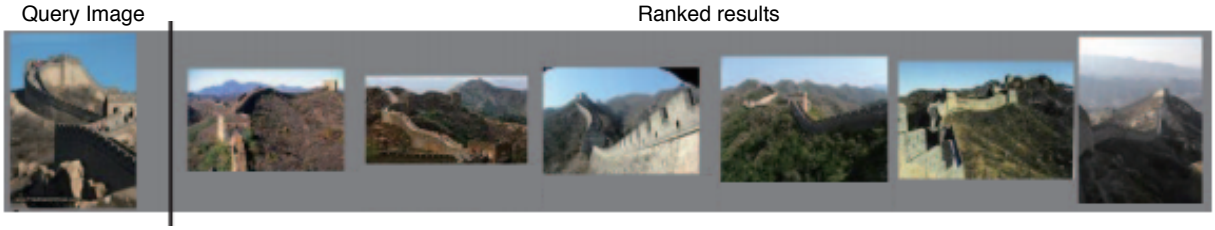
Image Similarity using Deep Ranking

Overview

The goal of this project is to introduce you to the computer vision task of image similarity. Like most tasks in this field, it's been aided by the ability of deep networks to extract image features.

The task of image similarity is retrieve a set of n images closest to the query image. One application of this task could involve visual search engine where we provide a query image and want to find an image closest that image in the database.

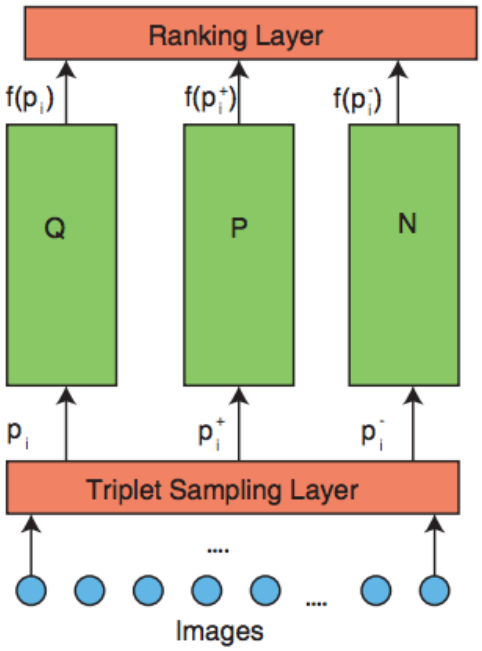


Your task, for this project, will be to implement a simplified version of the pipeline introduced in "Learning Fine-grained Image Similarity with Deep Ranking". We strongly encourage you to read this pager - https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42945.pdf.

Here's another good resource - https://medium.com/@akarshzingade/image-similarity-using-deep-ranking-c1bd83855978.

Project Description

You will design a simplified version of the deep ranking model as discussed in the paper. Your network architecture will look exactly the same, but the details of the triplet sampling layer will be a lot simpler. The architecture consists of 3 identical networks (Q,P,N). Each of these networks take a single image denoted by p_i, p_i^+, p_i^- respectively.



p_i: Input to the Q (Query) network. This image is randomly sampled across any class.

p_i^+: Input to the P (Positive) network. This image is randomly sampled from the ***SAME*** class as the query image.

p_i^-: Input to the N (Negative) network. This image is randomly sample from any class **EXCEPT** the class of p_i.

The output of each network, denoted by $f(p_i)$, $f(p_i^+)$, $f(p_i^-)$ is the feature embedding of an image. This gets fed to the ranking layer.

Ranking Layer

The ranking layer just computes the triplet loss. It teaches the network to produce similar feature embeddings for images from the same class (and different embeddings for images from different classes). g is a gap parameter used for regularization purposes.

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

D is the ***Euclidean*** distance between $f(p_i)$ and $f(p_i^{+/-})$.

$$D(p,q) = D(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

g is the gap parameter. We use the default value of 1.0, but you can tune it if you'd like (make sure it's positive).

Testing stage

The testing (inference) stage only has one network and accepts only one image. To retrieve the top n similar results of a query image during inference, the following procedure is followed:

1. Compute the feature embedding of the query image.
2. Compare (euclidean distance) the feature embedding of the query image to all the feature embeddings in the training data (i.e. your database).
3. Rank the results - sort the results based on Euclidean distance of the feature embeddings.

Triplet Sampling Layer

One of the main contributions of the paper is the triplet sampling layer. Sampling the query image (randomly) and the positive sample image (randomly from the same class as the query image) are quite straightforward.

Negative samples are composed of two different types of samples: in-class and out-of-class. For this project, we will implement out-of-class samples only. Again, out-of-class samples are images sampled randomly from any class except the class of the query image.

Other Notes

1. We recommend you use the ResNet architecture you implemented in your previous homework.
2. Use the data loader - it'll help a lot in loading the images in parallel (there is a **num_workers** option)
3. Sample your triplets beforehand, so during training all you're doing is reading images.
4. Make sure you load your model with pre-trained weights. This will greatly reduce the time to train your ranking network.
5. Blue Waters training time is approximate 24-36 hours, so please start early.

Dataset

For this project, we will use the Tiny ImageNet (https://tiny-imagenet.herokuapp.com/) dataset (located at **/projects/training/bauh/tiny-imagenet-200/**). This dataset consists of 200 different classes with 500 images each. The image size is 64x64. The dataset also includes a validation set of 10000 images (with classes).

You will use the training set to train your model and the validation set to test it.

Questions/Deliverables

1. Your code

2. Report your accuracy - Target accuracy: 60% or higher

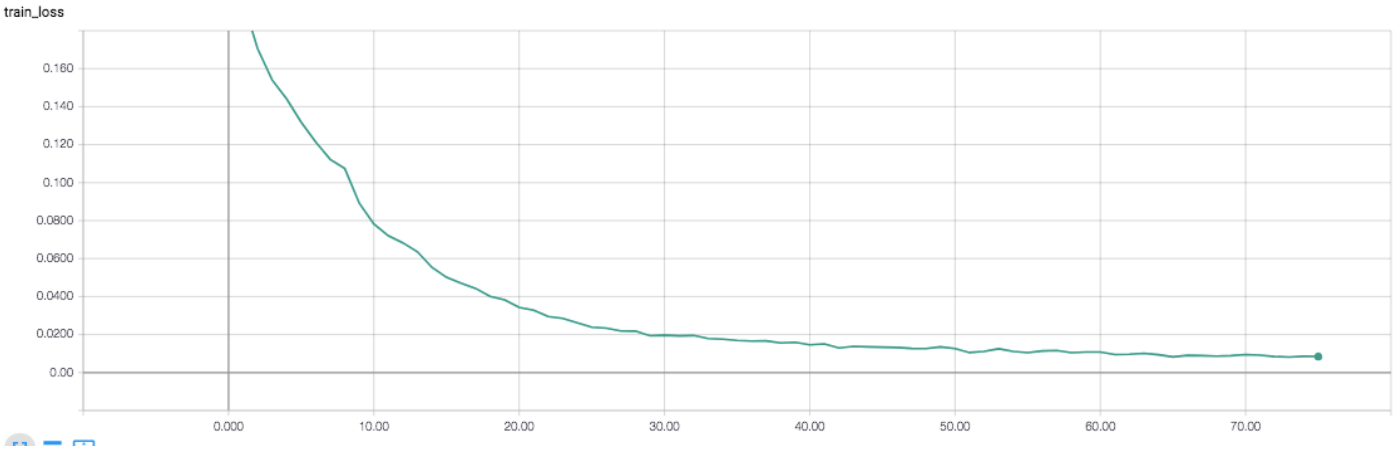
3. Describe your implementation

4. Quantitative results
   - Show a plot of your training loss
   - Include a table of similarity precision for both your training and test
     - The percentage of triplets being correctly ranked

5. Sample 5 different images from the validation set (each from a different class)
   - Show the top 10 ranked results from your pipeline.
   - Show the euclidean distance of the ranked results from the query image
   - Show the bottom 10 ranked results (along with their euclidean distance)

6. Describe at least one way in how you can improve the performance of your network
   - Be as descriptive as possible
   - Hint: a plausible answer is in the paper. Make sure you understand what they did before you write it down as your answer

Our implementation



Architecture: ResNet101
Initial learning rate: 0.001
Accuracy: 67%

Examples: