Object Classification

Emanuele Giuseppe Maria Maita EMANUELE.MAITA@OUTLOOK.IT

1. Model Description

The model used to tackle this task is a convolutional neural network, which consists on several layers of computation used to extract features with increasing complexity from the input image, a fully connected layer that combines them and a classifier which, given such features, computes probabilities of membership to a certain class (see Fig. 1).

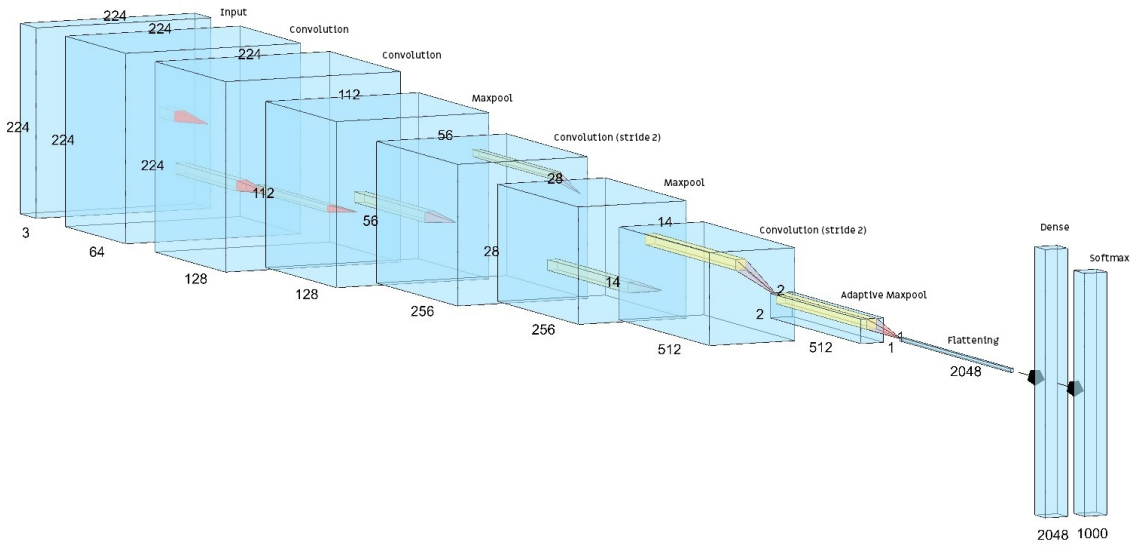

Figure 1: Representation of the proposed model.

Following the pre-processing/augmentation phase (see below), the input comes as a 224x224 color image. The architecture presents 4 convolutional layers, each one followed by the ReLU activation function and sharing the same kernel size (3x3) and padding (1). The number of feature maps grows by a factor of two: 64, 128, 256, 512. From the second convolutional layer onwards, they are followed by max pooling (the last layer presents adaptive max pooling to give to the final convolutional output 2x2 $W \times H$ dimensions). While the first and second layer have stride 1, the last two present stride 2 (as a consequence, the features' width and height decrease faster towards the end). Finally, the features are flattened prior to the dense layer and then passed to the softmax output layer of the 8 classes.

2. Dataset

The dataset consists on 7068 images of different sizes representing one of the following objects: gasoline can, hummer, pebbles, pliers, rope, screw driver, toolbox, wrench. Some images features distractors of some kind (mostly humans interacting with the object, see Fig. 2).

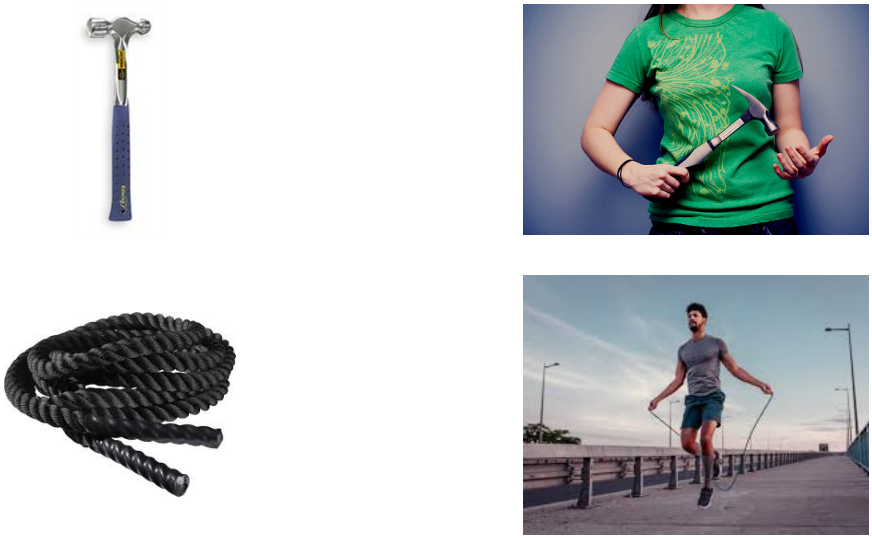

Figure 2: Images labelled with "hammer" (first line) and "rope" (second line). To the left more "representative" images, to the right images with (human) distractors.

There is a non-negligible amount of class imbalance: the ratio between the most populated class and the least populated one is around 8:1 (see Fig. 3).

The average dimensions of the images are 282x339 pixels (WxH); the minimum width is 88 and the minimum height is 36 pixels; the maximum width is 6720 and the maximum height is 8688 pixels.

The dataset was randomly splitted in training, validation and test set with proportion 80/10/10, i.e. with 5656 samples used as train set, 706 samples used as validation set and another 706 samples used as test set.

3. Training procedure

Input images fed to the network during training are first processed/augmented through the following pipeline:

- resizing to 300x300;
- random cropping to 224x224;
- normalization with mean 0.5 and standard deviation 0.5;
- random horizontal flipping with probability 0.4;

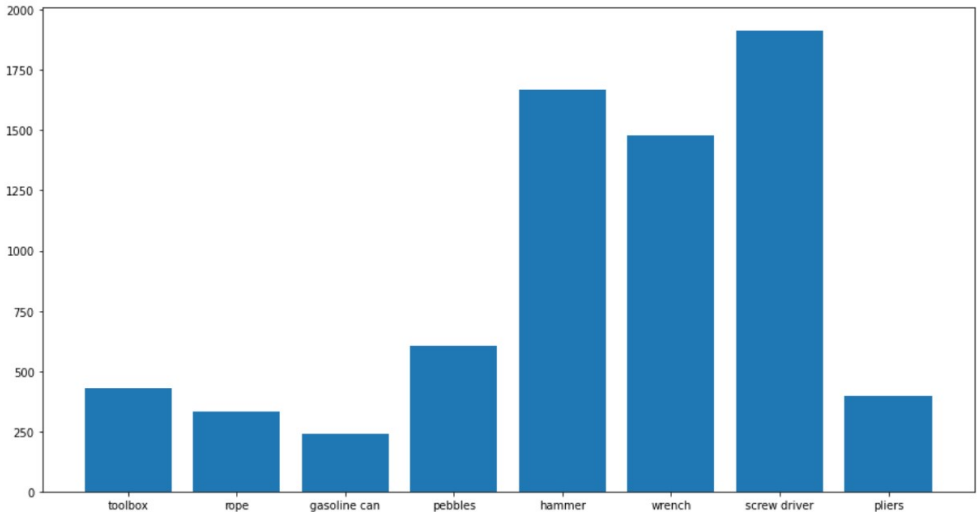

Figure 3: Number of examples per class.

- random vertical flipping with probability 0.4.

Each network is trained for 100 epoch with batch size equal to 8, using an Nvidia GeForce GTX 1050 Ti GPU with 4 GB of RAM. The last batch with less than 8 element is discharged during training.

The chosen loss function for training is cross entropy, optimized using stochastic gradient descent with learning rate 0.01. The performance of the model is evaluated using standard accuracy.

4. Experimental Results

The baseline network for the experiment is a convolutional neural network with the following structure:

- convolutional layer with 64 feature maps of kernel size 3x3, stride 1, padding 1 and ReLU activation function;
- adaptive max pooling, with output dimensions 64x2x2 (DxWxH).
- after flattening the prior output, a dense layer with 1024 neurons;
- 8 neurons softmax classifier layer.

From the baseline, convolutional layers followed by ReLU activation function and max pooling are added one at a time (see Table 1). For each new convolutional layer, the number of feature maps are twice the prior layer and the size of the kernel and padding hyperparameters are kept constant. Stride for the first new layer is 1; for each new layer it's equal to 2. Max pooling is done with kernel size 2 and stride 2. The pooling step at the end of the last convolutional layer is specified by adaptive max pooling to get 2x2 dimensions of the output. For the third model onwards, the final dense layer presents more number

of neurons than 1024: in particular, is the same number of the dimensions of the flattened output of the convolution layers (2048, 4096, 8192 respectively).

For each architecture, the best model is chosen according to the stopping rule: the weights configuration with minimum validation error among the 100 epochs. The recorded performance is given by the accuracy on an independent test set for the chosen configuration.

Model	Accuracy
Baseline Net	58.29%
+ Layer 1	63.32%
+ Layer 2	67.13%
+ Layer 3	67.70%
+ Layer 4	65.17%
+ Layer 5	66.29%
Final model	67.70%

Table 1: Test performances.

The selected model has a training loss of 0.6052, a validation loss of 1.0114 and a test loss of 1.0010. Besides the mentioned test accuracy, the training accuracy is 0.7726 and the validation accuracy is 0.6629. The confusion matrix (see Fig. 4) on test set highlights e.g. the model disorientation in distinguishing hammers, wrenches and screw drivers from each other.

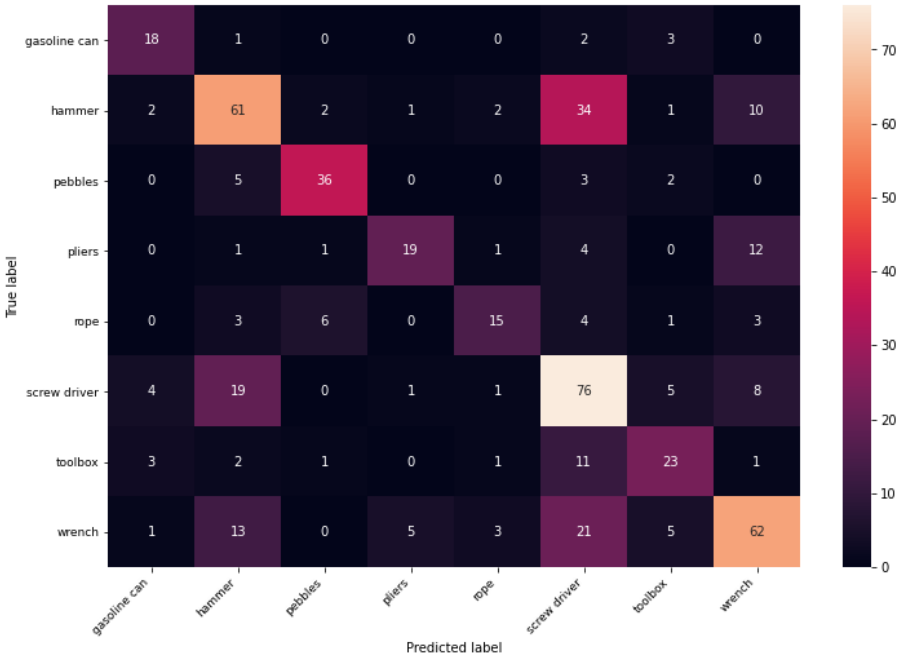

Figure 4: Confusion matrix on test set.