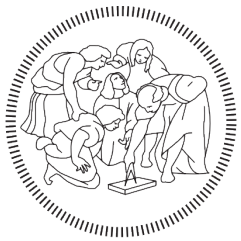

POLITECNICO
MILANO 1863

# Homework 1

Image Classification with convolutional neural networks

*Artificial Neural Networks and Deep Learning*


**Authors**

Ali Arslan - CP: *10807090*

Sorrentino Alessandro - CP: *10746269*

Menta Andrea - CP: *10636205*

**General approach:** We started by creating a fully customized CNN. Once found an base structure that had a good accuracy/training-time trade off we started by tuning the hyper-parameters in order to reach the maximum accuracy possible on the validation set (avoiding overfitting) and improving our model iteratively[1]. After getting a first satisfying accuracy on test we moved to transfer learning in order to increase the accuracy of our network even more.

**Dataset:** The dataset has been divided 3 sets: *training*, *validation* and *test*. Our custom python script divided uniformly the training dataset into training and validation with a 80% - 20% relation. Each class in training and validation roughly contains the same amount of images in order to avoid imbalances.

```
python3 dataset_creator.py -n 4491 -reproduce -DEBUG
```

Data augmentation was used in all the experiments with: shifting, rotation, flipping, height and width shift (fill mode), zooming, horizontal flip, brightness adjust and shearing.

# 1 Fully custom CNN

Each CNN block is characterized by 3 main components

- 2D Convolution (3x3)
- Activation Function (ReLu)
- Max Pooling (2x2)

**Model A (test accuracy $\approx 66\%$)** As base model, we started by creating a simple structure:

- 2 convolutional blocks
  - 1st with 32 filters
  - 2nd with 64 filters
- Flatten
- Dense layer of 64 neurons (ReLu)
- Dense layer of 3 output neurons (Softmax)

[1] Only the most important test have been reported for brevity

The learning rate was set ot 0.001, the batch size was 32 and, as method to prevent overfitting, the early stopping method was used with patience of 10 epochs. All the images in the dataset were resized to 256x256. The final result is an accuracy of 68% on the validation set and a 66% on the test set.

**Model B (test accuracy $\approx 76\%$)**[2] The batch size was decreased to 16. A third convolutional block with 128 filters was added after the second one. A dropout layer was added after each max pooling layer in the convolutional part with a 20% probability. A batch normalization layer was added in between of each convolutional and activation layers. In the FC part the number of neurons was increased to 128 and a dropout layer was added with a 50% probability in order to induce exploration (but causing a more noisy descent towards the optimal). Because of the noise caused by all the dropout layers, the epochs of training were set to 200 and the patience for the early stopping to 30. Only the best weights are kept if the early stops the execution. The weights were initialized with the he_uniform. All the images in the dataset were kept to 256x256. The final result is an accuracy of 77% on the validation set and a 76% on the test set.

## 2 Transfer Learning

After reaching good results in the first phase, in which we focused on *Full Custom Models*, we decided to try *Transfer Learning*.

**Model A (test accuracy $\approx 86.4\%$)** We started by choosing 3 pre-trained candidate models for the CNN part in order to choose the fittest:

- `VGG16`
- `MobileNetV2`
- `ResNet50`

We used the same base configuration for all: *batch size* of 32, same *data augmentation* (except for the *preprocessing function*, proper of each model), no fine tuning, same *fully connected layers*, *learning rate* of 0.001 and *early stopping* with patience 10 to avoid overfitting.
The fully connected part is so structured:

- Flatten layer
- Dense layer of 512 neurons ($ReLU$)

[2] First model delivered

- Dense layer of 256 neurons (*ReLU*)
- Dense layer of 128 neurons (*ReLU*)
- Output layer of 3 neurons (*Softmax*)

The higher accuracy and smoother convergence made us lean towards the `VGG` structure. At the end of the tuning we managed to get an accuracy of 82.7% on the validation set and a 83.4% on the test set.

**Model B (test accuracy $\approx 95.5\%$)**[3] After some intermediate tests we decided to move from `VGG16` to `VGG19` which provided us more performance. We found significant benefits using greater image size (from 256x256 to 612x612). The batch size was decreased to 16, the learning rate was kept to 0.001 and the patience of the early stopping was also kept at 10 epochs. The FC was taken from the previous model and modified as follows. We used the *LeakyRelu* instead of the normal *Relu* and the weights initialization was changed from *xavier* to *He uniform*. As we observed in the Model C of the fully custom network, adding some more non linearty using *dropout* (with descending dropout rate) and *batch normalization* was helpful in increasing the accuracy. We also did fine tuning on the last block of the convolutional part.

The fully connected part is so structured:

- Flatten Layer
- Dense Layer of 1024 neurons (*He uniform*)
- Batch Normalization Layer
- Activation Layer (*LeakyReLU*)
- Dropout (0.7)
- Dense Layer of 512 neurons (*He uniform*)
- Batch Normalization Layer
- Activation Layer (*LeakyReLU*)
- Dropout (0.5)
- Dense Layer of 256 neurons (*He uniform*)

[3] Second model delivered

- Batch Normalization Layer
- Activation Layer (*LeakyReLU*)
- Dropout (0.3)
- Output Layer of 3 neurons (*Softmax*)

At the end of the tuning we got an accuracy of 94.2% on the validation set and a 95.5% on the test set.

**Model C (test accuracy $\approx 94.4\%$)**[4] We also explored another model, the `InceptionResnetV2` which gave us comparable performance with faster training with respect to the `VGG19`. We used an hybrid approach between full-custom and transfer learning, taking the *"imagenet"* network weights as an initialization and re-training all the model to adapt it to our task. We used Global Average Pooling at the end of the convolutional part and a simple FC structure[5]. We also used a couple of tricks in order to avoid *Plateau* (*ReduceLROnPlateau* callback) and improve test accuracy (*Test Time Augmentation*). At the end we reach an accuracy of 94.9% on the validation set and a 94.4% on the test set.

[4]Third model delivered

[5]See the notebook comments for further specifications