# Siamese Architecture For Template Matching

Andrea Bennati Alessio Della Libera

## I. Introduction

The goal of this project is to perform template matching using a Siamese Architecture in order to identify multiple templates in different target images. We choose to apply this method in order to identify different kind of fruits. This method could be applied in the context of grocery store to identify if in a fruit box there are missing fruits, if there are fruits that should not be in that box, or if there are rotten fruits.

## II. Proposed approach

A Siamese Architecture is meant to learn a function $f_W$ parameterized by $W$, that "encodes" inputs $x_i$ in a feature space. The distance between points in these features space should semantically approximate the distance in the original space, such that if the inputs belong to the same class, this distance should be small, otherwise this distance should be large. This means that similar images will be mapped close in the features space. The function $f$ can be any function, but since we are dealing with images, we used a Convolutional Neural Network (CNN).
Once we have trained our architecture, given a template image, we extract patches from the target image and compute the distance between the template and each patch. If this distance is below a threshold, this means that that patch is of the same class of the template (and thus we draw a bounding box around that patch).

### A. Dataset

In this project we used the Fruits-360: dataset [1] that contains images with dimension 100x100x3 pixels. The dataset contains 131 different fruits and a total of 90483 images ($\sim$400 images per fruit). All the images of each class are taken from a single fruit from 360 degrees around it in all the directions. The images have a white background. Figure 1 shows some example of images of the dataset. Due to time constraints, we took a subset of the entire dataset for a total of 75 classes. Then we choose $10\%$ of this classes as testing (7 classes) and the remaining classes for training/validation (68 classes). For the validation we took $10\%$ of the images from the training dataset. This means the testing set contains classes never seen during training.

### B. Architecture

The first pipeline that we used for the Siamese Architecture is the following (see Fig. 2 for a visual representation of the architecture):

- Take in input a pair of images $(x_i, x_j)$ and a label $y$ (0 if the images are of the same class, 1 otherwise)
- Compute the encoding $f_W(x_i), f_W(x_j)$
- Compute the distance between encoding: $D_W(x_i, x_j)$
- Minimise the contrastive loss function

Another architecture we tried is the triplet architecture. The pipeline is similar to the previous one:

- Take in input an anchor ($a$), positive ($p$) and negative ($n$) image
- Compute the encoding $f_W(a), f_W(p), f_W(n)$
- Compute the distance between encoding: $D_W(a, p)$ and $D_W(a, n)$
- Minimise the triplet loss function

In both cases the networks share the same weights. See Table 1 for details about the Neural Network architecture.

### C. Loss Functions

*1) Contrastive Loss:* The contrastive loss used in the first architecture is defined as follows [2]:

$$L(W, y, x_i, x_j) = \frac{1}{2}(1-y)D_W^2 + \frac{1}{2}y \max\{0, m - D_W\}^2$$

Where $D_W = ||f_W(x_i) - f_W(x_j)||_2$, $y$ is the label, $m$ is the margin.

*2) Triplet Loss:* The triplet loss used in the second architecture is defined as follows [3]:

$$L(a, p, n) = \frac{1}{2}\max\left(0, m + D_W^2(a, p) - D_W^2(a, n)\right)$$

Where $D_W(a, p) = ||f_W(a) - f_W(p)||_2$ and $D_W(a, n) = ||f_W(a) - f_W(n)||_2$

## III. Experiments

We achieve our best result using the first architecture. We train the model for 5 epochs, with training batches of size 32. We used Adam optimizer with a learning rate of $0.001$. Figure 3 shows the distance and the prediction of a batch of images during validation and testing. In order to choose the best value for the threshold, we also compute the average distance in case of positive pairs and the average distance in case of negative pairs. This way we were able to identify the best value for the threshold, given a particular margin.

### A. Evaluation

Once we have a trained model, in order to evaluate it, for each pair of images in the validation/testing dataset we compute the encoding then we compute the distances between these encoding and finally we predict 0 if this distance is below a threshold 1 otherwise. We choose the threshold as explained in the section Experiments.

## IV. Results and discussion

We have been able to achieve a validation accuracy of 98% and a testing accuracy of 91% with the first Siamese Architecture (using a margin of 1.0 and threshold of 0.6).
We decide to apply this method to a real world images we took. In figure 4 and 5 we can see that we able to identify all the template that we chose. On the other hand, in figure 6 we report some missing matches. These failure are import to understand the behaviour of our model and how this model can be improved to identify template more robustly. We think that the possible causes of the errors that we obtained in our experiments are due to the dataset that we used. The Fruits-360 dataset has a lot of images per class but all those images are taken from a single fruit from 360 degrees around it. Given that each kind of fruit may have different color or shape between one and another this may affect the prediction. Furthermore the training images have white background; we tried to test this architecture also with images with different background and the prediction weren't so great. To conclude we think that with a better and more varied dataset the Siamese architecture could achieve good performances for the template matching task.

| Layer Name | Output Size |
|---|---|
| Conv1 | 16x96x96 |
| Conv2 | 34x48x48 |
| Conv3 | 64x18x18 |
| Conv4 | 128x5x5 |
| Linear1 | 1024 |
| Linear2 | 256 |

TABLE I: Schema of the architecture. We have four step of convolutional layer each one activated by a Relu function and then followed by a Max Pooling. After those we have two fully connected linear layer each one activated with a Sigmoid function.

Fig. 1: Examples of the images of the dataset.

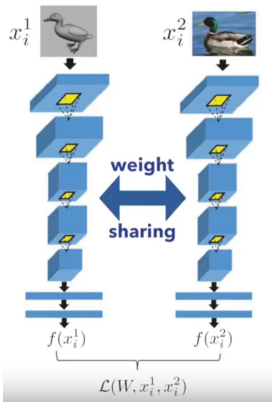


Fig. 2: Visual representation of the Siamese Architecture


Fig. 3: Batch of the validation result (left) and the test result (right). To have an immediate visual understanding of the results we printed the index of the classes, the distance computed by the net, the label assigned and the prediction obtained. As we can see pair images of the same class ($label = 0$) have distances smaller that pair of images of different classes.

Fig. 4: Templates extracted from the target images. Each of these templates are extracted from one of the target images. Then each of these templates are matched with all the patches extracted from all the target images. If the distance between a template and a patch is below the threshold, then that patch is associated to the same class of the template matched.

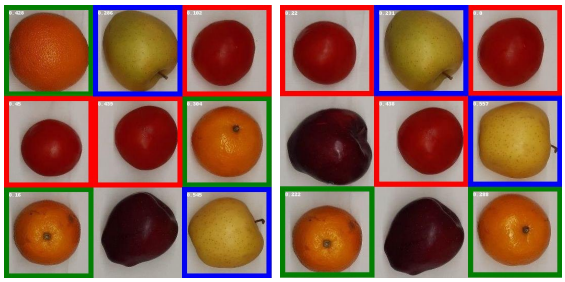

Fig. 5: In order to test the matching we picked some templates (Figure 4) randomly from one of the target images that we shoot. Once that the template was extracted, we extracted different patches from each of the target images using a simple sliding window approach. In the end to test the predicted class of each patch we measured the distance between each patch extracted and the template and we assign each patch to the closer template class. This is an example of positive result achieved. As we can see we have that all the matches are found correctly. A red bounding box surrounds each of the tomatoes, a blue one each of the yellow apples and a green one each of the oranges.

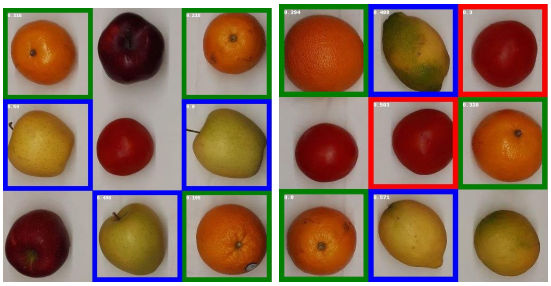

Fig. 6: In this example we got templates that are miss classified. In the image on the right we have two lemons classified as yellow apples and we miss the classification of a tomato, while on the image on the left we miss again a tomato but the remaining of the predictions are correct.

## REFERENCES


[1] Fruits-360: A dataset of images containing fruits and vegetables, https://github.com/Horea94/Fruit-Images-Dataset

[2] Dimensionality Reduction by Learning an Invariant Mapping, Raia Hadsell, Sumit Chopra, Yann LeCun, CVPR 2006, http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf

[3] FaceNet: A Unified Embedding for Face Recognition and Clustering, Florian Schroff, Dmitry Kalenichenko, James Philbin, CVPR2015