CONTENT-BASED IMAGE RETRIEVAL (CBIR) WITH DEEP LEARNING

Lampros Lountzis

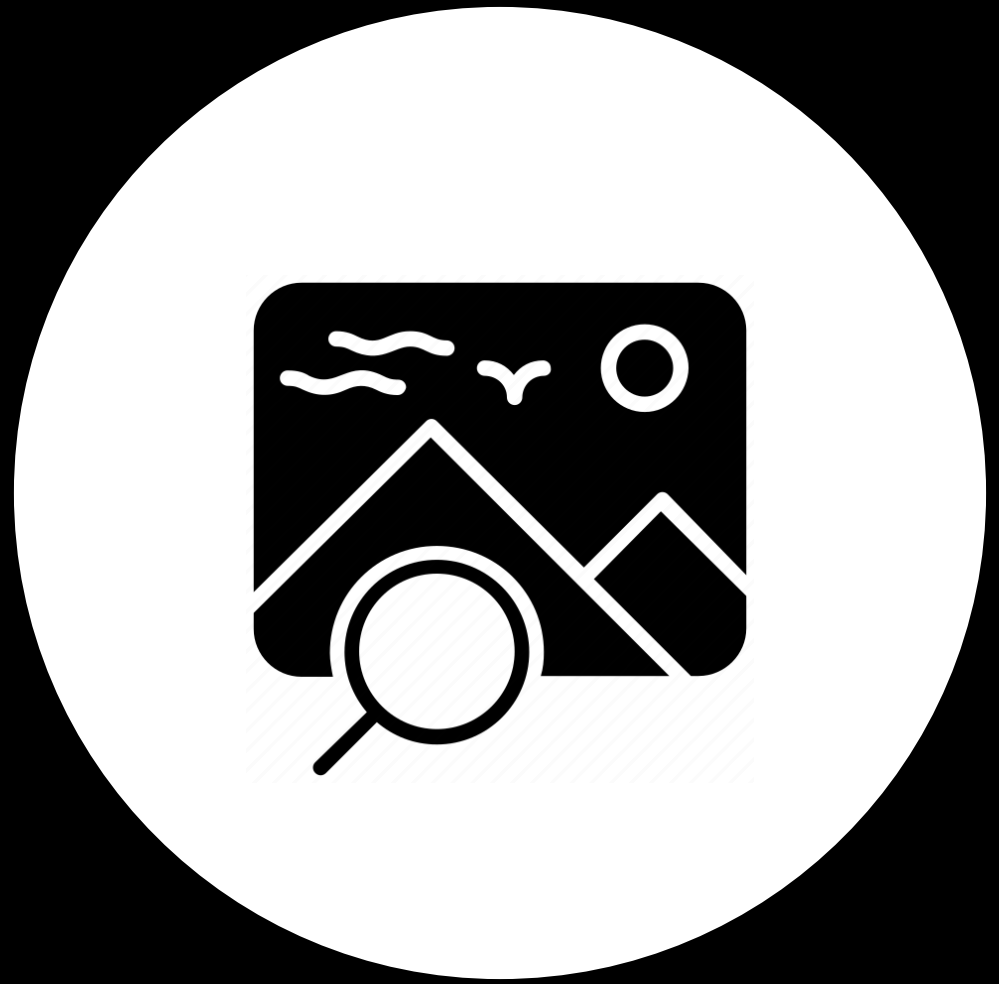


Table of Contents

1. INTRODUCTION
    1. The problem
    2. Challenges deep-dive
2. SEARCH ENGINE
    1. IR system architecture
    2. CIFAR-10 data
    3. Elasticsearch search engine
3. BAG OF VISUAL WORDS
    1. Visual descriptors
    2. BOVW model
    3. Features
    4. Machine Learning models

Table of Contents

4. VISUAL EMBEDDINGS
    1. Model architecture
    2. Visual embeddings
    3. Deep Learning models
5. RESULTS
    1. Results
    2. Discussion
    3. Ideas and further discussion
6. DEMO

INTRODUCTION

The problem

Problem statement

Although we communicate in a variety of ways with each other, our favorite way to do so is via the written word.

However, when you think, do you think in words or images ?

Pictures sometimes are easier to recognize and process than words. What is more, they can be a way of communicating something that's impossible to verbalize, like thoughts, feelings, memories.

So, how can we improve information retrieval and accessibility via images ?

Context

Content-Based Image Retrieval is, the problem of searching for digital images in large databases based on their contents, rather than the metadata (keywords, tags, etc.), using computer vision techniques.

Computer Vision deals with how computers can gain high-level understanding from digital images or videos.

Deep learning is concerned with algorithms inspired by the structure and function of the brain.

Challenges deep-dive

Challenge 1

Image data:

- What are the **attributes** of the images (e.g., size, color-space, format) ?
- What **kind of objects** and **how many of these objects** do the images contain ?
- Is the search engine **topic specific** or is it a **general-purpose** IR system ?

Challenge 2

Retrieval method:

- The method is **heavily dependent** on the techniques of computer vision (classification, object detection, etc.), that are used to extract information from images.
- Should we use **machine learning** methods or **deep learning** models ?
- What kind of **features** should we extract ?

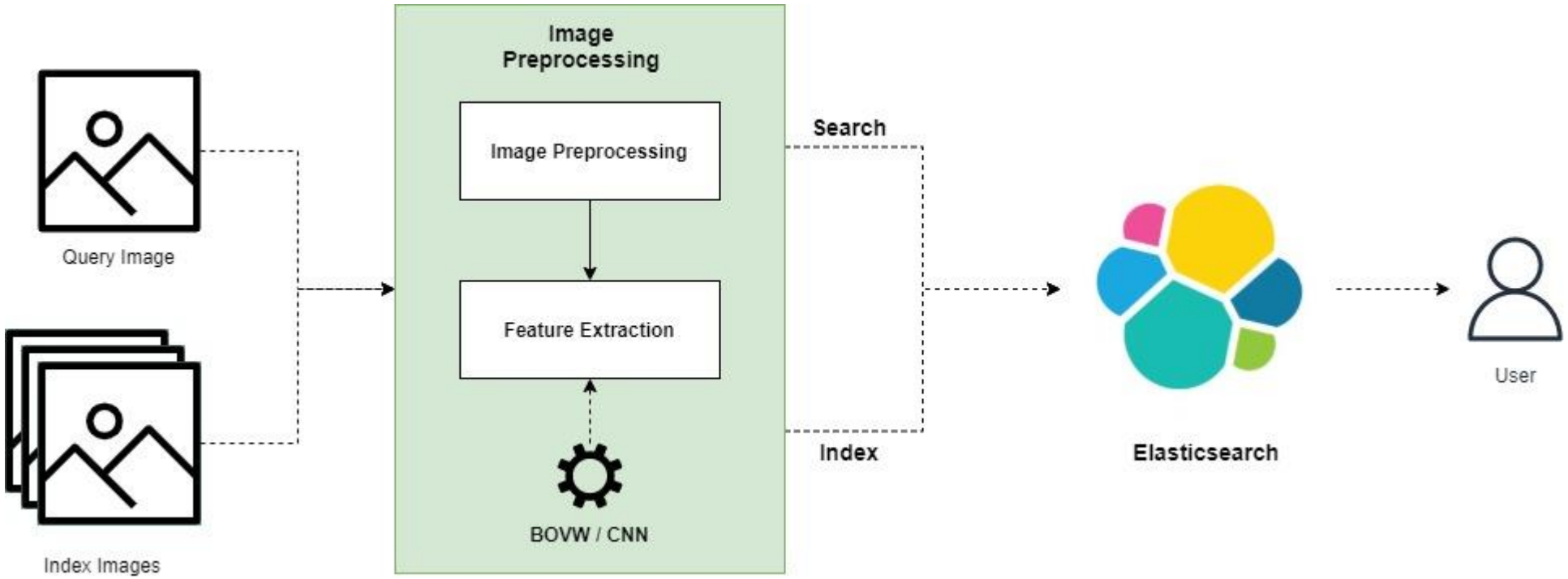
Challenge 3

Performance:

- The IR system's performance is **dependent** on the computer vision method used to extract visual information.
- How do we **measure** the IR systems performance ?
- The search engine's efficiency is also affected by the **libraries and tools** used for IR and CV.

SEARCH ENGINE

IR system architecture



CIFAR-10 data

- We're using the **CIFAR-10** corpus (by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton) which consists of image data, about 50000 training images and 10000 test images.
- Each image is a **32x32 color image**.
- The dataset contains **10 classes**, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
  - The classes are completely mutually exclusive (e.g. there is no overlap).
- We consider the training images as **index images**, meaning that these will be indexed in our IR system. In the same manner, we consider the test images as **query images**.
- The **query relevance** is defined as follows: **each query image (test image) is related with a set of indexed images (training images) where the relevance relationship depends on the class label.**
  - For example, a query image that is a car is associated with indexed images that belong to the car class.

Elasticsearch search engine

The Information Retrieval system (search engine) was created using the **Elasticsearch** service, in **Python**.

Queries and Documents

- The image document and queries consist of the following fields: **id**, **filename**, **path** (absolute path to file), **features** (dense vector of image features as found by the underlying computer vision model) .
- **The image documents are retrieved and ranked using the features vector**. In order to accomplish this, we compare the image-query feature vector with the image-document feature vector using the **cosine similarity.** Specifically

  cosineSimilarity(query.features, doc.features) + 1.0

  where we add 1.0 to the cosine similarity to prevent the score from being negative.

Elasticsearch configuration

- We configure the Elasticsearch client to run on localhost (port 9200) with a timeout of 60sec and retry on timeout.
- We create only **one node per cluster**, since this is not a system ready for production.

Elasticsearch search engine

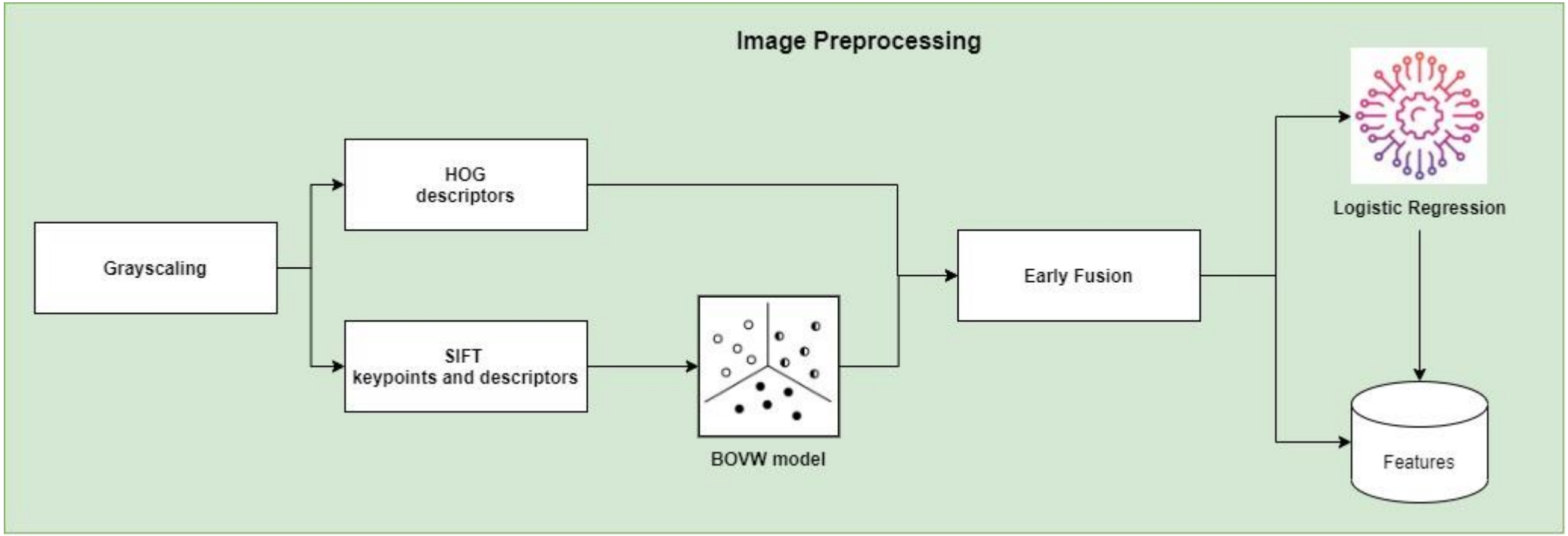
- We have decided to **not create any replicas** for the indexes, since this is a demo application.
- Each index is **distributed in 30 shards** (each shard is an instance of a Lucene index, that indexes and handles queries for a subset of the data in an Elasticsearch cluster) so that we can handle the big volume of the CIFAR-10 data.
  - We gain better performance in indexing and searching operations.
- The **mapping** for the image-documents that the indexes can handle, consists of the fields of the documents and queries that we have described before. The only **field that can be used for retrieval is "features"** which is of **dense vector** type and the dimensions are dependent on the CV method. However, the size of the dense vector is always smaller than 2048 since this is the maximum size Elasticsearch can handle.

Evaluation

- To evaluate our IR system, we have used the **trec_eval** tool and its metrics.
- Specifically, the search engine is evaluated on **the top 100 retrieved image documents**, and we consider the **mean average precision (MAP)**.

BAG OF VISUAL WORDS

Model architecture

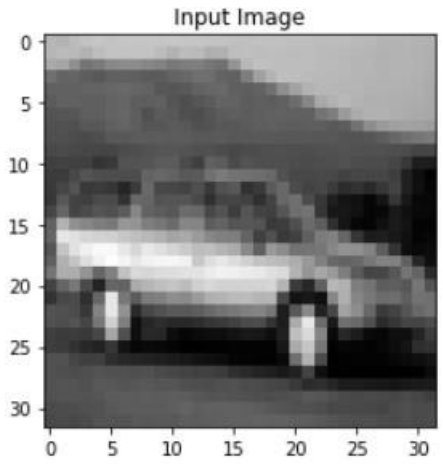


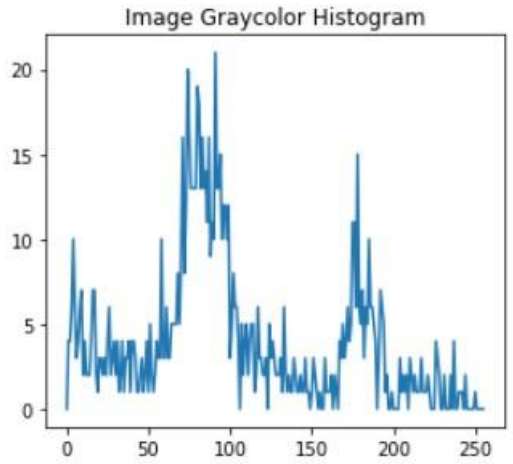
Visual descriptors

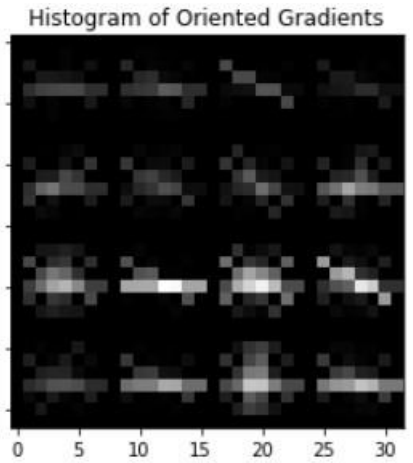
Before using machine learning models, we need to extract from the CIFAR-10 images the keypoints and descriptors.

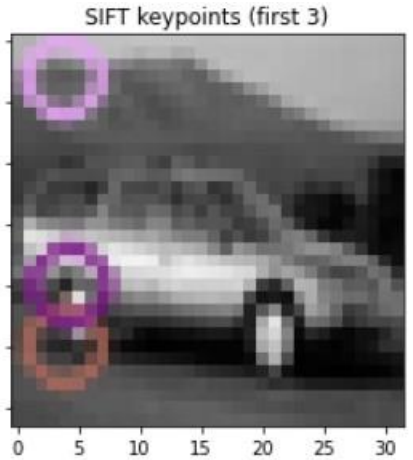
We extract from the images the following **visual descriptors**:

- Gray Color Histogram (GCH),
- Histogram of Oriented Gradients (HOG),
- Scale-Invariant Feature Transformation (SIFT).









BOVW model

The idea of the **Bag of Visual Words (BOVW)** is adapted from **Natural Language Processing (NLP) Bag of Words** model.

- In Bag of Words model, we count the number of appearance of each word in a document, use the frequency of each word to know the keywords (features) of the document, and make a frequency histogram from it. We treat a document as a bag of words.

The general idea of **Bag of Visual Words (BOVW)** is to represent an image as a **set of features**. Features consists of **keypoints and descriptors**.

- Keypoints are special because no matter what transformation we apply to an image (rotate, shrink, expand, etc.) the keypoints will always be the same.
- A descriptor is the description of the keypoint.

We use the keypoints and descriptors to construct **visual vocabularies** and then we quantize the image features.

By doing so, we have successfully represented images as a **frequency histogram of features** that are in the images.

With the use of visual vocabularies, later, we can perform many tasks, such as classification, retrieval and more.

BOVW model

Since GCH and HOG features are already in histogram forms, we'll only create visual vocabularies using the SIFT keypoints and descriptors.

To create visual vocabularies, we make use of **clustering models**. Then the **centroid** of each cluster is considered a **visual word**, and all centroids together form the visual vocabulary. Provided that the training set is sufficiently representative, the visual vocabulary will be "universal". Finally, a vector quantizer takes a feature vector and maps it to the index of the nearest visual word in the visual vocabulary.

We have selected **mini-batch k-means** (with batch size of 64 images) to find the visual vocabulary. Also, we have tested **vocabulary sizes between 100 and 1000 visual words**, since the CIFAR-10 images are small (32x32 pixels).

- vocabulary too small: the visual words won't be representative.
- vocabulary too large: overfitting to the training set.

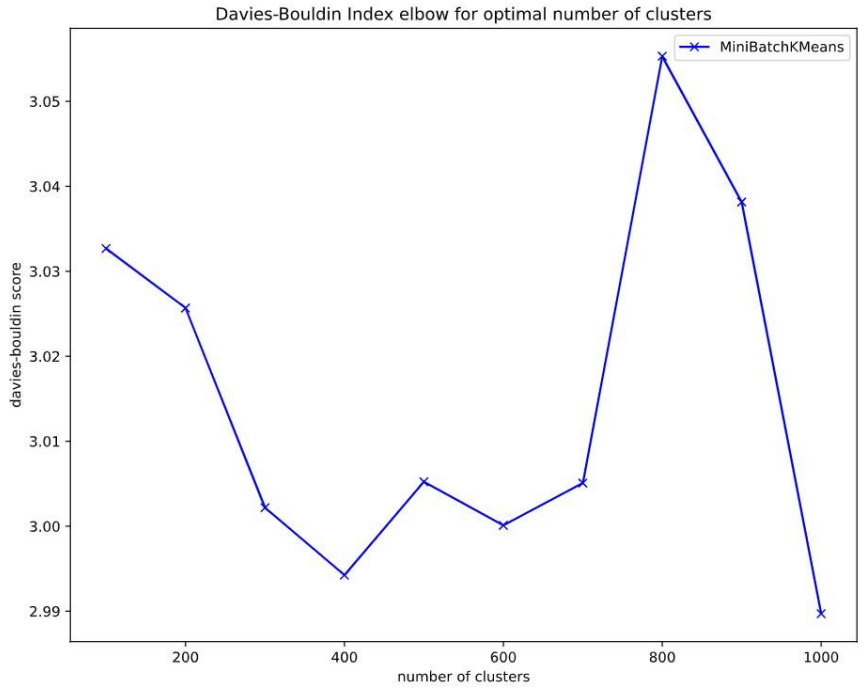
Note: Due to lack of computational recourses we were unable to test more clustering models.

BOVW model

To **evaluate** our Bag of Visual Words model, we'll use the **Davies-Bouldin Index** metric.

- The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score.
- Lower values indicate better clustering.

According to the results, a **vocabulary size of 400 visual words is sufficient**.



Features

Having extracted the visual vocabulary and quantized the vectors of SIFT descriptors, we are ready to decide what features are we going to feed our classification models and the search engine.

We have used **early fusion** to combine the GCH descriptors, HOG descriptors and SIFT descriptors after quantization, before we feed them to our classification model. The combinations are the following:

- HOG and GCH descriptors,
- HOG and SIFT descriptors,
- HOG, GCH and SIFT descriptors.

We have decided to test only combinations that include HOG descriptors, because Histogram of Oriented Gradients is a state-of-the-art feature for machine learning applications and thus should be a de-facto feature.

- N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005

Machine Learning models

The idea of this part is that **the final feature vector for each image will be a fusion of image features and the predicted class one-hot vector.**

The models we have tested are the following:

- Gaussian Naïve Bayes,
- Logistic Regression,
- k Nearest Neighbors,
- Linear Support Vector Machines,
- Random Forest.

Moreover, we have tuned their hyperparameters exhaustively using **grid-search** and **5-fold cross-validation.**

The **primary metric** used to access our models is **accuracy**, since CIFAR-10 is a well-balanced dataset. Also, we have considered **precision**, since the models will be used for the image retrieval task.

As the results show, **Logistic Regression utilizing HOG and SIFT features is the top model** since it accomplishes the highest accuracy and precision score and simultaneously doesn't overfit.

Machine Learning models

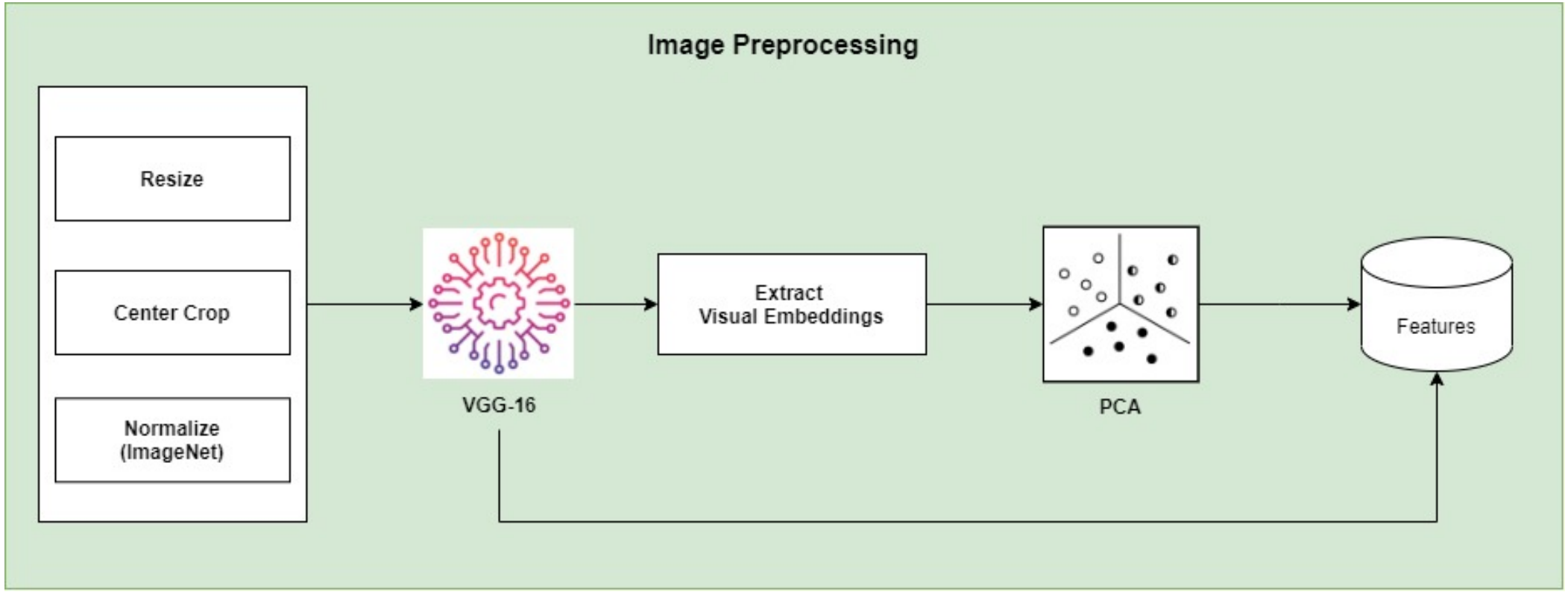
	Accuracy					
	HOG + GCH		HOG + SIFT		HOG + GCH + SIFT	
	Train	Test	Train	Test	Train	Test
Gaussian Naïve Bayes	37%	37%	39%	37%	40%	37%
Logistic Regression	56%	52%	60%	56%	61%	56%
kNN	100%	23%	100%	19%	100%	23%
Linear SVM	54%	51%	58%	56%	58%	55%
Random Forest	100%	54%	100%	54%	100%	54%

Machine Learning models

	Precision					
	HOG + GCH		HOG + SIFT		HOG + GCH + SIFT	
	Train	Test	Train	Test	Train	Test
Gaussian Naïve Bayes	39%	38%	41%	38%	40%	37%
Logistic Regression	55%	52%	59%	56%	61%	55%
kNN	100%	24%	100%	22%	100%	24%
Linear SVM	54%	50%	58%	55%	59%	56%
Random Forest	100%	54%	100%	53%	100%	54%

VISUAL EMBEDDINGS

Model architecture



Visual embeddings

Contrary to machine learning, in deep learning we don't have to handcraft features. **The feature extraction is performed by the deep learning models**. Deep neural models learn high-level features in the hidden layers.

- First, the image goes through many convolutional layers.
- In those convolutional layers, the network learns new and increasingly **complex features** in its layers.
- Then the transformed image information goes through the fully connected layers and turns into a classification or prediction.

The extracted features from the deep learning models we have utilized, are the visual embeddings. **Visual embeddings** refers to the collection of features of the last fully connected layer (prior to a loss layer) appended to a CNN.

- The size of this fully connected layer determines the dimension of the embedding vector space.

The visual embeddings are learned by jointly training the feature extractor with the embedding layer and the classifier (next slides describe the methods we used to tune and train deep learning models). We register a **forward hook** to our deep learning models, so that we can access this vector space (features) and utilize them for retrieval later.

Finally, since Elasticsearch can only handle vectors with maximum size of 2048, we **apply dimensionality reduction with Principal Component Analysis (PCA) and reduce the size of the visual embeddings to 2000 features**.

Deep Learning models

The idea of this part is that **the final feature vector for each image will be a fusion of the visual embedding and the predicted class one-hot vector.**

Before passing the image data to the deep learning models (batch size 64) we apply some **transformations**:

- **resizing**,
- **center cropping** and
- **normalization** using mean and standard deviation from ImageNet.

The models we have tested are the following:

- Neural Network,
- Convolutional Neural Network (custom),
- Alexnet,
- VGG-16,
- Inception v1,
- ResNet-50.

Deep Learning models

We jointly train the deep learning models and the visual embedding layer on the **classification task**. Thus, we choose **cross-entropy loss** as the cost function. Moreover, we make use of **Adam optimizer** and **ReduceLROnPlateau scheduler** (reduces learning rate when accuracy has stopped improving). Finally, we apply additional regularization by optimizing the **weight decay**.

Considering the pretrained models, we have decided to use **transfer learning**. We believe that the features the pretrained models have acquired, during training on ImageNet, can provide better results than training those networks from scratch. That is because the features learned in the first convolutional layers are quite generic.

Since CIFAR-10 data are similar to ImageNet, we use the **pretrained models as feature extractors**, meaning that we freeze all the layers of the pretrained model and only train the classifier part.

Note: Due to lack of computational recourses we only train the classification dense layer (softmax layer).

The **primary metric** used to access our models is **accuracy**, since CIFAR-10 is a well-balanced dataset. Also, we have considered **precision**, since the models will be used for the image retrieval task.

Deep Learning models

Considering hyperparameter tuning, we have used **Ray tune** which is a popular framework for scalable hyperparameter tuning. During tuning we utilize **early stopping** with **ASHAScheduler** (a better version of HyperBand scheduler) for eliminating bad trials based on validation accuracy.

- The **grace period** is selected in such way that we won't fall in a local optima.
- Not only do we tune learning rate and weight decay, but also, we tune the number of layers, nodes and dropout probability in the cases of the Neural Network and the custom CNN.

As the results show, **VGG-16** is the top model since it accomplishes the highest accuracy and precision score and simultaneously doesn't overfit.

Note: Since we were only able to train the softmax layer we believe that this might had an impact on the results. Normally, we would have expected that the ResNet-50 would be the winner since contrary to the other models it utilizes state-of-the-art techniques.

Deep Learning models

	Accuracy		Precision	
	Train	Test	Train	Test
Neural Network	67%	55%	67%	55%
CNN	78%	67%	78%	67%
AlexNet	89%	84%	90%	85%
VGG-16	89%	85%	89%	85%
Inception v1	84%	81%	84%	82%
ResNet-50	90%	85%	90%	85%

RESULTS

Results

	Accuracy		Precision	
	Train	Test	Train	Test
BOVW	60%	56%	59%	56%
VGG-16	89%	85%	89%	85%

	Mean Average Precision (MAP) @ 100
BOVW	0.0019
VGG-16	0.0088

Discussion

Machine Learning vs Deep Learning

As expected, **the VGG-16 model has surpassed the Bag of Visual Words model**. 🏆

This is because the VGG-16 model (pretrained on ImageNet and fine-tuned on CIFAR-10) has learned **complex features** that are representative of the data and thus it can distinguish better the CIFAR-10 classes and classify more accurately the images.

In contrast, **the BOVW model is dependent on feature engineering· the better the handcrafted features are the more predictive power the machine learning model will have.** Some of the produced features in the BOVW model are **local features**, meaning that they contain specific information about the images they were extracted from. Because of that, only a small proportion of extracted features is representative of the data and thus our machine learning model cannot generalize.

Search engine:

Similarly as before, **the IR system dependent on VGG-16 gives the best map@100 results**. 🏆

Since VGG-16 can distinguish better the CIFAR-10 classes, our IR system will be more accurate (higher precision). So, during retrieval there are **more true positives** (relevant images with respect to the query) leading the set of retrieved documents. One could say that the sorting by the similarity function (cosine in our case) is better.

Ideas and further discussion

As stated before, due to lack of computational resources, we were unable to replace the classification layers in the pretrained CNNs with our own and fine-tune the model. If we had in our disposal a virtual machine with more CPU/GPU cores to fine-tune the models, then we would have achieved better results.

What's more, we believe that **the choice of training the models in the classification setting has prevented us from gaining the best results**. The aim of the cross-entropy loss is to categorize features into predefined classes and thus the performance of such a network is poor when compared to losses incorporating similarity (and dissimilarity) constrains in the embedding space during training.

A better alternative would be to use:

- **contrastive loss**, which optimizes the training objective by encouraging all similar class instances to come infinitesimally closer to each other, while forcing instances from other classes to move far apart in the output embedding space.
- **triplet loss**, which forces data points from the same class to be closer to each other than they are to a data point from another class, by considering both positive and negative pair distances at the same point

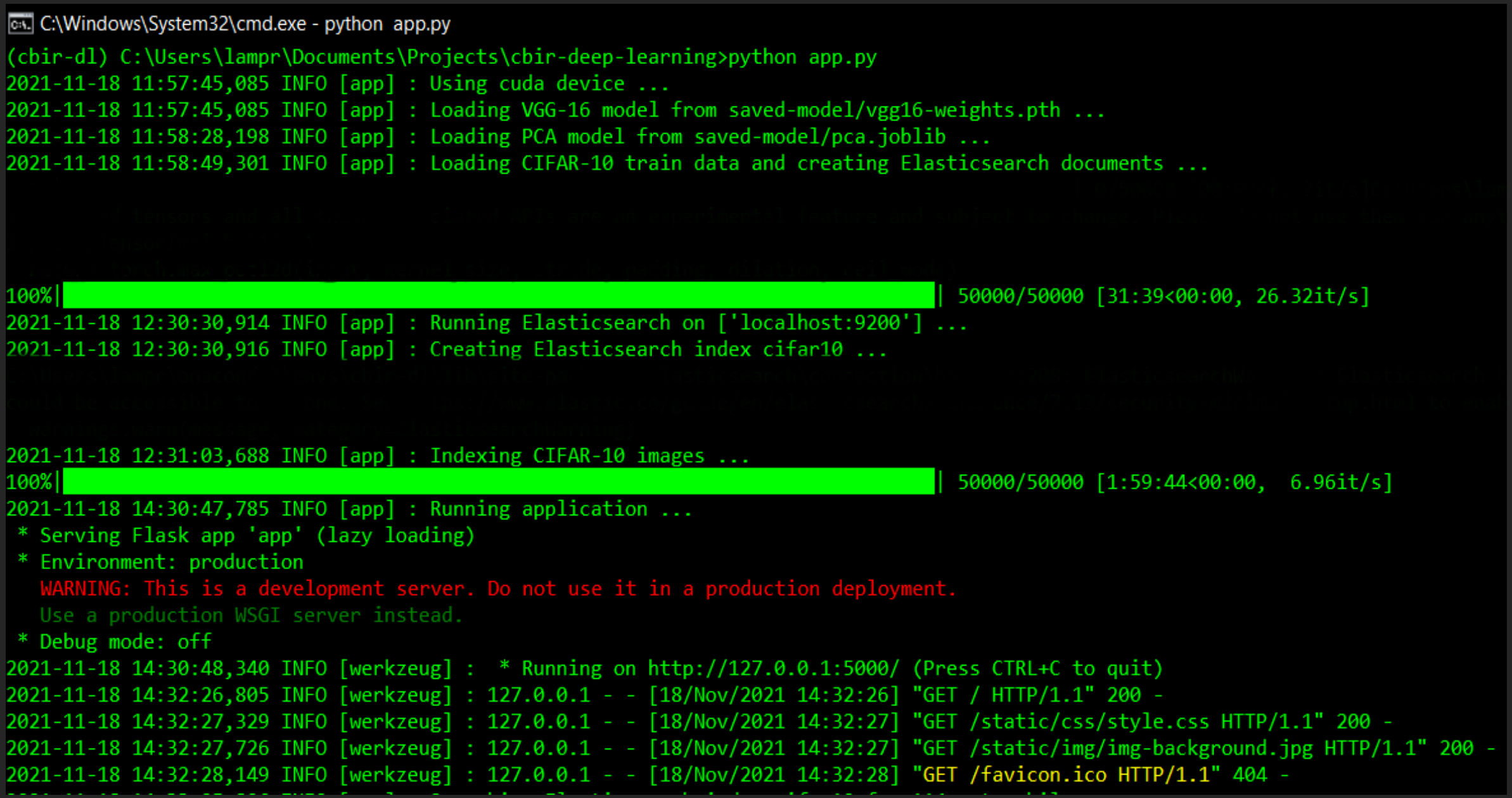
Ideas and further discussion

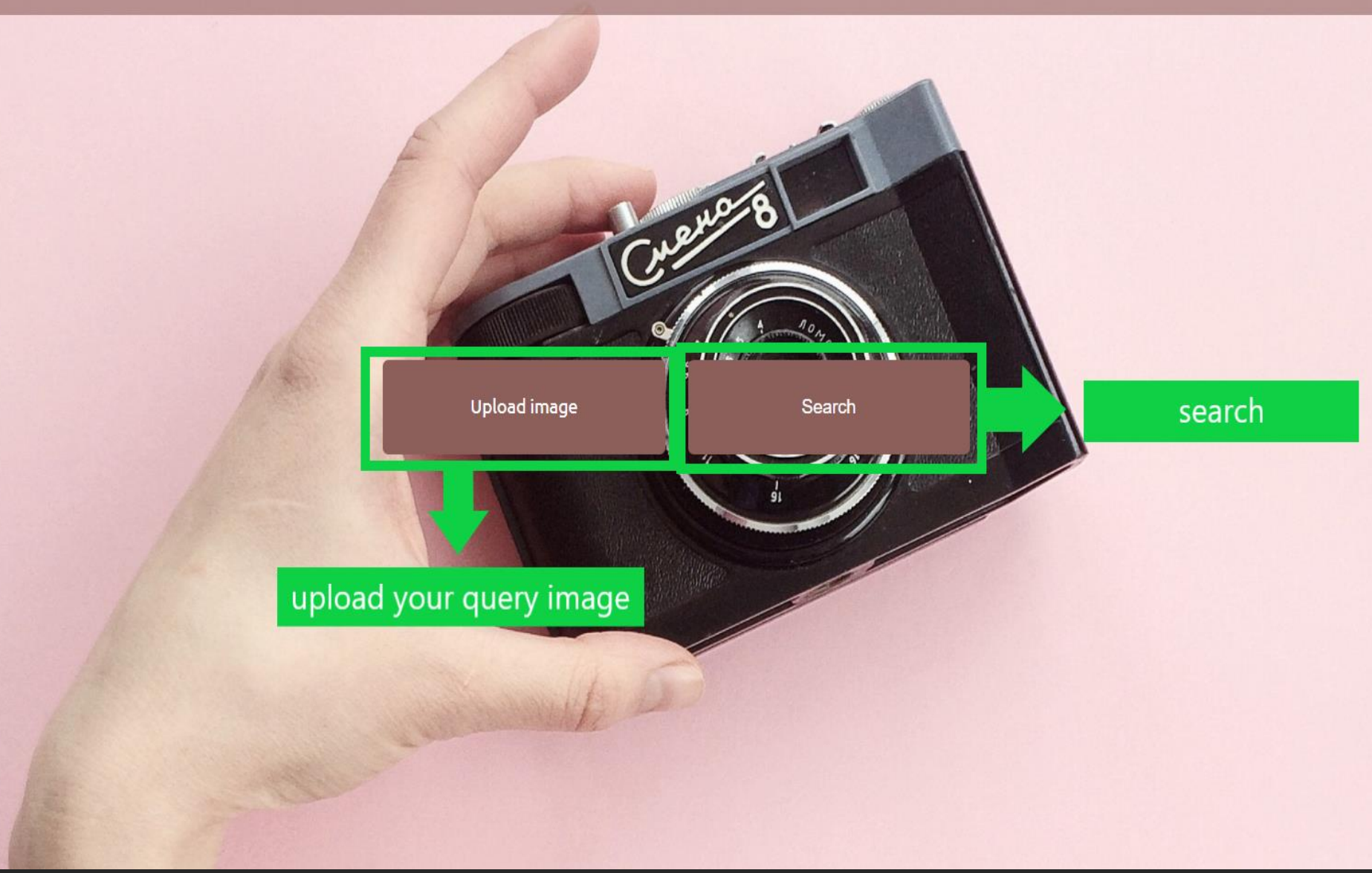
If we were to choose **triplet loss** as our loss function, the we should also consider an appropriate method for **mining informative points** in order to improve training convergence and computational complexity.

Popular sampling approaches are:

- **Batch all (BA)**: uses all positive and valid triplets; that is, we're not performing any ranking or selection of triplets (all samples are equally important).
  - Can potentially lead to information averaging out since many valid triplets are trivial, and only few are informative.
- **Batch hard (BH)**: considers only the hardest data for an anchor. For each possible anchor in a batch, it computes the loss with exactly one hardest positive item and one hardest negative item.
  - BH is robust to information averaging out, because trivial (easier) samples are ignored.
  - BH is difficult to use with outliers.
- **Batch weighted (BW)**: a sample is weighted based on its distance from the corresponding anchor, thereby giving more importance to informative (harder) samples than trivial samples
- **Batch sample (BS)**: uses the distribution of anchor-to-sample distances to mine positive and negative data for an anchor.

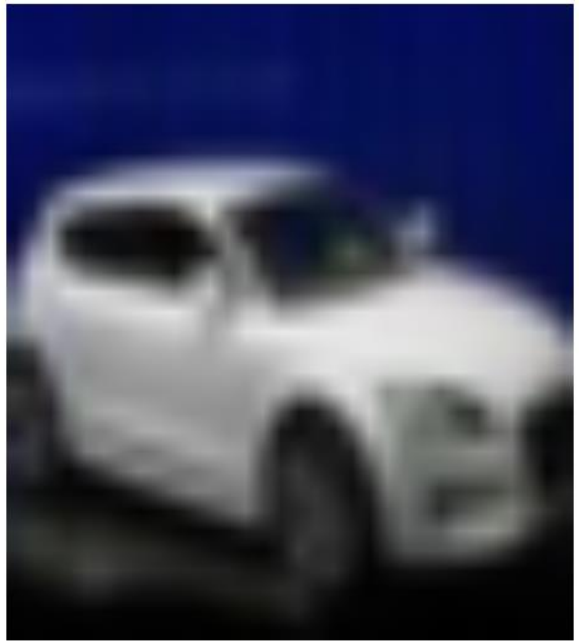
DEMO


C:\Windows\System32\cmd.exe - python app.py
(cbir-dl) C:\Users\lampr\Documents\Projects\cbir-deep-learning>python app.py
2021-11-18 11:57:45,085 INFO [app] : Using cuda device ...
2021-11-18 11:57:45,085 INFO [app] : Loading VGG-16 model from saved-model/vgg16-weights.pth ...
2021-11-18 11:58:28,198 INFO [app] : Loading PCA model from saved-model/pca.joblib ...
2021-11-18 11:58:49,301 INFO [app] : Loading CIFAR-10 train data and creating Elasticsearch documents ...
100%| 50000/50000 [31:39<00:00, 26.32it/s]
2021-11-18 12:30:30,914 INFO [app] : Running Elasticsearch on ['localhost:9200'] ...
2021-11-18 12:30:30,916 INFO [app] : Creating Elasticsearch index cifar10 ...
2021-11-18 12:31:03,688 INFO [app] : Indexing CIFAR-10 images ...
100%| 50000/50000 [1:59:44<00:00, 6.96it/s]
2021-11-18 14:30:47,785 INFO [app] : Running application ...
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
2021-11-18 14:30:48,340 INFO [werkzeug] : * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
2021-11-18 14:32:26,805 INFO [werkzeug] : 127.0.0.1 - - [18/Nov/2021 14:32:26] "GET / HTTP/1.1" 200 -
2021-11-18 14:32:27,329 INFO [werkzeug] : 127.0.0.1 - - [18/Nov/2021 14:32:27] "GET /static/css/style.css HTTP/1.1" 200 -
2021-11-18 14:32:27,726 INFO [werkzeug] : 127.0.0.1 - - [18/Nov/2021 14:32:27] "GET /static/img/img-background.jpg HTTP/1.1" 200 -
2021-11-18 14:32:28,149 INFO [werkzeug] : 127.0.0.1 - - [18/Nov/2021 14:32:28] "GET /favicon.ico HTTP/1.1" 404 -


Upload image
Search
search
upload your query image

ID: 21073

Filename: 21073-automobile.png



Score: 1.6039015

result
(retrieved image)

ID: 13523

Filename: 13523-automobile.png

Thank you 😃