Salvatore Romano SALVOROMANO23@GMAIL.COM

Using Faster-R CNN for image classification

1. Model Description

The model that I decided to use is **Faster R-CNN** (Ren et al., 2016), a deep convolutional neural network used mainly for object detection; it can be considered the extension of other method like R-CNN and Fast-R CNN. The main point of this model is that it is able to build a region-proposal network that can generate region proposals that are fed to the detection model (Faster R-CNN) to inspect the objects.

Moreover, together with the Faster R-CNN, is used a **ResNet50 FPN V2** as a sort of "backbone" for the object detection model; this means to use the backbone as feature extractor. ResNet50 FPN v2 is a deep neural network that combines **Residual Networks** (ResNet) and **Feature Pyramid Networks** (FPN).

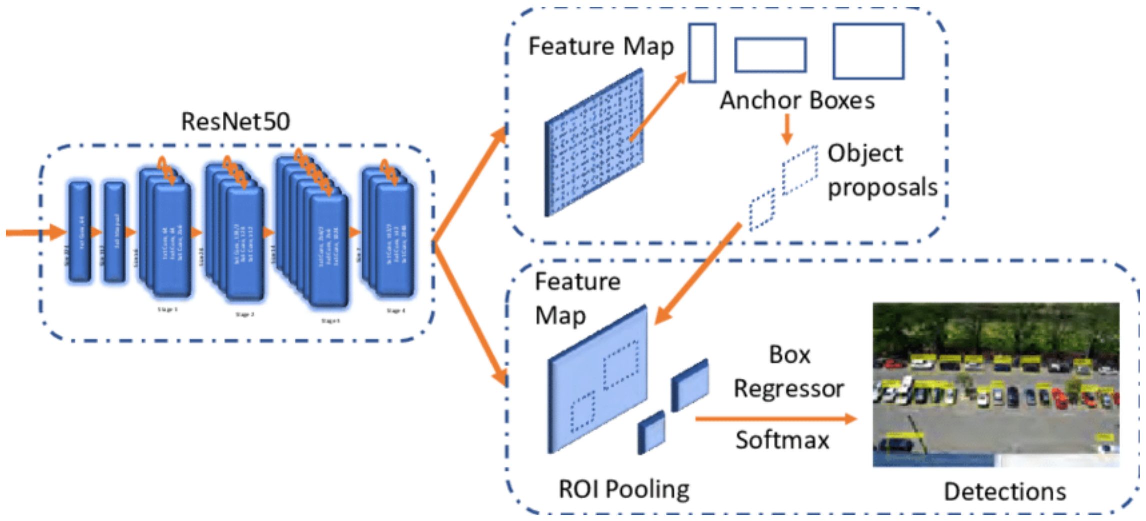

Figure 1: Faster R-CNN architecture containing a ResNet50 backbone

As represented in Figure 1 it is possible to see the complete architecture of Faster R-CNN.

The image first passes through the backbone network to get a feature map, and the ground truth bounding boxes are projected onto the feature map. So the output for this step is a spatially dense *Tensor* representing the learned features of the image; each point of the feature map will be handle as an *anchor* by the FPN network that will generate multiple boxes of different size. This allows the model to detect objects of various sizes and aspect ratios. There will be positive or negative anchor boxes based if they will overlap to the projected ground truth boxes. A 1x1 convolutional network is used to predict the offset from ground truth boxes, and then the offset is applied to bring the anchor boxes closer to the ground truth boxes. Now the anchor boxes are called region proposal cause they are transformed using the predicted offset and the network is the **Region Proposal Network** (RPN). The second stage for the Faster R-CNN is to take as input the region proposal generated; now it is predicted the category of the object in the region proposal using a simple convolutional network. In this phase is used a **Region of Interest** (ROI) pooling layer to resize the region proposals before passing them through the network that will learn to predict multiple categories. At the end, for each region proposal, a feature vector is extracted. This vector is then fed to 2 sibling fully-connected layers: the first FC layer is named *cls*, and represents a binary classifier that generates the objectness score for each region proposal (i.e. whether the region contains an object, or is part of the background) instead the second FC layer is named *reg* which returns a 4-D vector defining the bounding box of the region. The first FC layer has two outputs, one for classifying the region as background, one for classifying the region as object. If the first output is 1 and the second is 0, the region proposal is classified as background and vice versa.

2. Dataset

The dataset that I used for this Homework has been provided by the Professor under a Kaggle competition (ConcettoSpampinato, 2023); it is a *hand-held* object dataset, characterized by the presence of domain shift between training and test data. The images in train set are 1600, 200 for each class; instead in test set there are 800 images. The original dimension of all the images is 350x350. There are 8 classes as you can see by inspecting the Figure 2: plug adapters, mobile phones, scissors, light bulbs, cans, sunglasses, balls and cups.

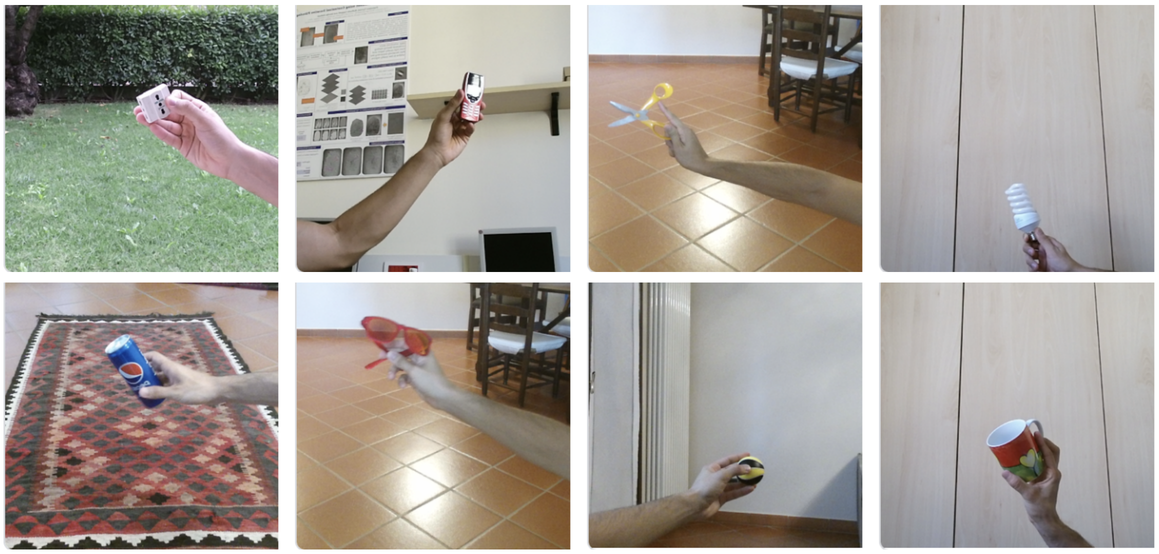

Figure 2: Dataset's classes

In particular the class 0 that corresponds to the plug adapter, has a the same background of all the images in the test set, this produces the so called domain shift, defined as a change in the data distribution.

As the problem of domain shift affects a lot the performance, also the bounding boxes were provided for this image classification task. The bounding boxes are in a *.csv* file, containing the following information: image_id, x1, y1, x2, y2, class. By having them, we can force the model to be focused only on the area inside the coordinates.

In order to improve the model's performance I used **Albumentation**, a popular library for image augmentation; the transformations used in training's images were flip with a chance of 50% and blur with a chance of 80%. Next, the train data loader has been created with a batch_size of 8, shuffle *True* and 4 as num_workers.

3. Training procedure

Now is important to define some of the methods used in the training process.

For training RPNs, it is assigned a binary class label (of being an object or not) to each anchor. The label is positive for two kinds of anchors: (i) the anchor/anchors with the highest **Intersection over Union** (IoU) overlap with a ground-truth box, or (ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box. The label is negative for a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective. Some RPN proposals highly overlap with each other. To reduce redundancy, it is adopted **non-maximum suppression** (NMS) on the proposal regions based on their *cls* scores.

The loss types used by Faster R-CNN, defined as composite loss function, combines the losses of both classification and bounding boxes regression:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right) \tag{1}$$

Here in 1, the classification loss L_{cls} is log loss over two classes (object vs. not object) and for the regression loss is used $L_{reg}\left(t_i, t_i^*\right) = R\left(t_i - t_i^*\right)$ where R is the robust loss function (smooth $L1$) of Fast-R CNN. The outputs of the *cls* and *reg* layers consist of $\{p_i\}$ and $\{t_i\}$ respectively.

Moreover, the chosen optimizer is the **Stochastic Gradient Descent** (SGD) with a learning rate of 0.005, momentum of 0.9 and weight decay of 0.0005. To improve the model's convergence during training, a learning rate scheduler has been used to reduce the learning rate of the optimizer by a factor of gamma every step_size epochs. The model has been trained for 5 epochs on a NVIDIA T4 GPU provided by Kaggle.

4. Experimental Results

Before getting into the experimental results, it is good to know that I tried different "models" for this Homework. The first attempt was about pretrained CNN, the **GoogLeNet** (Szegedy et al., 2014) applied to the raw version of the train set, but the results in test were not good due to the presence of domain shift. So all the predictions were 0. Considering the scarcity of results, I decided to use Faster-R CNN, with the following trials:

1. Faster-R CNN with ResNet50 FPN as backbone, usage of bounding boxes and image augmentation
2. Faster-R CNN with ResNet50 FPN v2 as backbone, usage of bounding boxes and image augmentation with Albumentaiton

Model	**BBs**	**Albumentation**	**Test accuracy**
GoogLeNet	No	No	13.25%
Faster R-CNN with ResNet50 FPN	Yes	No	87.25%
Faster R-CNN with ResNet50 FPN v2	Yes	No	93.75%
Faster R-CNN with ResNet50 FPN v2	Yes	Yes	99%

Table 1: Models' performance on test set

By looking at the Table 1 we can see that the GoogLeNet performed the worst cause weren't adopted solutions against domain shift. By changing completely architecture the increase in term of accuracy is obvious, but the computational time was very high, something around 40 to 50 minutes for the training phase of Faster-R CNN with ResNet50 FPN v2.

References

Giovanni Bellitto Simone Palazzo ConcettoSpampinato, Federica Proietto. Ai@unict 2023, 2023. URL https://kaggle.com/competitions/aiunict-2023.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.