# COMPUTER VISION AND PATTERN RECOGNITION PROJECT
## CNN CLASSIFIER

Alessandro Cesa

January 2024

# Contents

# Introduction

The final project for the course in Computer Vison and Pattern Recognition consists in a classification task on a provided dataset containing 15 categories of images [2]. The classification has to be carried out by ploying a Convolutional Neural Network, built following given specifications and steps described in the project assignement.

# Problem statement

The project requires the implementation of an image classifier based on Convolutional Neural Networks, following three major steps.

1. Build from scratch a Convolutional Neural Network following this architecture:

   Convolutional Neural Network Architecture

   | **Layer** | **Size** |
   |---|---|
   | Image Input | $64 \times 64 \times 1$ |
   | Convolution | $8 \times 3 \times 3$ |
   | ReLU | - |
   | Max Pooling | $2 \times 2$ |
   | Convolution | $16 \times 3 \times 3$ |
   | ReLU | - |
   | Max Pooling | $2 \times 2$ |
   | Convolution | $32 \times 3 \times 3$ |
   | ReLU | - |
   | Fully Connected | 15 |
   | Softmax | - |
   | Classification Output | - |

   Following these specifications:

   - Rescale the images to 64x64
   - Split the provided training set in 85% for actual training set and 15% to be used as validation set;
   - Employ the stochastic gradient descent with momentum optimization algorithm, using the default parameters of the library you use, except for those specified in the following;

- Use minibatches of size 32 and initial weights drawn from a Gaussian distribution with a mean of 0 and a standard deviation of 0.01; set the initial bias values to 0;

It's required to reach an accuracy of 30%

2. Improve the previous result, according to the following suggestions (not all the following will necessarily result in an increase of accuracy, but you should be able to obtain a test accuracy of about 60% by employing some of them):

   - **Data Augmentation:** Given the small training set, data augmentation is likely to improve the performance. For the problem at hand, left-to-right reflections are a reasonable augmentation technique. By augmenting the training data using left-to-right reflections, you should get an accuracy of about 40%.
   - **Batch Normalization [1]:** Add batch normalization layers before the ReLU layers.
   - **Change Convolutional Filters:** Change the size and/or the number of the convolutional filters. For instance, try increasing their support as you move from input to output: $3 \times 3$, $5 \times 5$, $7 \times 7$.
   - **Optimization Parameters:** Play with the optimization parameters (learning rate, weights initialization, weights regularization, minibatch size, etc.). You can also switch to the Adam optimizer.
   - **Dropout:** Add some dropout layers to improve regularization.
   - **Ensemble of Networks:** Employ an ensemble of networks (five to ten), trained independently. Use the arithmetic average of the outputs to assign the class, as in [3].

3.

# Implementation

# Results

# Bibliography

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[2] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.