# Assignment 4

Daniele Gilio

May 31, 2020

## 0.1 Introduction

This assignment is about image classification. We are going to build a classifier that is able to distinguish between 12 flower species. The dataset we will be working on contains 80 images per species in the training set and 20 per species in the test set.

## 0.2 Low Level Classifier

The first approach we tested was using a classifier that used *Low Level Features*, which are specifically "crafted" functions that extract features from images. We tried using only one low level feature and then we tried using them all. With regards to the classifier we tested both a Single Layer Perceptron and a Multi Layer Perceptron. The first one had only the input layer, which dimensions matched the ones of the low level features, and the output layer made of 12 neurons. The MLP had the same input layer than the sequence $[512, 128, 32, 12]$, representing the width of each subsequent layer.

### 0.2.1 Single Feature

The Single Feature strategy obtained a very low accuracy score. We choose to use the *RGB Co-occurence Matrix* amongst all the possible ones because it was the best performing one. As we can see in Table 1 the MLP got a

| | SLP | MLP |
|---|---|---|
| Test Accuracy [%] | 45.83 | 60.83 |

Table 1: Single Low Feature Results

much higher score as expected. Both networks were trained for 500 epochs with a batch size of 16. The learning rate was set at $10^{-3}$ and was not modified during training.

### 0.2.2 Multiple Features

After the poor results obtained we decided to use all the available Low Level Features which are:

- Color Histogram
- Edge Direction Histogram
- Co-occurence Matrix
- RGB Co-occurence Matrix

In order to use them together we just concatenated them a big feature vector. The hyperparameters stayed the same for the MLP but we increased the epochs of training to 800 for the SLP. This time the result were a bit better but we still consider them insufficient, as we can see in Table 2.

| | SLP | MLP |
|---|---|---|
| Test Accuracy [%] | 59.583 | 69.583 |

Table 2: Multiple Low Features Results

## 0.3 Neural Features

### 0.3.1 PVMLNet as Feature Extractor

A leap forward performance wise is to use a CNN (Convolutional Neural Network) as a feature extractor. This way the valuable features are autonomously learnt by the network. In order to do this we used a pre-trained network called "PVMLNet", which structure we can see in Figure 1. This network was trained on a subset of the ImageNet Dataset. This means that we are using the features that the network learnt on that dataset hoping that they are general enough to work on ours. To use PVMLNet as a feature extractor we need to remove at least the last two layers, and use the activations of the last remaining layer as the feature vector. We basically pass the each item in the dataset through the PVMLNet and record the activations of the last active layer as its features. Once we processed the whole dataset we train a SLP for the final classification. The first test was performed using the layer before the last two as the feature vector constructor (we will indicate this

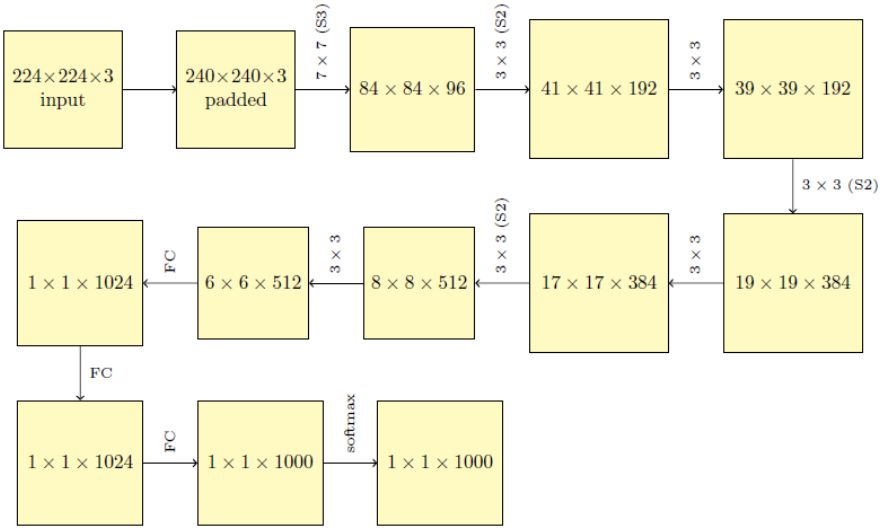


Figure 1: PVMLNet Structure

layer as $-3$, the last layer with this notation would than be $-1$ and the one before $-3$ would be $-4$). We then used layers $-4, -5$ and $-6$ [1]. We can see all this results in Table 3. The SLP was trained for 500 epochs with

| | Layer -3 | Layer -4 | Layer -5 | Layer -6 |
|---|---|---|---|---|
| Test Accuracy [%] | 88.75 | 92.083 | 93.75 | 94.167 |

Table 3: PVMLNet Feature Extractor + SLP Results

a batch size of 50. The learning rate was set at $10^{-3}$ and updated at each epoch[2]. Table 3 seems to indicate that the further back we go into PVMLNet the better the result. We were not able to prove this result due to hardware limitations (we ran out of memory during layer $-7$ testing). Looking at Figure () we can see that we probably would have encountered the law of diminishing returns.

### 0.3.2 Transfer Learning

Another way to use PVMLNet as a feature extractor is via *Transfer Learning*. This method consists in directly replacing the trained SLP to the removed layers in the CNN. This may seem unnecessary since this method yields the same results as the one described above. The major advantage of Transfer Learning is the possibility of fine tuning the resulting network. This process is nothing but an additional training performed with a very low Learning Rate. The choice of this hyperparamter is crucial since in this case is very easy to fall into overfitting. Our results are presented in Table **??**.

| | Layer -3 | Layer -4 | Layer -5 | Layer -6 |
|---|---|---|---|---|
| Test Accuracy [%] | 89.167 | 91.666 | 93.75 | 94.167 |

Table 4: Fine Tuned PVMLNet+SLP Results

## 0.4 Data Augmentation

## 0.5 Results Analysis

## 0.6 Conclusions

[1] Note that in order to use those layers we need to delete all the ones after them from PVMLNet and adjust the SLP input layer accordingly.

[2] As always with update it with $\mathrm{LR} = \mathrm{LR}_0 \,/\, \sqrt{t}$, where $t$ is the current epoch number.