Introducing Machine Learning Concepts by Training a Neural Network to Recognize Hand Gestures*

Alessandro Giusti, David Huber, Luca M. Gambardella
Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland

Abstract

We present an interactive guided activity to introduce supervised learning by training a deep neural network (treated as a black box) to recognize "rock paper scissors" hand gestures from unconstrained images. The audience is actively involved in acquiring a varied and representative dataset, on which the rest of the activity is based. Covered concepts include the training/evaluation split, classifier evaluation, baseline accuracy, overfitting, generalization, data augmentation.

Introduction

Machine Learning (ML) techniques play an increasingly important role in society and everyday life, yet most people, with the exception of practitioners of the field, are unaware of ML's main ideas, capabilities and limits. We present an interactive, guided experimental activity which assumes no background knowledge, during which the audience is introduced to supervised deep learning and some of its core concepts in a learning-by-doing fashion. The activity consists in building from scratch a system that solves a challenging visual pattern recognition task, namely classifying "rock paper scissors" hand gestures from pictures; the process encounters unanticipated setbacks and challenges, which prompt the discussion of core ML concepts (**bold** in the description below).

The activity can be easily tuned to different levels of detail, requires from 30 minutes to 2 hours, and has been tested successfully on small groups (10-20 people each) ranging from secondary education students to adults. The audience participates in the activity by shooting training pictures with cellphones, and testing the classifier once it's learned. In the proposed form, the activity does not involve coding, but could be easily adapted to a longer, hands-on experience for university Computer Science students.

Downloads and extra material related to this activity are available at http://bit.ly/2iW7zaA.

*This project has been partially supported by the Swiss National Center of Competence Research (NCCR) Robotics through the Swiss National Science Foundation

Activity Description

1. Initially, a short (5-10 minutes) presentation introduces what a classification problem is, and how a **supervised classification pipeline** works i.e.: we obtain a training dataset comprised by many instances with known ground truth, use this dataset to train a classifier, and once this classifier is trained, apply it to unseen instances to estimate their class.

2. We introduce the problem we want to solve: given a picture of a hand, estimate its class as one of: rock, paper or scissors. We show a set of 100 pictures to be classified (our evaluation set), and quickly scroll through them to: show that this problem is generally easy for humans; show that the three classes are evenly represented, so the **baseline accuracy** would be 33%; exemplify the expected framing (the hand is always well centered in the photograph and in the foreground); show that rotation of the hand is arbitrary, the background may be uneven, lighting and subjects are heterogeneous (adults and kids, male and female, different skin colors).

3. In order to train a classifier, we need a training dataset, which we don't yet have: so we ask the audience to acquire it. We underline that for each picture they shoot we need to know the class, and in practice we want to end up with three folders full of pictures, one for each. We remark the importance of having a large, correct, and **representative training dataset**: the pictures they shoot should be as varied as possible and cover the range of conditions represented in the evaluation pictures. The time required to acquire the data is at least 10 minutes (motivated users shoot up to 20 pictures per minute per device, but on average we observe about 5 pictures per minute per device), but can become a longer assignment if there is a coffee break in the middle or the activity is split in two days. We show the audience that we collected their pictures in three folders[1], one for each class, and show how large the dataset is. We expect between 80 and 200 images per class.

4. We now switch to a Jupyter notebook (Kluyver and others 2016) running Python, in which the rest of the activity takes place; we don't write code live nor discuss the code contents, but we execute one cell at a time and discuss results.

[1] supplementary material includes software to facilitate the logistics of this step

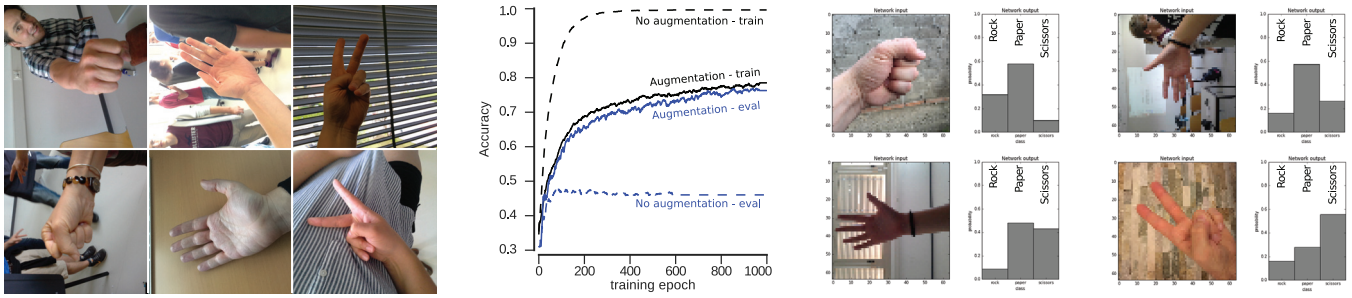

Figure 1: Left: example training images. Center: training (black) and evaluation (blue) accuracy without (dashed) and with (continuous) data augmentation. Right: visualization of results on evaluation images (1 wrong, 3 correct).

5. We load all pictures and display one with a viewer widget, zooming in until individual pixels are visible; we discuss how images are represented. We quickly scroll through the images of the dataset, underlining that we know the class of each.

6. We discuss whether we really need so many pixels to represent an image, explaining that choking the classifier with redundant data won't help: we need a **compact representation**. We scroll through the images at different resolutions and ask: are the classes still recognizable to us as humans? We settle on 64x64 pixels as a choice that is compact but still recognizable.

7. We are now ready to train a deep neural network on this data. We don't discuss the network internals, but treat it as a black box with an input (64x64 RGB triplets) and an output (3 activations for rock, paper, scissors respectively). We start training the net (using the Keras (Chollet and others 2015) high-level frontend) and while it trains, we briefly discuss the concept of **gradient descent**.

8. We recall how the accuracy of a classifier is computed on a dataset, then we show a live graph (black, dashed in Figure 1) of the accuracy of the network on the training data, as the network is trained epoch-by-epoch. We verify its starting point, which will be around 33%, is the baseline we can expect when we just give chance answers. After a few minutes, the training accuracy reaches 100%. Everybody in the audience seems satisfied.

9. We now apply the network on the 100 evaluation images, which the network has never seen. For each image, we show the input of the net and its output. To the audience's dismay, many answers are wrong. We go through a few and compute a numerical accuracy value on the whole evaluation set (less than 50%). We reopen the plot of the training accuracy, which had reached 100% after a few epochs, and reveal a new (blue, dashed in Figure 1) plot, which shows how the evaluation accuracy changed during the training. It grows a little then shows no signs of increase (or even starts decreasing). We remark the importance of the **training-evaluation split**. We explain what has happened: **overfitting**. The network has memorized rules to classify correctly the training samples (after all, they are not so many) but those rules are instance-specific and do not capture the problem we want to solve. One way to deal with overfitting is to train for a shorter time, but in our case it would not work, as shown in the evaluation accuracy plot. We are dealing with a really hard task that probably requires 10x or even 100x more training data. Should we spend some days acquiring more pictures?

10. We introduce the idea of **data augmentation**: we could generate more training examples synthetically, given the limited ones we have. If we have a picture of "rock" and rotate it 30 degrees, it will still be "rock", but will be a new instance for the net. Same if we flip the image, zoom in or out a little, or perturb the color balance, brightness, or contrast. We propose that every time we show a picture to the net, we will apply all these perturbations at random, so the network will never see the same image twice. For a single picture in the training set, we perturb it 50 times and quickly scroll through the resulting images.

11. We train the network again, now with data augmentation. We show that the training accuracy (black line) grows much more slowly than before, which is good news: the net can not just memorize all training samples. More importantly, the evaluation accuracy (blue line) is now growing steadily. After a few minutes, we stop the training.

12. We apply the trained net on the evaluation images; now, about 75% are classified correctly. For each image, we display the network outputs as three bars and discuss the concept of **probabilistic outputs**. For images that are classified incorrectly, often one can find and discuss a good reason (unclear picture content, confusing background, weird lighting). We remark that given the difficulty of the problem, and the limited size of the training set, we are happy with the result. If needed, we load a classifier that was previously pre-trained on a larger training set for a long time, and demonstrate its results.

13. Finally, we leave the classifier running on the presenter's laptop and show the live classification from a camera feed. The audience can freely experiment with the classifier and discuss its performance.

References

Chollet, F., et al. 2015. Keras: Deep learning library for theano and tensorflow. https://keras.io.

Kluyver, T., et al. 2016. Jupyter notebooks-a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87–90.