MCV-M3: Image Classification (week 4)

Jonatan Poveda, Martí Cobos

Image Classification Pipeline

Implement a CNN based on a pretrained model

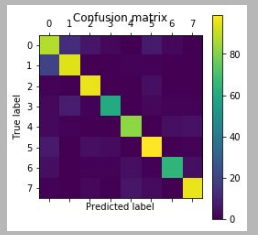
Input Images and Labels

CNN

CNN based on pretrained VGG16

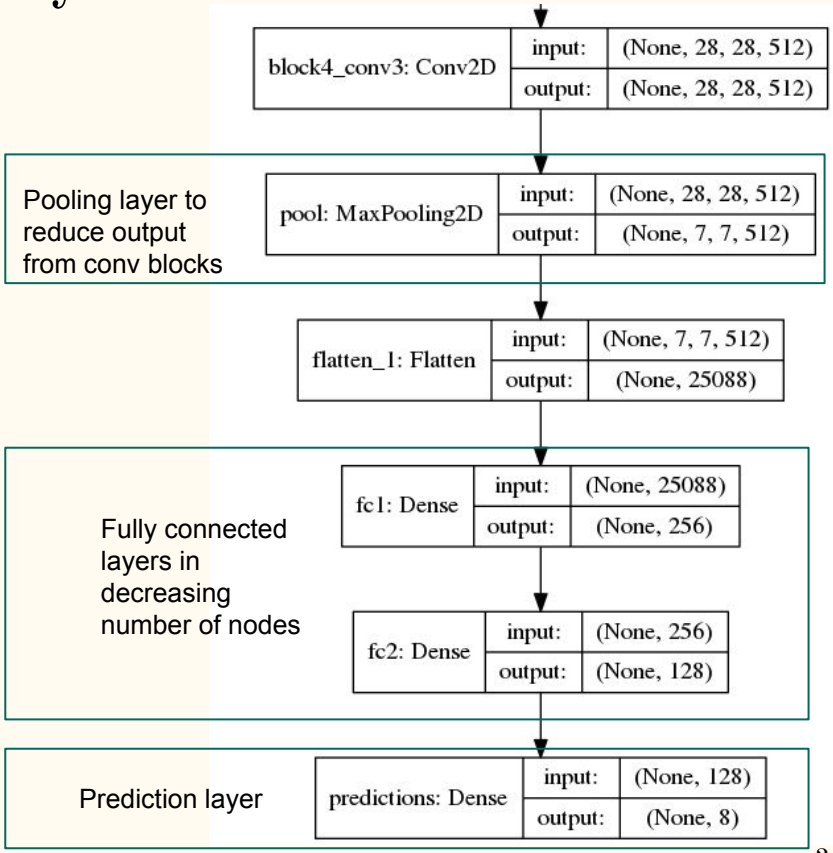
Evaluate Code Performance

Compute metrics using ground truth information



Task 1: Set a new model from a layer below block4 including at least a fully connected layer + a prediction layer

- The last convolutional blocks of VGG 16 is eliminated. A pooling layer to reduce the number of outputs from the convolutional blocks, two fully connected layers and a softmax layer are added at the end of the third block as shown on right image.
- The neural network is trained in 2 stages:
  - 1st stage: Freeze layers until the end of block 3, and train last fully connected layers.
  - 2nd stage: Unfreeze all layers to train all the CNN.
- The learning rate on the 2nd training stage is reduced in order to avoid accuracy drop when changing the layers trained.



Task 2: Apply the model to a small set of data

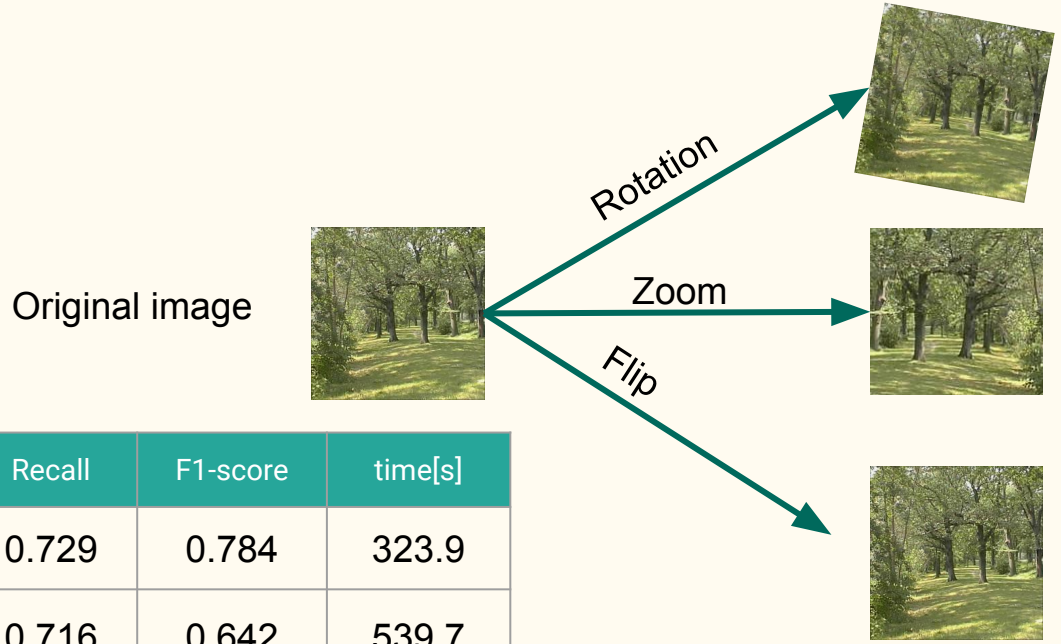
- The initial train set has been reduced to 400 images (50 images of each class)
- The script on the right has been used to select randomly 50 images of each class and generate a new training set

```python
import os
import shutil
import random
import numpy as np

ROOT_PATH = os.path.dirname(os.path.dirname(__file__))
DATA_PATH = os.path.join(ROOT_PATH, 'data')
TRAIN_PATH = os.path.join(DATA_PATH, 'train')
REDUCED_TRAIN_PATH = os.path.join(DATA_PATH, 'reduced_train')

folders = os.listdir(TRAIN_PATH)

for folder in folders:

    files = np.random.choice(os.listdir(os.path.join(TRAIN_PATH, folder)), 50)
    for image in files:
        file_to_copy = os.path.join(TRAIN_PATH, folder,image)
        path_to_copy = os.path.join(REDUCED_TRAIN_PATH, folder, image)
        shutil.copy(file_to_copy,path_to_copy)
```

Task 3: Introduce and evaluate the usage of data augmentation

In order to increase the number of train images, with just 400 images, data augmentation has been implemented. The following techniques have been implemented:

- Random rotation (15º)
- Random width shift (90%)
- Random zoom (30%)
- Random horizontal flip
- Filling with reflection

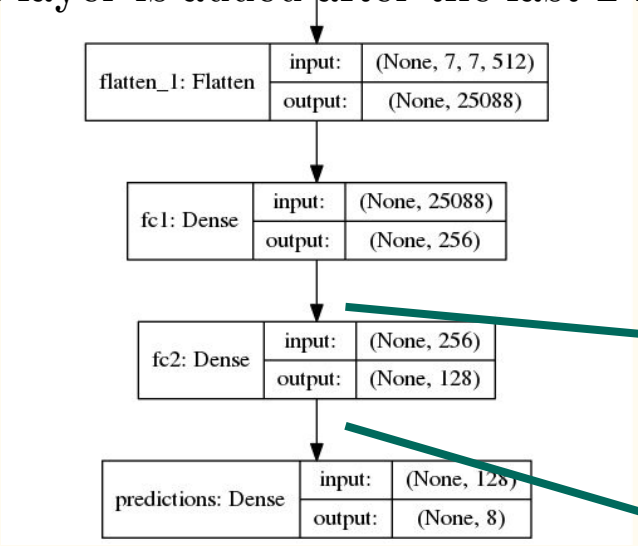


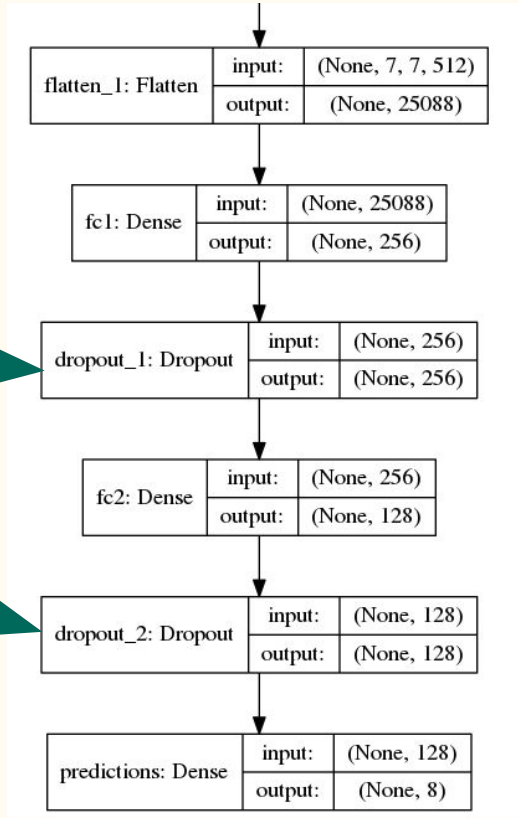
Results on test image set:

Image based	Accuracy	Precision	Recall	F1-score	time[s]
Normal	0.747	0.780	0.729	0.784	323.9
With data augmentation	0.674	0.639	0.716	0.642	539.7

Task 4: Introduce and evaluate the usage of a dropout layer

- A dropout layer is added after the last 2 fully connected layers:





Results on test image set:

Image based	Accuracy	Precision	Recall	F1-score	time[s]
With data augmentation	0.674	0.639	0.716	0.642	539.7
Dropout Added	0.489	0.465	0.403	0.461	521.6

Task 5: Apply random search on per model hyperparameters

- We had troubles in order to adapt the initial CNN. By the time those issues were fixed, we could not perform a large random search of the hyperparameters due to a lack of time.
- The following parameters were added to the random search:
  - Batch size (possible values: 16, 32, 64)
  - Fully connected layer 1 size (possible values: 512, 1024, 2048, 4096)
  - Fully connected layer 2 size (possible values: 128, 256, 512, 1024)
  - (We didn't try to search over the Adadelta optimize parameters because it was discouraged bu Keras library)
- Best results were achieved with a batch size of 32 and the size of the fully connected layers set to 1024. Evaluation results over the training set are shown on the following table:

Image based	Accuracy	Precision	Recall	F1-score	time[s]
Best random search results	0.557	0.467	0.580	0.466	292

Note: Random search did not improve previous due to lack of time to implement a larger search

Classification performance evaluation (on test images)

Image based	Train Loss	Train Acc.	Val loss	Val Acc.	# ephocs	time[s]
Method 1	0.1453	0.9447	0.2301	0.9306	20	112
Method 2	0.0296	0.9904	0.9028	0.7656	20	324
Method 3	0.6920	0.7550	0.9047	0.6589	20	540
Method 4	1.9153	0.2696	1.6319	0.5234	20	522

Algorithm performance evaluated on test images using the following architectures:

- Method 1: block4+4-pool+4096fc+1024fc (original training dataset, 1881 images)
- Method 2: block4+4-pool+256fc+128fc + training norm. (reduced dataset, 400 images)
- Method 3: method 2 + data-augmentation (reduced dataset)
- Method 4: method 3 + dropout in both FC (reduced dataset)

Conclusions

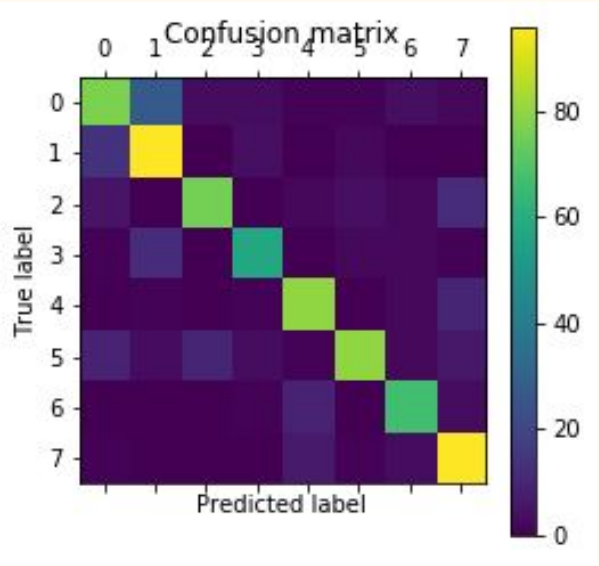
- Training a CNN is a very high time consuming time. By the time our framework was correctly set up, we could not search for the best hyperparameters with random search.
- In order to achieve good results data needs to be normalized. If it is not normalized, accuracy is around 15%.
- The dropout has worsen accuracy on both train and set images. Probably because the low number of parameters on the fully connected layers implemented (half are deactivated in the learning process), or because it would need more steps to learn.
- A visualization of the layers should be implemented in order to analyze better the results achieved.
- Best results achieved using method 2(block4+4-pool+256fc+128fc + training norm. (reduced dataset, 400 images)

Pixel based	Accuracy	Precision	Recall	F1-score	time[s]
Method 2	0.747	0.780	0.729	0.784	323.9

Conclusions

Best method Confusion Matrix plot

- Method 2



Future work

- Extend the number of epochs when using dropout so each node sees the same number of images than when no dropout is done.
- Implement visualization to get insights to know what is happening when applying dropout to the FC layers. It should improve test score due to its generalizing effect.