SKU Recognition for retail environment

Bonci Stefano[1] and Galeazzi Margherita[2]

[1]Master's Degree students in Computer Science and Automation engineering, Università Politecnica delle Marche, Ancona, Italy

***Abstract*— The aim of this paper is to illustrate a possible way to solve a problem of image recognition on retail products.**

We trained a CNN based on MobileNetV3Large and we used triplet loss as loss function for the reasons explained in the article.

The results obtained are satisfying in both test sets used; the first one had more images and they were of good quality, the second one contained less images and difficult to recognize because of their low quality. Our project is available at the following GitHub repository.

I. Introduction

In this paper we present a possible way to implement image recognition on retail products. The aim of our project was to recognize the class 'product' given an input image; in particular we were interested in associating the SKU corresponding to each product given in input. The SKU is a unique code that identifies a specific item and it represents the label of the images in our dataset. However, differently from how it seems, this wasn't a classic task of classification; that because in this field we don't have a fixed number of classes, but they can increase over the time because of the introduction of new products. So, as explained in the next paragraphs, we want to express each image with an embedding vector with the following features:

• Two examples with the same label have their embeddings close together in the embedding space

• Two examples with different labels have their embeddings far away. To reach this scope we used as loss function the 'Triplet loss' and not the 'softmax cross entropy loss' that is commonly used in this type of tasks. Our work also consisted in finding a network that was efficient in terms of both accuracy and inference time, and we have found a good compromise in a CNN that is based on MobileNetV3 available in the Keras Application package.

II. State of the Art

There are few articles in the literature about image recognition on retail products with similar features with our task and a major part of them was based on an interesting paper about a Face recognition problem. This article of Schroff *et al.* [1] present a possible solution to our problem about the presence of a variable number of classes using an innovative network called FaceNet. As a matter of fact, in face recognition we have to understand if an input image contains a face which is related to a certain class or if it is completely new. FaceNet's article proposes the use of triplet loss as a possible way to solve this problem because it doesn't give as output a probability but embedding vectors with specific features that have been mentioned in the Introduction I. and that will be deepened in the paragraph *Triplet loss* A..

III. Methods

A. Triplet loss

This section explains how triplet loss works so it will be clear why we decided to use it instead of softmax cross entropy loss. The goal of the triplet loss, as said before, is to make sure that:

• Two examples with the same label have their embeddings close together in the embedding space

• Two examples with different labels have their embeddings far away.

So the only requirement is that given two positive examples of the same class and one negative example, the negative should be farther away than the positive by some margin.

This type of loss is so called because it is defined over triplets of embeddings:

• an *anchor*(a).

• a *positive*(p), that belongs to the same class of the anchor.

• a *negative*(n), that belongs to a different class than the anchor.

The loss of a triplet (a, p, n) is defined as:

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0) \qquad (1)$$

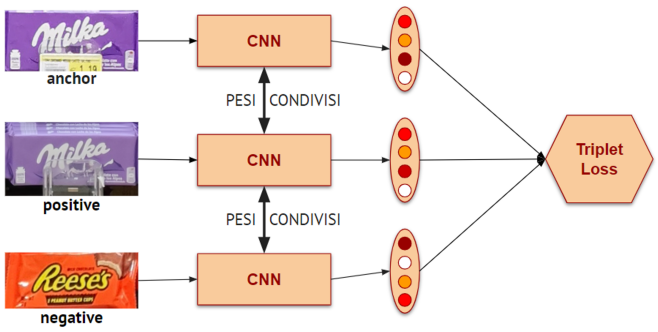

Fig. 1. anchor, positive and negative example

a.1. Triplet mining

Three types of triplets exist:

• *Easy triplets*: triplets where the loss is 0, because

$$d(a,p) + margin < d(a,n) \tag{2}$$

• *Hard triplets*: triplets where the negative is closer to the anchor than the positive because

$$d(a,n) < d(a,p) \tag{3}$$

• *Semi-Hard triplets*: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss because

$$d(a,p) < d(a,n) < d(a,p) + margin \tag{4}$$

This categories can be extended to the negatives, so we have: *hard negatives, semi-hard negatives or easy negatives*.

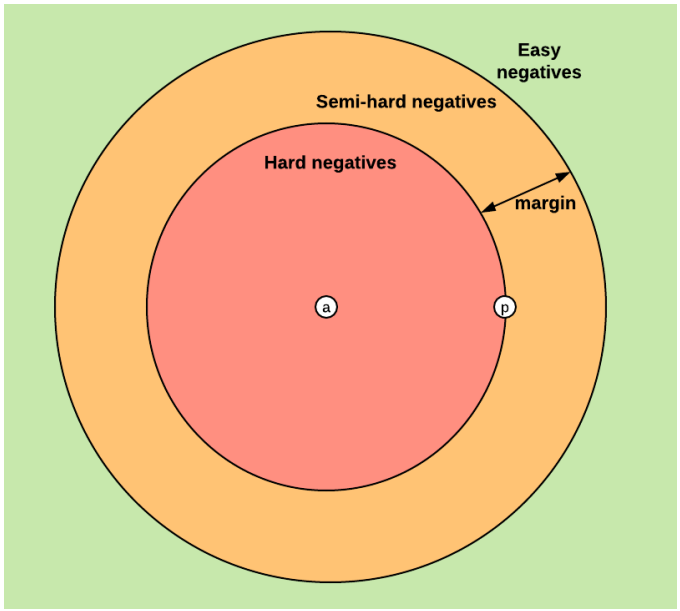

Fig. 2. The three types of negatives, given an anchor and a positive

There are also two different ways to 'mine' triplets, that are:

• *offline triplet mining*: generate triplets offline every n steps, using the most recent network checkpoint and computing the argmin and argmax on a subset of the data.

• *online triplet mining*: generate triplets online. This can be done by selecting the hard positive/negative exemplars from within a mini-batch.

A more complete discussion on triplet loss is available in the article in this link [2] and in the article of Hermans *et al.* called *In Defense of the Triplet Loss for Person Re-Identification* [3]

a.2. Implementation

In our case triplet loss has been implemented using a functionality from the repository *TensorFlow Addons*[4] called *tfa.losses.TripletHardLoss*. It computes triplet loss with hard negative and hard positive mining. The margin we have chosen is 1.0 and we set the parameter *soft=True*.

B. CNN

Regarding the network used to train the model we used the MobileNetV3Large [5] adding some layers. We have choosen this this network because it is one of the fastest available and, as said before, one of our aims is to reduce inference time as much as possible. MobileNet was pre-trained with imagenet dataset.

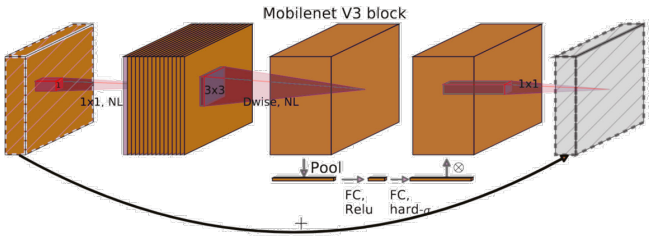

Fig. 3. MobileNetV3 Block Scheme

So the complete network consists on:

- Layers of MobileNetV3Large
- One *Flatten layer*
- One *Dense layer*

IV. Experimental protocol

A. Datasets

We trained our model using two different datasets. The first one had 32.212 images and the other 47.553 images. In both cases 80% of images has been used for training and 20% for validation. The models trained on the first dataset have been tested on a test set of about 1.000 images of good quality while the models trained on the second dataset have been tested on a test set of about 350 images with a very low quality.

B. Model training

We trained different models for each dataset using different values for the following parameters:

- *batch size*
- number of *epochs*
- *learning rate*

We also tried to use other networks like VGG16 and EfficientNet but in both cases we didn't obtain good results, specially for inference time.

The best results for the training set composed by 32.212 images were obtain with this configuration:

- *Triplet loss* as loss function
- *embedding size* = 256
- *batch size* = 64 and 128 (we obtain similar results)
- number of *epochs* = 50
- *learning rate* = 0.0001

The best results for the training set composed by 47.553 images were obtain with this configuration:

- *Triplet loss* as loss function
- *embedding size* = 256
- *batch size* = 128
- number of *epochs* = 50
- *learning rate* = 0.0001

C. Results

The results obtained for the first dataset using a batch size of 64 images are:

- *Accuracy* = 0.8703170028818443
- *Inference time on gallery** = 59.45407509803772 s
- *Inference time on test set* = 1.8961317539215088 s

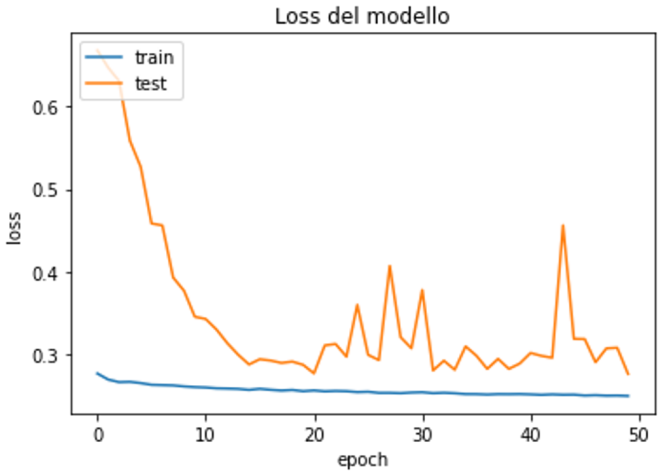

Fig. 4. Loss function graph

The results obtained for the first dataset using a batch size of 128 images are:

- *Accuracy* = 0.8837656099903939
- *Inference time on gallery** = 44.52806520462036 s
- *Inference time on test set* = 2.0112876892089844 s

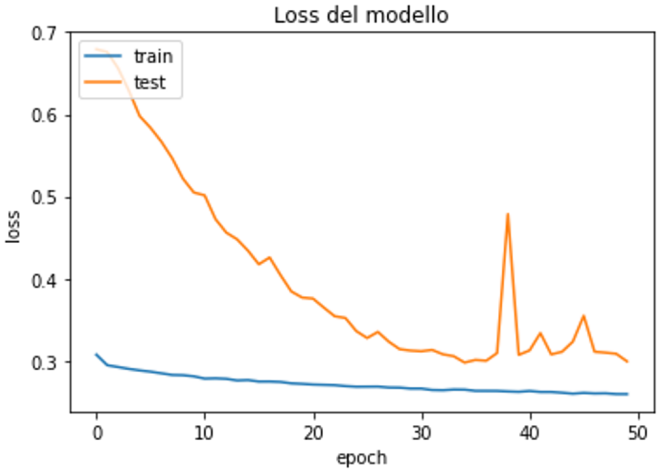

Fig. 5. Loss function graph

The results obtained for the second dataset are:

- *Accuracy* = 0.7759103641456583
- *Inference time on gallery** = 76.46771430969238 s
- *Inference time on test set* = 1.427288293838501 s

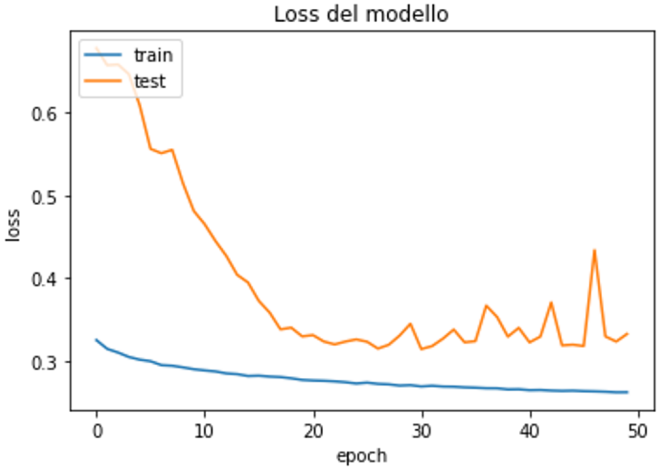

Fig. 6. Loss function graph

*Gallery is a dataset with the same images of the training set

V. Conclusions

In this paper we presented a method to implement image recognition on retail products when we don't have a fixed number of classes. We have achieved with good results our goal of training a model that performs satisfactorily in terms of accuracy and speed. From the results shown in the previous paragraph we can note that the best models are those trained with the first dataset. That's because, as we said, they have been

tested using a better test set with more images and with a better quality. Therefore also the third model can be considered efficient.

REFERENCES

[1] Florian Schroff, Dmitry Kalanichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: (2015). DOI: 10.1109/CVPR.2015.7298682.

[2] Olivier Moindrot. *Triplet Loss and Online Triplet Mining in TensorFlow*. 2018. URL: https://omoindrot.github.io/triplet-loss.

[3] Alexander Hermans, Lucas Beyer, and Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". In: (2017). DOI: 10.48550/ARXIV.1703.07737.

[4] *TensorFlow Addons*. https://github.com/tensorflow/addons. 2019.

[5] *MobileNetV3Large*. https://github.com/keras-team/keras/blob/v2.9.0/keras/applications/mobilenet_v3.py. 2021.