Image Ranking

Aidana Karipbayeva Askar Mukhanov Bryan Lunt

1. Introduction

Search-by-example is a relatively new field that involves finding images similar to the query image. It has made a dramatic impact on modern image search engines. A key component of this method is an image similarity metric.

Most of the image similarity models are category-level image similarity. This means that two images are considered to be similar if they belong to the same class. One major drawback of these models is that they fail to rank images based on visual and semantic similarities. In other words, a "blue car" is more similar to a "black car" than a "red car" is.

The work by Wang et al. [1] tackles this problem using a Triplet Sampling Layer. The layer samples a triplet: a query image, a positive image (image from the same class as a query image), negative image (an image from a different class than a query image). The model learns the fine-grained image similarity information.

In this project we implemented a DeepRanking model similar to that proposed by Wang et al. The model to be implemented consists of three main components: a Triplet Sampling layer, a deep neural network, and a Ranking Layer. The Triplet Sampling layer samples a triplet of images. Each image in a triplet is independently fed to a deep neural network that maps an image to a point in Euclidean space. The Ranking layer on the top evaluates a hinge-like loss of a triplet. The hinge-like loss is as follows:

$$l(q,p,n) = max\{0, g + D(f(q), f(p)) - D(f(q), f(n))\} \quad (1)$$

where q, p and n are query, positive and negative images. $f(.)$ is the image embedding function that maps an image to Euclidean space. $D(f(q), f(p))$ is Euclidean distance between points $f(p)$ and $f(q)$:

$$D(f(q), f(p)) = ||f(q) - f(p)||_2^2 \quad (2)$$

Finally, g is a gap parameter that regularizes the gap between the distance of image pairs.

In the original work a negative image in a triplet could be sampled from the same class as the query image. However, in our implementation a negative image is sampled from all classes but the class of the query image.

2. Details of the Approach

The first component of the system is a Triplet Sampling layer. The dataloader indexes samples by class and at training time presents the model with batches of query image (Q), positive images chosen at random from the same label as the query (P), and negative images chosen at random from any other label (N). Our implementation generates triplets on-the-fly, giving better coverage of the $4,965,050,000,000$ possible triplets in the training data.

The second component in the system is the model, which takes an input RGB image of any size and creates a vector representing some summary properties of the image. The details of individual models used in our experiments will be covered in another subsection.

The last component of our system is a Ranking layer that evaluates the training function in eq (1)

The system uses batched training with an algorithm such as SGD or Adam, training via backpropagation through the Ranking layer and model. Batches are drawn from the Triplet Sampling layer.

Finally, we created a rough image search product by using the vector representations of all training images as keys in a high dimensional spatial database. Specifically, we used a K-Nearest Neighbors database from scikit-learn, though there are other options better suited to a real-world product.

2.1. Original Network

The original multiscale network has three branches (Figure 1). The first branch is a pre-trained ResNet-18. ResNet networks are deep neural networks that were trained to classify objects on ImageNet dataset. Thus ResNet-18 can learn fine-grained image features and, is a viable network for our task. ResNets are trained on images with size of 224x224. Therefore, we decided to upsample ResNet-18

inputs as it may help to achieve better performance. Other two branches are much more shallow, and have only convolutional layers. These two parts have less invariance and capture the visual appearance.

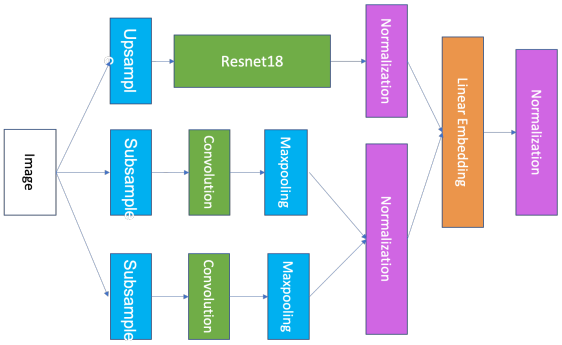

Figure 1: Architecture of the original network

2.2. Our Networks

In addition to multiscale network from the paper we also trained three neural networks with different structures. An architecture of our first network (Figure 2) is similar to that of the original network. The only difference is that this network has only one shallow part. This part contains two convolutional, activating and pooling layers.

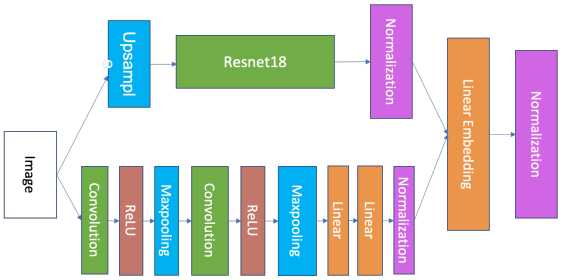

Figure 2: ResNet-18 and one shallow branch

We wanted to see if having more shallow branches helps to capture the visual appearance better. Our second model (Figure 3) has same structure as the original model but with additional shallow part.

Since we aim to build an image classifier, we decided to try ResNet-18 without upsampling inputs or including additional parts and layers. This is our final model.

2.3. Dataset

The TinyImageNet dataset was used for training and testing purposes. It contains 100,000 64x64 pixel color images as training data, and 10,000 as test data. This dataset consists of 200 different classes with 500 images in each.

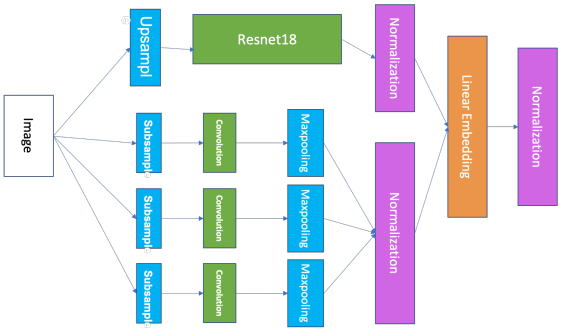

Figure 3: ResNet-18 and three shallow branches

2.4. Training

Models were trained using the Adam algorithm and a weight decay of 0.00001 with an adaptive learning rate starting at 0.0005. Batch size was 200 and models were trained for roughly 50k batches. This required roughly 20 hours of total wallclock time each using a Tesla K20X GPU on the NCSA BlueWaters supercomputer. A gap parameter of 2.0 was used in the loss function.

Models were trained on 80% of the training data, with 10% used for training-time cross-validataion. Training time for each model was approximately 21 hours. Training loss of the original model is shown in Figure 4.

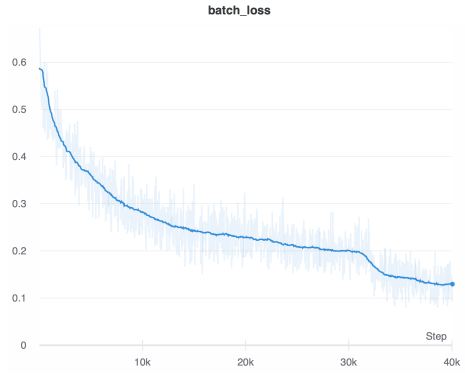

Figure 4: Training loss

3. Results

3.1. Accuracy Metrics

Accuracy on triplet, *KNN-1* and *KNN-30* were used to evaluate performance of models from previous sections.

Accuracy on triplet is defined as the number of correct triplets divided by the total number of triplets in a subset. A triplet is said to be correct if embedding of a positive image

is closer to an embedding of a query image than does an embedding of a negative image:

$$D(f(q), f(p)) < D(f(q), f(n))$$

KNN-1 is defined as the fraction of query images whose nearest neighbor in Euclidean space is from the same class over total number of query images in a subset.

KNN-30 is defined as the fraction of query images whose thirty nearest neighbors in Euclidean space contains an image from the same class over total number of query images in a subset. This metric was chosen and implemented to allow direct comparison with Dharawat al. who previously completed the same class project.

3.2. Model Performance Comparison

Performance of models implemented are shown in Table 1. It can be seen that a simple network having only pre-trained ResNet-18 outperforms more sophisticated networks. Multiscale networks with ResNet-18 and additional shallow parts have comparable accuracy on training and testing triplets. It also seems that having more shallow parts negatively affects *KNN-1* and *KNN-30*.

Model	Accuracy on training triplets	Accuracy on testing triplets	KNN-1	KNN-30
ResNet18	95%	86%	18%	40%
Original Network	91.5%	80.4%	8%	27.4%
ResNet18 + one conv branch	93%	80.7%	12.1%	29.1%
ResNet18 + three conv branches	92.2%	80.5%	9.8%	25.4%

Table 1: Comparison of models

Apparent discrepancies between test triplet accuracy and KNN-1 performance are easily explained by the difference in number of out-of-class examples. While triplet accuracy gives one positive and one negative example, KNN-1 accuracy requires that one of the N positive examples have better similarity than one of the $(199)N$ negative examples.

3.3. Ranked Examples

Five randomly selected query images with top and bottom 10 ranked results are shown below. Figures also display Euclidean distances between query and ranked images. Ranked images are labeled as O if they are of the same class, and X otherwise.

These images were obtained by model that has only one shallow part. This model has lower accuracy power than ResNet-18 only model. However, it captures the visual appearance better, therefore, produces better ranked images.

Figure 5: First query image

Figure 6: Second query image

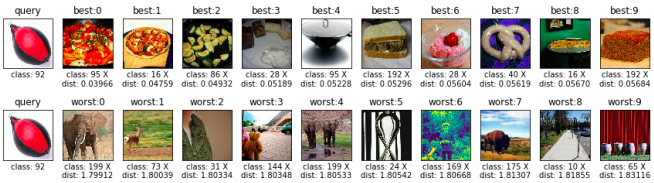

Figure 7: Third query image

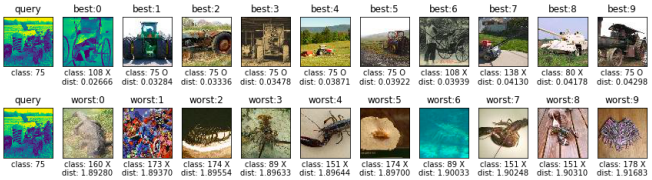

Figure 8: Fourth query image

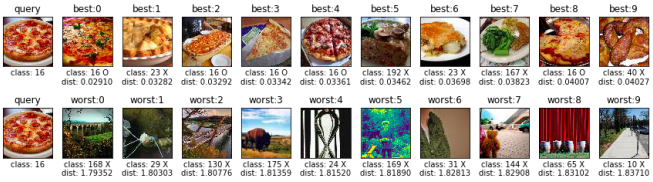

Figure 9: Fifth query image

4. Discussion

For models that output a normalized vector, that is, a point on the unit sphere, it may make more sense to use

cosine similarity as the distance function in the triplet loss score. Euclidean distance makes more sense in a linear Euclidean space.

The goal of models in this paper is to classify images, something ResNet-18 is trained for. That is the reason we believe ResNet-18 without additional branches outperformed all other networks. Therefore, to get more accurate results we may want to train deeper networks: ResNet-50 and/or ResNet-101.

References

[1] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen and Y. Wu. *Learning Fine-grained Image Similarity with Deep Ranking*. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42945.pdf

[2] GitHub repository https://github.com/UIUC-CS547-2021sp-Group36/project