Supervised Learning project
Image Classification with SIFT/BOW and CNN

Matteo Falcone
MSc Artificial Intelligence for Science and Technology
Student ID: 865448
Email: m.falcone16@campus.unimib.it

Abstract—**The aim of this project is the classification of several images of food in 251 different classes. The techniques used to perform this task are two: Convolutional Neural Network and Scale Invariant Feature Transform (SIFT) with Bag of Words (BOW), or Bag of visual features in this context, coupled with a traditional SVM classifier. At the end there is an evaluation of the performance of both and a comparison between these two strategies.**

1. Introduction

In this report it is discussed and commented all the process of multiclass classification with imbalanced dataset. Multiclass classification is the process of assigning labels to data objects with more than two classes. Each entity is assigned to one class without any overlap. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. These two characteristics for this task can make less effective the classification for a Machine Learning (ML) algorithm, especially dealing with imbalanced dataset makes harder to classify correctly the minority class.

The first step of this development is the data preprocessing which is the process of evaluating, filtering, manipulating, and encoding data so that a machine learning algorithm can understand it and use the resulting output. Concerning the images, this concept translates into resizing and normalizing the pixels of the pictures, but also to make them suitable for the specific task, for example the conversion from gray scale to RGB or viceversa.

At this point there are a lot of machine learning algorithms that are capable to handle this kind of tasks. One of the most effective one is in the area of deep learning, in particular convolutional neural networks showed a lot of potential in this area, which made the concept of transfer learning so popular. In addition to artificial neural networks, also other approaches can be very effective, for example Support Vector Machines (SVM) or Random Forest(RF) are two powerful algorithms with many state of art's implementations. The key difference between deep learning and traditional classifier is the way to extract features: in the first case it is performed directly by the network thank to its convolutional layers that "look" into the image and extract the important features, while traditional classifiers cannot perform this operation. For this reason it is essential to rely on another algorithm to automate the feature extraction. The algorithm of feature extraction discussed in this report is called Scaled Invariant Feature Transform (SIFT) to whom is also associated a Bag of visual features or Bag of Words (BOW).

Both these approaches lie on strong theoretical foundations that makes them reliable and robust methods for classification and regression tasks, but this makes them also very difficult to handle. Propitiously there are many algorithms embedded in the state of art which help us to deal with these high level structures. This project has been developed in Python using some of the most famous libraries for machine learning such as Pytorch and Sklearn, used respectively for CNN and SVM, but also with powerful tools for image manipulation like OpenCV, PIL and Torchvision, which is included in the Pytorch library. In order to manage computations and dataframes, NumPy and Pandas were used.

2. Dataset

The dataset contains a total of 120,216 images for train/validation set and 12,170 images of food for the test set from 251 classes.

2.1. Data preprocessing

Before talking about the algorithms, it is important to concentrate on the preprocessing of the images. This is one fundamental part for every machine learning task, especially when dealing with images, which have to be readable by the machines. The main transformations applied to the dataset are resizing and normalization of the pixels. These two operations are very important, especially when the dataset is large, since this shrewdness can facilitate many calculations for the algorithm. The parameters chosen for this purpose are the following:

- Size = (256,256)

- Normalization:
  Mean = (0.485, 0.456, 0.406)
  std = (0.229, 0.224, 0.225)

These are common parameters in many ML application, above all the normalization parameters are the same as ImageNet. They are usually chosen to use transfer learning since there are a lot of high level architectures already trained on millions of images, so training a classifier using the same parameters is much more effective. While developing this project, many comparisons were done with pretrained models, but will not be discussed since the objective was to train a CNN from scratch. This comparisons have been made only to have a boundary in terms of results.

2.2. Data Augmentation

Data Augmentation is one of the most prevalent strategies in deep learning projects involving image data. This process involves applying various transformations to existing images and incorporating these modified versions into the dataset. As a result, the training dataset effectively expands in size.

Depending on the specific augmentation techniques used, synthetic images can be generated from the original training set, potentially increasing the dataset's size by 10 to 100 times. A significant advantage is that these transformations can be performed in real-time during the model's training process, eliminating the need to store the altered images locally and thereby conserving disk space. In addition to enlarging the dataset, data augmentation helps mitigate overfitting, enhances the model's performance on new, unseen data, and does not require additional efforts for image labeling. [3]

In this project have been applied many types of data augmentation using the torchvision library, such as:

- Horizontal/vertical flip
- Rotation (range 0 -180)
- Perspective
- Affine transformation

These transformations are applied directly in the training process so that it was not necessary to store additional space. Every operation has a probability to happen of 0.5, except for rotation that operates in a range from 0 to 180 degrees. This is one of the countermeasures adopted to prevent overfitting of the CNN, which occurred after a few epochs and the net could not generalize well in the validation set.

3. SIFT and BOW feature extraction

In this section it is described the first method used for image classification, which is divided in two main steps: feature extraction with SIFT and BOW and the training of a traditional classifier. Before entering in the core aspects of this analysis, it is necessary to understand the process of resampling of the dataset applied before the algorithm. It is

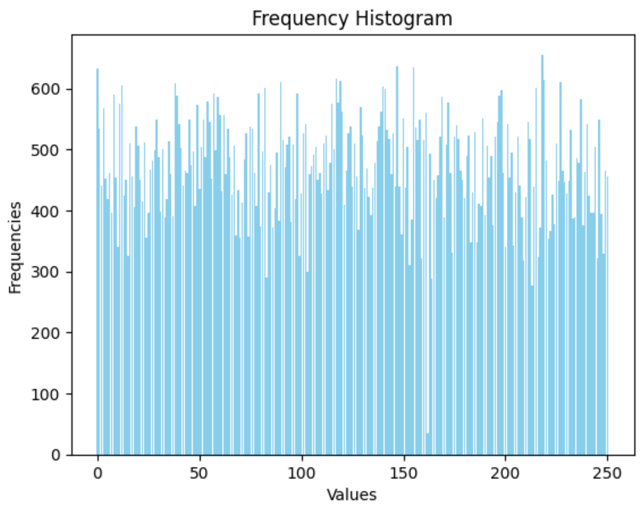

Figure 1. Histogram of the absolute frequency of each class in the dataset

showed in figure 1 the number of images for each of the 251 classes. It is clear that there is a strong imbalance between the classes since the most represented one counts more than 600 samples while the least has 34 images. This can lead to some difficulty to recognize the images belonging to the minority class.
Another, more important, reason that made it necessary to resample the dataset is the computational cost of the feature extraction: since the dataset contains thousands of images, this operation is really expensive. Furthermore for every image are extracted at least 200 features and in some cases even 600. As a result, the algorithm exceeded the available space in the RAM and restarted.

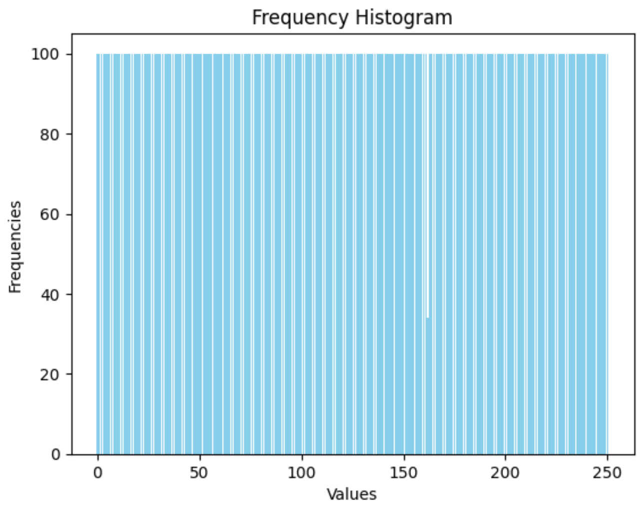

Figure 2. Resampled histogram

For all these reasons, a resampling of maximum 100 images for each class has been made as showed in figure 2

3.1. SIFT

The Scale Invariant Feature Transform (SIFT) [4] algorithm is a widely used method for detecting and describing local features in images that are invariant to scale, orientation, and other transformations. This section explains the key aspects of the SIFT algorithm, including its scale-space representation, keypoint localization and orientation assignment, and the computation of the keypoint descriptor. The SIFT algorithm uses a scale-space representation to detect keypoints that are invariant to scale and orientation. This is achieved by computing the difference-of-Gaussian function to identify potential interest points. The scale-space representation is created by applying a Gaussian blur to the image at multiple scales, which helps to reduce noise and enhance the detection of features.
Keypoints are localized and oriented based on local image

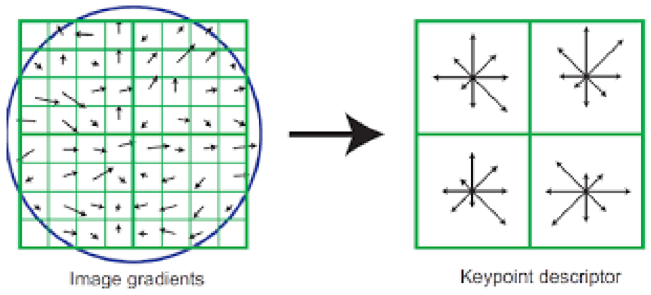

Figure 3. Visualization of how descriptors are computed by the algorithm. This image belongs to the article "Distinctive Image Features from Scale-Invariant Keypoints " by David G. Lowe [4]

gradient directions. The orientation assignment provides invariance to these transformations, ensuring that the features are robust to changes in viewpoint and illumination. The keypoints are displayed as vectors indicating scale, orientation, and location. The keypoint descriptor is formed from a vector containing the values of all the orientation histogram entries, corresponding to the lengths of the arrows in Figure 3.
The algorithm extract a total of 128 features for each keypoint since it takes advantage of a 4x4 array of histograms with 8 orientation bins in each (4x4x8=128).

3.2. Bag of visual words (BoVW)

In computer vision, bag of visual words (BoVW) is one of the methods used for building image vocabularies. We can use BoVW for content-based image retrieval, object detection or, in this case, image classification.

The Bag of Visual Words (BoVW) [5] model is analogous to the Bag of Words (BoW) model used in Natural Language Processing. In BoW, text is broken down into individual words or sub-words, which are then compiled into an unordered list. Similarly, BoVW represents images through patches, extracting unique patterns or visual features from these patches. After these visual features are extracted, a codebook, also known as a dictionary or vocabulary, is constructed. This codebook functions as a repository of all the visual words, similar to how a traditional dictionary contains definitions of words.

The codebook enables the conversion of a potentially infinite range of visual features into a predetermined set of visual words. This is achieved by grouping similar visual features into clusters, with each cluster assigned a central point that represents the visual word for that group. The standard method for clustering visual features into visual words is k-means clustering.

K-means clustering divides the data into k clusters, where k is a user-defined parameter. Once the data is grouped, k-means calculates the mean for each cluster, known as the centroid. This centroid acts as the representative visual word for that cluster. The algorithm then iteratively reassesses each visual feature, determining the closest centroid. If a feature's nearest centroid changes, the feature is reassigned to the new nearest centroid. This process continues for a set number of iterations or until the centroids stabilize. In this code it is implemented k-means++ which is a more efficient variant of the original algorithm that does not initialize randomly the centroids, but try to choose them as far as possibile from each other. This procedure stops if one of this two conditions is satisfied: 10 attempts has been made or if the specified accuracy has been reached.
Selecting the number of centroids, k, is more of an art

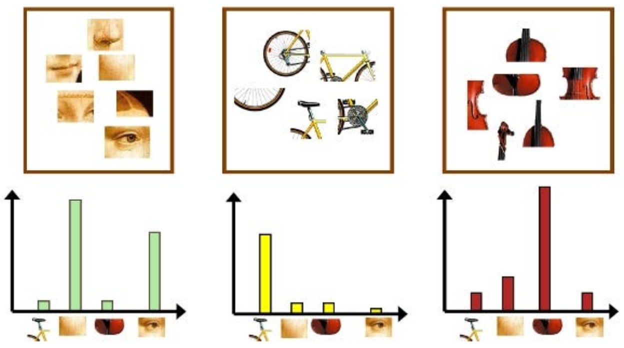

Figure 4. Schematic representation of the learning process using bag of visual features. Image from Medium: bag of visual words(BoVW

than a science, involving some degree of trial and error. One must consider how many visual words are necessary to cover the various relevant visual features in the dataset. To begin, one might start with a smaller k value (e.g., 100 or 150) and rerun the process multiple times, adjusting k until the model performs satisfactorily. For instance, choosing $k = 150$ results in the generation of 150 centroids, thereby defining 150 visual words. For computationally issues this has not been done, but a fixed number of $k = 100$ was chosen because exceeding this value would have run out of memory on the notebook. This has been the strongest limitation for this task since 100 centroids means to have less than half of the number of classes as visual features [6].

When mapping new visual feature vectors to the nearest centroid (visual word), this process categorizes visual features into a limited set of visual words. This procedure is called vector quantization. Consequently, images that share a large number of visual words, such as two images of churches, will be considered similar. In contrast, an image of a church and an image of a dog will share fewer visual words, indicating dissimilarity. After this step, images are represented by a varying number of visual words. The subsequent step involves transforming these visual words into image-level frequency vectors. The frequency of visual words can be counted and visualized using histograms, where the x-axis represents the codebook and the y-axis denotes the frequency of each visual word in the codebook for a given image.

4. Classification with Support Vector Machine

After the feature extraction, the actual classification can start. Now to every image is assigned a record of length 100 which describes which visual elements are present. Now, to distinguish the images in their respective class, it was implemented a **Support Vector Machine**(SVM). SVMs are a group of supervised learning techniques employed for classification, regression, and outlier detection.

The advantages of SVMs include their effectiveness in high-dimensional spaces and their efficiency even when the number of dimensions surpasses the number of samples. SVMs are also memory efficient as they utilize a subset of the training points, known as support vectors, in the decision function. However, SVMs have certain disadvantages. In situations where the number of features is significantly greater than the number of samples, it is crucial to avoid overfitting by carefully choosing the kernel functions and the regularization term. Furthermore, SVMs do not inherently provide probability estimates; these are obtained through an expensive five-fold cross-validation process. [10]

In order to select the better hyperparameters, it was implemented a grid search, which is a common technique used in machine learning. It involves defining a set of hyperparameters and systematically working through all possible combinations of these to determine which combination provides the best performance for a given model. The selected parameters for the grid are the following:

- C : [1.0, 2.0, 3.0]
- kernel: [linear, Radial basis function, polynomial]
- degree: [2, 3, 4]

The result of this analysis returned a value for the best hyperparameters of $C = 2.0$ and rbf kernel. The degree was ignored since it was meaningful only for the polynomial kernel.

5. Convolutional Neural Network

A **convolutional neural network** or **CNN** is a particular type of artificial neural network inspired by the structure of the human retina, in which sensory neurons have a so-called **receptive field**: a limited region in which they respond to a visual stimulus. In a CNNs this mechanism is mimicked by connecting each neuron in the first hidden layer to a small number of connected input neurons.
There are multiple neurons with the same receptive field

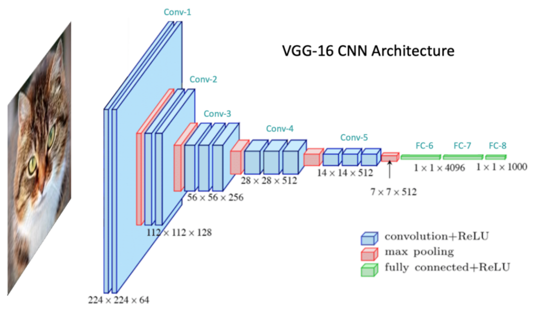

Figure 5. Scheme of the architecture of a Convolutional neural network Learn OpenCV

that respond to different stimuli and compute different features. In a successive layer is computed a **pooling** over small regions: only the maximal activation of the neuron in this region is passed on. The idea is to maintain the knowledge that a feature has been detected and only swell the information exactly where it was found since the relative position of features is more important than their exact position in the image.
In a CNN there are multiple stages of:

- convolution
- maximum pooling
- REctified Linear Units (RELUs) to modulate the activation of neurons

and such stages build a hierarchy of image features that are used in a final fully connected layer or multiple layers to perform classification.

5.1. Architecture

The CNN used to perform the image classification task is the following:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 20, 83, 83]	4,880
MaxPool2d-2	[-1, 20, 41, 41]	0
Conv2d-3	[-1, 40, 17, 17]	64,840
MaxPool2d-4	[-1, 40, 8, 8]	0
Conv2d-5	[-1, 80, 8, 8]	28,880
MaxPool2d-6	[-1, 80, 4, 4]	0
Dropout-7	[-1, 1280]	0
Linear-8	[-1, 500]	640,500
Dropout-9	[-1, 500]	0
Linear-10	[-1, 346]	175,350
Linear-11	[-1, 251]	88,101

Total params:	999,543
Trainable params:	999,543

To enter more in the details of this network let us talk about thew convolutional layers: it is formed by 3 convolutional layers that extract all the relevant features from the image using 2 different kernel sizes.

- **First convolutional layer**: kernel size 9, stride 3, no padding
- **Second convolutional layer**: kernel size 9, stride 2, no padding
- **Last convolutional layer**: kernel size 3, stride 1, padding 1

In the first layer the network extract the details of the pictures with a 'large' kernel of size 9x9 that capture coarse details of the objects with a stride of three, which means that the kernel moves of three steps each time. In the second layer we still have a kernel size of 9x9, but the stride is set to 2 for capturing finer details. At the end there is a smaller kernel of size 3x3 that extracts in depth the features with stride 1 and padding 1. At every stage it is applied both max pooling of size 2x2 and a leaky ReLU activation function. The first one takes the maximum between every 4 inputs in order to diminish the number of parameters, while the leaky ReLU is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. When the feature extraction is over, there are three more fully-connected layers that work as a classifier ending up in a sequence of 251 outputs which will be the prediction. There are two more details to consider: the dropout layers and the softmax.

The term dropout refers to dropping out the nodes in a neural network in order to prevent overfitting and regularize the network. In fact, the CNN learns through the epochs to classify the images in the train set again and again. At a certain point it becomes too specific and learns too much details on those images without generalize the information. For this purpose, it was already mentioned the data augmentation, but also dropout is a wise solution to avoid this problem. The softmax is an activation function widely used for classification tasks. It takes in a vector of raw outputs of the neural network and returns a vector of probability scores. The equation of the softmax function is given as follows:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

In this network it is used the log softmax, whose equation is the following:

$$\log(\text{softmax}(\mathbf{z}))_i = \log\left(\frac{e^{z_i}}{\sum_j e^{z_j}}\right)$$

6. Training Process

The training process for our neural network model was conducted using the PyTorch framework. The key components of the training process include the loss function, optimizer, and learning rate scheduler. Below, we describe each of these components and the overall training loop.

6.1. Loss Function

It was used the Cross Entropy Loss, which is appropriate for multi-class classification problems. This loss function combines the log softmax and negative log likelihood loss in one single class. It can be defined as follows:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for the i-th class. [7]

6.2. Optimizer

The Adam optimizer was employed to update the model parameters. Adam is an adaptive learning rate optimization algorithm designed for training deep neural networks. The learning rate was set to 0.0001. The Adam update rule for parameter θ_t at iteration t is:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

where α is the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected first and second moment estimates. [8]

6.3. Learning Rate Scheduler

To adjust the learning rate during training,Cosine Annealing scheduler was adopted. This scheduler adjusts the learning rate following a cosine function, allowing it to decrease from the initial learning rate to a minimum value (in this case, 10^{-8}) over a specified number of iterations. This can help improve the convergence and generalization of the model. [9]

6.4. Training Loop

The training process was performed over multiple epochs, and the key steps in each epoch are as follows:

Training Phase: The model is set to training mode, which enables dropout and other useful functions for ML. The gradients of the model parameters are zeroed before the forward pass is performed to compute the outputs and the loss. After this, the backward pass is performed to compute the gradients and the optimizer updates the model parameters based on the computed gradients while learning rate is updated using the scheduler. During this process labels, ground truth and loss values are stored to keep track of the progresses during this phase.

Validation Phase: The model is now set to evaluation mode, which disables dropout to test the complete power of the network. A forward pass is performed with gradient computation disabled while the ground truth labels and predictions are stored for accuracy computation. Validation loss is computed and stored.

Early Stopping: After each epoch, the average validation loss is computed. If the validation loss improves, the model state is saved. If no improvement is observed for a predefined number of epochs (patience=10), training is stopped early.

The training has been performed for $\sim$100 epochs, then the network reached the tenth level of patience and the training stopped. The validation loss and accuracy at the best epoch (not the last) are 3.51 and 0.21 respectively.

7. Evaluation metrics

The final part of this project is the evaluation of the results. There are a lot of evaluation metrics suitable for supervised classification tasks, they typically involve comparing the predictions made by the model with the ground truth labels that are provided in the dataset.
These measures are based on the following classification:

- **True Positive (TP)**: The number of instances correctly predicted as positive.
- **False Negative (FN)**: The number of instances incorrectly predicted as negative when they are actually positive.
- **False Positive (FP)**: The number of instances incorrectly predicted as positive when they are actually negative.
- **True Negative (TN)**: The number of instances correctly predicted as negative.

7.1. Accuracy

Accuracy is a measure of the overall correctness of the model. It is the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

7.2. Precision

Precision, also known as positive predictive value, is the ratio of correctly predicted positive instances to the total predicted positives. It indicates how many of the predicted positive instances are actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

7.3. Recall

Recall, also known as sensitivity or true positive rate, is the ratio of correctly predicted positive instances to the total actual positives. It measures the ability of the model to identify all positive instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

These metrics provide different perspectives on the performance of a classification model. While accuracy gives an overall performance measure, precision and recall provide insights into the model's ability to correctly identify positive instances and avoid false alarms, respectively.

For the CNN there is also an additional metric: the top-5 accuracy. Conceptually it is not different from the traditional accuracy, but in this case it takes in account the first 5 ”guesses” of the network instead of the first one.

8. Results

In this section are displayed all the results of the classification and the values of the metrics discussed above.

- **SIFT/BOW**

Accuracy score (%):	6.1
Precision score (%):	5.2
Recall score (%):	5.8

- **CNN**

Accuracy score (%):	23.1
Top-5 Accuracy score (%):	49.0
Precision score (%):	22.1
Recall score (%):	22.2

9. Conclusion

In conclusion, it is clear that the CNN performed much better in solving this task than the other method based on SIFT and BOW feature extraction. The reasons for this difference of performance are principally two:

1) Computational cost
2) Different feature extraction optimization

The first one is more related to this specific analysis, since the implementation and fine tuning of the SIFT algorithm has been more time consuming and also the computational cost was really high. This method required a larger amount of space with respect to the CNN since this could take advantage of a data loader that did not gave any problem in the storage. Furthermore, CNN libraries allow to perform the training in GPU, while it is not possible to do the same with the other strategy. The second reason why CNN performs better is related to the structure of the two algorithms: in the first case, the feature extraction and the training are two separated things and do not communicate, while in CNN's are all connected and learn to optimize both feature extraction and classification simultaneously through backpropagation, directly from raw pixel data. This allows the network to learn the most relevant features for the specific task at hand. This aspect made CNN's a most effective method in computer vision.

It would be possible to improve this project in many ways. The SIFT/BOW feature extractor can be made much more effective by increasing the number of visual words in the dictionary, a richer vocabulary would lead to a better description of the images. Also the classifier performance be improved by loading more images, if it were possible to handle them without running out of RAM. The CNN could be improved by adding batch normalization to regularize the training process and use a higher learning rate.

Disclosure statement

The authors declare that this report is entirely original and does not contain any plagiarism. The research explained in this report was conducted by the authors themselves, and all the sources have been cited in the Reference Section. None of the content was generated using automated language models.

References

[1] Data Preprocessing in Machine Learning: Steps and Best Practices

[2] Machine Learning — Multiclass Classification with Imbalanced Dataset

[3] Data augmentation in image classification

[4] Distinctive Image Features from Scale-Invariant Keypoints

[5] Bag of Visual Words

[6] Bag of visual words in a nutshell

[7] Cross-Entropy Loss - PyTorch

[8] Adam Optimizer - PyTorch

[9] Cosine Annealing Scheduler - PyTorch

[10] Support vector machines - Sklearn

[11] Evaluation Metrics — Supervised ML