Filippo Ficarra: Lab 4 - Object Recognition

fficarra@student.ethz.ch, 22–938–062.

08/11/2023 - 16:39h

1 Introduction

The first part of the lab focuses on image recognition in a more classical way,l using a method borrowed from NLP. The Bag of word method is a method for Image recognition it is adapted this way:

- Divide the images in subimages
- Create a codebook of explenative images
- Correlate images to the one in the codebook to classify them

The second method uses a Deep Learning architecture to extract information and classify the images. This is done through the usage of a simplified version of the VGG architecture.

2 Bag of Words

2.1 Grid

The image is divided into a grid. The grid is created with the following steps:

- leave a border on all the edges. In this case border = 8
- compute the width and height of the grid
- take all the points that start from border to the image width border and analogously for the height
- create an array of points from the step before

2.2 Histogram of Oriented gradients

In order to extract the feature we use the HOG algorithm. The variant used in this exercise doesn't use the magnitudes of the gradient but just their orientation.
TO COMPLETE

2.3 Codebook

The book is generated gathering all the descriptors for all the sub images taken from the grid of all images and clusterizing them using the K-means cluster algorithm. The code was already implemented and the following code was added:

Listing 1: Codebook Generation

```
1  grid = grid_points(img, nPointsX, nPointsY, border)
2  descriptors = descriptors_hog(img, grid, cellWidth, cellHeight)
3  vFeatures.append(descriptors)
```

2.4 Bag-of-Words histogram

In the function create_bow_histograms() we compute for the descriptors of all the images the histogram that relates each feature to a the centroids. The function relies on the function bow_histogram() function that computes the histogram of the features of an images with respect to the centroids.

2.5 Nearest Neighbour

The classification is done using the findnn function to find both the distances of the positive BoW and the negative BoW. The smallest distance is the class we predict.

3 CNN-based Classifier

3.1 Training

The model is a simplified version of the VGG architecture. The architecture given was the following:

Block Name	Layers	Output Size
conv_block1	ConvReLU (k=3) + MaxPool2d(k=2)	[bs, 64, 16, 16]
conv_block2	ConvReLU (k=3) + MaxPool2d(k=2)	[bs, 128, 8, 8]
conv_block3	ConvReLU (k=3) + MaxPool2d(k=2)	[bs, 256, 4, 4]
conv_block4	ConvReLU (k=3) + MaxPool2d(k=2)	[bs, 512, 2, 2]
conv_block5	ConvReLU (k=3) + MaxPool2d(k=2)	[bs, 512, 1, 1]
classifier	Linear+ReLU+Dropout+Linear	[bs, 10]

Figure 1: VGG simplified architecture

The model was trained with the following parameters:

Argument	**Value**	**Description**
batch_size	128	Batch size
log_step	100	How many steps to log once
val_step	100	Validation step
num_epoch	50	Maximum number of training epochs
fc_layer	512	Number of features in the first linear layer in VGG
lr	0.0001	Learning rate

As shown in the table above the model was trained for 50 epochs with batch size 128.
Furthermore the training has been performed on Euler with 1 gpu RTX 4090.
Below we can find the graphs for the train losses in which we can see that the loss progressively drops from an initial value of 2 to a value around 1.3, while the validation accuracies pass from around 45% to a final value of 81.92%

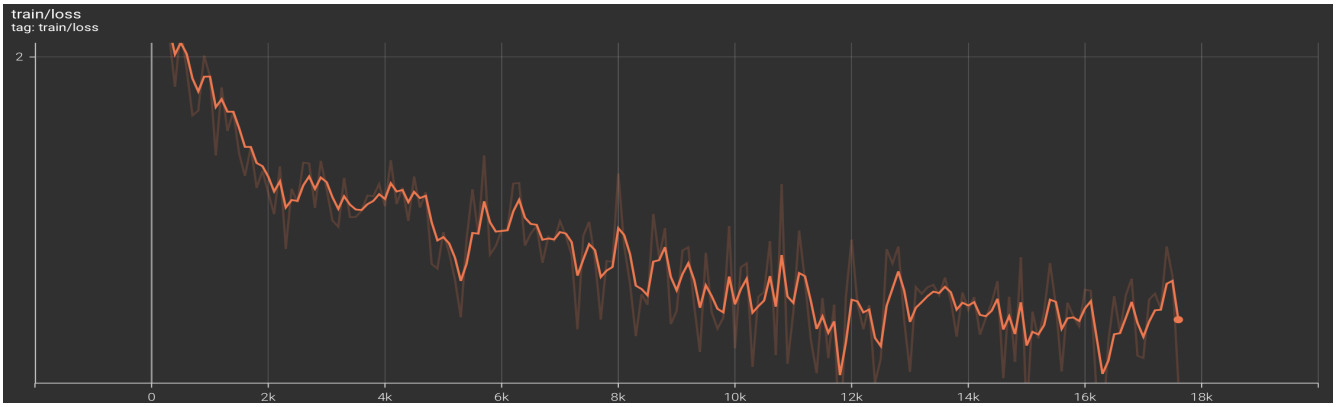

Figure 2: Train Losses

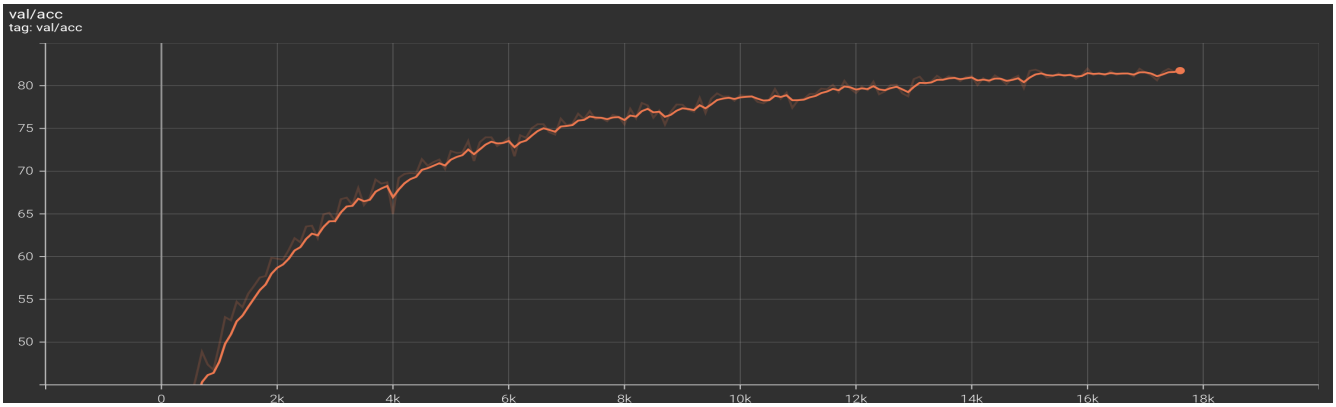

Figure 3: Validation Accuracies

3.2 Testing

The model after training with 50 epochs achieved a test accuracy of 80.86 on the CIFAR10 dataset. The test script was run locally on cpu for convenience and gave the following output:

```
79it [00:08,  9.11it/s]
test accuracy: 80.86
```