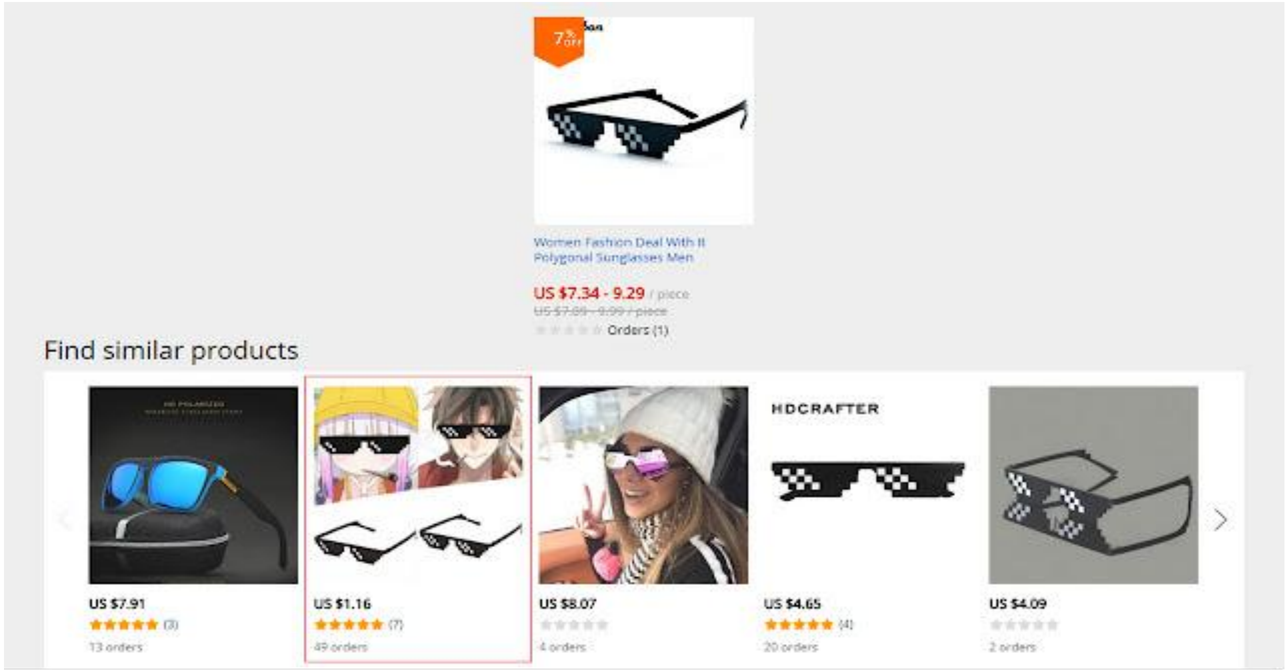
Metric learning in online shop use-case

Task: finding similar products by image



7% OFF

Women Fashion Deal With It Polygonal Sunglasses Men

US $7.34 - 9.29 / piece

~~US $7.89 - 9.99 / piece~~

Orders (1)

Find similar products

US $7.91 (3) 13 orders

US $1.16 (7) 49 orders

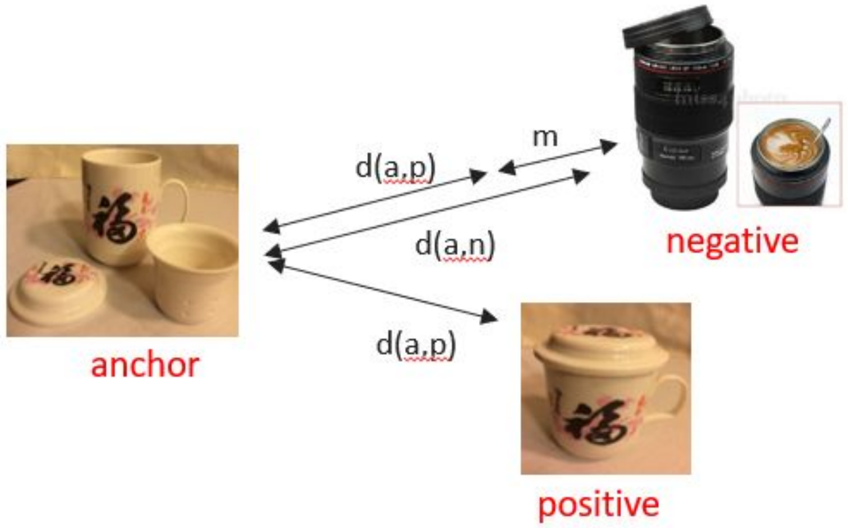
US $8.07 4 orders

US $4.65 (4) 20 orders

US $4.09 2 orders

Used method: similarity learning

Rather than learning to classify the objects we learn to measure the similarity between two objects using **Deep Metric Learning**. So we can be free from problems connected with classification such as close to unlimited amount of classes or little amount of examples for certain classes.

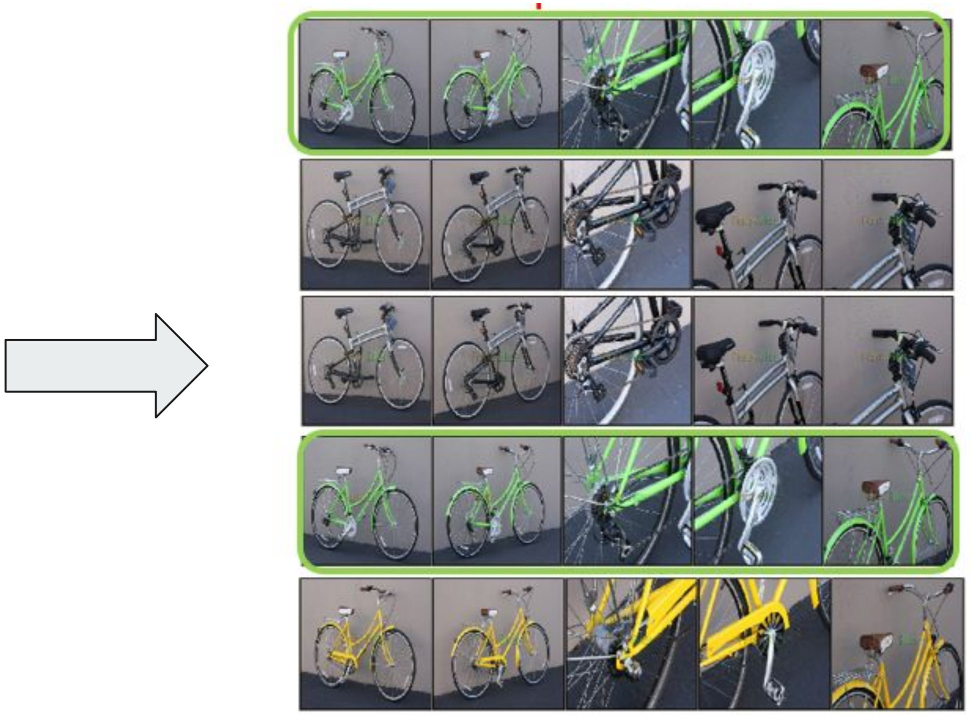


The key is using hard triplet loss to form embedding space of objects in our dataset:
L(a,p,n) = max(d(a,p) – d(a,n) + m,0)

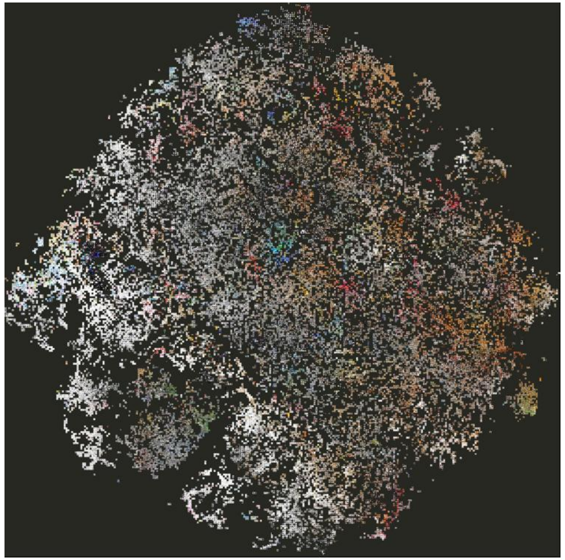
Special formation of batches is also required

As a result, model has to be able to find photos of the same object given the query photo in k top

Similarity learning: input and output



Stanford Online Products dataset



There are 12 categories of products in this dataset: bicycle, cabinet, chair, coffee maker, fan, kettle, lamp, mug, sofa, stapler, table and toaster;

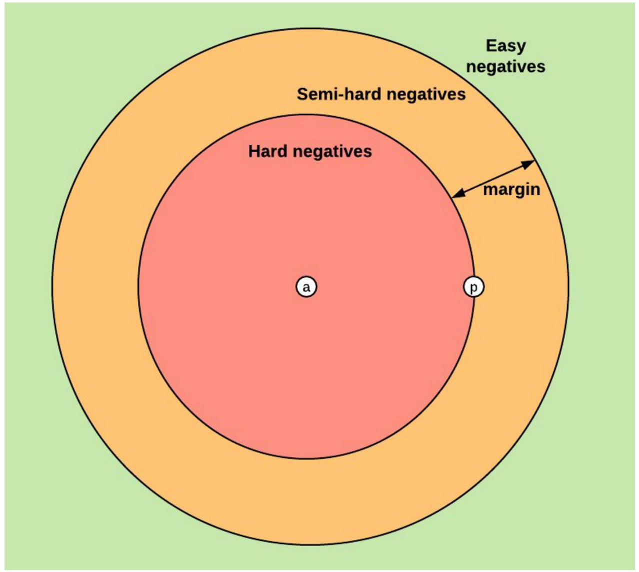
120,053 images in the dataset in total;

22,634 lots/products in the dataset in total.

Forming triplets for loss function

Based on the definition of the loss, there are three categories of triplets:

- easy triplets: triplets which have a loss of 0, because *d(a, p) + margin < d(a, n)*
- hard triplets: triplets where the negative is closer to the anchor than the positive, i.e. *d(a, n) < d(a, p)*
- semi-hard triplets: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss: *d(a, p) < d(a, n) < d(a, p) + margin*



We implemented hard triplet loss which ignores easy and semi hard triplets and uses only hard triplets. It takes less time for model to train.

Mining strategy: offline vs online

Offline mining

finding triplets at the beginning of each epoch for instance

This technique is not very efficient since we need to do a full pass on the training set to generate triplets. It also requires to update the offline mined triplets regularly.

Online mining

computing useful triplets on the fly, for each batch of inputs

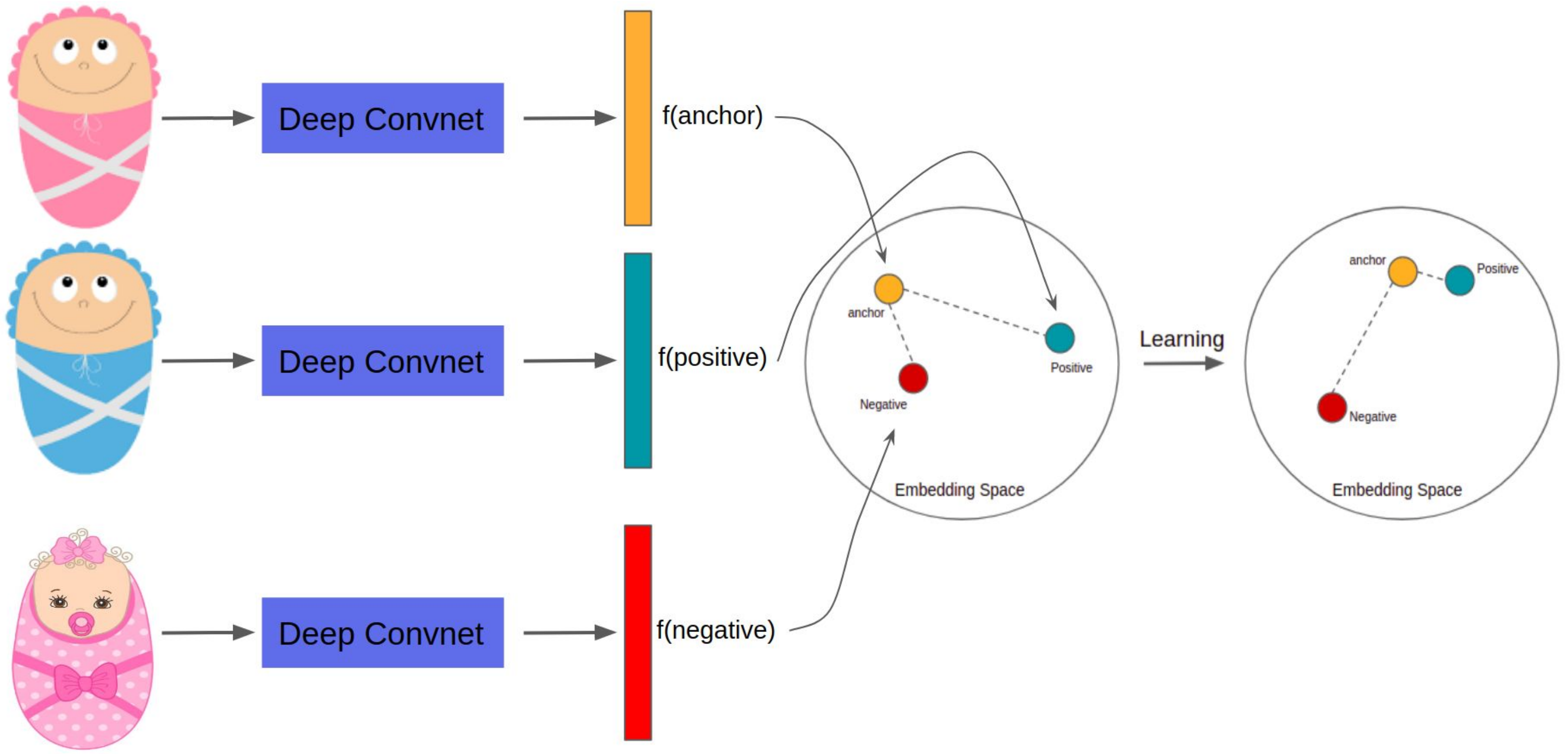
This technique gives you more triplets for a single batch of inputs, and doesn’t require any offline mining. It is therefore much more efficient.

Mining strategy: offline vs online

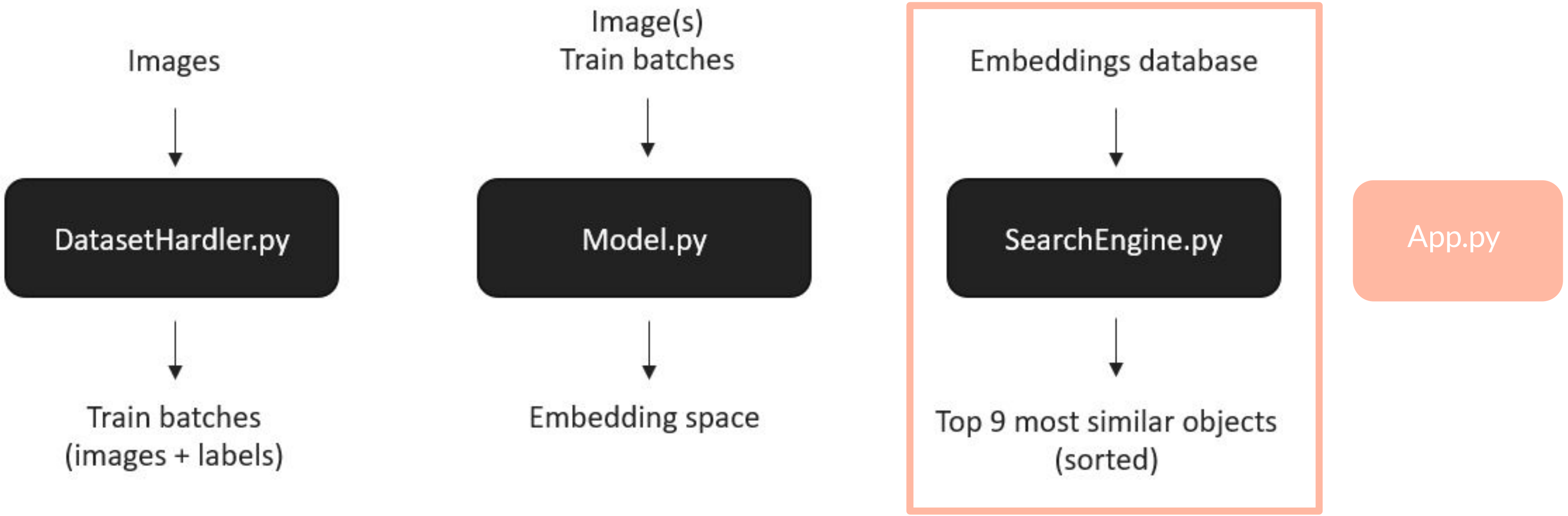
Initially, we followed the first strategy. We were generating triplets ourselves, but almost all of them came out easy triples, which slowed down our learning. Therefore, we have abandoned offline mining.

Then we moved on to online mining, in which we do not form triplets ourselves, but only form batches. The network forms triples itself and chooses hard ones, thus making learning process faster.

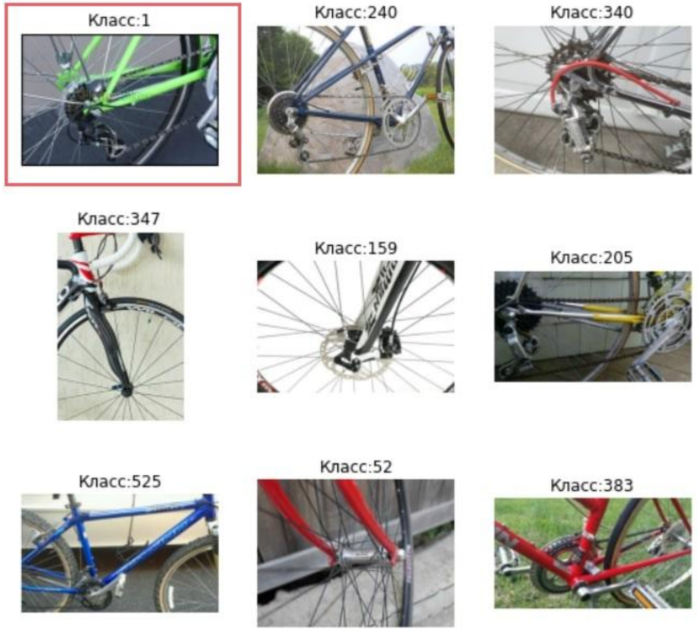
Triplet network



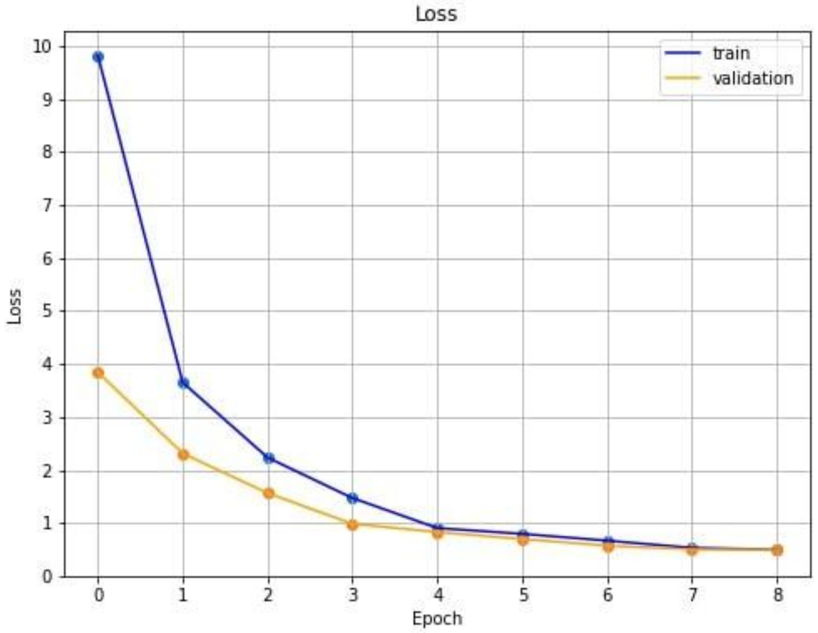
Project structure



Results



Working example. The first picture is anchor, the others are predicted and sorted from the closest to the farthest.



Loss function visualized

Possible improvement

It could be beneficial to split the database of embeddings into classes.
So, having previously predicted an image class, we will search for similar images only within its class:

- This will decrease the search space, thus evidently increasing its speed.

- Predictions are likely to become more accurate, since objects from other classes will not be taken into account (and not returned as a result)

Also other, more up-to-date, architectures could be used (currently ResNet50 is used)

Thank you for your attention!