



## Lab 5: Deep Learning with Spark

The objective of this lab is to detect image similarity using PySpark. Identifying images that are similar to one another is intuitive for humans, but it is a computationally complex task. At a large scale, it becomes even more challenging. In this lab, we will introduce an approximate method for finding similar items called Locality Sensitive Hashing (LSH) and apply it to images. We will use deep learning to convert image data into a numerical vector representation. PySpark's LSH algorithm will then be applied to the resulting vectors, allowing us to find similar images for a given input image.

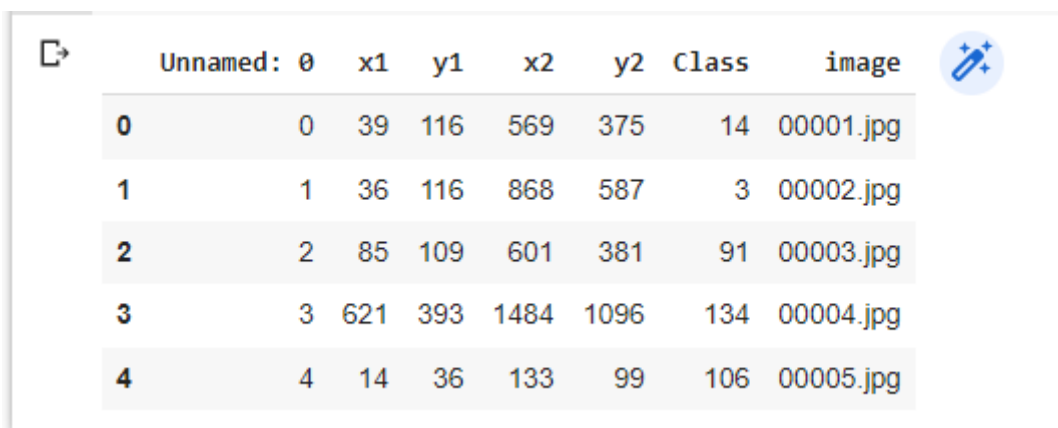
First, we will download and prepare our dataset. The dataset used for this task is the Cars dataset, published in 2013 by the Stanford AI Lab. PyTorch<sup>1</sup> will be used for image preprocessing. This will be followed by converting our input data (images) into a vector representation (image embeddings).

We will then import the resulting embeddings into PySpark and transform them using the LSH algorithm. Finally, we will take a new image and perform a nearest neighbor search using our LSH-transformed dataset to find similar images.

### Data Preparation

We will use the Stanford Cars dataset<sup>2</sup>.

You can obtain a CSV file containing labels for the training dataset here:  
<https://github.com/BotechEngineering/StanfordCarsDatasetCSV>.



	Unnamed: 0	x1	y1	x2	y2	Class	image
0	0	39	116	569	375	14	00001.jpg
1	1	36	116	868	587	3	00002.jpg
2	2	85	109	601	381	91	00003.jpg
3	3	621	393	1484	1096	134	00004.jpg
4	4	14	36	133	99	106	00005.jpg

Ignore all columns except **Class** and **Image**. The other columns are related to the original research project from which this dataset was derived and will not be used for our task.

<sup>1</sup> PyTorch is a library for building deep learning projects.

<sup>2</sup> Stanford Cars Dataset: <https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset>

## Resizing Images with PyTorch

Before proceeding further, we need to preprocess our images, as deep learning models (neural networks) require the input to meet certain requirements.

We need to apply a series of preprocessing steps, called transformations, to convert the input images into the appropriate format for the models. In our case, the input images must be in JPEG format with dimensions 224 x 224 pixels, as this is a requirement for the ResNet-18 model that we will use in the next section.

Q1)- [Perform this transformation using the Torchvision package in PyTorch.](#)

## Deep Learning Model for Vector Representation of Images

Convolutional Neural Networks (CNNs) are the standard neural network architectures used for prediction when the input observations are images. We will not use them for a prediction task, but rather to generate a vector representation of the images. Specifically, we will use the ResNet-18 architecture.

Residual Network (ResNet) was introduced by Shaoqing Ren et al. in their 2015 paper "Deep Residual Learning for Image Recognition". The 18 in ResNet-18 represents the number of layers in the neural network architecture.

## Image Embeddings

An image embedding is a representation of an image in a vector space. The basic idea is that if a given image is close to another image, their embeddings will also be close.

We can obtain image embeddings from ResNet-18 by taking the output from its second-to-last fully connected layer, which has a dimension of 512. Then, we create a class C that, given an image, can return its vector representation.

Q2)- [Create class C.](#)

Q3)- [Using class C, calculate the image embeddings for all images.](#)

Q4)- [Store the image embeddings in a CSV file.](#)

Now that we have the required image embeddings, we can import them into PySpark.

## Importing Image Embeddings into PySpark

Q5)- [Create a Spark DataFrame that contains the image embeddings.](#)

Q6)- [The PySpark LSH implementation requires a vector column as input. Perform a transformation to represent the relevant columns as a single vector.](#)

## Image Similarity Search Using PySpark LSH

Locality Sensitive Hashing (LSH) is an important class of hashing techniques commonly used for clustering, approximate nearest neighbor search, and outlier detection with large datasets. Locality-sensitive functions take two data points and decide whether or not they should be considered a candidate pair.

The general idea of LSH is to use a family of functions ("LSH families") to hash data points into buckets so that data points close to each other are likely to end up in the same buckets, while data points that are far apart are more likely to end up in different buckets. Data points that fall into the same bucket are considered candidate pairs.

In PySpark, different LSH families are implemented in distinct classes (e.g., MinHash and BucketedRandomProjection), and APIs for feature transformation, approximate similarity join, and approximate nearest neighbor search are provided within each class.

Q7)- Create a model using the `BucketedRandomProjection`.

Q8)- Transform the `DataFrame` using the newly created LSH model. The resulting `DataFrame` will include a hash column containing a hashed representation of the image embeddings.

With our LSH-transformed dataset ready, we will test our work in the next section.

## **Nearest Neighbor Search**

Q9)- Select a random image.

Q10)- Convert this image into a vector format.

Q11)- Perform an approximate search for the five nearest neighbors.

Q12)- Display the images that correspond to the five neighbors.