from the buffer c_j, accept it with probability $\min(1, r_{i,i+}/r_{i+})$.
        if *Sample in-class samples* **then**
            Uniformly sample a query image sample p_i^- from the buffer c_j, accept it with probability $\min(1, r_{i,i-}/r_{i-})$.
        else
            Uniformly sample a negative image sample p_i^- from all the images in the other buffers.
        end
        Accept the triplet if the margin criteria is satisfied.
    end
end

Algorithm 1: Online triplet sampling algorithm.

gradient of the features with respect to the parameters of the GMM; Finally, we whiten the gradient to produce the Fisher vector. Similarity between Fisher vectors are judged by their normalized L2 distance. We tried histograms of 128, 512 and 2048 bins. The best performing SIFT-like Fisher vectors use 128 bins.

- Histogram of Oriented Gradients (HOG): A similar implementation to [1] is used. We first resize each image to a desired size (96 x 96), then divide each image into cells (e.g. with cell size 16 x 16). The histogram of oriented gradients features are extracted for each cell, with 32 floats for each cell. Descriptors from different cells are then combined. For image similarity, L1 distance can be computed between two images (L1 distance is found to out-perform norm L1 distance for this application).
- SPMK Texton features with max pooling: Spatial Pyramid Matching Kernel (SPMK) is a way to aggregate features from coarse to fine spatial scales [5, 8]. For each image, 3 x 3 Texon is first extracted, with a dictionary size of 1024. Spatial Pyramid Matching Kernel is then used to aggregate the texton histogram.

5. More Ranking Examples

More ranking examples of the ConvNet, OASIS feature (L1HashKPCA features with OASIS learning) and Deep Ranking are shown in Fig. 3 and Fig. 4.

References

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, pages 886–893. IEEE, 2005.

[2] P. S. Efraimidis. Weighted random sampling over data streams. *arXiv preprint arXiv:1012.0256*, 2010.

[3] A. Finkelstein and D. Salesin. Fast multiresolution image querying. In *Proceedings of the ACM SIGGRAPH Conference on Visualization: Art and Interdisciplinary Programs*, pages 6–11. ACM, 1995.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[5] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178. IEEE, 2006.

[6] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157. IEEE, 1999.

[7] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, pages 3384–3391. IEEE, 2010.

[8] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009.

Figure 3. Comparison of the ranking samples of ConvNet, OASIS feature and Deep Ranking.

Figure 4. Comparison of the ranking samples of ConvNet, OASIS feature and Deep Ranking.