Компьютерное зрение

Практический курс

Савельева Юлия Олеговна
i.o.saveleva.kpfu@gmail.com

2-й семестр, 13.03.2021 г.

Image Retrieval

Подсчет точности

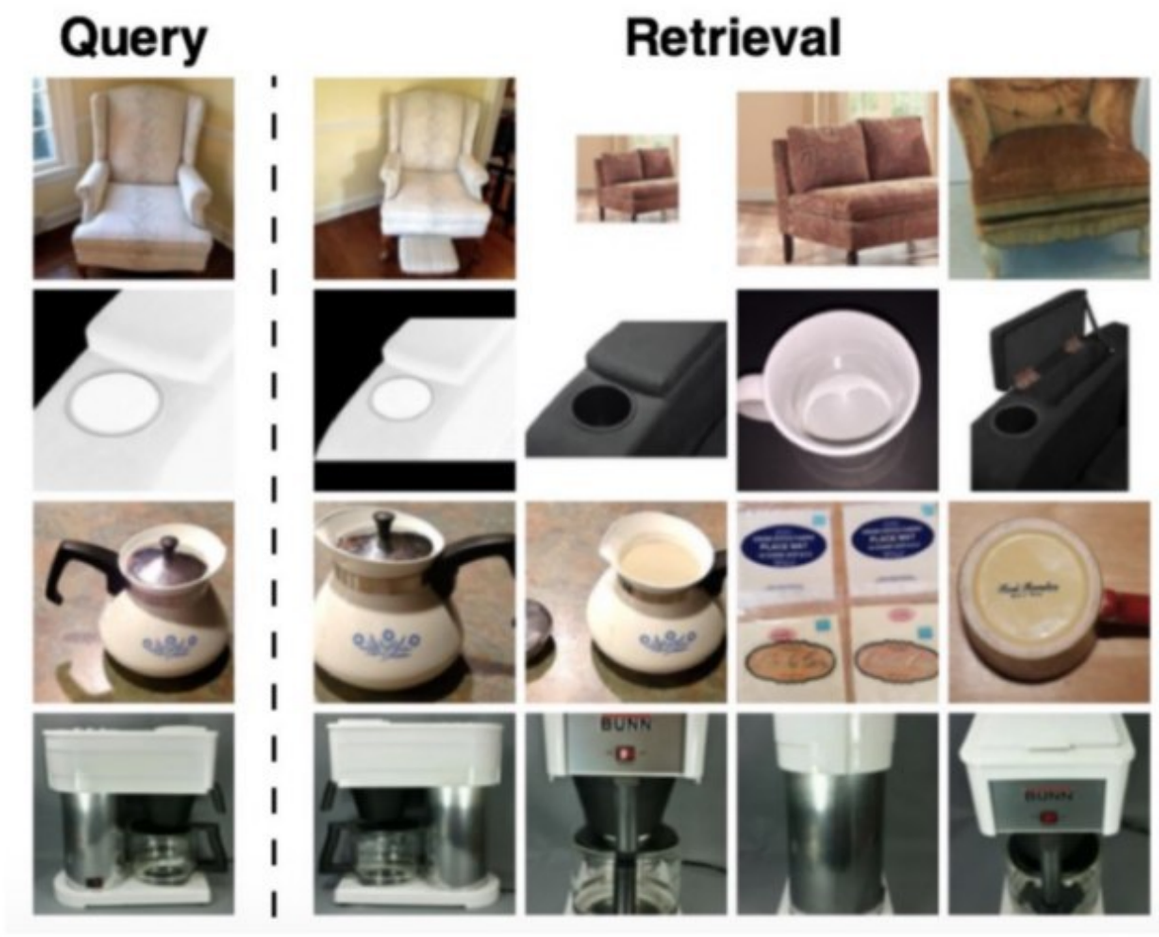

Image Retrieval

Подсчет точности

Разбиение на query и retrieval для валидационной и тестовой выборок вы можете найти на GitHub:
query_retrieval_valid_dataset.pickle
query_retrieval_test_dataset.pickle

Pretrained on ImageNet with random fc top-1 accuracy: 48.98%.
Trained with Triplet loss top-1 accuracy: **71.89%** ≤ эту точность нужно получить на максимальный балл.
Для выставления оценки необходимо сообщить точность на тестовой выборке.

Файлы содержат:

- пути до query картинок,
- общие для всех query пути до retrieval картинок

и имеют следующую структуру:

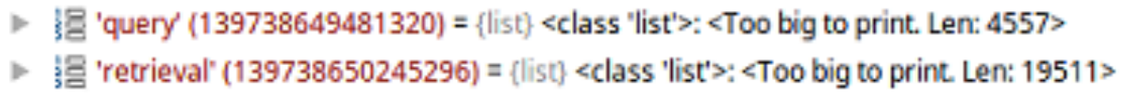

Image Retrieval

Подсчет точности

```
import nmslib

Input: query_labels, query_embeddings, retrieval_labels, retrieval_embeddings
index = nmslib.init(method='hnsw', space='l2')
index.addDataPointBatch(retrieval_embeddings)
index.createIndex()
search_count = 1
matches = zeros(len(query_labels))
for i in range(len(query_labels)):
    quary_label = query_labels[i]
    ids, distances = index.knnQuery(query_embeddings[i], k=search_count)
    matches[i] = any(quary_label == retrieval_labels[ids])
accuracy = np.sum(matches[i]) / float(len(query_labels)) * 100.
```

Image Retrieval

Cross-Batch Memory for Embedding Learning

- Обучить модель с Triplet loss как обычно на 1000 итераций
- Создать класс «памяти», который хранит эмбеддинги (и их лейблы) с прошлых итераций (количество хранимых эмбеддингов определяется параметром % от обучающей выборки)
- Проконтролировать, чтобы в памяти не лежали старый и новый эмбеддинг для одного и того же элемента обучающей выборки
- На каждой итерации считается
  Loss = Triplet_loss(batch_embs, batch_labels) + XBM_Triplet loss(batch_embs, batch_labels, memory_embs, memory_labels)
- XBM_Triplet_loss(b_e, b_l, m_e, m_l):

```
joint_l = concat(b_l, m_l)
joint_A = dists(concat(b_e, m_e))
for label in unique(b_l):
  same_indices = where(joint_l == label)
  negative_indices = where(joint_l != label)
  anchors, positives = all_combinations(same_indices)
  ap_dists = joint_A[anchors, positives]
  for a, p, ap_dist in (anchors, positives, ap_dists):
    loss = ap_dist - joint_A[a, negative_indices] + α
    semi_hard = random_choice(0 < loss < α)
    if semi_hard is not None:
      triplets.append((a, p, semi_hard))
```

Дедлайн 27.03.2020 23:59