Master's degree in Computer Engineering for Robotics and Smart Industry

Machine Learning & AI Project

SOLVING A CLASSIFICATION PROBLEM THROUGH SUPERVISED LEARNING:

Implementation and comparison of different *Machine Learning* and *Deep Learning* techniques to classify 6 classes of images

Academic year: 2020-2021

Authors: Michele Sandrini (VR465391), Fabio Castellini (VR464639)

Index

1. MOTIVATION AND RATIONALE

The proposed project aims at implementing, testing and comparing different ***classification algorithms***, in combination with feature extraction and reduction methods. Machine learning approaches such as k-Nearest Neighbors and multiclass Support Vector Machine are exploited, as well as deep learning neural networks, such as *ResNet18* and *VGG16*.

The project addresses the classification of ***6*** different ***natural scene images***. The chosen dataset hasn't an immediate industrial application of the resulting predictions, but it can be surely seen as a research and academic challenge, that can lead to improvements of the classification techniques in the future. Moreover, the same techniques can be applied to classify other datasets, that can be more application based, from an industrial point of view.

As it will be explained further, the six classes have some *similarities* and *ambiguities*, so that a study on *feature selection*, *extraction* and *hyperparameter tuning* of the algorithms can be done.

2. STATE OF THE ART

Since the breakthrough of AlexNet[1], Convolutional Neural Networks (*ConvNets*) have been the dominating model architecture for computer vision[2] and image classification. Many models have been presented, especially in the last decade, producing a higher and higher *top-1 accuracy*, tested for example on the ImageNet dataset[3]. Currently the best model is CoAtNet-7 that reached a 90.88% accuracy[4].
According to the "Recent Advances in Convolutional Neural Networks" (2017)[5] paper, several deep learning concepts can be exploited to solve our classification problem. Like ReLU (*Rectified Linear Unit*), *Softmax* loss function, *data augmentation* (random crop: RandomResizedCrop(), mirroring: RandomHorizontalFlip()), *Stochastic Gradient Descent* and *batch normalization.*

To notice that our problem is a ***supervised learning classification***, which relies on annotations, that are not easy to collect, and for this reason more recent models aim at reaching high accuracies with fewer and fewer data (*few-shot transfer*) or using *pseudo-labeling techniques*[6].

Moreover we are interested in classic machine learning techniques, to classify our images and to compare image classification algorithms based on traditional machine learning with deep learning.
The "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning"[7] study shows that usually traditional machine learning has a better solution effect on small sample datasets, while deep learning frameworks have higher recognition accuracy on large sample datasets. However, a machine learning method like SVM, has a remarkable smaller training time. Thus, our dataset, with more than 2000 images per class, can be considered a *large* dataset and we found out that CNN reaches noticeably higher precisions than SVM.

About the machine learning techniques, we will use SVM classifier with some kernels, RBF (*Radial Basis Function*), polynomial, sigmoid and linear[7] and KNN (*k-Nearest Neighbors*) with different k values and metrics: Euclidean, Cosine, Jaccard, Mahalanobis[8]. According to a recently published paper[9], KNN has the main advantage of being simple and relatively efficient, but several variants of the standard algorithm can be explored, for instance Locally adaptive KNN and Weight adjusted KNN.

If standard machine learning techniques lead to low accuracy results, a possible solution[10] could be to use neural network ability to extract complex and meaningful features, and to use them as inputs for the machine learning models.

3. OBJECTIVES

The main objectives of the project are:

a) To **implement**, **test** and **compare** several **supervised learning techniques for image classification**, starting from some classical machine learning (KNN and SVM) methods up to Convolutional Neural Networks (VGG16 and ResNet18), **tuning** the algorithm's hyperparameters, based on certain evaluation metrics;

b) to **use different** kinds of **features** to assess which one produces the best accuracies for each method;

c) to analyze which are the **features** that yield the **best results for** each single **class**.

In doing so, our last goal is:

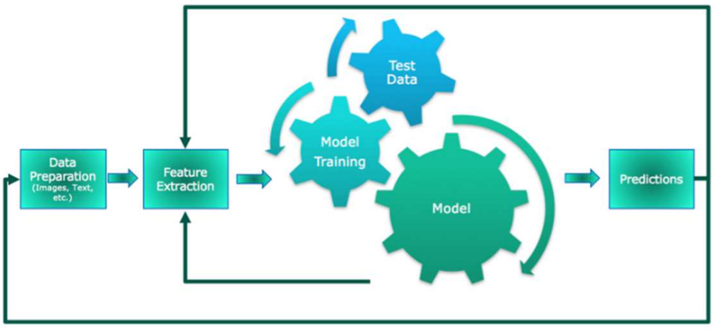
d) to determine the ***most accurate approach*** to solve our image classification problem.

4. METHODOLOGY

4.1. THE MACHINE LEARNING PIPELINE

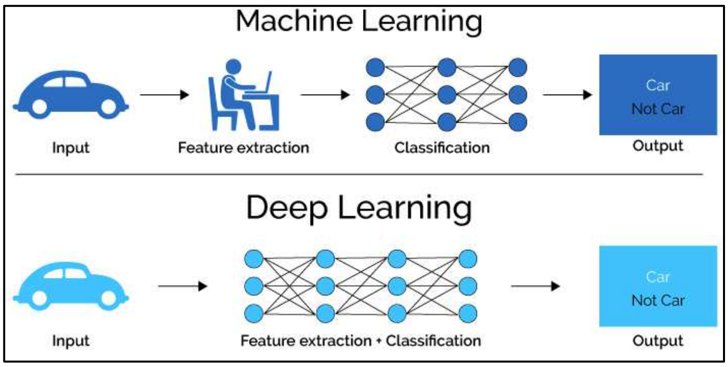
To solve the classification problem, the standard Machine Learning pipeline has been used. It's composed by several steps:

- ***Data preparation***
- ***Feature extraction***
- ***Feature selection***
- ***Model selection***
- ***Model training***
- ***Prediction***
- ***Model testing***



4.2. MACHINE LEARNING VS DEEP LEARNING

The main difference from Machine Learning and Deep Learning approaches is the prior knowledge. In fact, if Neural Networks are used, not only *feature selection* and *model selection*, but also *feature extraction* and *prediction* processes are considerable prior knowledge from the CNN's point of view. Features are constantly learned during model training, through layers of convolutional filters, core of CNN architectures.



4.3. DATASET

An image dataset of *"Natural Scenes around the world"* has been chosen[11]. According to the *kaggle.com* page from which the dataset was downloaded, this collection of images was published by Intel to host an image classification challenge. In particular, it contains around **17k** images of size **150x150** pixels distributed under ***6 categories***: *buildings, forest, glacier, mountain, sea, street.* Of the 17k, 14034 images were used for model training and 3000 for model testing, following the 80/20 rule of thumb for splitting train and test sets.

Class: BUILDINGS

Class: FOREST

Class: GLACIER

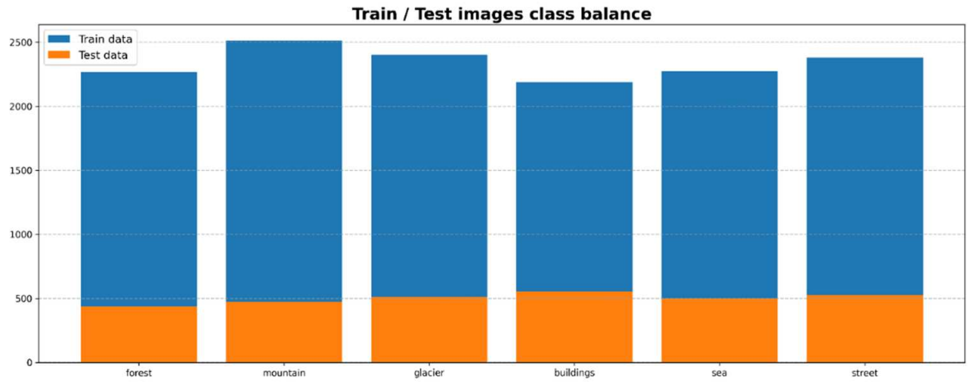
Class: MOUNTAIN

Class: SEA

Class: STREET

Revising the dataset, some images seem ambiguous due to the presence of animals, humans or objects related to the environment they're in, but they can be misleading for the overall training process. Moreover, being glaciers, mountains and forests so related, sometimes those images are not so trivial to distinguish, even by humans. Because of the similarity between classes, correctly classifying this dataset isn't as trivial as it seems at first glance. It was chosen, to understand how to better solve the classification problem, focusing not only on the type of ML/DL algorithm but also on the *feature extraction methods*.

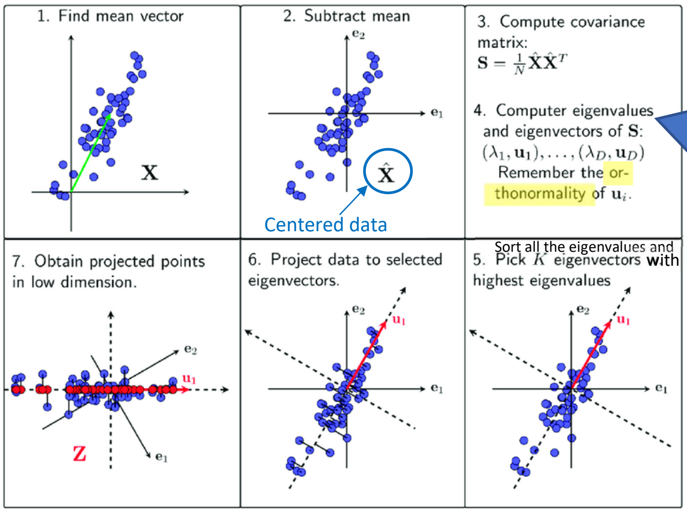
On the following histogram the number of images per class for training and testing are shown. As can be seen, the six classes are *well balanced*. This analysis was performed at the beginning, being class imbalance a problem that must be solved using different performance metrics and approaches.



4.4. PCA: FEATURE REDUCTION

Once the features are extracted from images, a dimensionality reduction algorithm is applied: PCA. *Principal Component Analysis* is one of the most popular ones and it's based on the observation that data is often not randomly distributed in space but near specific lines[13]. For this reason, the algorithm projects data into a smaller dimension vectorial space, whose axes retain most of the original information. A trade-off between wanted variance and cardinality of the feature space must be found.

To present the algorithm steps, we use a schematic image[14] [15]:

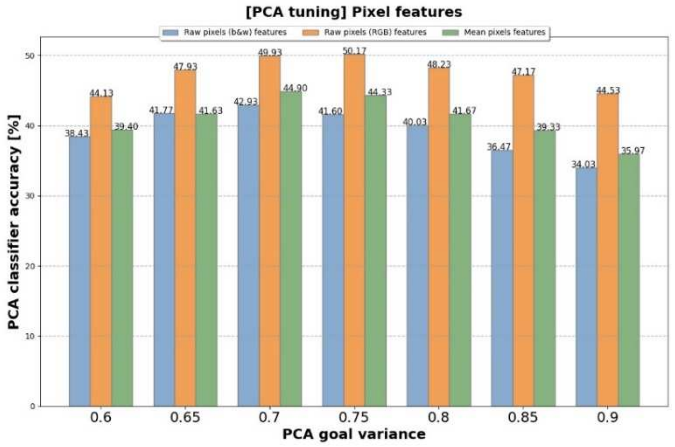

1. Find mean vector
$\mathbf{X}$
2. Subtract mean
$\mathbf{e}_2$
$\mathbf{e}_1$
$\hat{\mathbf{X}}$
Centered data
3. Compute covariance matrix:
$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$
4. Computer eigenvalues and eigenvectors of $\mathbf{S}$:
$(\lambda_1, \mathbf{u}_1), \ldots, (\lambda_D, \mathbf{u}_D)$
Remember the orthonormality of $\mathbf{u}_i$.
7. Obtain projected points in low dimension.
$\mathbf{e}_2$
$\mathbf{u}_1$
$\mathbf{Z}$
$\mathbf{e}_1$
6. Project data to selected eigenvectors.
$\mathbf{e}_2$
$\mathbf{u}_1$
$\mathbf{e}_1$
Sort all the eigenvalues and
5. Pick K eigenvectors with highest eigenvalues
$\mathbf{e}_2$
$\mathbf{u}_1$
$\mathbf{e}_1$

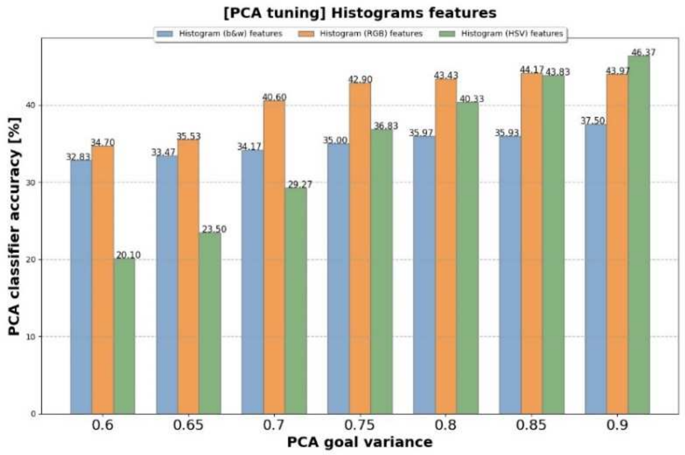
NOTICE:
Eigenvectors are the **Principal Components (PC)**, which represent the *directions of the data* that explain a ***maximal* amount of *variance* (= **minimal residual sum of square**)**, that is to say, the lines that capture most information of the data.

Eigenvalues give the *amount of variance carried in each Principal Component* [16]

In the project, a ***tuning** process* based on the PCA classifier's accuracy was performed, in order to choose the optimal number of Principal Components (*eigenvectors*), or equivalently, the optimal relative total variance they carry. We started from the PCA code we implemented from scratch during laboratory sessions and we used the scikit learn PCA library, just to verify the results' correctness.

The following figures show the PCA tuning process' results, both for pixel and histogram features. As mentioned, the aim is to retrieve the optimal variance amount (spanning from 0.6 to 0.9 with an increment of 0.05) that leads to the highest PCA classifier accuracy.



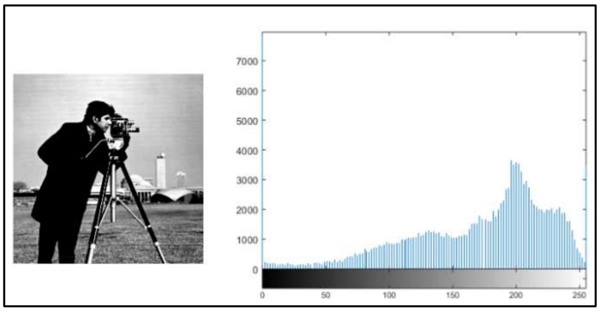


4.5. KINDS OF FEATURES

In this project, we exploited different typologies of feature, to assess their efficiency in describing the images of "NatureDataset".

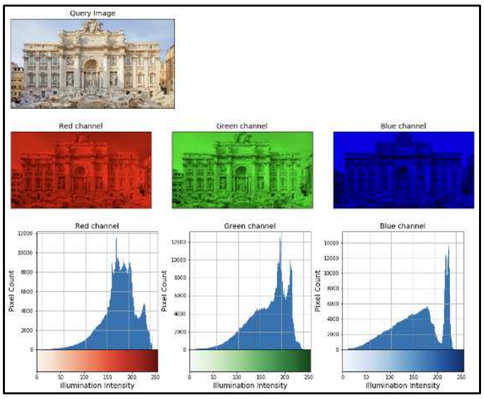
a) Greyscale histograms (or B&W -Black & White- histograms or Intensity histograms)

This histogram is a graph showing the number of pixels in an image at each different intensity value found in it. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 bins showing the distribution of pixels amongst those grayscale values. Intensity values can also be mapped in a range from 0 to 1 of floats.



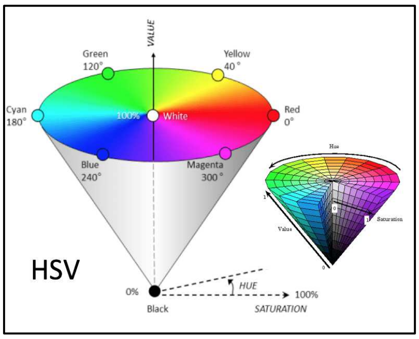
A ***color space*** is a coordinate system where a single point represents a distinct color value. There are several well-known color spaces that are used to represent pixels of an image. Inside this project we chose *RGB* and *HSV* histograms that are the most common ones.

b) RGB histograms



The RGB color space contains three color components: *Red*, *Green* and *Blue*.
The RGB histogram is a graph showing the distribution of *each primary channel's brightness* level in the image. On the right image case, the horizontal axis indicates the color's brightness level (brighter on the left and darker on the right), while the vertical axis indicates how many pixels exist for each level.

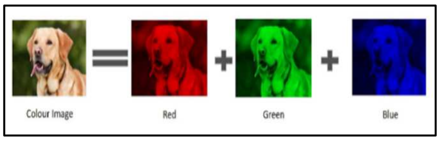
c) HSV histograms



The HSV (*Hue*, *Saturation*, *Value*) color space was designed to represent colors as similar as possible to ***human vision perception***. In this model, color space is represented as a three-dimensional *hexacone*, where the central vertical axis represents intensity which takes a value between 0 and 255 (from black to white). Hue is defined as an angle in the range [0,2π] relative to the red axis with red at angle 0, green at 2π/3, blue at 4π/3, and red again at 2π[17].

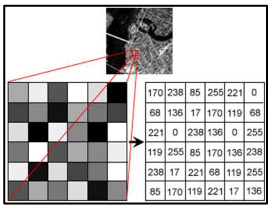
RGB and *HSV histograms*, encode just the *color content* information (or intensities in the greyscale) in the image, disregarding all the other information. On the other hand, from *RGB pixels* can be retrieved also "*colors' position*" inside the image. *Greyscale* images instead contain only the ordered pixels' intensities.

d) RGB pixels



Images in RGB model, are 3 *R*x*C* matrices. They can be seen as the "overlapping" of three different images of the same size, each one containing the image content respectively of *Red*, *Green* and *Blue* channels. Each value is in range [0,255], or equivalently, normalized in range [0,1].

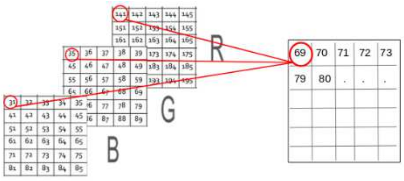
e) Greyscale (or B&W -Black & White- or intensity) pixels



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Greyscale images are *R*x*C* matrices (*R* and *C* are numbers of pixels in Rows and Columns), in which every element assumes a value ([0,255]) representing the pixel's intensity, where 0 is black and 255 is white.

f) Mean pixels



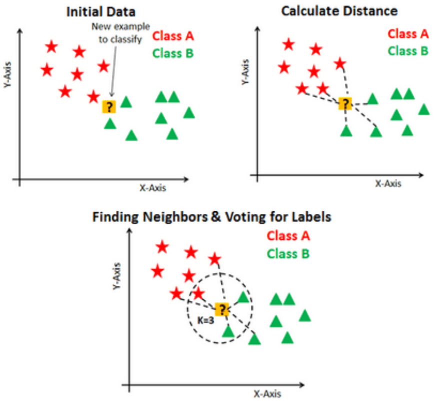
```
ImageRGB                                # Matrix NxMx3
for i in (1,N):
    for j in (1,M):
        NewImg(i,j) = ( ImageRGB(i,j,R) + ImageRGB(i,j,G) +
                        ImageRGB(i,j,B) ) / 3

Mean_pixels_image = NewImg      # Matrix NxMx1
```

To compute these mean pixels images, as depicted in above picture and pseudocode, we start from RGB images. *For each pixel* the ***mean*** *between the three channels* is computed and becomes the new image's pixel value. The final image will have dimension NxMx1.

4.6. K-NEAREST NEIGHBORS ALGORITHM

K-Nearest Neighbors is a *non-parametric classifier algorithm*, meaning that no assumption on the probability density function's shape is made. Also, it is used in *supervised learning* conditions: ground truth labels are known. It is relatively easy to implement and understand, flexible and gives reasonable accuracies, even though finding distance between points can be computationally expensive[8].



The algorithm can be summed up in 3 steps:

Given a set of X training examples and a new example to classify:

- compute the ***distance*** (using a certain distance metric, e.g.: Euclidean) from the new point to all the other points and remember the *K* nearest points' classes;
- find the ***most frequent class*** "C" that appears inside the new point's "neighborhood";
- ***classify*** the new example as "C".

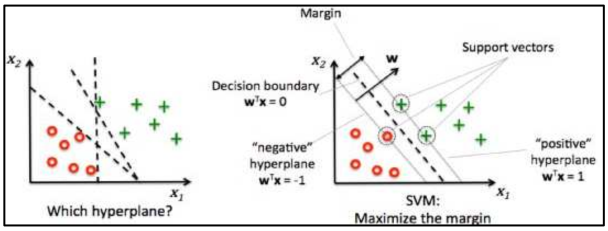
The result depends on two hyperparameters: the *distance metric* and *K*.
The first defines how to compute the distance between the new example to classify and the other points. One of the most common metrics is the *Euclidean distance*. *Cosine distance*, for example, considers the angle between feature vectors and has the advantage to be independent on their magnitude[8]. The value of *K*, on the other hand, defines how many points should be considered as neighbors of the new example to be classified.

4.7. SUPPORT VECTOR MACHINE ALGORITHM

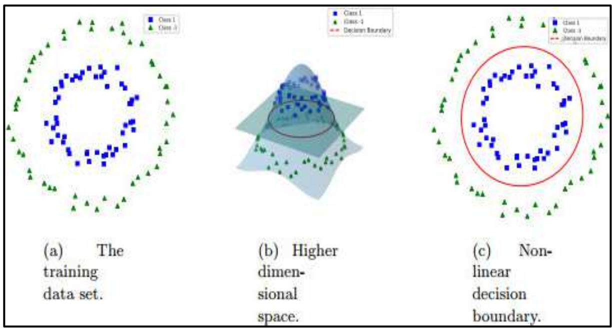
Support Vector Machine (*SVM*) is a supervised machine learning algorithm that can be used for classification problems. It is a binary classification technique that uses the training dataset to predict an optimal hyperplane in an *n*-dimensional space, called *decision boundary*, to separate this dataset into two classes.
To perform this task, ***Support Vectors*** are used: they're the closest data points to the separating hyperplane, for each class; all other points instead, don't affect the decision boundary choice. The distances between hyperplane and support vectors are called ***margins***. The goal of a support vector machine is to maximize those *margins*, in order to determine the optimal decision boundary.

We mainly focused on the ***kernel*** hyperparameter choice:

- ***Linear SVM classifier:*** used if data is linearly separable; the optimal decision boundary is a *linear hyperplane* in the feature space;



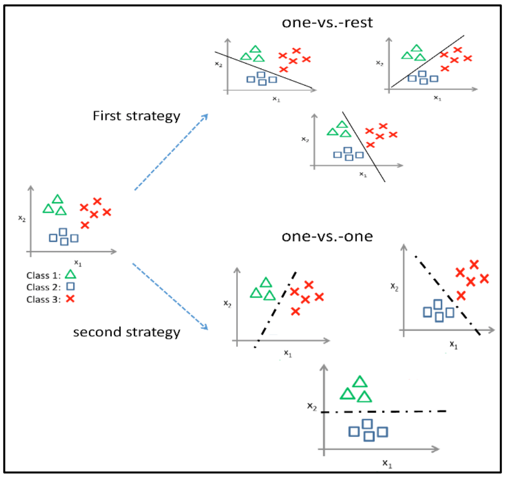
- ***Non-linear SVM classifier:*** used for non-linearly separable data. ***Kernel trick*** is used, where kernel is a function that maps data points into the higher-dimensional space, such that they become linearly separable. Then, as shown in the nearby figure, the hyperplane that separates points in a higher-dimension problem space is mapped back to the original problem space, resulting in a *non-linear* solution.



(a) The training data set. (b) Higher dimensional space. (c) Non-linear decision boundary.

SVM doesn’t support multiclass classification natively.
However there exists different approaches to use binary classification algorithms for multi-classification problems, like ***One-vs-Rest*** and ***One-vs-One*** strategies:

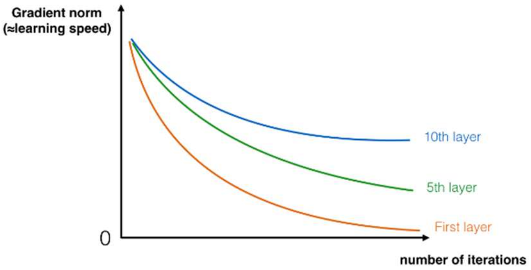
- The ***One-vs-Rest*** strategy tries to find the decision boundary between a class and all the others (N classifiers needed; $N =$ number of classes);
- The ***One-vs-One*** strategy tries to find decision boundaries for each possible pair of classes. This approach is less effective from a computational point of view (*N·(N-1)/2* classifiers needed).

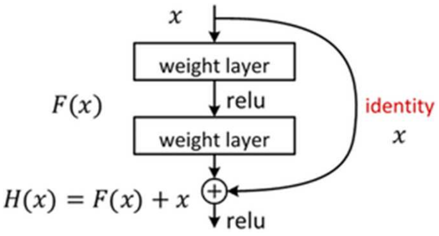


4.8. CNN COMPARISONS: VGG16 VS RESNET18

VGG stands for *Visual Geometry Group* and is one of the most popular “deep” Convolutional Neural Network (CNN) architectures with multiple layers. It consists of 16 convolutional layers, but many other variants exist; it was created in 2014 out of the need to reduce the number of parameters inside the layers and improve training time[18].
R*esNet* stands for Residual Neural Network and is another type of CNN. It was created to solve the main problem of VGG and other similar architectures: adding more layers to a CNN leads to better performances, but VGGs, due to the ***gradient vanishing problem***, can’t have a high number of layers[19].

The weights of a neural network are updated through the ***backpropagation*** algorithm, which makes a minor change to each weight so that the model’s loss decreases. Indeed, according to the chain rule, the derivatives of each layer are multiplied down the network (from the final to the initial layer) to compute the derivatives of the initial layers. However, if a VGG architecture is used, increasing the number of layers, results in the gradient becoming smaller and smaller, meaning that the initial layers’ weights will be less and less affected by changes. In the end, training time will increase significantly, as shown in the following images.

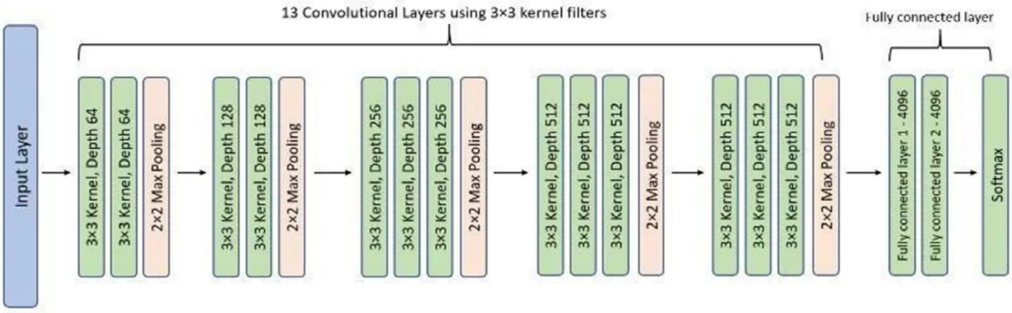




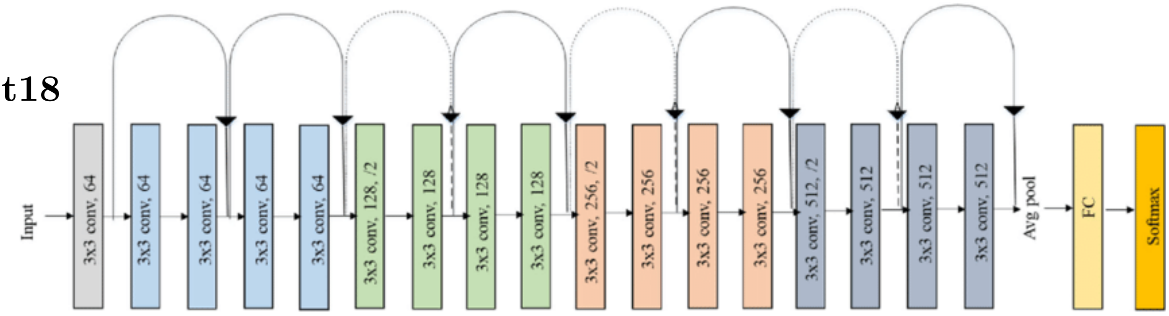
So, the *gradient vanishing problem* can be solved if the local gradient is always higher than 1, that is achieved by ResNets using the identity function. As the gradient is backpropagated, it does not decrease in value because the local gradient can’t be less than 1.

In the following images the VGG16 and ResNet18 architectures are shown:

VGG16



ResNet18



5. EXPERIMENTS & RESULTS

As mentioned before, in our experiments we used several features, in combination with some algorithm's hyperparameters. Regarding the machine learning approaches (KNN, SVM), we tested the following ***feature-hyperparameters*** combinations:

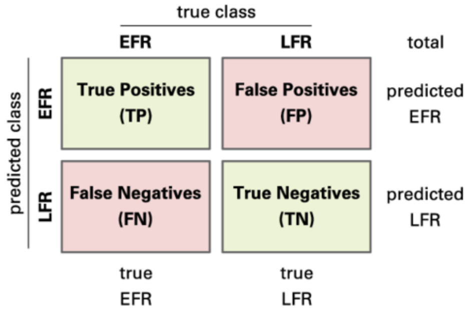
a) FEATURE: greyscale image histograms with (§)(*)
b) FEATURE: RGB image histograms with (§)(*)
c) FEATURE: HSV color image histograms with (§)(*)
d) FEATURE: RGB image pixels with (§)(*)
e) FEATURE: greyscale image pixels with (§)(*)
f) FEATURE: RGB channel mean for each pixels with (§)(*)

(§) KNN *K value*: 1, 3, 5, 7 and *distance metric*: Euclidean, Cosine, Jaccard, Mahalanobis
(*) SVM KERNEL: *RBF*, *polynomial*, *sigmoid* and *linear*.

5.1. EVALUATION PROTOCOL

For each combination of feature and hyperparameter, we compute the overall ***accuracy***, ***precision***, ***recall*** and ***f1-score***. We analyze the best and worst *accuracy* results, assessing in this way which combination model-feature-hyperparameter achieves best classification performances.

In particular, model's *accuracy* represents the total number of correct predictions with respect to the total number of predictions; *precision* defines how trustable the model is when predicting a class. Instead, *recall* of a class expresses how well the model is able to detect that class. *F1-score* of a class is given by the harmonic mean of precision and recall, combining them into one metric.



		true class		
		EFR	LFR	total
predicted class	EFR	True Positives (TP)	False Positives (FP)	predicted EFR
	LFR	False Negatives (FN)	True Negatives (TN)	predicted LFR
		true EFR	true LFR	

$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

$$\mathrm{CA} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$

$$\mathrm{F}_1 = \frac{\mathrm{2TP}}{\mathrm{2TP+FP+FN}}$$

For each kind of features, we keep the most accurate model among all the tested hyperparameters and we analyze its confusion matrices, in combination with the per class precision and recall, provided by the scikit-learn function *"classification_ report"*[12]. Our aim is to answer questions like: does the model easily misclassify a class with another one? Is there a class that is more/less easily recognized than others? Is the class well detected but also misclassified? Does the model struggle to detect a class, being trustable when it does?

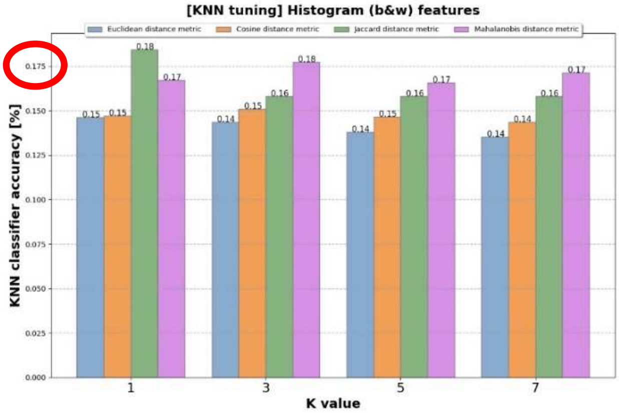
5.2. K-NEAREST NEIGHBORS TUNING PROCESS & RESULTS

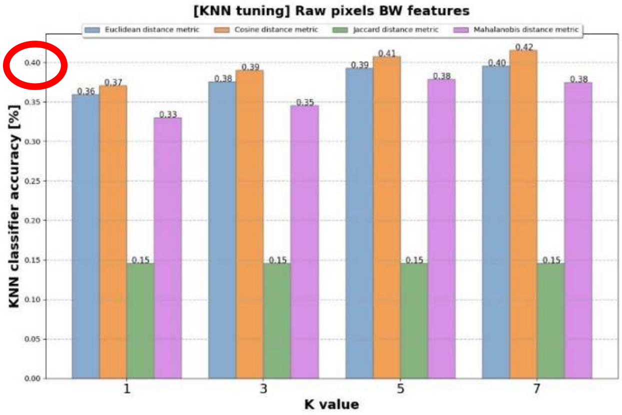
As mentioned before, several feature extraction methods were used. In the following pages will be compared the performances achieved by KNN using different ***distance metrics*** and ***K***-values, for every feature type.
To determine the two main parameters inside the KNN algorithm, an *automatic tuning process* was implemented. Given a type of feature, a fixed image resolution (32x32) and the best PCA variance parameter (previously computed through the PCA tuning process), all the possible combinations of K (1, 3, 5, 7) and distance metric (Euclidean, Cosine, Jaccard, Mahalanobis) were tested.

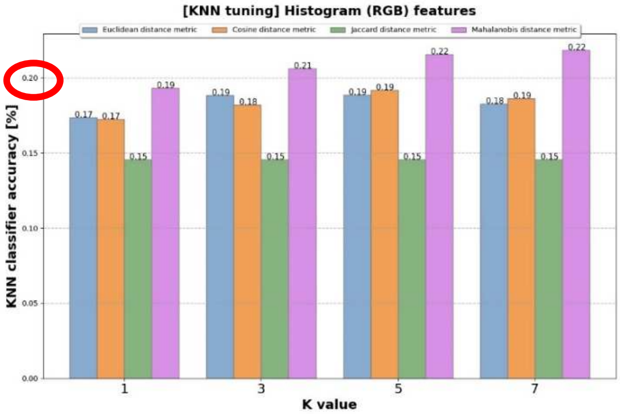
In particular, the kinds of *distance metrics* that have been considered are:

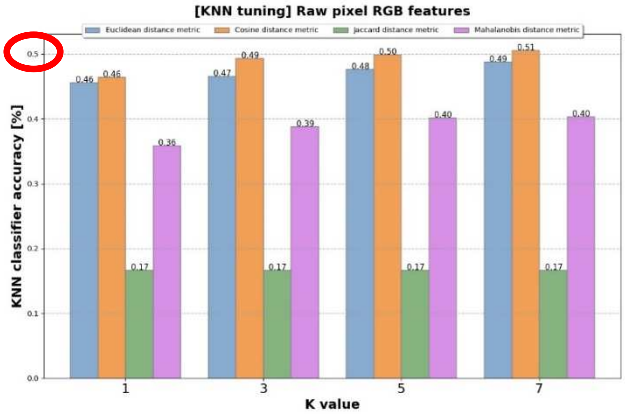
- Euclidean $d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$
- Cosine $\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$
- Jaccard $J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$
- Mahalanobis $D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}.$

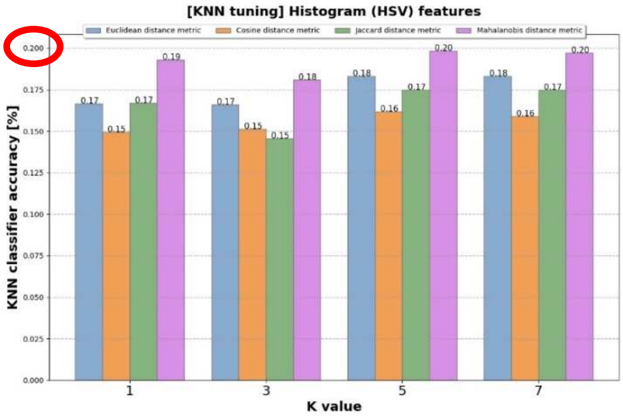
The following histograms show how the above-mentioned parameters impact on the KNN performances:

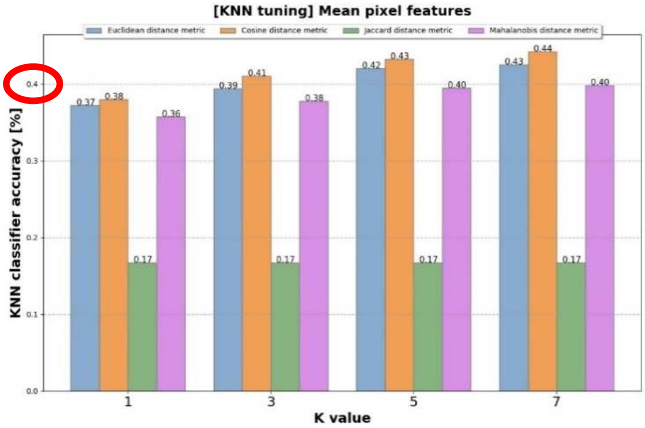




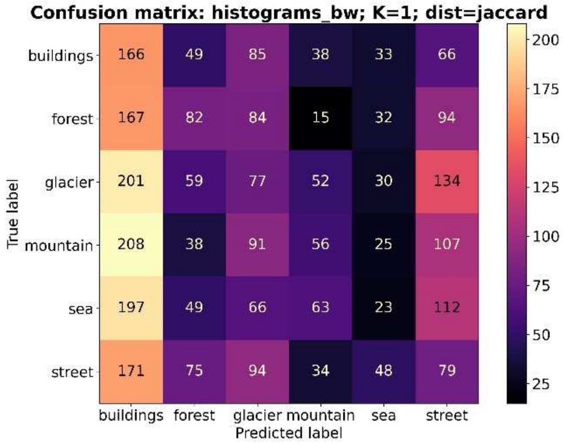


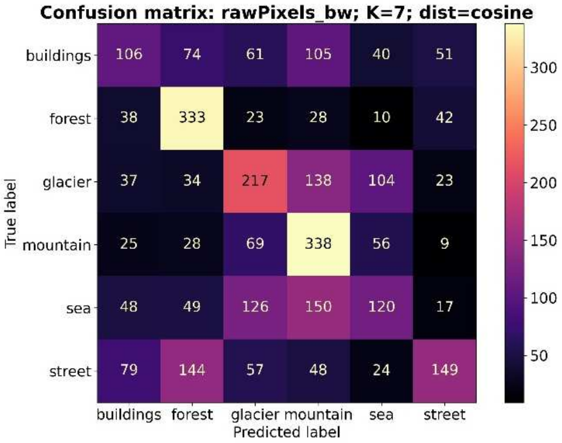


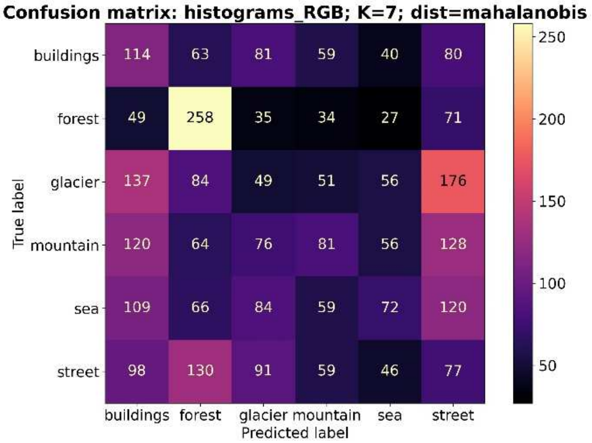


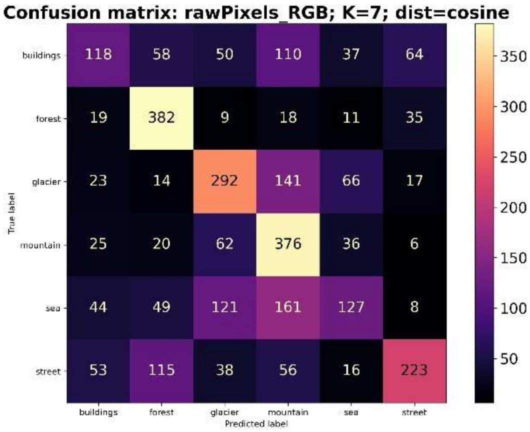


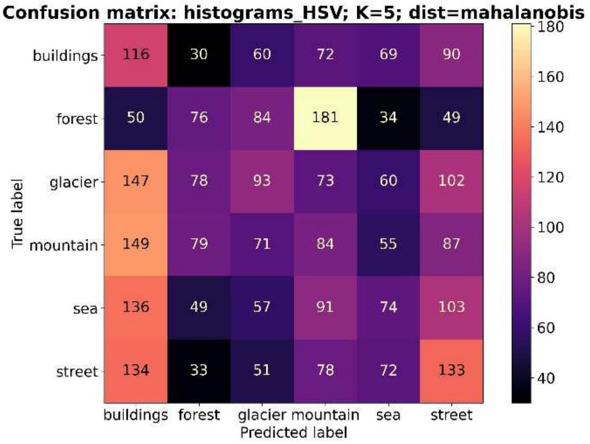
The best confusion matrices for each feature extraction method:

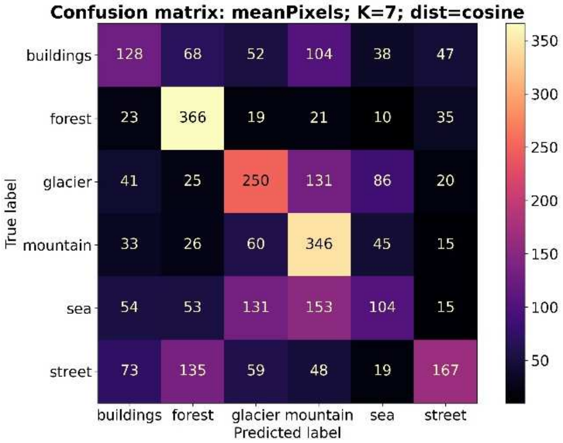

Confusion matrix: histograms_bw; K=1; dist=jaccard


Confusion matrix: rawPixels_bw; K=7; dist=cosine


Confusion matrix: histograms_RGB; K=7; dist=mahalanobis


Confusion matrix: rawPixels_RGB; K=7; dist=cosine


Confusion matrix: histograms_HSV; K=5; dist=mahalanobis


Confusion matrix: meanPixels; K=7; dist=cosine

Comments about the previous results:

Looking at the accuracy plots, ***pixel features*** *perform far better than histogram features*. In particular, the colour (RGB or HSV) information benefits the overall KNN performances but doesn't outperform black and white only features (both pixels and histograms).

With our dataset the colour information should be useful to solve the classification problem, but a lot of *images are ambiguous* if only colours are considered. To explain the far better performances of pixels, we can say that the position of pixels inside the image, helps a lot. This information isn't preserved looking only at histograms. Also, being colours useful, HSV and RGB histograms' results are slightly better than B&W ones.

Moreover, according to the KNN tuning process, the best hyperparameters found are *Cosine distance* for pixels, *Mahalanobis distance* for histograms, while the *K value* varies *from 5 to 7* (excluding black and white histogram features).

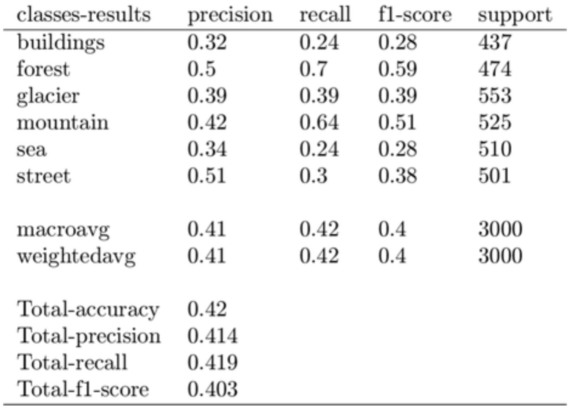
Looking at the confusion matrices, “streets” are often seen as “forests” or “buildings”; “sea” is often seen as “mountain” or “glacier”; “glacier” and “buildings” are sometimes classified as “mountain”. Considering the KNN algorithm, which is based on the distance between samples, within a certain neighbourhood, this behaviour makes sense. In fact, if there’s *similarity between the classes of images*, it will be more likely to have variety among the neighbors, due to the common features. Also, performances are highly affected by the distance metric and some state-of-the-art papers[8] suggest more complex approaches, to determine the distance between points. Further on, more representative features will be exploited, using a CNN as a feature extractor.
Looking at the histograms’ confusion matrices, lots of mistakes are made and probably, not only because of the ambiguity among the images, but also because of the *not so suited feature type*.

Classification Reports

Using the *sklearn.metrics.classification_report*[12] function, *precision, recall, f1-score* and *support* were collected, for every single class. Moreover, the overall *accuracy, precision, recall, f1-score, macro* and *weighted averages* are displayed. To notice that only the “***best scenario***” classification reports are shown: meaning that *K* and *distance* parameters were the best ones we got after the tuning process (based on the KNN’s overall accuracy), given a certain feature.
The following tables show in a quantitative manner, evaluation metrics for each of the six classes of the Nature Scenes dataset. As mentioned in the evaluation protocol, precision and recall can be also analyzed to understand how sure the model is about its predictions and there are 4 situations:

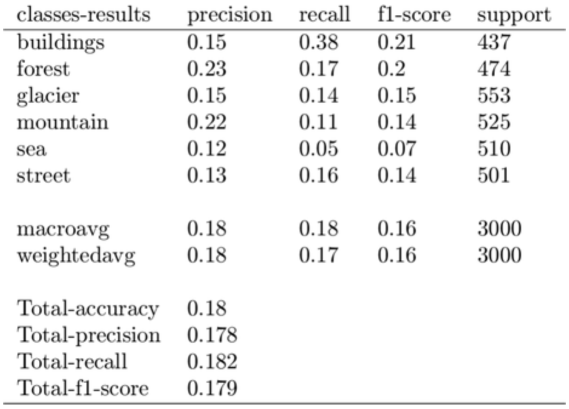
- *high recall & high precision:* the class is perfectly handled by the model;
- *low recall & high precision:* the model can’t detect the class well but is trustable when it does;
- *high recall & low precision:* the class is well detected but the model also misclassifies it;
- *low recall & low precision:* the class is poorly handled by the model.

B&W pixels; K=7; distance=cosine



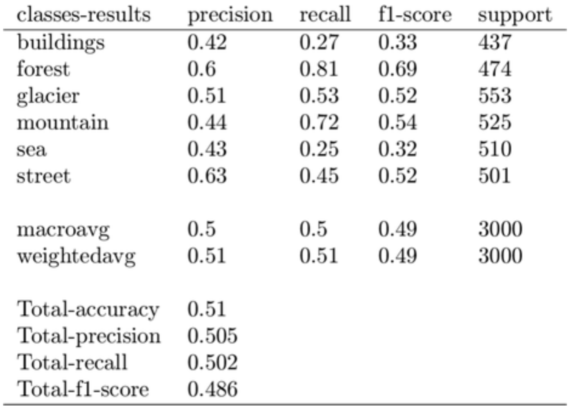
classes-results	precision	recall	f1-score	support
buildings	0.32	0.24	0.28	437
forest	0.5	0.7	0.59	474
glacier	0.39	0.39	0.39	553
mountain	0.42	0.64	0.51	525
sea	0.34	0.24	0.28	510
street	0.51	0.3	0.38	501
macroavg	0.41	0.42	0.4	3000
weightedavg	0.41	0.42	0.4	3000
Total-accuracy	0.42			
Total-precision	0.414			
Total-recall	0.419			
Total-f1-score	0.403			

B&W histograms; K=1; distance=Jaccard



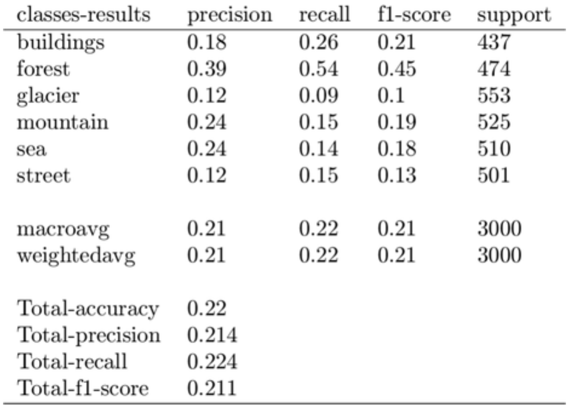
classes-results	precision	recall	f1-score	support
buildings	0.15	0.38	0.21	437
forest	0.23	0.17	0.2	474
glacier	0.15	0.14	0.15	553
mountain	0.22	0.11	0.14	525
sea	0.12	0.05	0.07	510
street	0.13	0.16	0.14	501
macroavg	0.18	0.18	0.16	3000
weightedavg	0.18	0.17	0.16	3000
Total-accuracy	0.18			
Total-precision	0.178			
Total-recall	0.182			
Total-f1-score	0.179			

RGB pixels; K=7; distance=cosine



classes-results	precision	recall	f1-score	support
buildings	0.42	0.27	0.33	437
forest	0.6	0.81	0.69	474
glacier	0.51	0.53	0.52	553
mountain	0.44	0.72	0.54	525
sea	0.43	0.25	0.32	510
street	0.63	0.45	0.52	501
macroavg	0.5	0.5	0.49	3000
weightedavg	0.51	0.51	0.49	3000
Total-accuracy	0.51			
Total-precision	0.505			
Total-recall	0.502			
Total-f1-score	0.486			

RGB histograms; K=7; distance=Mahalanobis



classes-results	precision	recall	f1-score	support
buildings	0.18	0.26	0.21	437
forest	0.39	0.54	0.45	474
glacier	0.12	0.09	0.1	553
mountain	0.24	0.15	0.19	525
sea	0.24	0.14	0.18	510
street	0.12	0.15	0.13	501
macroavg	0.21	0.22	0.21	3000
weightedavg	0.21	0.22	0.21	3000
Total-accuracy	0.22			
Total-precision	0.214			
Total-recall	0.224			
Total-f1-score	0.211			

Mean pixels; K=7; distance=cosine

classes-results	precision	recall	f1-score	support
buildings	0.36	0.29	0.32	437
forest	0.54	0.77	0.64	474
glacier	0.44	0.45	0.44	553
mountain	0.43	0.66	0.52	525
sea	0.34	0.2	0.26	510
street	0.56	0.33	0.42	501
macroavg	0.45	0.45	0.43	3000
weightedavg	0.45	0.45	0.43	3000
Total-accuracy	0.45			
Total-precision	0.447			
Total-recall	0.452			
Total-f1-score	0.434			

HSV histograms; K=5; distance=Mahalanobis

classes-results	precision	recall	f1-score	support
buildings	0.16	0.27	0.2	437
forest	0.22	0.16	0.19	474
glacier	0.22	0.17	0.19	553
mountain	0.15	0.16	0.15	525
sea	0.2	0.15	0.17	510
street	0.24	0.27	0.25	501
macroavg	0.2	0.19	0.19	3000
weightedavg	0.2	0.19	0.19	3000
Total-accuracy	0.19			
Total-precision	0.198			
Total-recall	0.194			
Total-f1-score	0.191			

Looking at the **pixels**' classification reports (on the left), a common pattern can be found, in fact, "buildings", "sea" and "street" have higher precision and lower recall, while "forest" and "mountain" have higher recall but lower precision. Regarding this point, all comparisons are made in relative and not absolute terms. "Glacier" has similar if not equal precision and recall values, always under 0.5, meaning that this class is poorly handled by the trained model. We can also notice that *mean pixel* features lead to results that are in between B&W and RGB pixel ones.

Also analyzing the **histogram**'s classification reports (on the right), some comments can be made. In fact, B&W and RGB histogram features don't help classifying "glacier" and "street" images. Precision and recall for the mentioned classes are very low and similar, as both reports and confusion metrices show. The "sea" class isn't well recognizable using B&W histogram features, being the recall near to zero ("sea" is predicted 191 times by the model, but only 23 are the true positive). More generally, we could say the histogram of features don't lead to good enough results.

A schematic table is shown, for clarity reasons:

Feature type High precision - low recall / low precision - high recall / Low precision - low recall

	Building	*Forest*	*Glacier*	*Mountain*	*Sea*	*Street*
B&W pixels	🟩	🟦	🟩	🟦	🟧	🟧
RGB pixels	🟧	🟦	🟩	🟦	🟧	🟧
Mean pixels	🟧	🟦	🟩	🟦	🟧	🟧
B&W histograms	🟦	🟩	🟩	🟩	🟩	🟩
RGB histograms	🟩	🟦	🟩	🟧	🟧	🟩
HSV histograms	🟦	🟩	🟩	🟩	🟩	🟩

🟧 *high precision & low recall:* the model can't detect the class well but is trustable when it does

🟦 *low precision & high recall:* the class is well detected but the model also misclassifies it

🟩 *low precision & low recall:* the class is poorly handled by the model

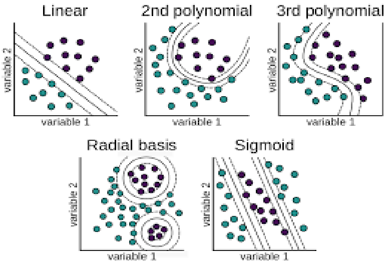
5.3. SUPPORT VECTOR MACHINE TUNING PROCESS & RESULTS

In this chapter, the results of the multiclass Support Vector Machine classifiers are presented. Since the binary nature of SVMs, we used the *One-Versus-Rest* approach[20] to face our multiclass problem.

The hyperparameter on which the SVM model has been tuned, is only the *kernel type*, while the *classification cost* has been kept constant (*C=1*), such as other kernel parameters.

In particular, the kinds of *kernels* that have been tested are:

- RBF (*Radial Basis Function*) kernel $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$
- Polynomial kernel $K(x,z) = (x^T z + 1)^p$
- Sigmoid $K(x,z) = tanh(a \cdot x^T z + b)$
- Linear $K(x,z) = x^T z$



This table resumes the accuracies results for each feature-kernel combination:

ACCURACY	**RBF**	**Polynomial**	**Sigmoid**	**Linear**
a) Greyscale histograms	0.14	0.10	0.16	0.18
b) RGB histograms	0.16	0.16	0.10	0.15
c) HSV histograms	0.17	0.16	0.17	0.17
d) RGB pixels	0.56	0.52	0.33	0.24
e) Greyscale pixels	0.47	0.41	0.27	0.15
f) RGB channels pixels mean	0.49	0.43	0.30	0.14

SVM EVALUATION METRICS HISTOGRAMS

Fig. a.1)

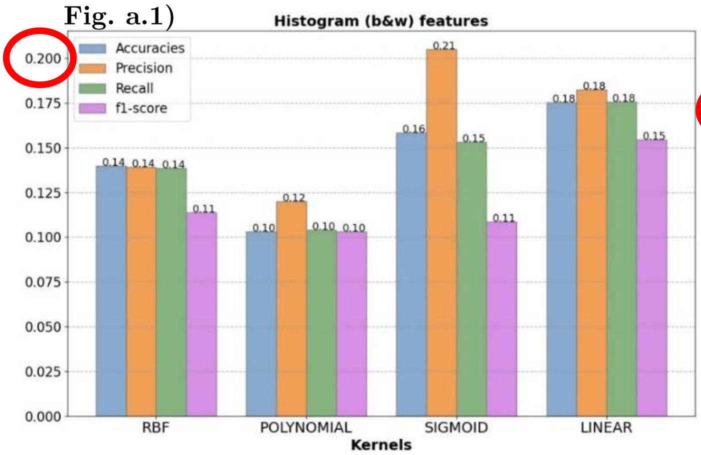


Fig. e.1)

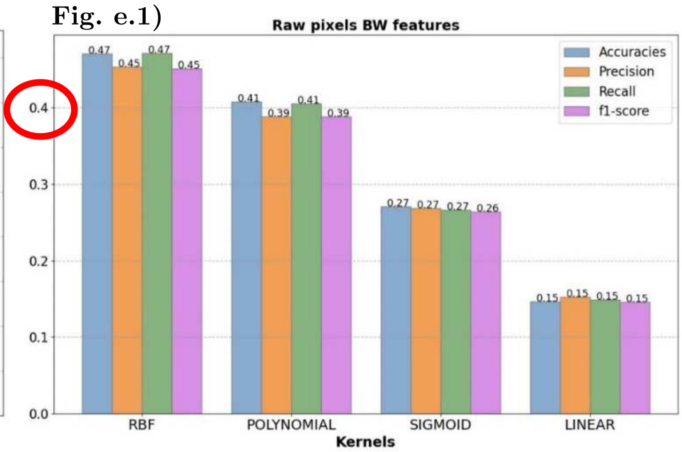


Fig. b.1)

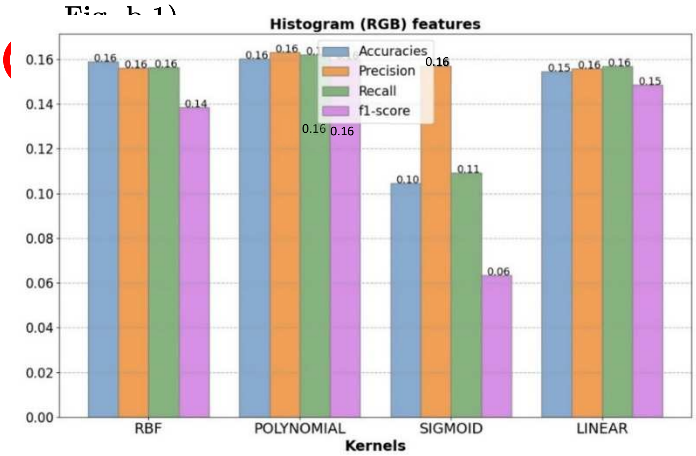


Fig. d.1)

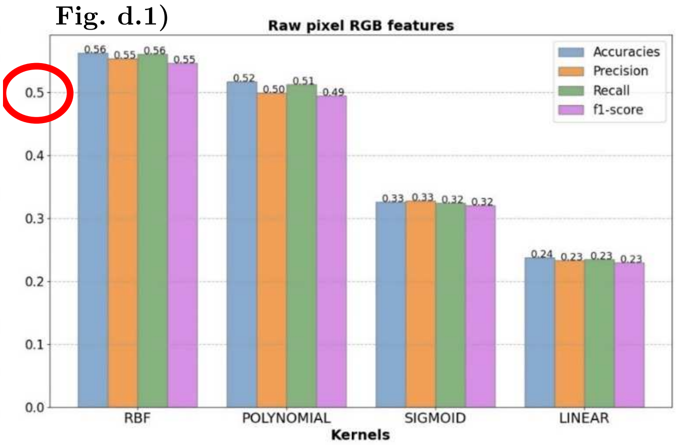


Fig. c.1)

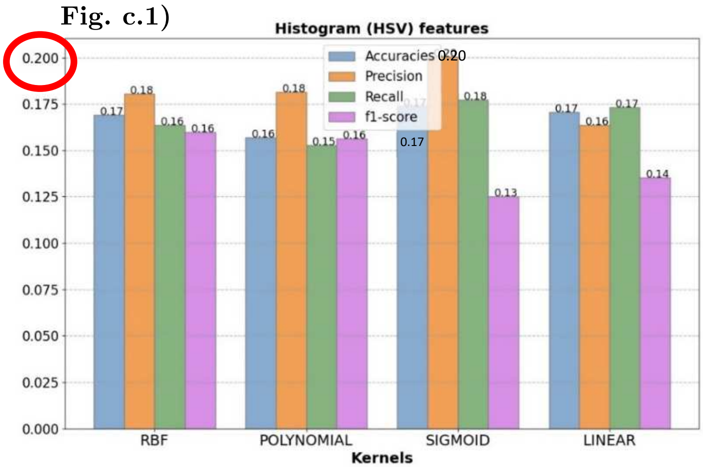
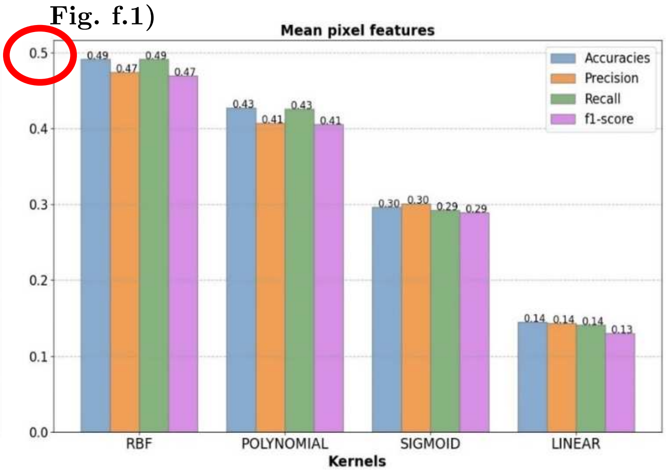


Fig. f.1)



SVM CONFUSION MATRICES (BEST KERNEL FOR EACH FEATURE KIND)

a) Black & White histograms (best kernel: linear)

Fig. a.2)

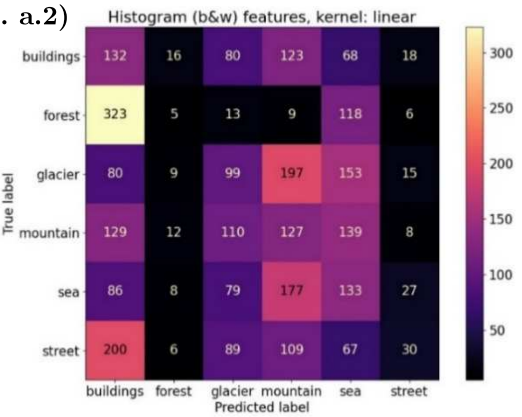
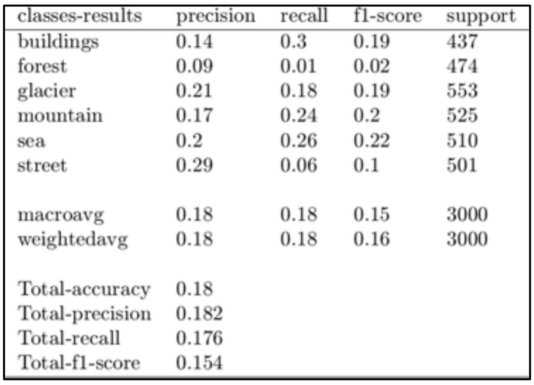


Fig. a.3)



classes-results	precision	recall	f1-score	support
buildings	0.14	0.3	0.19	437
forest	0.09	0.01	0.02	474
glacier	0.21	0.18	0.19	553
mountain	0.17	0.24	0.2	525
sea	0.2	0.26	0.22	510
street	0.29	0.06	0.1	501
macroavg	0.18	0.18	0.15	3000
weightedavg	0.18	0.18	0.16	3000
Total-accuracy	0.18			
Total-precision	0.182			
Total-recall	0.176			
Total-f1-score	0.154			

b) RGB histograms (best kernel: polynomials)

Fig. b.2)

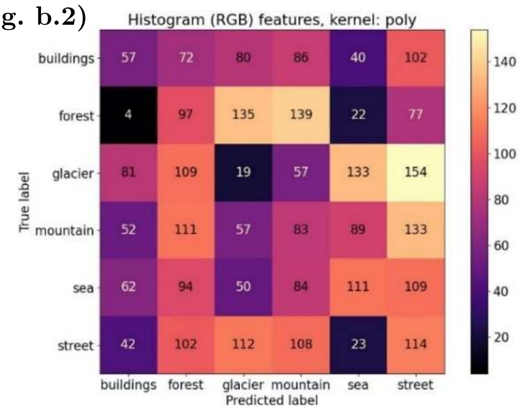
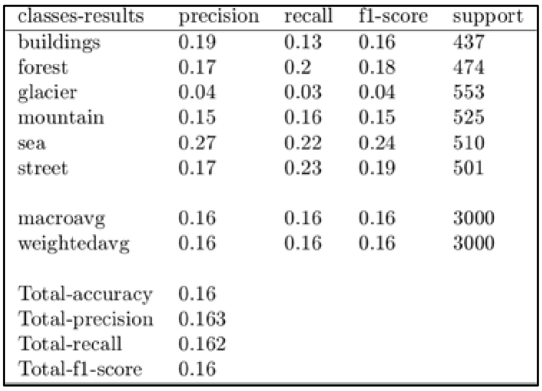


Fig. b.3)



classes-results	precision	recall	f1-score	support
buildings	0.19	0.13	0.16	437
forest	0.17	0.2	0.18	474
glacier	0.04	0.03	0.04	553
mountain	0.15	0.16	0.15	525
sea	0.27	0.22	0.24	510
street	0.17	0.23	0.19	501
macroavg	0.16	0.16	0.16	3000
weightedavg	0.16	0.16	0.16	3000
Total-accuracy	0.16			
Total-precision	0.163			
Total-recall	0.162			
Total-f1-score	0.16			

c) HSV histograms (best kernel: sigmoid)

Fig. c.2)

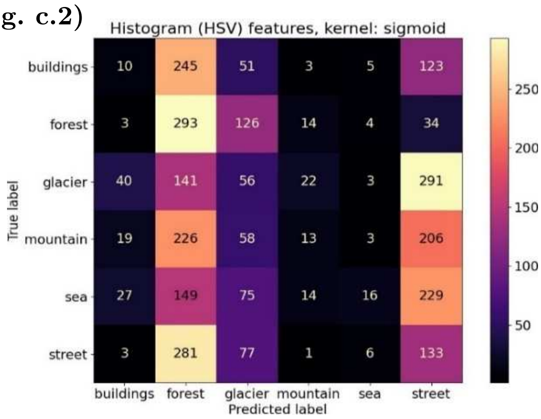
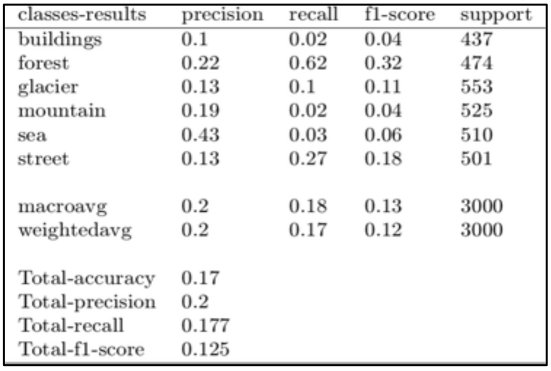


Fig. c.3)



classes-results	precision	recall	f1-score	support
buildings	0.1	0.02	0.04	437
forest	0.22	0.62	0.32	474
glacier	0.13	0.1	0.11	553
mountain	0.19	0.02	0.04	525
sea	0.43	0.03	0.06	510
street	0.13	0.27	0.18	501
macroavg	0.2	0.18	0.13	3000
weightedavg	0.2	0.17	0.12	3000
Total-accuracy	0.17			
Total-precision	0.2			
Total-recall	0.177			
Total-f1-score	0.125			

d) RGB pixels (best kernel: RBF)

Fig. d.2)

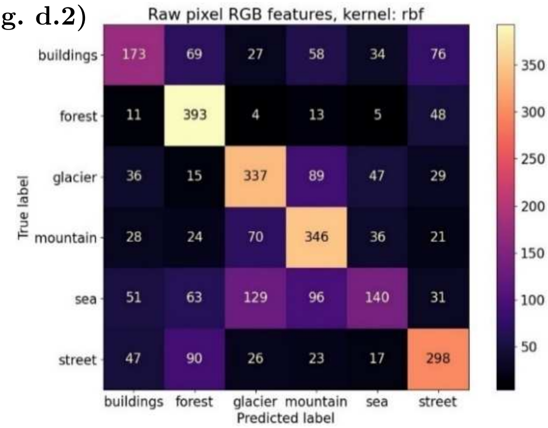
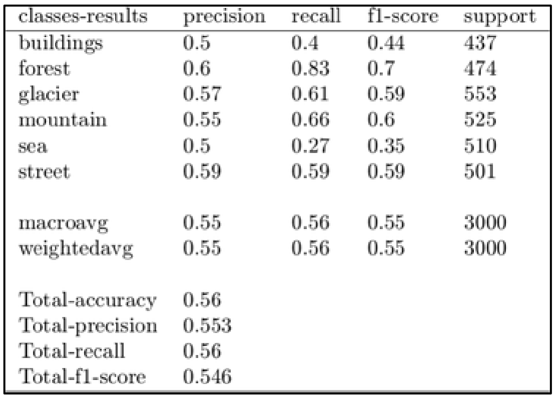


Fig. d.3)



classes-results	precision	recall	f1-score	support
buildings	0.5	0.4	0.44	437
forest	0.6	0.83	0.7	474
glacier	0.57	0.61	0.59	553
mountain	0.55	0.66	0.6	525
sea	0.5	0.27	0.35	510
street	0.59	0.59	0.59	501
macroavg	0.55	0.56	0.55	3000
weightedavg	0.55	0.56	0.55	3000
Total-accuracy	0.56			
Total-precision	0.553			
Total-recall	0.56			
Total-f1-score	0.546			

e) Black & White pixels (best kernel: RBF)

Fig. e.2)

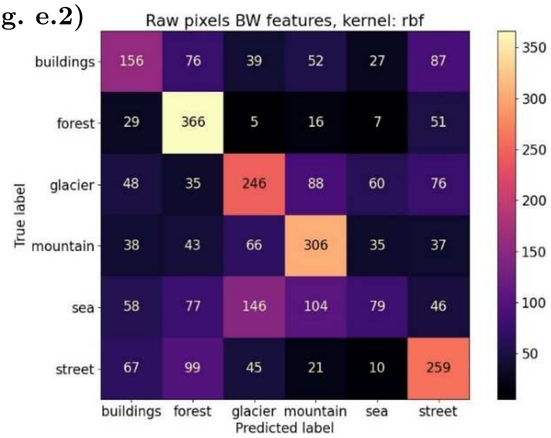
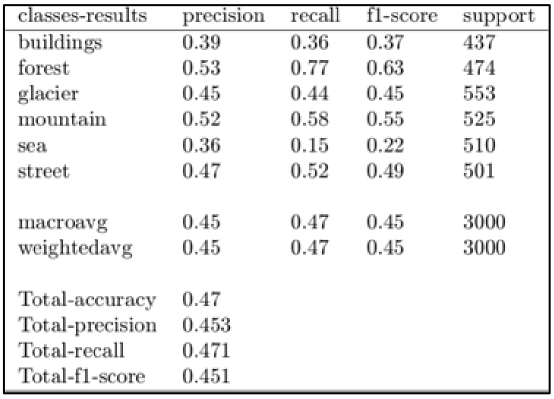


Fig. e.3)



classes-results	precision	recall	f1-score	support
buildings	0.39	0.36	0.37	437
forest	0.53	0.77	0.63	474
glacier	0.45	0.44	0.45	553
mountain	0.52	0.58	0.55	525
sea	0.36	0.15	0.22	510
street	0.47	0.52	0.49	501
macroavg	0.45	0.47	0.45	3000
weightedavg	0.45	0.47	0.45	3000
Total-accuracy	0.47			
Total-precision	0.453			
Total-recall	0.471			
Total-f1-score	0.451			

f) Mean pixels (best kernel: RBF)

Fig. f.2)

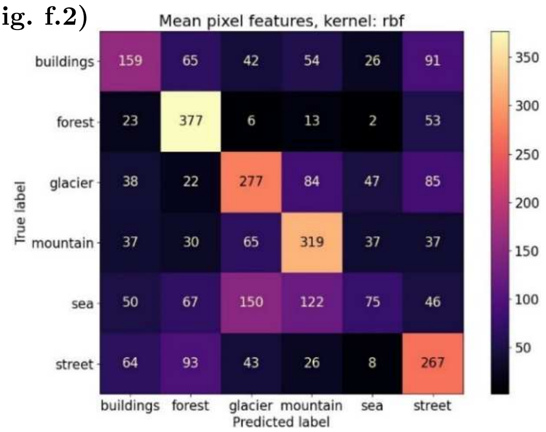
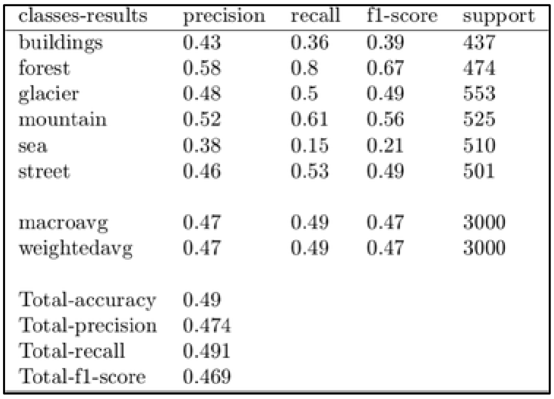


Fig. f.3)

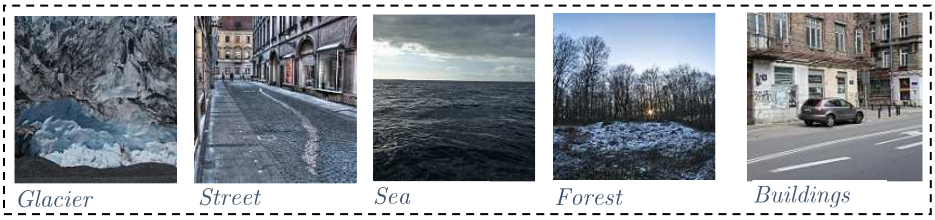


classes-results	precision	recall	f1-score	support
buildings	0.43	0.36	0.39	437
forest	0.58	0.8	0.67	474
glacier	0.48	0.5	0.49	553
mountain	0.52	0.61	0.56	525
sea	0.38	0.15	0.21	510
street	0.46	0.53	0.49	501
macroavg	0.47	0.49	0.47	3000
weightedavg	0.47	0.49	0.47	3000
Total-accuracy	0.49			
Total-precision	0.474			
Total-recall	0.491			
Total-f1-score	0.469			

Comments about the previous results:

From histogram accuracies, the first thing to notice are the low scores reached by all kind of histograms, with respect to pixels features. Indeed, histograms accuracies are all about 20% versus pixels' accuracies that goes from 47% for Black & White pixels to 56% for RGB ones. What is interesting is that RGB and HSV histograms, with respect to Black & White, store also the "quantity" of color of the images besides intensities. For this reason we'd have expected better results than grayscale histograms, but this is not the case.

A possible explanation for histograms poor results, can be deduced by looking at the pictures below: *focusing only on the color of the images, we can notice that some classes can often have a similar "average" color*.



Being the respective feature vectors near in the feature space, they can't be easily separated with a hypersurface, also with the ***kernel trick*** implemented by SVM method. For instance, these images show how many classes can share bluish/greyish shades. From that, we can deduce that *features coding only the intensity/color quantity of our images, are not enough to reach good classification accuracies in our dataset*, despite being easy to obtain and understand.

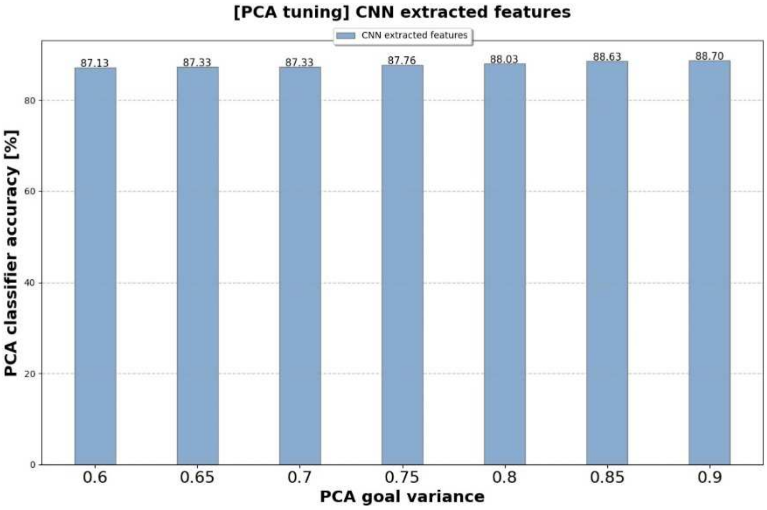
Pixels, on the other hand, also store the "*positional*" information of intensities/colors inside the pictures, and they perform quite better. For instance, *RGB pixels reach highest accuracy*, with an RBF kernel. Looking at the relative confusion matrix (Fig d.2), we have a further confirm of the high accuracy: all classes improved with respect to other feature cases, as the well-defined principal diagonal shows. However, even in this case, some classes are still misclassified: the main mistakes, are made with "seas", confused with "glacier" and "mountains". Despite pixels are more informative than histograms, the intrinsic ambiguity between images of those classes can lead to poor results. For instance: "mountain glaciers" and "mountains" can be very similar, as well as "sea glaciers" can look like "seas". Another consistent mistake is between "buildings" and "streets", but again, such images can be often related, like the following images show. Therefore, these kinds of mistakes are *acceptable* for RGB pixels, since images belonging to these classes can appear very similar and also a human might be in doubt.

A drawback of RGB pixels, is the ***high feature vector dimensionality***, depending on the images' resolution. In this sense, PCA helps a lot to reduce the feature space size, but alternatives such as Black & White pixel or mean pixel features are valid for our dataset. They are capable of reaching similar accuracies to RGB pixels, having lower features' space dimension.

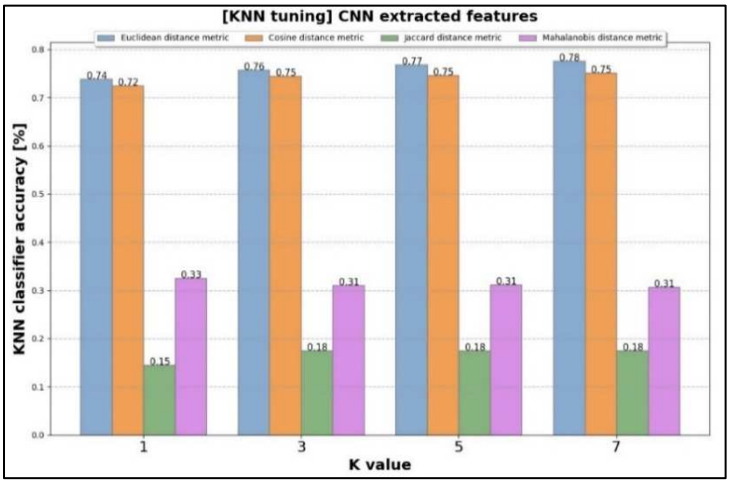
5.4. CNN AS A FEATURE EXTRACTOR

To improve the overall performances of machine learning algorithm, a possibility is to exploit "*better features*" extracted using a Convolutional Neural Network. In this case, a ResNet18 pretrained on ImageNet was used to perform the ***feature extraction***. A *512*-length array for each image is extracted from the last convolutional layer and it is provided as input to KNN and SVM. As it will be clearly shown, performances are much better than using raw pixels or histogram features. After loading the feature vectors, their dimensionality was reduced using *PCA* (choosing the best variance, according to the tuning process) and the best hyperparameters of the KNN and SVM were chosen, according to the best achievable accuracy.

PCA tuning process for dimensionality reduction of the feature vectors:

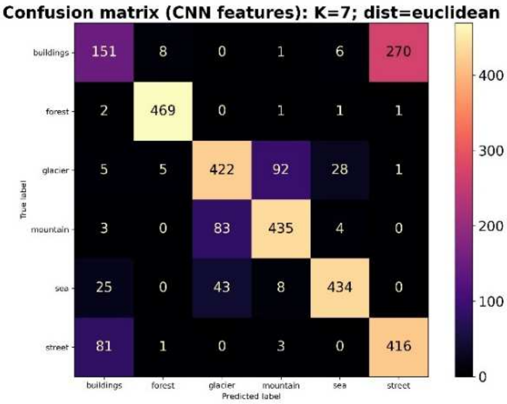


5.4.A. CNN extracted features to the KNN algorithm results

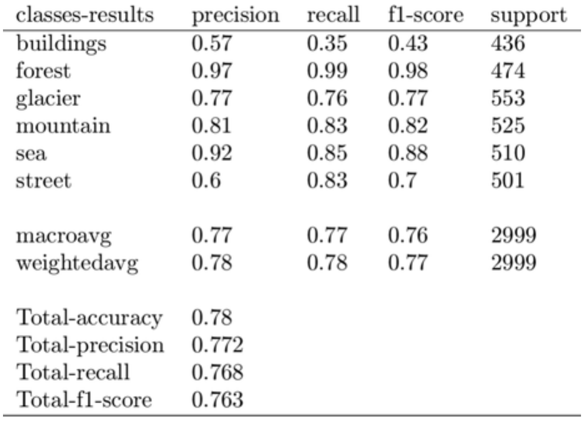


What can be easily noticed from the plot, *Jaccard* and *Mahalanobis* distances are not suitable metrics for this kind of features. Also, increasing the K parameter, accuracy doesn't significantly improve, as opposed to what we've observed with the other "elementary" features. Probably, this behavior is due to the fact that *CNN extracted features are more reliable*, so including feature vectors that are farther doesn't dramatically increase the overall accuracy.

Looking at the confusion matrix and the classification report, there is a noticeable improvement due to the better quality of the features that have been used. The "glacier" class, for example, has always been difficult to recognize but in this case it isn't. The "buildings" and "street" classes have a respectively *high precision–low recall* and *low precision–high recall* behavior. Instead, all the other classes are well recognizable by the model.

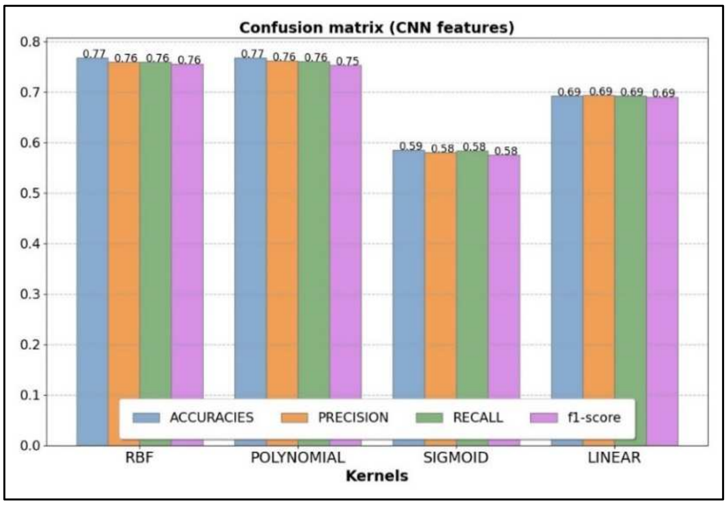

Confusion matrix (CNN features): K=7; dist=euclidean

True label \ Predicted label	buildings	forest	glacier	mountain	sea	street
buildings	151	8	0	1	6	270
forest	2	469	0	1	1	1
glacier	5	5	422	92	28	1
mountain	3	0	83	435	4	0
sea	25	0	43	8	434	0
street	81	1	0	3	0	416



classes-results	precision	recall	f1-score	support
buildings	0.57	0.35	0.43	436
forest	0.97	0.99	0.98	474
glacier	0.77	0.76	0.77	553
mountain	0.81	0.83	0.82	525
sea	0.92	0.85	0.88	510
street	0.6	0.83	0.7	501
macroavg	0.77	0.77	0.76	2999
weightedavg	0.78	0.78	0.77	2999
Total-accuracy	0.78			
Total-precision	0.772			
Total-recall	0.768			
Total-f1-score	0.763			

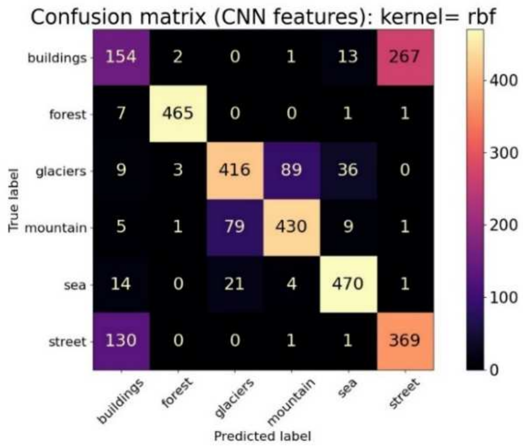
5.4.B. SVM classification using feature vector extracted from CNN



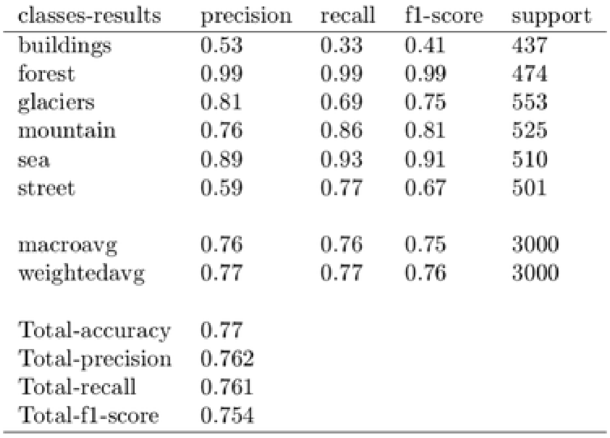
From the evaluation metrics histograms, the kernels that lead to the best accuracies are RBF and Polynomial. Since RBF reaches a slightly higher accuracy, as usual, its confusion matrix and classification report are shown thereafter, to assess per-class performances.

It can be clearly seen that the results are far better than the previous feature kinds, that at most reached 56% accuracy (RGB pixel features). This is due to the ability of CNN to extract more significative features, increasing the accuracies of Machine Learning methods, such as KNN in previous section.

It's interesting to make a comparison between this case and the SVM RGB pixel one. In particular, the best improvement regards the class "sea" that is recognized with really higher precision (89%) and recall (93%), with respect to RGB pixels (50% precision and 27% recall). On the other hand, in common with that kind of features, *the easiest class to identify and discern is "forest"*.



	buildings	forest	glaciers	mountain	sea	street
buildings	154	2	0	1	13	267
forest	7	465	0	0	1	1
glaciers	9	3	416	89	36	0
mountain	5	1	79	430	9	1
sea	14	0	21	4	470	1
street	130	0	0	1	1	369



classes-results	precision	recall	f1-score	support
buildings	0.53	0.33	0.41	437
forest	0.99	0.99	0.99	474
glaciers	0.81	0.69	0.75	553
mountain	0.76	0.86	0.81	525
sea	0.89	0.93	0.91	510
street	0.59	0.77	0.67	501
macroavg	0.76	0.76	0.75	3000
weightedavg	0.77	0.77	0.76	3000
Total-accuracy	0.77			
Total-precision	0.762			
Total-recall	0.761			
Total-f1-score	0.754			

The biggest problem of this method, in common with KNN, is that several "buildings" are misclassified with "street", implying in this case the by far smallest recall value. Two interesting related fact: first, RGB pixels commit a lot less mistakes between these two classes, even if they are a "simpler" kind of feature than CNN's extracted ones. Second, the viceversa is less problematic, i.e. "street" images are not so easily misclassified with "buildings".

Other remarkable mistakes, are as usual between "glaciers" and "mountain", even if less evident with these kind of features. To notice also that for both these CNN features and RGB pixels, the best kernel is RBF.

5.5. CONVOLUTIONAL NEURAL NETWORK RESULTS

To improve image classification performances, deep learning architectures, such as ***Convolutional Neural Networks***, can be exploited. In this work, we decided to use two of the most common architectures: VGG16 and ResNet18.

Being the required training/testing time pretty long, if compared with the previously discussed machine learning algorithms, both the train and test sets were *reduced* during the training process. This reduction has not affected the CNN model's accuracy and general performances overcame KNN's and SVM's ones. Despite we used a reduced version of the original dataset during training, *the final confusion matrices and evaluation metrics, were computed testing on the whole testset*, to be directly comparable with the previous results.

So, to sum up:

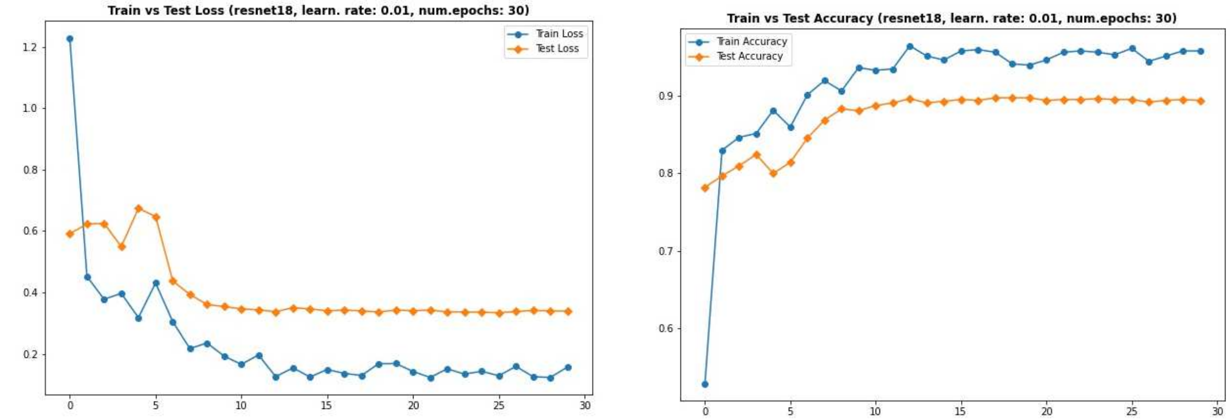
- *Model training phase of VGG16/ResNet18:* 600 train images (100 for each class)
- *Model testing phase after every training epoch:* 900 test images (150 for each class)
- *Final model testing to get confusion matrices, etc.:* 3000 test images (all the available)

For both the VGG and ResNet, the study was conducted starting from a pretrained model on the ImageNet dataset. The only hyperparameters that were tuned are the ***learning rate*** and the ***number of epochs***. Therefore, momentum, learning rate decay steps, optimizer, batch size were kept fixed to reasonable values, according to the available *Pytorch* tutorials[21]. In particular, 3 values of learning rate were chosen and the model training was performed until the 30th epoch. Based on the accuracy achieved testing every intermediate model on the 900 test images, the best model was saved. This way, we can analyze the model's behavior with respect to different learning rates and number of epochs, but we save only the best performing one.

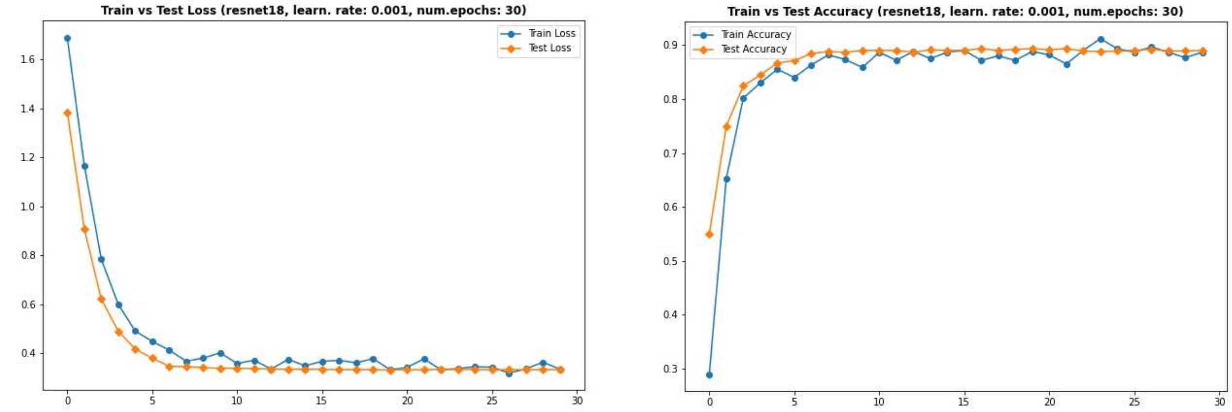
5.5.A. RESNET18 RESULTS

	First experiment	Second experiment	Third experiment
Learning rate	0.01	0.001	0.0001
Total number of epochs	30	30	30
Best model's number of epochs	12	16	14
Best model's accuracy	89%	89%	64%

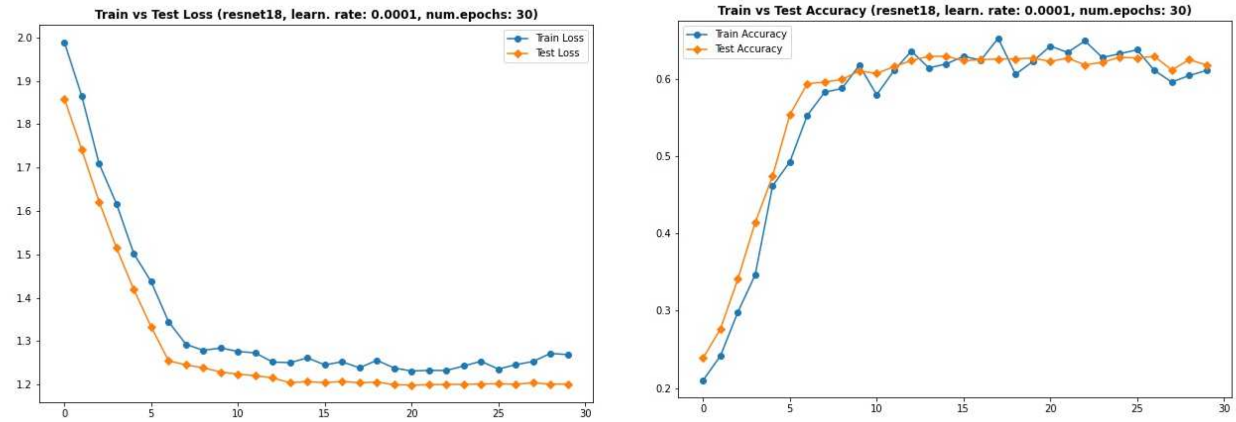
1) Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):



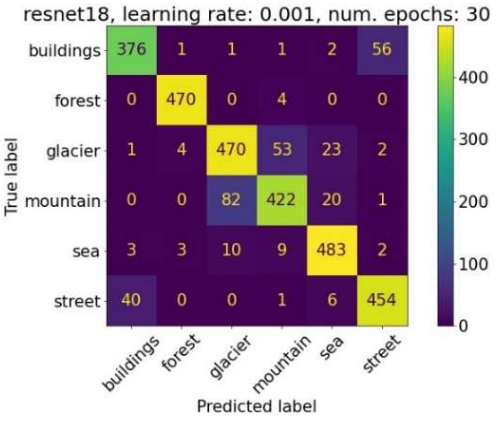
2) Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):



3) Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):



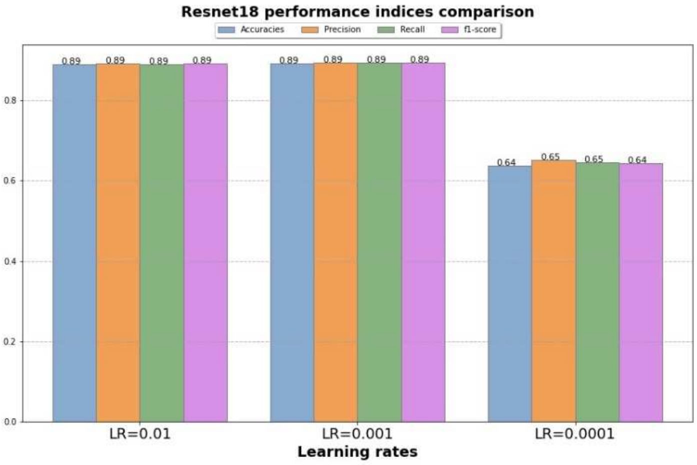
Confusion matrix and classification report for the ResNet18 best model (learning rate=0.001):


resnet18, learning rate: 0.001, num. epochs: 30

	buildings	forest	glacier	mountain	sea	street
buildings	376	1	1	1	2	56
forest	0	470	0	4	0	0
glacier	1	4	470	53	23	2
mountain	0	0	82	422	20	1
sea	3	3	10	9	483	2
street	40	0	0	1	6	454

classes-results	precision	recall	f1-score	support
buildings	0.9	0.86	0.88	437
forest	0.98	0.99	0.99	474
glacier	0.83	0.85	0.84	553
mountain	0.86	0.8	0.83	525
sea	0.9	0.95	0.93	510
street	0.88	0.91	0.89	501
macroavg	0.89	0.89	0.89	3000
weightedavg	0.89	0.89	0.89	3000
Total-accuracy	0.89			
Total-precision	0.893			
Total-recall	0.893			
Total-f1-score	0.892			

Comparison *accuracy*, *precision*, *recall* and *f1-score* of the 3 “best” ResNet18 trained models:

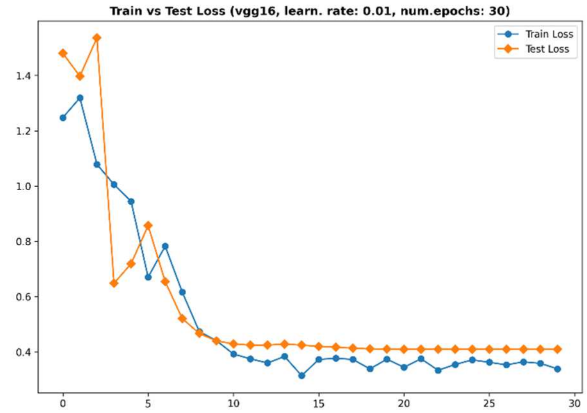


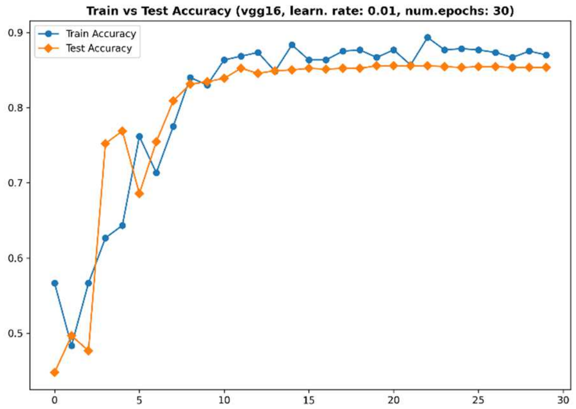
Looking at the metrics comparison histograms (on the right), the **2nd** scenario appears to be the best one, having more stable *accuracy* and *loss* progresses, high accuracy values and the best confusion matrix. Analyzing the confusion matrices, a neural network (*ResNet18*), that is far better performing than SVM and KNN, still confuses mountains with glaciers or streets with buildings, as a human being would. The achieved results are in line with what expected from theory and with other studies on our Nature Dataset.

5.5.B. VGG16 RESULTS

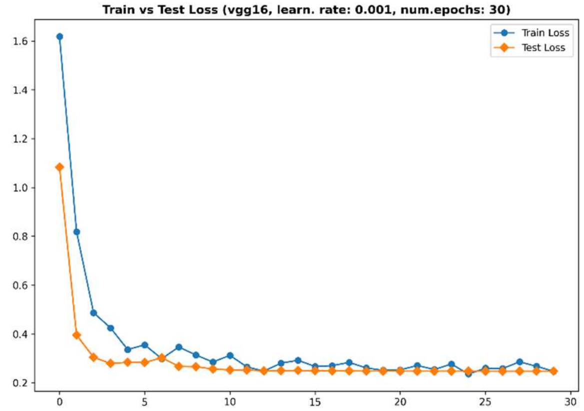
	First experiment	Second experiment	Third experiment
Learning rate	0.01	0.001	0.0001
Total number of epochs	30	30	30
Best model’s number of epochs	19	14	18
Best model’s accuracy	86%	90%	84%

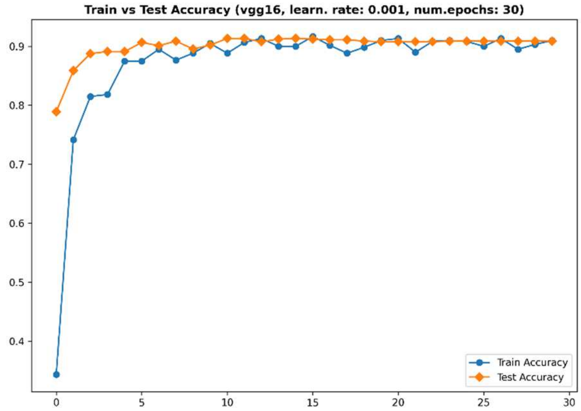
1) 1st exp. (learning rate 0.01): Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):



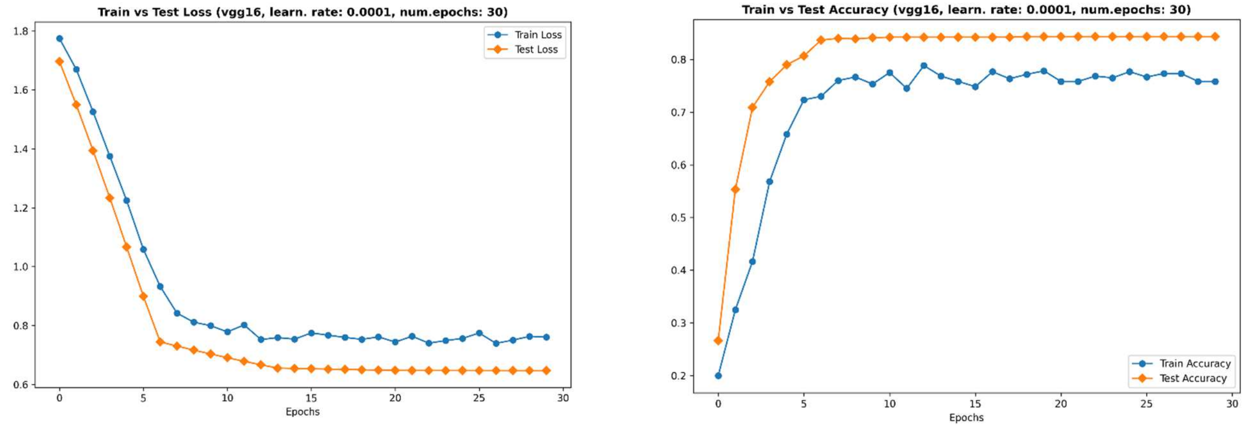


2) 2nd exp. (learning rate 0.001), Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):



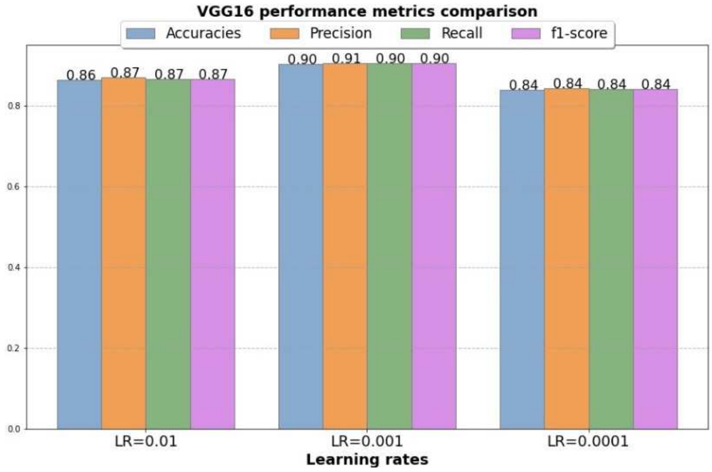


3) 3rd exp. (learning rate 0.0001): Train vs test ***Loss*** progress (left) and Train vs test ***Accuracy*** progress (right):

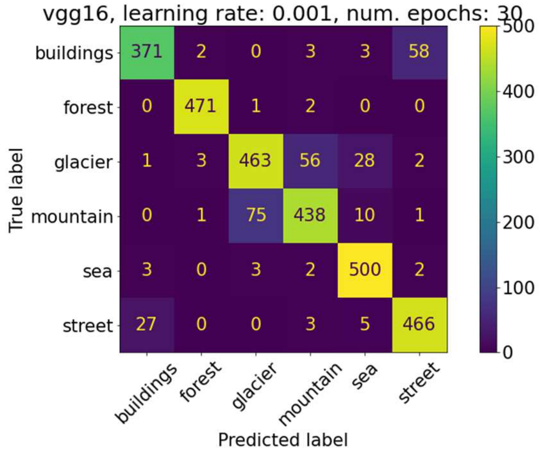


Comparison *accuracy*, *precision*, *recall* and *f1-score* of the 3 "best" VGG16 trained models:

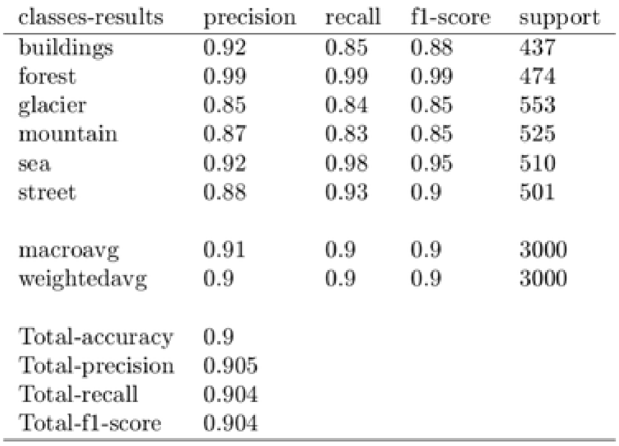
By looking at the histogram on the right, with evaluation metrics comparison, we see that in VGG16 case, ***the highest accuracy model*** is found during ***2nd experiment,*** with a starting learning rate of 0.001. With its **90%** of accuracy, this *ConvNet* is *the absolute best classifier among all our models*. This accuracy is substantially higher than machine learning methods (around 50%) and slightly better than the 89% of the ResNet18.



Confusion matrix and classification report for the VGG16 best model (learning rate=0.001):



	buildings	forest	glacier	mountain	sea	street
buildings	371	2	0	3	3	58
forest	0	471	1	2	0	0
glacier	1	3	463	56	28	2
mountain	0	1	75	438	10	1
sea	3	0	3	2	500	2
street	27	0	0	3	5	466



classes-results	precision	recall	f1-score	support
buildings	0.92	0.85	0.88	437
forest	0.99	0.99	0.99	474
glacier	0.85	0.84	0.85	553
mountain	0.87	0.83	0.85	525
sea	0.92	0.98	0.95	510
street	0.88	0.93	0.9	501
macroavg	0.91	0.9	0.9	3000
weightedavg	0.9	0.9	0.9	3000
Total-accuracy	0.9			
Total-precision	0.905			
Total-recall	0.904			
Total-f1-score	0.904			

Thanks to the confusion matrices and the classification report, we can have a further confirm of our previous considerations and results, about the inner ambiguity of some classes. From these data, we have that the most distinguishable class is by far the "forest" one, followed by "seas". The challenging part instead, is to distinguish "buildings" from "streets", and "glaciers" from "mountains", which are for their nature, intrinsically ambiguous and linked classes, like already pointed out in ResNet18.

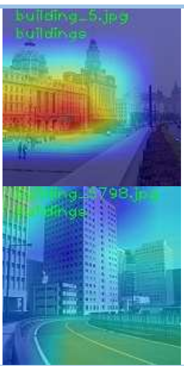
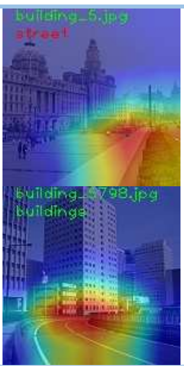
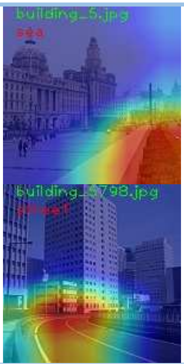
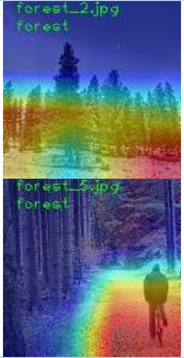
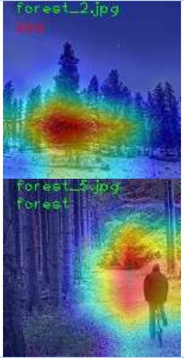
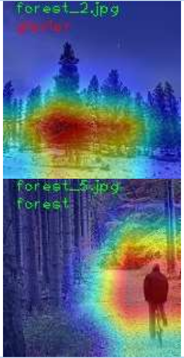
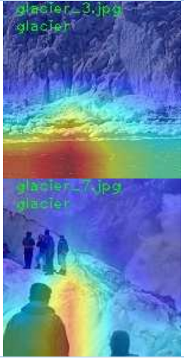
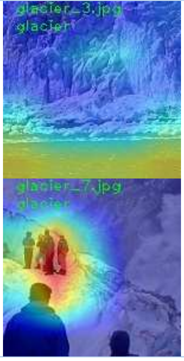
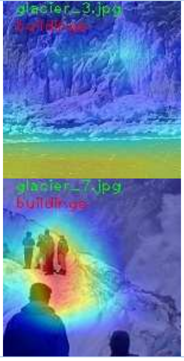
However, please note the remarkable less mistakes made by both *trained* CNNs, between classes "buildings" and "street" with respect to KNN and SVM with features extracted by the *pre-trained* ResNet18, as a fixed feature extractor. In fact, from the above VGG16 confusion matrix, we can notice that the number of misclassified "buildings" with "streets", is only 58, against the 270 of KNN and SVM with CNN features!

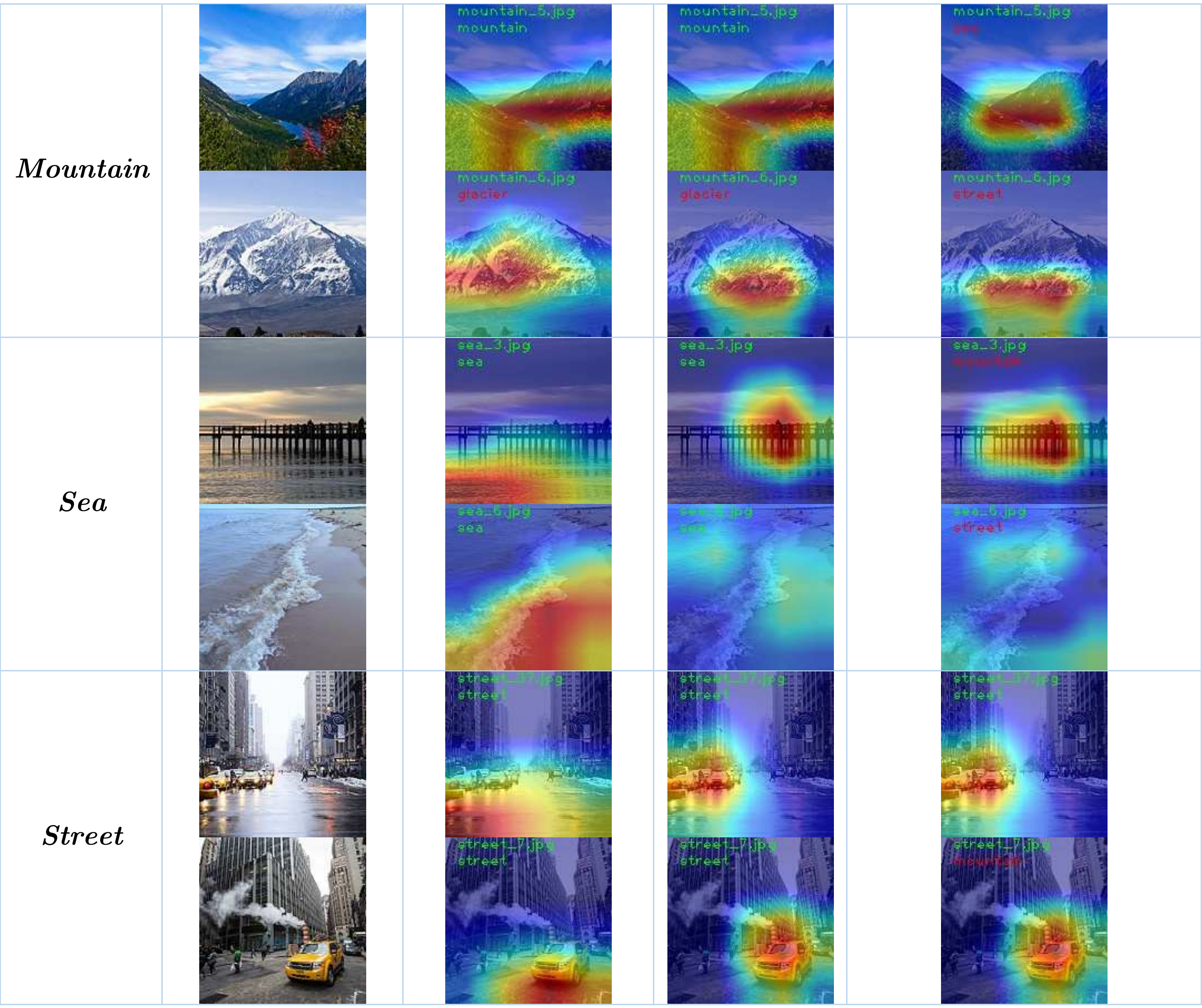
5.6. Grad-CAM: CNNs FEATURE INTRPRETATION

By using deep models, we sacrifice interpretability to achieve greater performance through greater abstraction. ***Grad-CAM*** (***Grad****ient-weighted* ***C****lass* ***A****ctivation* ***M****apping*) is a technique for producing *'visual explanations'* for decisions from a large class of Convolutional Neural Network (CNN)-based models, making them more transparent, without requiring architectural changes or re-training. It uses the *gradients* of any target concept (classes in a classification network) flowing into the *final convolutional layer* to produce a coarse localization map highlighting the ***important regions in the image*** for predicting the concept. As a result, important regions of the image which correspond to any decision of interest are visualized in high-resolution detail even if the image contains evidence for multiple possible concepts.[22]

Therefore, we exploit Grad-CAM technique, to try to understand on which kinds of feature the trained CNNs focus, to reach higher accuracy values than all other models. We also analyze the features extracted by the ResNet18, pretrained on ImageNet, that we use as fixed feature extractor for KNN and SVM.

GRADCAM RESULTS on some sample images for each class

CLASS	***Original image***	***VGG16***	***ResNet18***	***ResNet18_ imageNet***
Buildings		 building_5.jpg buildings building_5798.jpg buildings		
Forest				
Glacier				



The right column collects predicted images of pretrained ResNet18 on ImageNet. This last column usually highlights <u>different regions</u> from the trained (on NatureDataset) VGG16 and ResNet18. Since they are trained on our dataset, as expected, they focus on more discriminative areas, making less mistakes. The main differences however, are visible in VGG column. It seems like <u>*VGG16*</u> tends to take into account <u>*wider areas*</u>, whereas ResNets point on different smaller and specific regions. For instance in class "street" ResNet focuses on cars while VGG on roads. Similar fact occurs with "sea" where VGG actually focuses on "water" while ResNet highlights the wooden dock.

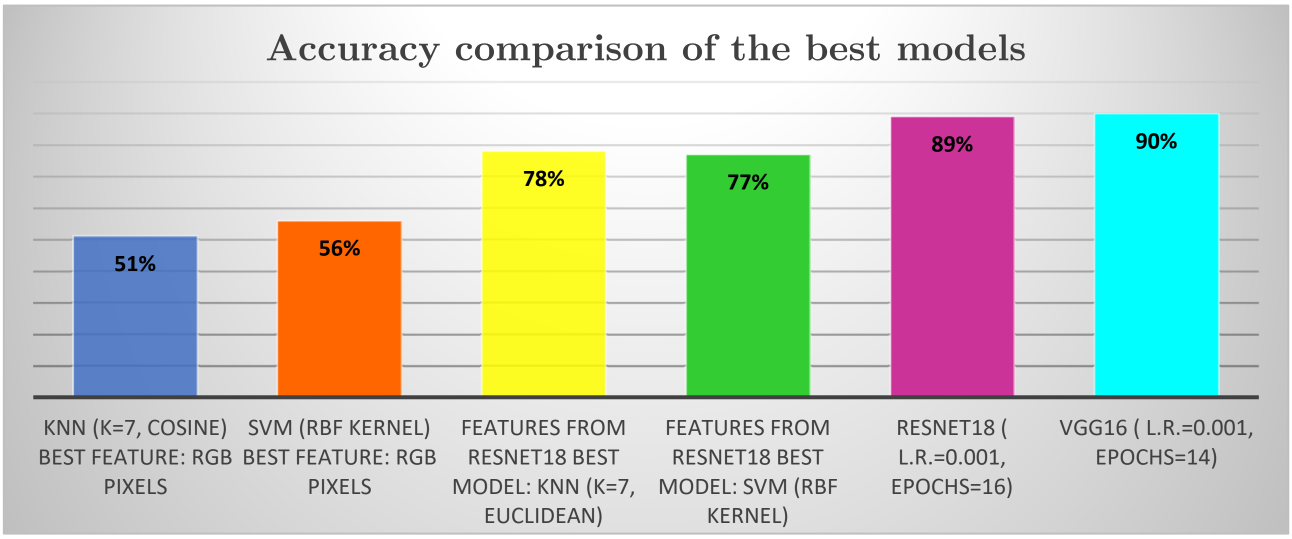
We can deduce two things: among our deep learning models, <u>*VGG16 seems to abstract and look for more general concepts*</u>, specific for classes: to recognize "sea" it searches for water; to find "streets" it looks for roads and so on. On the other hand, ResNet often tries to exploit characteristic objects and structures of classes' panoramas. Cars are related to "streets", "forests" to single trees, dock can remind a "sea". However, since our dataset is strongly related to panormas, VGG seems to have reached a better understanding of the real features that discern the scenes. For this reason, we can suppose that between our two CNNs, ***VGG16 is the most robust***. This could be important, because looking at the accuracies, the VGG16 and ResNet18 trained on our dataset, achieved in practice the same results. In fact, this is one of the benefits of using Grad-CAM, i.e. helping users successfully discern a 'stronger' deep network from a 'weaker' one even when both make identical predictions.[22]

5.7. FINAL RESULTS: BEST MODEL

In this project we studied several supervised learning models to perform image classification. What we found out is that ***for our dataset, deep learning methods such as Convolutional Neural Networks achieve***

better accuracies than machine learning methods. A sort of *"hybrid" pipeline* was also exploited, extracting feature vectors using a CNN and applying SVM and KNN machine learning algorithms for the final classification. In this case performances are quite better than using other kinds of "elementary" features (pixels and histograms). CNNs are the best, as far as accuracy goes, but the "hybrid" pipeline allows to reduce a lot training time.

Next figure, shows graphically combinations method-feature-parameter with best accuracies:



Consequently, as previous histogram shows, our study has produced the following results:
about *machine learning* methods with "manually extracted" features:

- **KNN**: the best accuracy is achieved with ***RGB pixels*** feature vectors, reduced with PCA algorithm, and with value ***K=7*** and ***Cosine*** as ***distance*** metric;
- **SVM**: the best accuracy is produced by an ***RBF*** (*Radial Basis Function*) ***kernel***, with ***RGB pixels*** feature vector again, reduced through PCA.

For *KNN and SVM classifiers, fed with features extracted from* pretrained *ResNet18* on ImageNet:

- **KNN**: ***the best accuracy*** reached with this kind of features has been ***78%***, with ***K=7*** and an ***Euclidean distance*** metric;
- **SVM**: feature extracted from ***ResNet18*** has produced a ***best accuracy of 77%*** with ***RBF kernel***.

For *deep learning* methods:

- **ResNet18**: the pre-trained ConvNet on ImageNet, has been trained for 30 epochs, an initial ***learning rate*** of ***0.001***, a momentum of 0.9 and a learning rate decay that reduced by 10 times the learning rate, every 7 epochs. The ***best model***, produced an ***accuracy*** of ***89%***, and has been ***obtained after 16 epochs***;
- **VGG16**: all the hyperparameters are the same of the ResNet18 and also in this ***case the best accuracy***, ***90%***, has been achieved again with a ***learning rate*** of ***0.001***. The best accuracy was obtained with the model trained until ***epoch 14***;

Therefore, among all models the ***VGG16 Neural Network has produced the absolute highest accuracy of 90%***, and so this is the ***best model*** we found to classify Nature Dataset's scenes.
Even though, VGG16 took longer time to be trained than ResNet and the two CNNs produce approximately the same results, VGG is the most reliable. Indeed, as we showed with Grad-CAM, it seems that ResNet18 uses features in image regions that likely are not so discriminant as VGG's ones, and so, there is the possibility that it doesn't generalize enough.

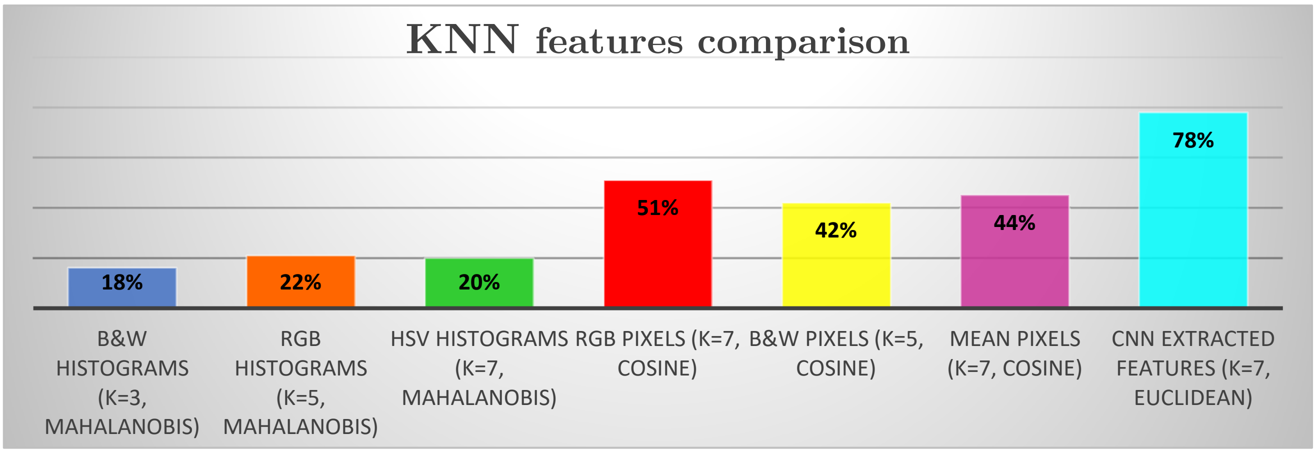
5.8. *FINAL RESULTS: FEATURE KINDS COMPARISON*

Another objective of our project was to analyze *which kinds of features are the best* for each machine learning method and *how these features perform in classifying each class.*

*The following tables resume **KNN BEST ACCURACIES** for all feature kinds:*

	FEATURES	Best K	Best Distance metrics	ACCURACY
KNN	B&W histograms	1	Jaccard	18%
	RGB histograms	7	Mahalanobis	22%
	HSV histograms	5	Mahalanobis	20%
	RGB pixels	7	Cosine	51%
	B&W pixels	7	Cosine	42%
	Mean pixels	7	Cosine	44%
	CNN extracted features	7	Euclidean	78%

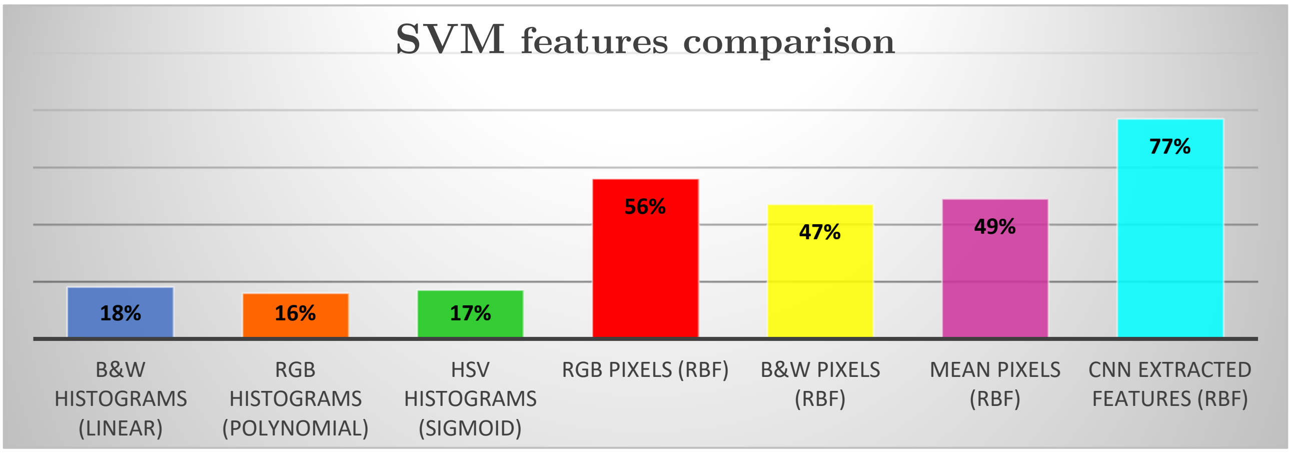
The following histogram depicts the same information but in a graphical way:



*The following tables resume **SVM BEST ACCURACIES** for all feature kinds:*

	FEATURES	Best KERNEL	ACCURACY
SVM	B&W histograms	linear	18%
	RGB histograms	polynomial	16%
	HSV histograms	sigmoid	17%
	RGB pixels	RBF	56%
	B&W pixels	RBF	47%
	Mean pixels	RBF	49%
	CNN extracted features	RBF	77%

The following histogram depicts the same information but in a graphical way:



For both KNN and SVM, the classification reaches the ***best accuracy*** with ***features extracted from ResNet,*** namely 78% with KNN and 77% SVM. To notice that this kind of feature *has not a direct interpretation.* However thanks to Grad-CAM we were able to have a qualitative idea of what they represents. In particular, we have seen that CNNs do not focus on the whole image, but only on discriminant part of them[22]. The difference with (RGB) pixels we used is substantial: in fact this last case, all pixels in the images have the same importance. This leads to consider also unimportant or even misleading image parts, inevitably worsening the results.

Let's focus now on all other features, that have a direct meaning and that we extracted manually. Among them, ***RGB pixels produces the best accuracies for both KNN and SVM.*** As previously pointed out, they collect both "*colors quantities"* and their *"positions".* However, it is significative to emphasize that this two information, have not the same importance for Nature Dataset classification: indeed greyscale (B&W) pixels, bringing only the *"positional"* information, reach considerably higher results than histogram features.
This let us conclude that, ***for Nature Dataset, pixel features information are definitely more discriminant than features that are only related to color quantities.***
This fact is strongly related to the answer of our last objective: *which is the best feature for every class.*

5.9. FINAL RESULTS: PER-CLASS PERFORMANCES

For both KNN and SVM ***the best results per class, are always achieved by RGB pixels***, **except for CNN features** that we don't take into account in this section, because of their lack of interpretability. RGB pixels allow to have a good precision and recall for all classes, proving that the highest overall accuracies with respect to other feature kinds, derive directly from better classification performances *of all classes.*

The following table shows the highest values of precision and recall for each class, disregarding results obtained with features extracted from the ResNet18:

CLASS	**highest PRECISION …**	**…obtained with features**	**…obtained with model**	**highest RECALL …**	**…obtained with features**	**…obtained with model**
buildings	50%	RGB pixels	SVM	40%	RGB pixels	SVM
forest	60%	RGB pixels	SVM/KNN	83%	RGB pixels	SVM
glacier	57%	RGB pixels	SVM	61%	RGB pixels	SVM
mountain	55%	RGB pixels	SVM	72%	RGB pixels	KNN
sea	50%	RGB pixels	SVM	27%	RGB pixels	SVM
street	63%	RGB pixels	KNN	59%	RGB pixels	SVM

From the previous table we can see that ***class "forest" is in average the most easily recognizable*** (83% recall). To notice that, usually, this happens also with histograms.
A possible interpretation, is that *"forests"* are often *linked to the color "green"* which is quite *specific for this class.* On the other hand, other classes ("buildings", "glaciers", "mountains", "seas" and "streets") usually have broader range of possible colors, which they may share with each other. For instance, "grey" color, can be easily associated to road in street, buildings' walls or rocks in mountains and glaciers. This color characteristic can help the classification, but it is not the only reason. ***Even in greyscale pixels, "forests" are usually easier to recognize than other classes***, meaning that even ***"shapes" play an important role.*** Let's think about the similarity between glaciers and mountain, or between building and street. They can reasonably look similar, instead "forests" contain characteristic details, such as trees, which are not shared with any other class.
Another interesting result from previous table, is that ***class "sea" is the hardest to recognize*** (27% recall maximum). This very small value comes mainly from the *misclassifications of "seas" with "glaciers"*, like it is evident from the confusion matrix of SVM with RGB pixels, that anyway is the best case. Again, the intuitive reason is the intrinsic apparent ambiguity between these classes, as we showed previously, leading to a deterioration in performances.

In conclusion, looking at the ***per-class precisions***, we can see that ***all classes have comparable values***. It is interesting to notice the relatively high precision of "seas" with respect to its recall, but it has the smallest per-class precision. From that, we can conclude that ***with RGB pixels, "sea" is the most problematic class and "forest" is the easiest to discern***.

However, as already said, feature extracted through CNNs perform always better than all other kinds of features, even in per-class performances. Likewise other feature types, for both KNN and SVM methods, ***"forest" is by far the easiest class to recognize*** (97% precision, 99% recall for KNN and 99% precision and recall for SVM!). What is different instead is the worst case: ***CNN features achieve the worst precision and recall results with "buildings"***. This comes from the strong tendency to misclassify images of this class with "streets", causing for these CNN features the lowest per-class precision (55%) and recall (35%) for both KNN and SVM. This fact is attributable to the tendency of pretrained ResNet18 on ImageNet, to focus mainly on road instead of buildings inside the images (see Grad-CAM "buildings" images), that leads to these kinds of misclassifications. On the other hand, *performances on class "sea" are extremely higher than RGB pixels' ones.* CNN features reach approximately a 90% precision and recall for both KNN and SVM in class "sea", with an impressive improvement of 40% precision and 60% recall!
To notice finally that, ResNet18 and VGG16 show very similar per-class performances to these features, except for class "buildings". ***CNN ability of discerning "buildings" from "streets", is the main reason for the consistent higher accuracy of Neural Networks, with respect to KNN and SVM accuracies reached with feature extracted from pre-trained ResNet18.***

6. CONCLUSIONS

In our project we addressed a *supervised learning image classification* problem, with a dataset of different panoramas. The dataset consists of six classes: "buildings", "forest", "glacier", "mountain", "sea" and "street", that for their intrinsic nature, can be often related and misleading. We used different methods: KNN and SVM testing different hyperparameters values to assess which combinations achieve higher performances, focusing on accuracy and per-class precision and recall. These machine learning algorithms need features extracted from images and we tried several feature kinds: from the simple histograms (Black & White, RGB, HSV), to more complex ones like pixels (Black & White, RGB, Mean of RGB channels), up to feature extracted from CNNs. We also used deep learning Neural Networks to solve this classification task, like VGG16 and ResNet18, tuning their hyperparameters.
Our main objective was to find which one of these models achieves the best accuracy, evaluating also advantages and disadvantages. CNNs have proved higher performances: ***VGG16 Neural Network has produced the absolute highest accuracy of 90%***, so this is the ***best model*** we found to classify Nature Dataset's scenes.
We've also focused on the study of feature performances, finding out that ***for our Nature Dataset, pixel features information are definitely more discriminant than features that are only related to color quantities***. Indeed, features extracted from CNNs reach the best accuracies for both KNN and SVM, by focusing on discriminant regions of the images, to recognize structures and shapes which are peculiar for each class. Instead, among "manually" extracted features, RGB pixels perform better. At the price of consistently lower accuracies (SVM 56%) than CNN features (77%), they gain interpretability.
Moreover, we analyzed which kind of feature optimally discerns each class, discovering that usually RGB pixels have the highest per-class precision and recall, only second to CNN features' ones. In particular, ***with RGB pixels, "sea" is the most problematic class and "forest" is the easiest to discern***. On the other hand, CNN features share the highest performances for class "forest" but ***achieve the worst precision and recall results with "buildings"***. This fact is strongly related to the similarity of images of these two classes, since they often both represent city where edifices and road are both often present. The same consideration holds for "glaciers" and "sea", that might lead also humans to doubt and mistakes.
In conclusion, we verified that the highest absolute accuracies of ***CNNs***, derive directly from their ***ability of discerning "buildings" from "streets" and "glaciers" from "mountains", that allows them to outperform all the other methods***, being probably able to outrun a human, since they focus on hidden and subtler features.

7. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1097–1105, 2012.

[2] Zihang Dai, Hanxiao Liu, Quoc V. Le, Mingxing Tan. CoAtNet: Marrying Convolution and Attention for All Data Sizes, 2021
https://paperswithcode.com/paper/coatnet-marrying-convolution-and-attention

[3] https://www.image-net.org/

[4] https://paperswithcode.com/sota/image-classification-on-imagenet

[5] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liub, Xingxing Wang, Li Wang, Gang Wang, Jianfei Cai, Tsuhan Chen. Recent Advances in Convolutional Neural Networks, 2017
https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A

[6] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, Quoc V. Le. Meta Pseudo Labels, 2021

[7] Pin Wang, En Fan, Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, 2020
https://www.sciencedirect.com/science/article/...

[8] Pádraig Cunningham, Sarah Jane Delany, k-Nearest Neighbour Classifiers
2nd Edition (with Python examples), 2020
https://arxiv.org/pdf/2004.04523.pdf

[9] Kashvi Taunk, Sanjukta De, Srishti Verma, Aleena Swetapadma, A Brief Review of Nearest Neighbor Algorithm for Learning and Classification, 2019
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9065747

[10] Stephen Notley, Malik Magdon-Ismail. Examining the Use of Neural Networks for Feature Extraction: A Comparative Analysis using Deep Learning, Support Vector Machines, and K-Nearest Neighbor Classifiers, 2018
https://arxiv.org/abs/1805.02294

[11] https://www.kaggle.com/puneet6060/intel-image-classification

[12] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

[13] Jolliffe, I. Principal Component Analysis; Springer: Berlin/Heidelberg, Germany, 2011.

[14] https://www.researchgate.net/figure/The-principle-component-analysis-PCA-procedure-source_fig2_337244216

[15] Geri Skenderi, Machine Learning & Artificial Intelligence, UniVR, 2021, Laboratory 4

[16] https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[17] https://www.researchgate.net/publication/243444000_Histogram_Generation_from_the_HSV_Color_Space

[18] https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

[19] https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

[20] https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[21] https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[22] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8237336