Image Search Engine

Research paper for course in Computer Intelligence

Božidar Mitrović
Nikola Andrić Mitrović

Abstract

Convolutional Neural Networks have been excelling at classification tasks.TODO. However, in this paper, we represent a network capable of learning similarities in images. We used a triplet network model, which aims to learn useful representations by distance comparisons proposed by Hoffer et al. [1.]. We then use these representations to find the most similar images to the input image from the dataset. In the end, we propose a way to speed up our model to make much faster predictions.

1 Introduction

With the recent great success of deep neural networks in computer vision, deep metric learning has attracted increasing attention and has been applied to various tasks. One of which is similarity learning. Similarity learning aims to learn an embedding space, where the embedding of the same classes in training are encouraged to be closer, while dissimilar ones are pushed apart from each other with the goal of generalizing embeddings. These embeddings are later used to distinguish never before seen images belonging to the classes used during training.

Many recent deep metric learning approaches are built on pairs of samples. Formally, their loss functions can be expressed in terms of pairwise cosine similarities in the embedding space. This group of methods is referred to as pair-based deep metric learning. We use a triplet loss approach that belongs to this group of methods. Another notable metric learning that belongs to the pair-based metric learning group is the Siamese Network in which a contrastive loss is used to train the network to distinguish between similar and dissimilar pairs of examples [2.]. A contrastive loss favors a small distance between pairs of examples labeled as similar and large distances for pairs labeled dissimilar. Triplet Network uses a similar strategy to learn the similarity between triplets of images.

2 Triplet Network

Given are triplets of objects (x , x^+, x^-), where x$^+$ and x$^-$ represent an image from the same class and an image from a different class respectably. The goal is to learn a function f such that for any new triplet of objects (x , x^+, x^-), it obeys $f(x, x^+) < f(x, x^-)$ [Figure 1].

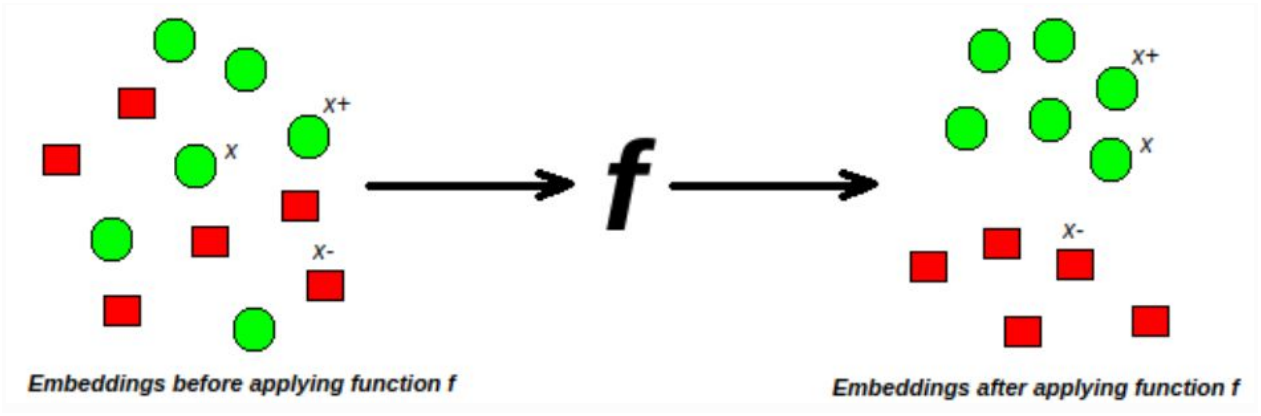
x+
x
x-
f
x+
x
x-
Embeddings before applying function f
Embeddings after applying function f

Figure 1

1.]A Triplet network consists of 3 instances of the same feedforward network with shared parameters. When fed with a triplet sample, the network outputs the embedding for each element in the triplet.

The task is to determine which of x$^+$ and x$^-$ is of the same class as x. Training is done by firstly computing the L2 norms for (x - x$^+$) and (x - x$^-$).

$$Norm1 = \|Net(x) - Net(x^+)\|_2 , \; Norm2 = \|Net(x) - Net(x^-)\|_2$$

In order to output a comparison operator from the model, a SoftMax function is applied to both norms - effectively creating a ratio measure.

$$d_+ = \frac{e^{Norm1}}{e^{Norm1} + e^{Norm2}} \text{ , } d_- = \frac{e^{Norm2}}{e^{Norm1} + e^{Norm2}}$$

For better results, we also apply MSE on the soft-max result.

$$Loss = \|(d_+, d_- - 1)\|_2^2$$

The objective is to make $Loss \to 0$ which happens when $\frac{Norm1}{Norm2} \to 0.$

By using the same shared parameters network, we allow the back-propagation algorithm to update the model with regard to all three samples simultaneously.

3 Dataset and preprocessing

Our model experimented with the CIFAR-10 dataset. It consists of 60000 32x32 color images of 10 classes (of which 50000 are used for training only, and 10000 for test only). Since we only worked with one dataset, data augmentation was applied because we found ourselves in need of more data. Dataset was sampled into batches, where each batch consisted of *batch size* triplets of images. Two of which are of the same class (x and x^+), and the third (x^-) of a different class [Figure 2]. Training epoch iteration consisted of *batch size* triplets (randomly chosen each epoch). The remaining 10000 images were used for testing and were not augmented or involved in the training in any way.

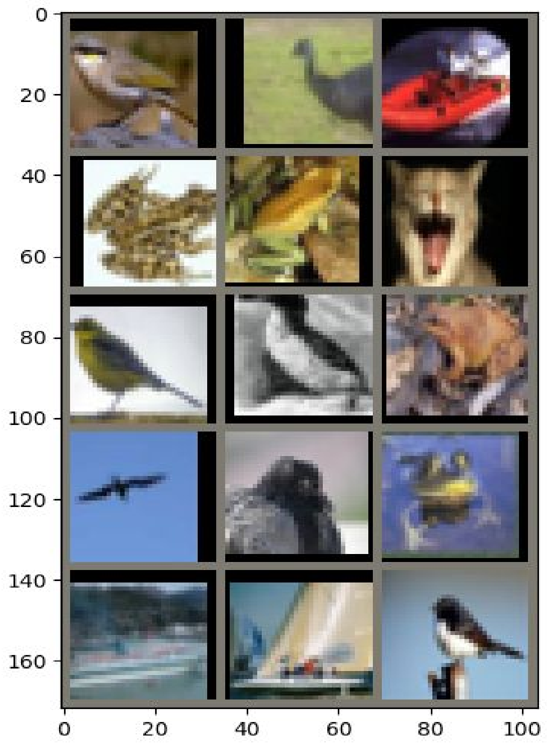

Figure 2

4 Model architecture

We used a deep neural network, consisting of 4 convolutional and 3 2x2 max-pooling layers. ReLU activation function is applied between two consecutive layers, and after the last layer, the Sigmoid function is used. Filter sizes for each of the 4 convolutional networks were as follows: {5,3,3,2}, and feature map dimensions: {3,64,128,256,128} where a 128 vector was the final embedded representation of the network. Usually, in convolutional networks, a fully-connected layer is used for classification. In our network, this layer is removed, as we are interested in a feature embedding only. Also, to control overfitting we added a Dropout between each of the convolutional layers with parameter p = 0.5 (dropout is not used on the last layer).

5 Results

Alongside dropout regularization to avoid overfitting we also used early stopping with a tolerance of 50 epochs. Both SGD and Adam optimizer achieved minimum loss after around 150 epochs. We measured accuracy by generating triplets from the test set and looking at the softmax output, summing the number of all the correctly predicted similar images (where the probability of the x^+ was lower than the probability of x^-, note that the desired softmax value of x^+ is 0). Accuracy won't always be the same since the triplets are randomly generated from the test set. Training with SGD resulted in 0.7387 average accuracy [Figure 3], while the best accuracy was achieved using Adam optimizer with a learning rate of 0.0005 and average accuracy of 0.915 where the best model was extracted from 146. Epoch [Figure 4].

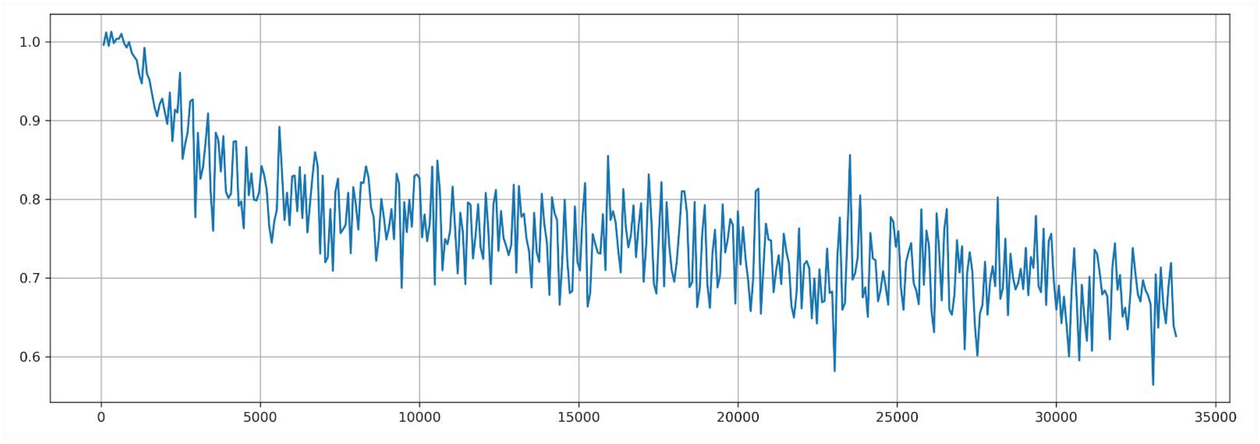

Figure 3

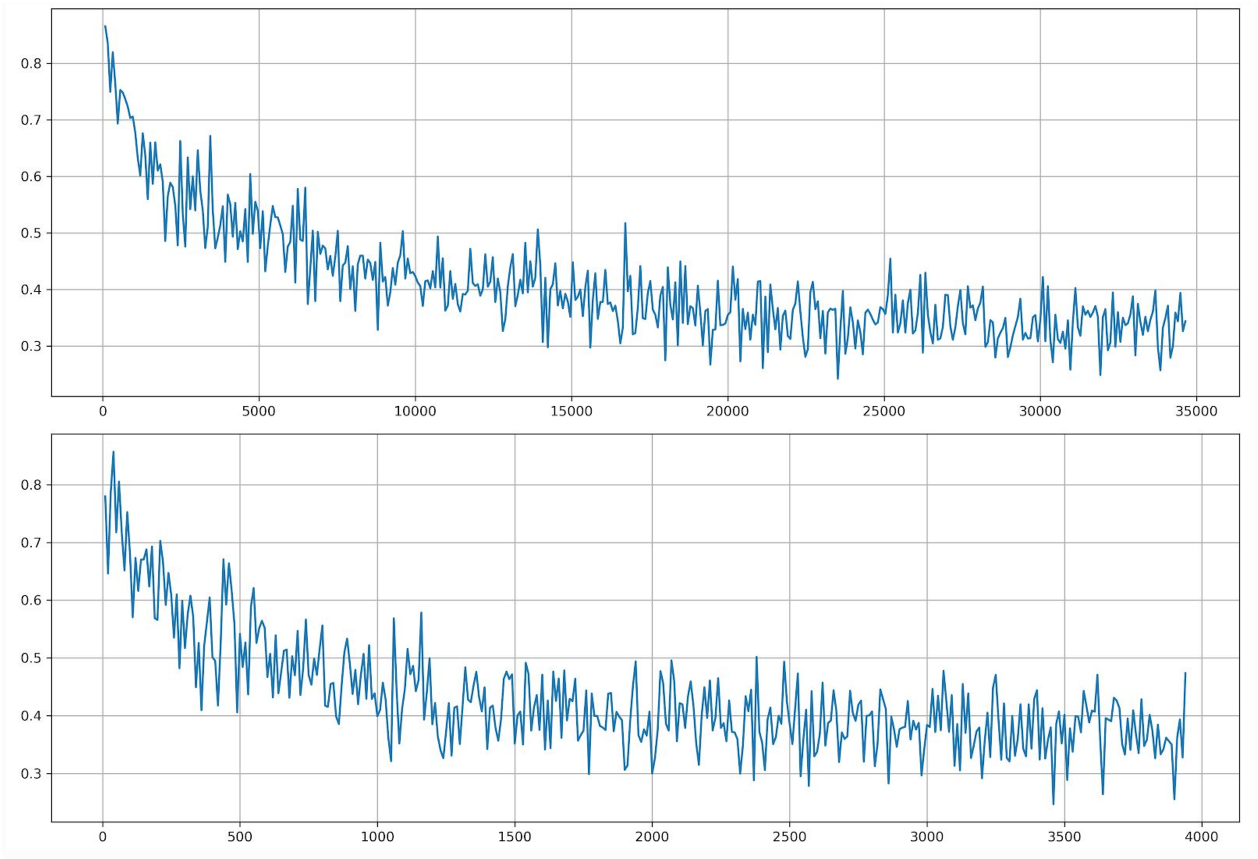

Figure 4, the upper plot is training loss, the lower plot is validation loss

Finally, we use the trained model to output the n most similar images to the input image [Figures 5, 6, note that the left topmost image is the input image]. The shortcoming of the model is that it is not able to distinguish between cats and dogs as they are similar to him since the model was not specifically trained to distinguish between them [Figure 7]. Similarly to human perception, the model can be tricked by images that resemble input image in some way [Figure 8, the second image is a sleeping cat but at the center of it horse-like shape can be interpreted].

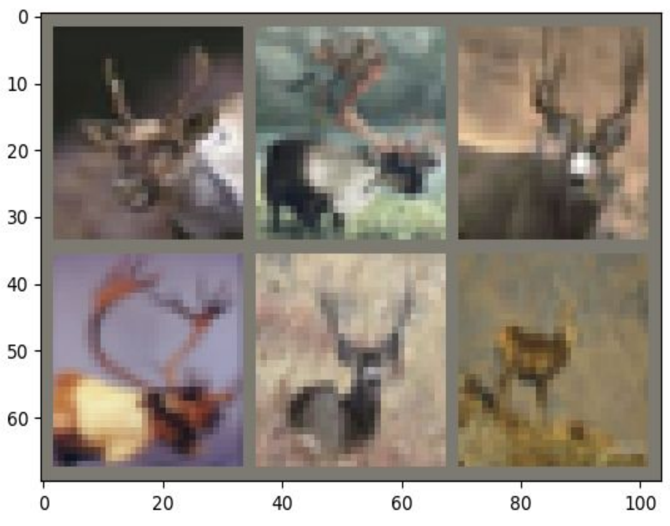

Figure 5

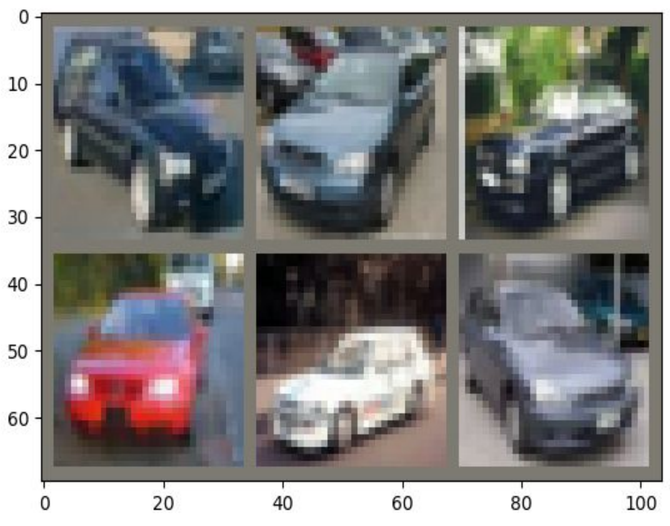

Figure 6

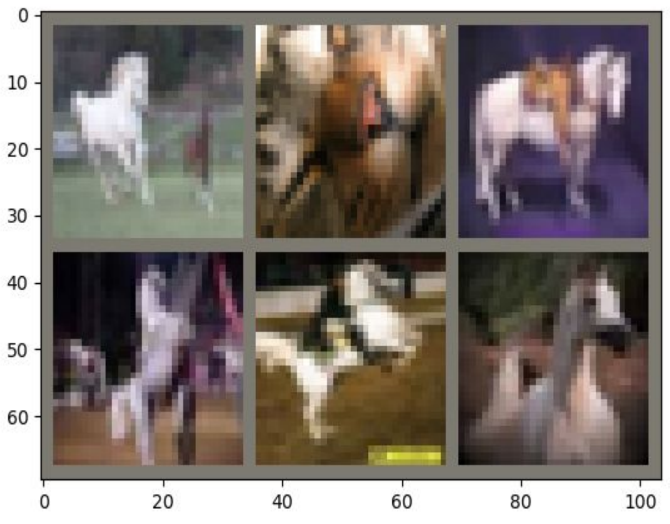

Figure 7

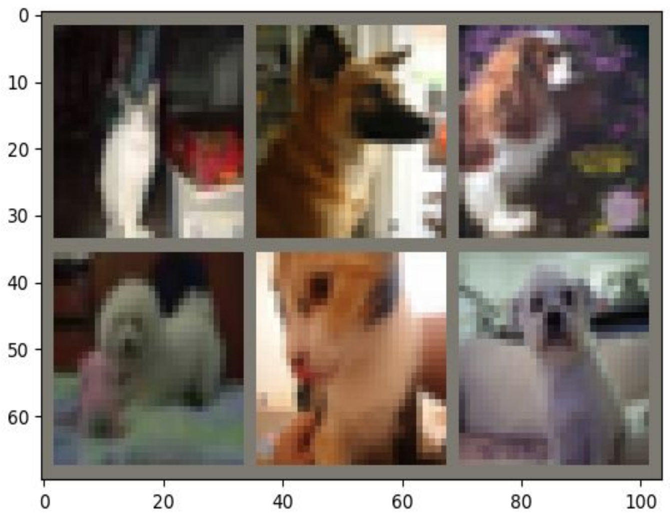

Figure 8

6 Conclusion

We represent a triplet network capable of finding the most similar images in a dataset. In the future it is possible to generate embedding once for the whole dataset we are working with and save them. This would allow us to find similar images much faster since there would be no need to pass them through the network every time. Another good quality of this approach is that it is easily scalable since with the expansion of the dataset we only need to pass new images through the network and save their embeddings alongside the old ones.

References:

1. Hoffer E., Ailon N. (2015) Deep Metric Learning Using Triplet Network. In: Feragen A., Pelillo M., Loog M. (eds) Similarity-Based Pattern Recognition. SIMBAD 2015. Lecture Notes in Computer Science, vol 9370. Springer, Cham.

2. S. Chopra, R. Hadsell and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 539-546 vol. 1, doi: 10.1109/CVPR.2005.202.

3. Sean Bell and Kavita Bala. 2015. Learning visual similarity for product design with convolutional neural networks. ACM Trans. Graph. 34, 4, Article 98 (August 2015), 10 pages.

4. 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. CoRR, abs/1404.4661, 2014.

5. Xun Wang, Xintong Han, Weilin Huang∗ , Dengke Dong, Matthew R. Scott Malong Technologies, Shenzhen, China, Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning

6. H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In CVPR, 2016.