Master in
Computer Vision
Barcelona

Image Retrieval

Module: C5

Group: 7

Students: Cristian Gutiérrez
Iñaki Lacunza
Marco Cordón
Merlès Subirà

Index

a) Image retrieval with pre-trained image classification model.

b) Train the model on metric learning (Siamese network).

c) Train the model on metric learning (Triplet network).

d) Visualize the learned image representation of each of the previous tasks a-c.

e) Image Retrieval on COCO with Faster R-CNN or Mask R-CNN.

Task (a): Image retrieval with pre-trained image classification model.

Employed Dataset: MIT_SPLIT

- Is a dataset composed by **2688 images** divided into **8 different classes:** coast, forest, highway, inside city, mountain, open country, street, and tall building.

- **1881 images** will conform the **train set** (catalogue of our retrieval system).

- **807 images** are used as **test set** and will be the query images, and we use them to retrieve the most similar images from the catalogue.

8 classes

Coast	Forest	Highway	Inside city	Mountain	Open country	Street	Tall building
244 train	227 train	184 train	214 train	260 train	292 train	212 train	248 train
116 test	101 test	76 test	94 test	114 test	118 test	80 test	108 test

Employed Metrics:

Precision at k=1 (Prec@1): Tells us if the first retrieved image for each query is correct or not.
Precision at k=5 (Prec@5): Tells us the percentage of images retrieved correctly in the 5 first images of each query image.
Mean Average Precision (mAP). Is the mean of the Average Precision (AP), the AP penalizes the models that are not able to sort the retrieved images with the correct images leading the set.

Pre-trained model: RES-Net 18

- We will load the model and its weights and we are going to **remove the last layer.** Moreover we will **include a embedding layer** with size **2096**.

- The **retrieval process** will be executed using the **K-NN** method. The number of neighbours (K) and the distance used will be obtained by an **Optuna hyperparameter optimization.**

Task (a): Image retrieval with pre-trained image classification model.

OPTUNA hyperparameter optimization:

After applying the optimization the higher results in terms of the sum of mapk1 + mapk5 are:

- KNN neighbors: 45
- euclidean metric: euclidean

Retrieval pipeline:

Like the model is pre-trained we are not going to train the model again. So we are going to use it to obtain the features of each image and later use these features to fit the KNN model with the train set images (catalogue). Once the KNN is fitted we can use it to obtain the K similar images of our test images (query).

Quantitative results:

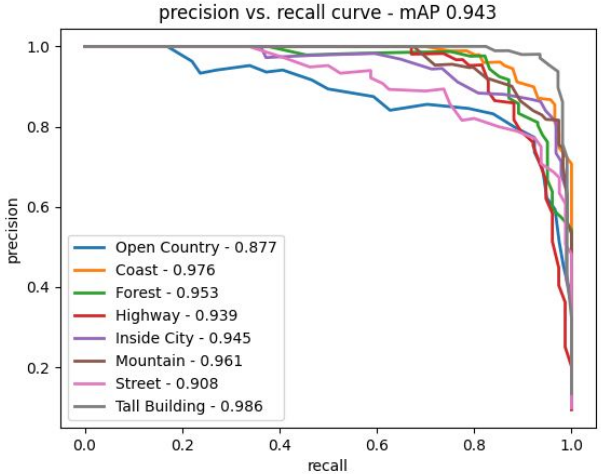
* (All runs of the project have been computed on a RTX 3090)

The metrics obtained after the inference in our retrieval system are:

Mean average precision: 0.8867601697348436
Precision@1: 0.8525402726146221
Precision@5: 0.8009913258983892

Run time: 3s *

So the results are very good, and if we analyze the **precision vs recall curve** class by class we can see similar results to the Week 1. The class open country is the lowest, the reason is the similar features of this class with others, especially with coast, highway and mountain images. The tall building class is the highest and is nearby perfect, this also has a lot of sense because the features of tall buildings are very different to the rest of classes so the distance between it and the rest will be higher so the retrieval will be easier.



Task (a): Image retrieval with pre-trained image classification model.

Qualitative results:

Query image: street

As we can see in this example the retrieval images are from the same class as the query so te system is working correctly. If we analyze the images we can propose that the region of the image that gives the relation between them is the sky and the building's top parts, since they are very similar in all the images.

Top-5 nearest neighbors and their distance:

street

11.559606552124023

street

11.675782203674316

street

11.990601539611816

street

12.204532623291016

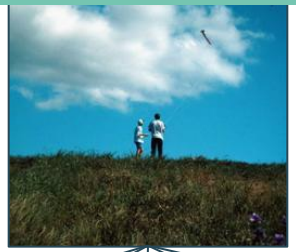
street

12.235445976257324

Task (a): Image retrieval with pre-trained image classification model.

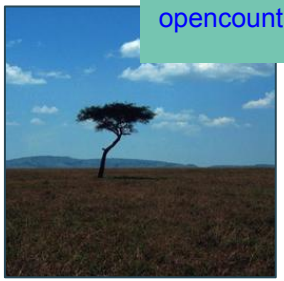
Qualitative results:

Query image: opencountry

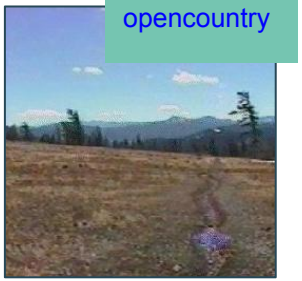


As we can see in this another example the opencountry class sometimes retrievals false neighbors. The main reason of these false matchings are the similarity between these classes in some aspects like the skyline or the presence of trees and grass in the road edges. But anyways the results are pretty good because we are not using any siamese or any training.

Top-5 nearest neighbors and their distance:



9.181400299072266



10.528480529785156

11.124317169189453

11.200284004211426

11.506510734558105

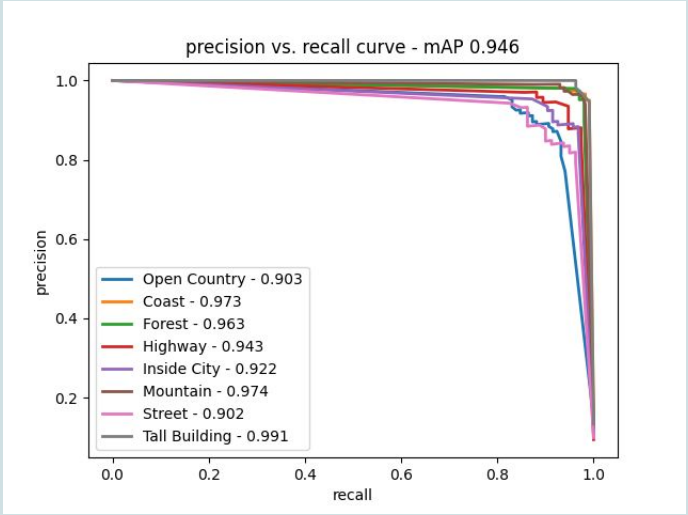
Task (b): Train the model on metric learning (Siamese network)

For this task we trained a Siamese network with pre-trained ResNet-18 to perform the image retrieval. We used **contrastive loss** as loss function. We have employed the `pytorch_metric_learning` library and especially the class `losses.`

Again we have applied an **Optuna hyperparameter optimization,** but this time to also obtain the best value for the training parameters: batch size, embed size. Moreover, we have included a pair miner to select hard examples to train the network. The network has been trained on 50 epochs.

The values obtained with the Optuna fine-tuning are:

- KNN neighbors: 45
- euclidean metric: euclidean
- batch size: 128
- embed size: 2096



The results obtained with this new implementation are:

Mean average precision: 0.9409382585284318
Precision@1: 0.9330855018587361
Precision@5: 0.9333333333333332 **Run time:** 127s

We have overscored the previous results with the implementation of the siamese network and now we have very high values. If we take a look to the precision vs recall curve, it follows the same pattern as in the previous task but with higher values in all classes. With the quantitative results we can confirm that we have improved our system, now we're going to take a look to the qualitatives ones.

Task (b): Train the model on metric learning (Siamese network)

Qualitative results:

Query image: street

The images are very similar to the previous task, but in the previous task the system already gave good retrievals for our query image, so we didn't expect any improvement. However, as we will see later when we show the results of the learned image representations, the distance between neighbors has been reduced.

Top-5 nearest neighbors and their distance:

street

5.87372913019383833

street

6.73773983923773689

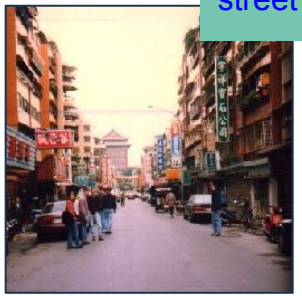
street

6.98891873732234847

street

7.73928301301933334.

street

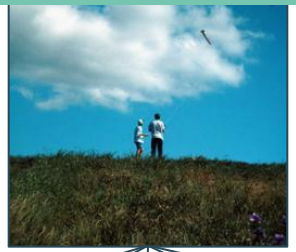


7.93456232232983654

Task (b): Train the model on metric learning (Siamese network)

Qualitative results:

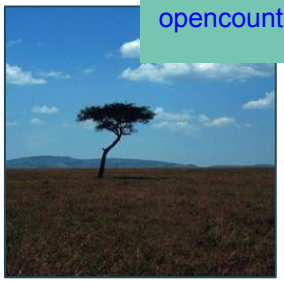
Query image: opencountry



Again very similar results to the first task, but in this example there are a true improvement because now the 5 retrieval images are from the same class and in the previous task not. Moreover, again the distances are shorter between neighbors.

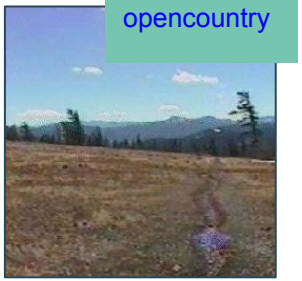
Top-5 nearest neighbors and their distance:

opencountry



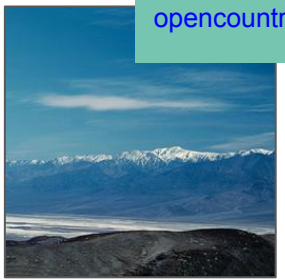
3.4567828372329387

opencountry



3.52848098765456

opencountry



4.1876567898622345

opencountry

5.39990055478754

opencountry

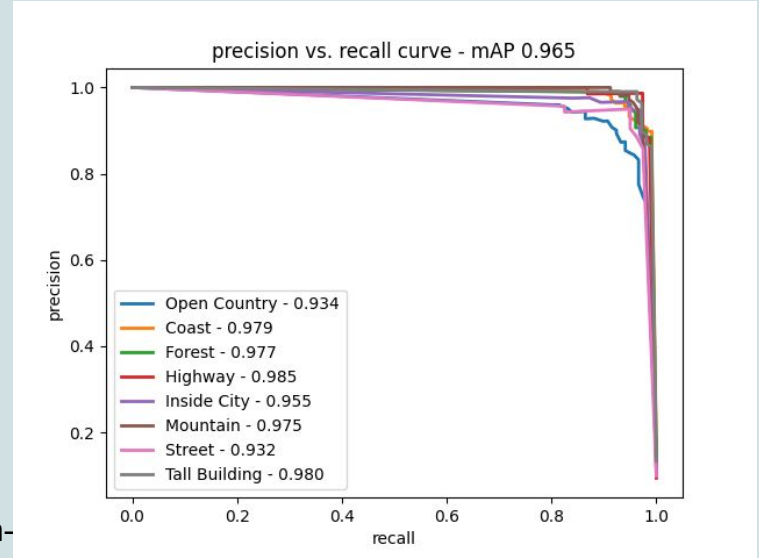
5.50129864357800

Task (c): Train the model on metric learning (Triplet network)

This task is very similar to the previous one, with the only difference that now we are going to use a triplet network instead a siamese one. The pipeline is the same but now we are use the **triplet margin loss** with a **margin of 0.1** as a loss function. The network has been trained on 50 epochs.

The hyperparameter employed are the same as in task b with siamese network:

- KNN neighbors: 45
- euclidean metric: euclidean
- batch size: 128
- embed size: 2096



The results obtained with this third implementation are:

Mean average precision: 0.9544750741731427
Precision@1:0.9467162329615861
Precision@5: 0.9449814126394054 **Run time:** 124s

Again we have overscored the previous retrieval system. We have very high values that assure us that practically all the retrieval images will be from the same class of the query image. The precision vs recall curve has the most of the values near to 1, and again the worst class is opencountry, but with a 0.93 that is a very very good result. So we can confirm that this is the best retrieval system and that works nearby perfection.

Task (c): Train the model on metric learning (Triplet network)

Qualitative results:

Query image: street

The best results are the same but the distances are much shorter between them. The principal difference between this and the previous method will be easily analyzed in the task d, when we plot the learning graphs.

Top-5 nearest neighbors and their distance:

2.9165337139938736

3.45678567873689

5.14678353673820378

5.4567890019362738

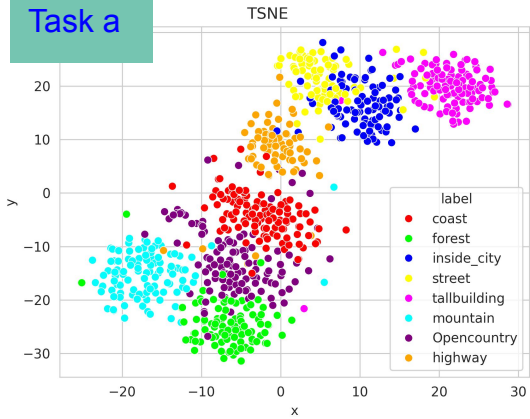
5.93456232232983654

Task (d): Visualize the learned image representation of each of the previous tasks a-c

After analyzing the qualitative and quantitative results in the 3 different retrieval systems that we have used, we will plot the visual representation of the metric learning.

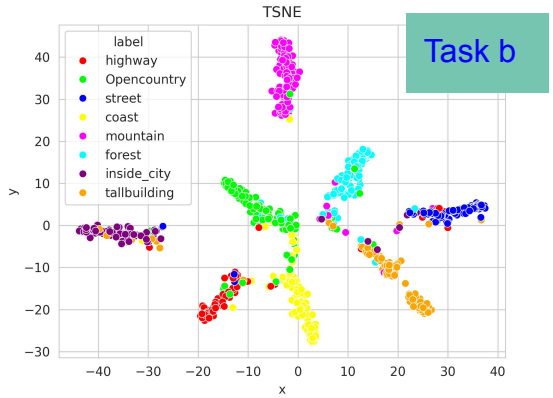
First, we show the pre-trained Resnet-18 model (task a). As we can see in the right graph the images form 8 groups, but these groups are not perfectly separated and there are some overlap between them. Moreover we can see a big correlation between the quantitative results, especially the open country images that like we have said are easily confused with other classes.

Task a



In the results of the second task using the siamese network, we can see how the results improve. Now the classes are more separated so the correct retrieval of images is easier now. Moreover the distance between same class images are shorter. There still are some errors between classes but we can see how the use of the siamese network gives better results.

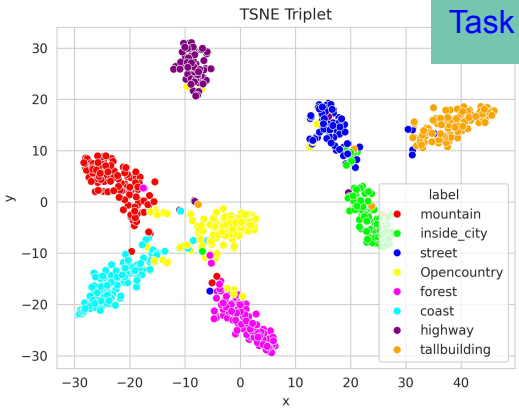
Task b



Task (d): Visualize the learned image representation of each of the previous tasks a-c

In the last retrieval system, we have obtained the highest results in terms of average precision. Analyzing the visual metric learning results we can see how now the classes are very well separated and there are hardly any false classification errors. Moreover the groups are more rounded than the siamese ones, where they were more elongated, so the distances between same class images are again shorter.

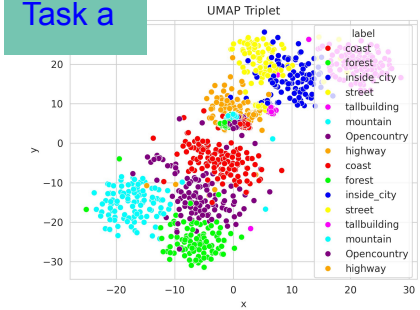
Task c



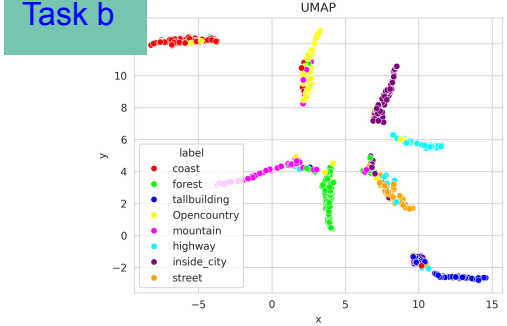
Other representation results:

Apart from the TSNE visualization method we have implemented other ones as the **U-MAP,** with very similar results.

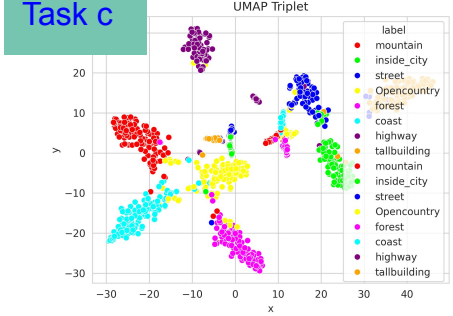
Task a



Task b

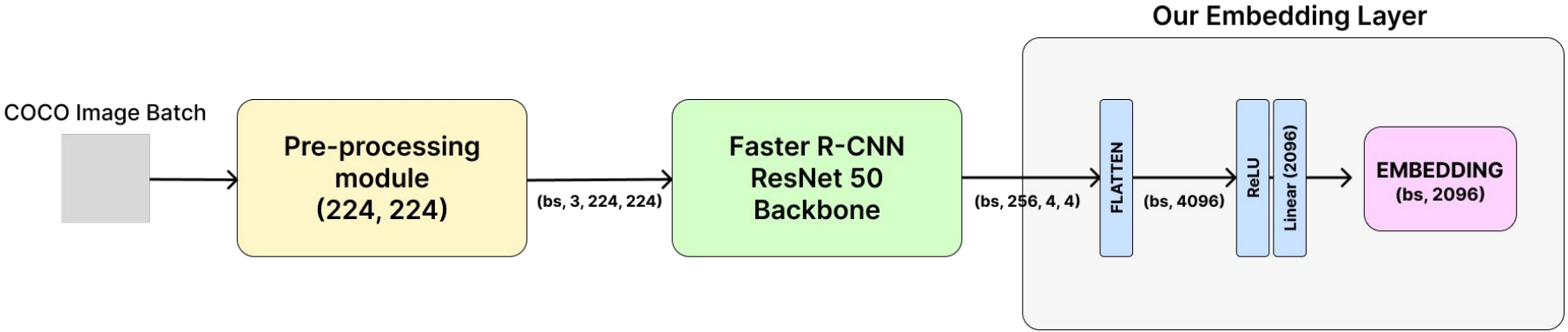


Task c



Task (e): Image Retrieval on COCO with Faster R-CNN

In order to perform Image Retrieval on COCO with the pre-trained Faster R-CNN, we made use of PyTorch's implementation, that uses a ResNet 50 as the backbone to generate features. To do this we had to follow a similar procedure as previous tasks and substitute the ROI heads Box predictor by our own Embedding Layer.



Note that in order to have the same output from Faster R-CNN feature extraction pipeline we had to resize all the COCO images to a common size of (224, 224).

We then apply a flatten onto the features and we pass them through and activation, and project them to an EMBED of 2096.

Task (e): Image Retrieval on COCO with Faster R-CNN

In order to apply Self-supervised (Contrastive) learning, we decided to build a **1:1 dataset**, that is, each anchor contains 1 positive and 1 negative sample.

Also, we applied hard-mining in an **offline manner** and stored the dataset into a `.pkl` file to make efficient runs to test different hyperparameters.

Our dataset contains `(anchor, positive, negative, anchor labels)`, we sampled as positive the sample that contained the maximum common labels with anchor and as negative the one that NOT contained any common label with anchor + positive labels.

Hard-mining Example: The sampling is done **randomly** to avoid repetitions, furthermore the negative sample must not contain any of the labels present in the anchor nor the positive sample.

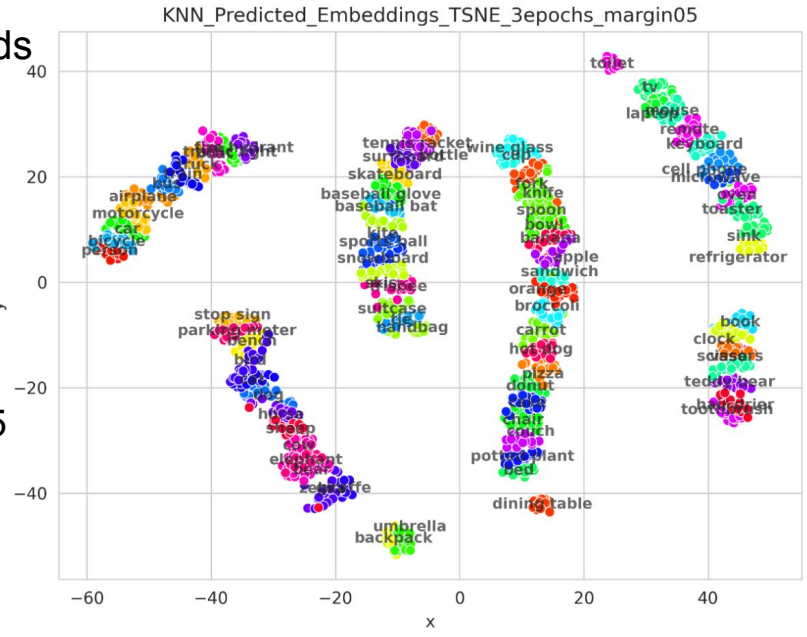
```
Random anchor:   19694  with ['1', '6']
Positive sample: 103356 with ['1', '6', '27']
Negative sample: 105008 with ['64', '65']
```

5 hours to build the whole contrastive dataset !!!! O(n^2)

Task (e): Image Retrieval on COCO with Faster R-CNN

The hard work done during the offline contrastive dataset building payed off really well. With just 3 epochs of training, 1:30h of training, we obtain this embeddings on the **validation set**.

To visualize it, we computed the centroids of each category and plotted the text for reference.



person	baseball glove	train
orange	chair	cat
dining table	laptop	giraffe
fork	donut	bird
bottle	bed	zebra
vase	sink	surfboard
pizza	tv	horse
airplane	toaster	fire hydrant
truck	scissors	teddy bear
stop sign	mouse	apple
motorcycle	keyboard	tennis racket
skateboard	sandwich	couch
bench	cup	toothbrush
backpack	wine glass	oven
refrigerator	broccoli	remote
clock	book	toilet
skis	baseball bat	boat
kite	bicycle	bear
handbag	tie	cow
suitcase	cell phone	frisbee
carrot	bus	elephant
spoon	dog	parking meter
knife	sports ball	hot dog
bowl	microwave	banana
umbrella	snowboard	sheep
traffic light	cake	hair drier
car	potted plant	

<u>Hyper-parameters:</u>

Embedding size: 2096
Triplet Loss margin=0.5
Epochs: 3
Training time: 1:30 h
Offline dataset: 5 h
LR: 2e-5
KNN n neigh = 1

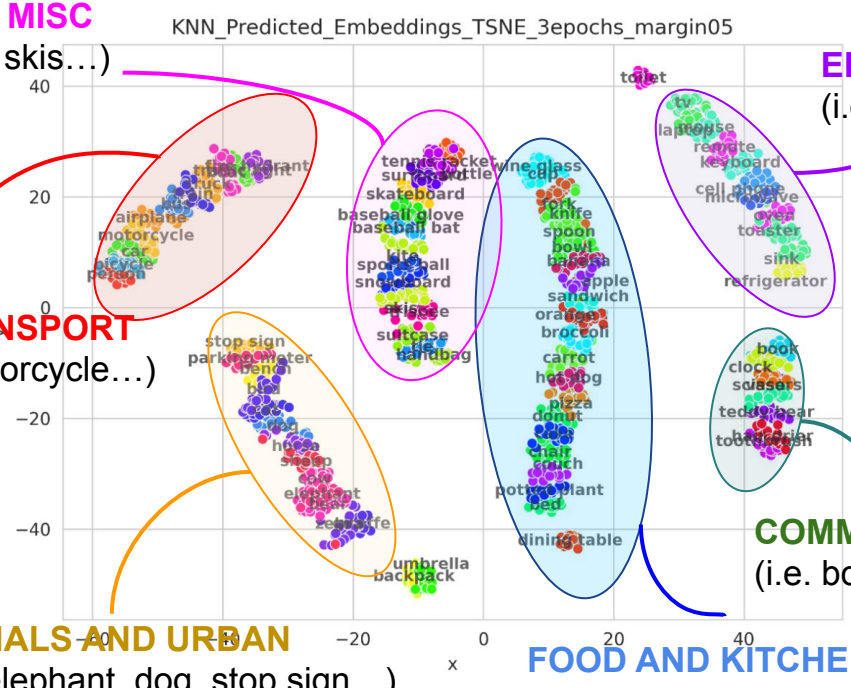
Validation Acc KNN = 97.65%

Task (e): Image Retrieval on COCO with Faster R-CNN

The resulting is our named groups seen in the embedding results for the validation set, notice that there are some outlier groups such as `toilet` or `{backpack, umbrella}`

SPORT ACTIVITIES AND MISC
(skateboard, baseball bat, skis…)

LIVING AND TRANSPORT
(person, truck, motorcycle…)



ELECTRONICS
(i.e. keyboard, cell phone…)

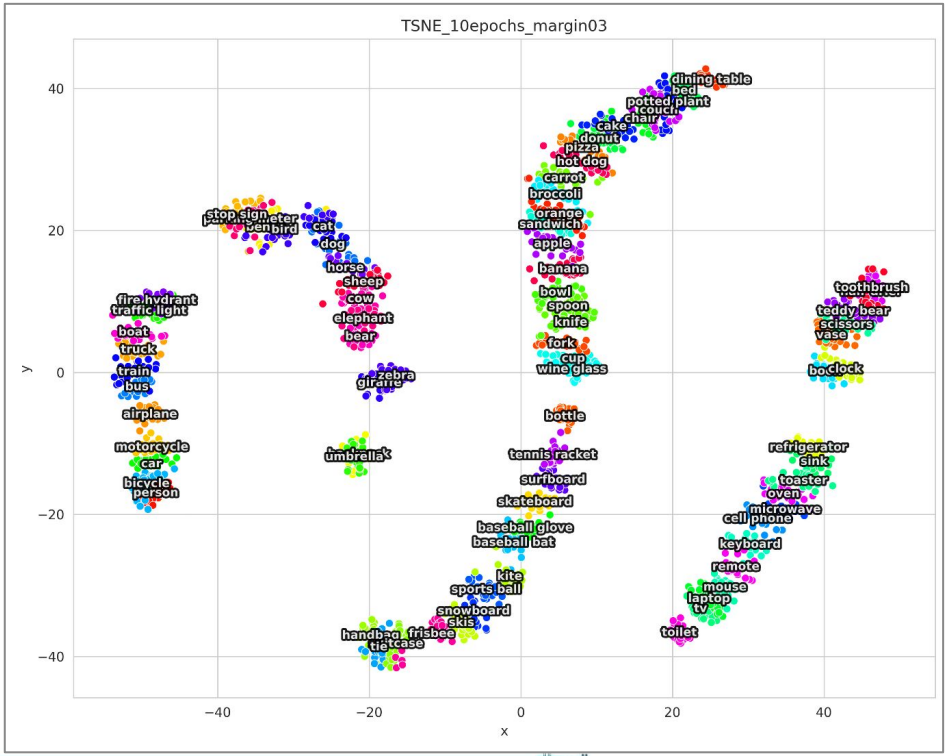
COMMON UTILITIES
(i.e. book, scissors, toothbrush…)

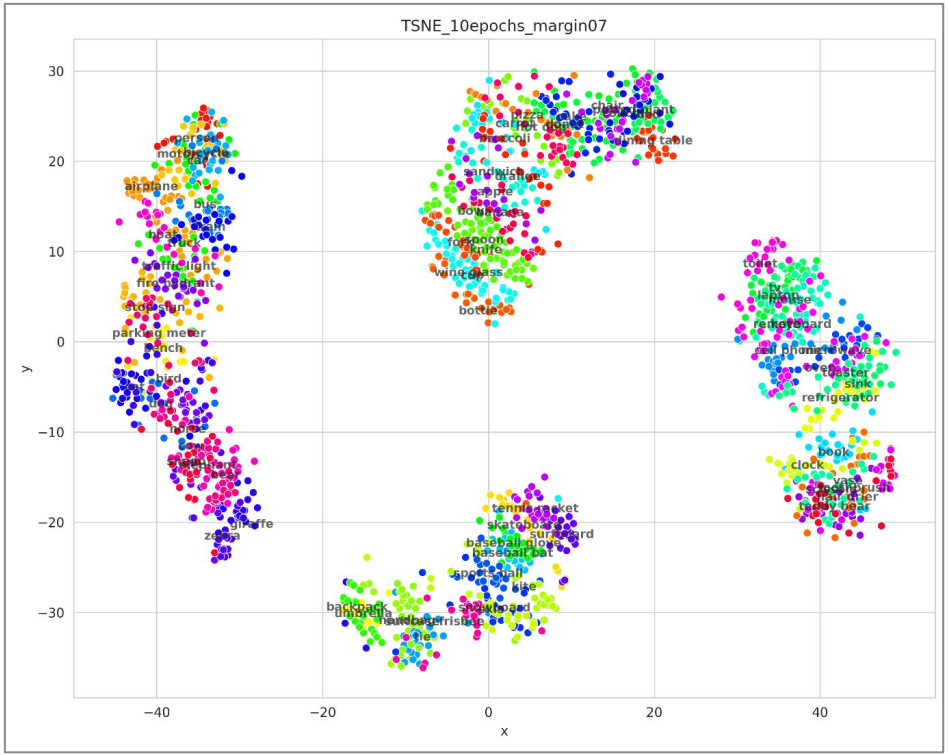
ANIMALS AND URBAN
(i.e. elephant, dog, stop sign…)

FOOD AND KITCHEN
(i.e. broccoli, knife, spoon…)

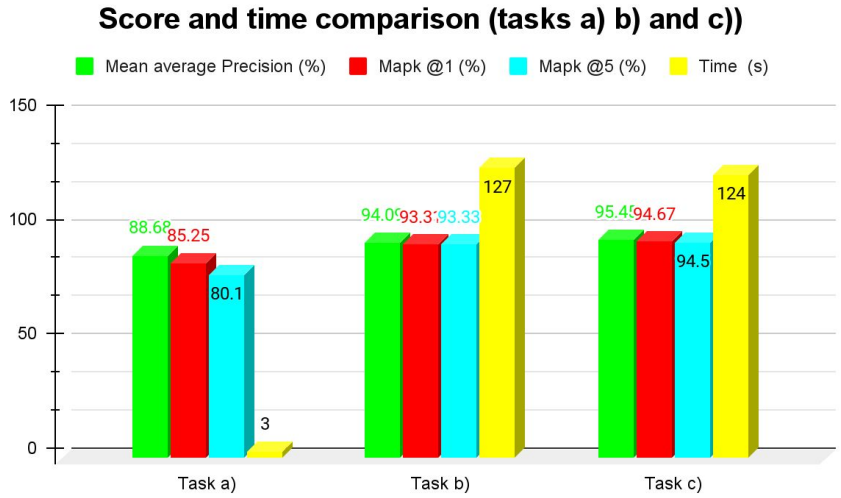
Task (e): Image Retrieval on COCO with Faster R-CNN

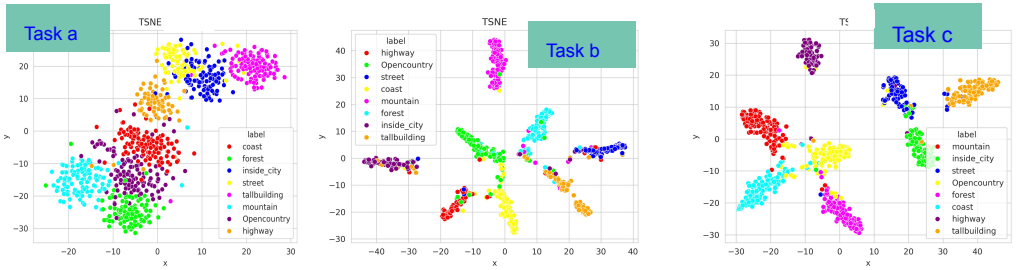
We tested with different `margin` values on the Triplet Loss, turns out this will determine how spread or narrowed are our final clusters! By setting a smaller margin value we have more robust clusters, that is, less internal overlapping between groups in the same group.





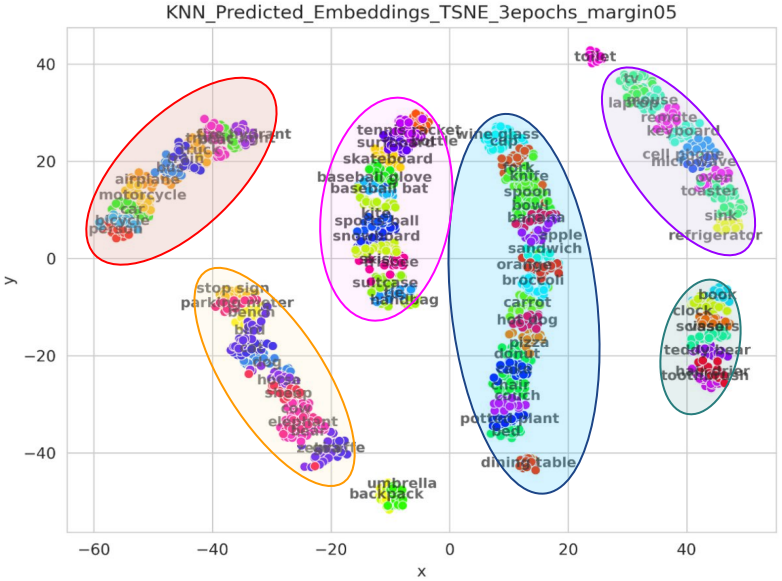
Summary Slide G7





Task e)

Val. accuracy: 97.65% Total run time: 6h 30min



Found super classes:

LIVING AND TRANSPORT
(person, truck, motorcycle…)

FOOD AND KITCHEN
(i.e. broccoli, knife, spoon…)

SPORT ACTIVITIES AND MISC
(skateboard, baseball bat, skis…)

ELECTRONICS
(i.e. keyboard, cell phone…)

COMMON UTILITIES
(i.e. book, scissors, toothbrush…)

ANIMALS AND URBAN
(i.e. elephant, dog, stop sign…)