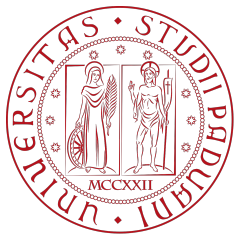
UNIVERSITÀ DEGLI STUDI DI PADOVA

Faculty of Engineering

Bachelor of Science in Computer Engineering

Tesi di Laurea Triennale

Automatic recognition of handwritten mathematical expressions



Advisor:
prof. Nanni Loris

Candidate:
Michelotto Federico

15 July 2019

In this thesis I present the project developed during the last months to create a machine learning system able to detect and translate in LaTeX handwritten mathematical expressions. This work takes inspiration from the mobilty Erasmus period that I made last semester going to San Sebastián, Spain, where I had the possibility to attend a machine learning class.

The dataset used in this project is provided by the 6th edition of the Competition on Recognition of Handwritten Mathematical Expressions (CROHME) made this year. I developed this project using the MATLAB environment and libraries. The final project will be released at https://github.com/federicomichelotto/CROHME

Contents

List of Figures

Chapter 1

Introduction

1.1 CROHME 2019 dataset

Handwritten formula data can be in two formats: online and offline. In the offline domain, data is simply an image. In the online domain instead, data is represented as a sequence of strokes, which are the (x,y) points recorded from a tablet or similar devices, and the strokes that belongs to the same symbol are grouped together. Note that, in the online domain, since you have a sequences of coordinates, you also know the evolution in time for each symbol. The format of the online data used in this competition is the InkML format, an XML-based markup language. An example of an online representation of a mathematical expression (ME) and the correspondent image are shown in Fig. 1.1 and Fig. 1.2.

To represent a ME in a concise way, it can be represented as a Symbol Relation Tree (SLT), where the nodes of the tree are the symbols in the ME, and the edges between the nodes describe the relationship between them. For each ME in the dataset, is provided a Label Graph (.lg) file that rapresents the corrisponding SLT, as shown in Fig. 1.3 and Fig. 1.4.

In the CROHME dataset there are 101 different symbols including digits, alphabet characters, operators and so on. Six spatial relationships are defined in the CROHME competition, they are: *Right*, *Above*, *Below*, *Inside*, *Superscript* and *Subscript*. The *Right* edge is used to describe the relationship in which the final node is at the right of the first one, that is, they are at the same level. The *Above* and *Below* edges are used when a fraction, a summation or a limit is present. The *Inside* edge is used to connect the root symbol to the symbol with the highest precedence inside the root . The *Subscript* and *Superscript* are used other than in the nominal case also when is present an integral symbol and you want denote the upper and lower limits of it.

The dataset contains 9993 handwritten MEs for the training stage, 986 for the validation and 1199 for the test. In addiction to the complete expressions, the competition provides also a large number of individual symbols, in this set there are also 'junk' symbols, that simply are a collection of strokes that do not represent any symbol. More precisely, there are 180440 (82150 junk) symbols in the training set, 18435 (8416 junk) symbols in

the validation set and 15483 symbols in the test set. About the isoleted symbols, it can be seen that the dataset is highly unbalanced, in fact, without considering the junk symbols, for each class there is an average of 973 instances, but 36 out of 101 classes have less than 200 instances.

```
<trace id="0">
11.6603 15.4164, 11.6643 15.4004, 11.6683 15.3883, 11.6643
   15.3723, 11.6563 15.3603, 11.6362 15.3522, 11.5841 15.3402,
   11.5239 15.3562, 11.4677 15.3763, 11.4356 15.4124, 11.4356
   15.4285, 11.4517 15.4525, 11.4717 15.4606, 11.5038 15.4806,
   11.5359 15.4766, 11.5801 15.4927, 11.6202 15.5208, 11.6443
   15.5689, 11.6443 15.593, 11.6322 15.609, 11.5801 15.6171,
   11.5199 15.609, 11.4677 15.593, 11.4196 15.5809, 11.4035 15.5689
</trace>
<trace id="1">
11.8248 15.7174, 11.8128 15.7174, 11.7887 15.7214, 11.7165
   15.7214, 11.6001 15.7294, 11.4597 15.7254, 11.3192 15.7294,
   11.231 15.7134, 11.1949 15.7053, 11.1828 15.6973, 11.1949
   15.6973
</trace>
<trace id="2">
11.5038 16.0023, 11.5199 15.9862, 11.5279 15.9742, 11.5199
   15.9461, 11.4998 15.93, 11.4717 15.918, 11.4155 15.918, 11.3554
   15.9421, 11.2912 15.9982, 11.2189 16.0785, 11.2069 16.1828,
   11.2631 16.2631, 11.3513 16.3152, 11.4276 16.3233, 11.4717
   16.3032, 11.4878 16.2671, 11.4717 16.2149, 11.4155 16.1628,
   11.3353 16.1387, 11.239 16.1106
</trace>
```

Figure 1.1: Strokes of an online ME.

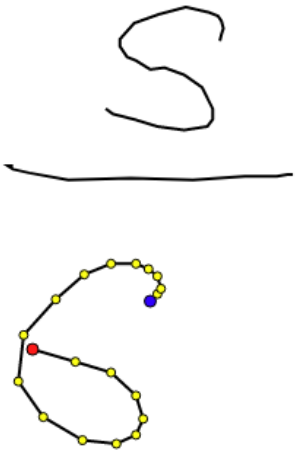


Figure 1.2: Reconstruction of the image from the original online data in (Fig. 1.1).

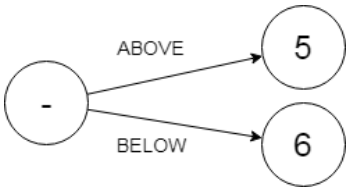


Figure 1.3: SLT of $\frac{5}{6}$

```
# Nodes:
N, 0, 5, 1.0
N, 1, -, 1.0
N, 2, 6, 1.0

# Edges:
E, 1, 0, A, 1.0
E, 1, 2, B, 1.0
```

Figure 1.4: Label graph representation of $\frac{5}{6}$; A stands for 'above' and B for 'below'.

1.2 Tasks

The CROHME competition has three main tasks:

- Task 1. Online Handwritten Formula Recognition
- Task 2. Offline Handwritten Formula Recognition
- Task 3. Detection of Formulas in Document Pages

In The first two tasks participating systems must produce one .lg file for every test ME, and the results are evaluated based on the number of correctly recognized formulas (expression rate). Since primitive level information (connected components, i.e. strokes) is not provided in task 2, the evaluation is based just on the correct symbol classification and correct relationships between the symbols, while in task 1 evaluation is based also on the correct correspondence of the connected components to symbols.

In task 3, for every document page, the participating systems must return the bounding boxes corresponding to the location of the formulas in the document. The evaluation is based on the intersection over union (IoU) of the regions proposed with the real regions. I decided to focus on the offline recognition of handwritten MEs, that is task 2. In the next section I will describe the problem and how I thought to proceed.

1.3 The problem

From a naive point of view the problem of recognize handwritten formula can be splitted in three parts:

1. Develop a system able to detect the regions that contains one and only one symbol.

2. Pass the regions found earlier through a classifier to determine which symbol corresponds to that region.

3. Develop a system able to classify the right relationship between two symbols, if any, and for each image deploy the corresponding SLT.

The main bottleneck of this procedure is the first stage, in fact if the detection returns wrong regions, inevitably, the job of the classifier will be compromised.

At the beginning of this project I didn't put much attention on this problematic, also because I had no idea on how I could do it differently, so I started working on the classification task. Relative to the first stage, I thought to use a sliding window algorithm, i.e. an algorithm that slides the image with a "window" of different size and ratio at every iteration and classify each region to detect if that region is a symbol or not.

As you can imagine a system like this, is not very efficient, in fact we can say that it is a brute force algorithm. But the main problem of this system is not the efficiency, since we are not talking about a real-time system, but I will discuss this later. (symbols with disconnected components!!!)

The competition put some constraints about the resolutions of the images rendered from the online data. They provided a script that automatically do this operation and the default values are 1000x1000 pixels for equations with 5 pixels padding and 28x28 pixels resolution for isolated symbols with the same padding. Obviously these parameters could be modified, and would be interesting to see how the classification would be affected reducing or increasing the size of the image.

1.3.1 Initial results

I will describe here the initial results regarding the classification task of isolated symbols. Since 2012 convolutional neural networks (CNN or ConvNet) have been widely used for image classification tasks, so I decided to start training a ConvNet with a layout similar to the one used in the VGGNet [8].

```
convolution2dLayer(3,32,'Padding','same')
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)
convolution2dLayer(3,64,'Padding','same')
convolution2dLayer(3,64,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)
convolution2dLayer(3,128,'Padding','same')
convolution2dLayer(3,128,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(1000)
fullyConnectedLayer(1000)
fullyConnectedLayer(1000)
fullyConnectedLayer(101)
softmaxLayer
```

Figure 1.5: ConvNet layout used.

The classifier was trained on 70000 images for 30 epochs with a batch size of 256. The ConvNet has been tested on a validation set of 15000 images picked randomly reaching an accuracy 92,55%. After having tried different layout with similar result, I used a technique called transfer learning [5] [10], using AlexNet as a pretreined network, and training the last three layers of the network on a training set of 20000 images for 30 epochs, with a batch size of 30 and a learning rate of 0.0001, obtaining an accuracy of 91%. These numbers are in line with the results presented in the last CROHME competition in 2006 [4].

Since the dataset is highly unbalanced I applied some image processing techniques to get an offline data augmentation, in order to increase the number of instances of the classes with a low number of samples (less than 1000). I applied geometric affine transformations, rotation and contrast with parameters selected randomly from a set of pretested values. An example of the result of this procedure is shown in Fig. 1.6. Then, I retrained the classifier with the augmented dataset, but in this case the accuracy was 88%. Since time constraints I wasn't able to test this procedure properly and further evaluation are then required.

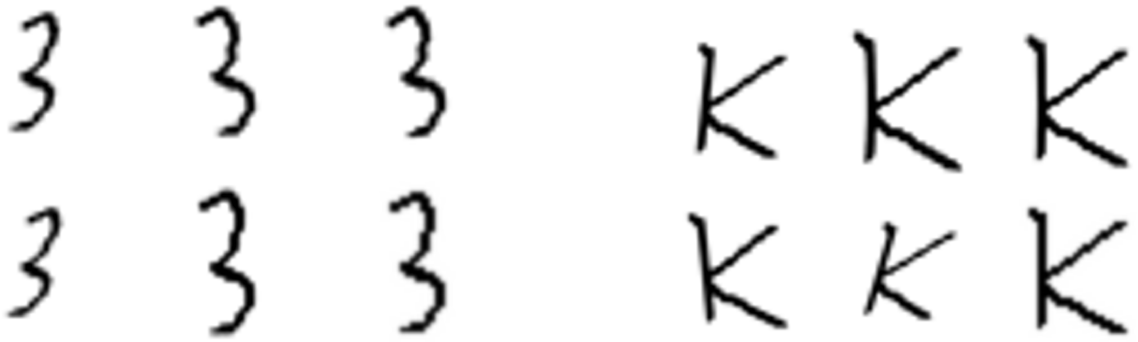


Figure 1.6: Five different image processing techniques applied randomly to the original symbol at the bottom right.

In the meanwhile, I started looking for a model more efficient for the detection of symbols in an image. A big constraint of the sliding window algorithm for this problem, is also due to the fact that it happens frequently to find symbols not written very well and whose components are not connected, and this would lead the detection model to return to many wrong detection.

Another issue about this approach is related to the root symbol, this because the isolated symbols that belongs to the root class, are with just the root itself, so without any argument inside, but, instead, in the complete formulas, the roots have the argument inside. So what we except, is a weak ability to detect root symbols in complete formulas. About this question, an in depth analysis should be done to a complete understanding.

During my research, I found a family of novel techniques that revolutionized the field of object detection. These techniques are based on convolutional neural networks, and since they are used to return a set of regions, they are called Region-CNN (R-CNN). In the next chapter I will discuss the foundation of each of these models.

Chapter 2

Region CNN

2.1 (Slow) R-CNN

The motivation behind the R-CNN proposed by Girshick et al. [2] was to consider the regions with some relevance, and to exclude the others instead to classify all the regions provided by a brute force sliding window algorithm. The detection system presented in the paper consists of four modules. The first generates category independent region proposals through an algorithm called "Selective Search" that uses computer vision techniques to provide a fixed set (~2000) of region proposals. These proposals define the set of region candidates that the detector will later evaluate. The second module is a ConvNet, used to extract a fixed-length feature vector from each region, for example Girshick et al. have used AlexNet as a pretrained ConvNet and then fine-tune the network on PASCAL VOC detection data (20 object classes, and 1 background class). The third module consists of a classifier that evaluate the feature vector produced in the previous stage. For example, in the paper was used a specific SVM for each class trained independently. The last module is a bounding box regressor (one for each class), that given a feature vector predict the correction relative to the region proposed initially.

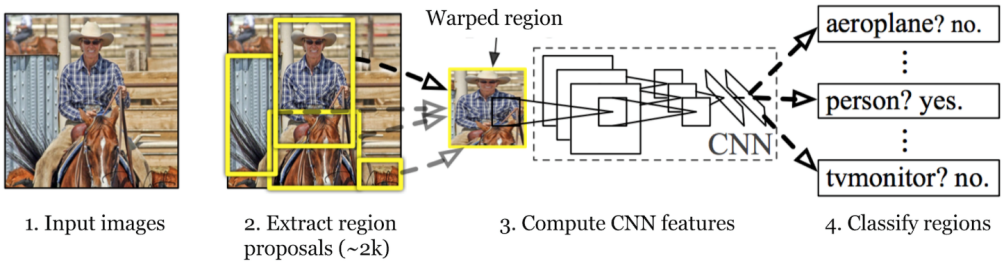


Figure 2.1: rcnn

Problems of this solution:

- Slow at test-time: need to run a full forward pass of CNN for each region proposal.

- Region candidates have to be converted to make them compatible with the input size of the ConvNet (227x227x3 for AlexNet).
- Region features (extracted from all the region proposals for each image) used for training occupy a lot of space: ~200GB for PASCAL dataset.

2.2 Overfeat

The main limitation of the R-CNN is that for every region proposal a full forward pass is executed, and for this reason, test a new image is very slow. The paper published in 2014 by Sermanet et al. [7] introduces for the first time a clear explanation of how and why ConvNets can be used efficiently to implement both a multiscale and a sliding window network. Convolution implementation of sliding window Consider the following CNN, and a fixed window of 14x14, the image below show how there is no need to compute the forward pass for every window, but it's enough to compute the forward pass only once on the entire image. The size of the output in this case is equal to 4, because 4 windows fits in the input image.

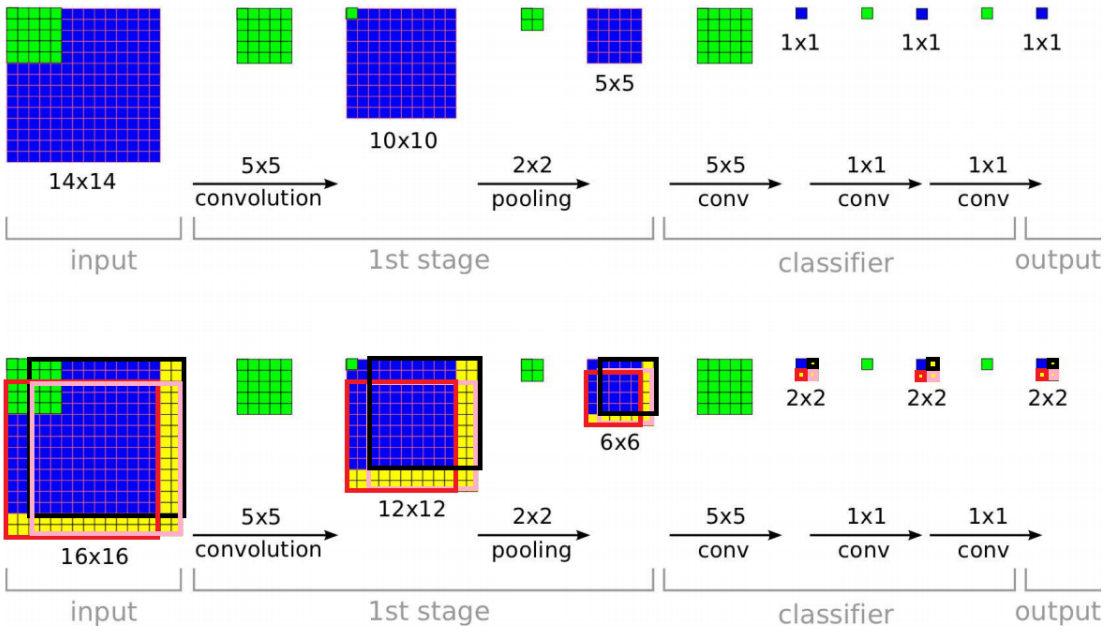


Figure 2.2: The efficiency of ConvNets for detection. [7]

Quoting from [7] (*3.5 ConvNets and Sliding Window Efficiency*):

> In contrast to many sliding-window approaches that compute an entire pipeline for each window of the input one at a time, ConvNets are inherently efficient when applied in a sliding fashion because they naturally share computations

common to overlapping regions. When applying our network to larger images at test time, we simply apply each convolution over the extent of the full image. This extends the output of each layer to cover the new image size, eventually producing a map of output class predictions, with one spatial location for each "window" (field of view) of input.

Note that the last layers of our architecture are fully connected linear layers. At test time, these layers are effectively replaced by convolution operations with kernels of 1x1 spatial extent. The entire ConvNet is then simply a sequence of convolutions, max-pooling and thresholding operations exclusively.

2.3 Fast R-CNN

Fast R-CNN has been published by Girshick et .al in 2015 as an evolution of the first version of the R-CNN. This version fixes the main bottleneck of the previous one, i.e. one pass forward for each proposal, exploiting the properties of the ConvNets shown in the overleaf paper.

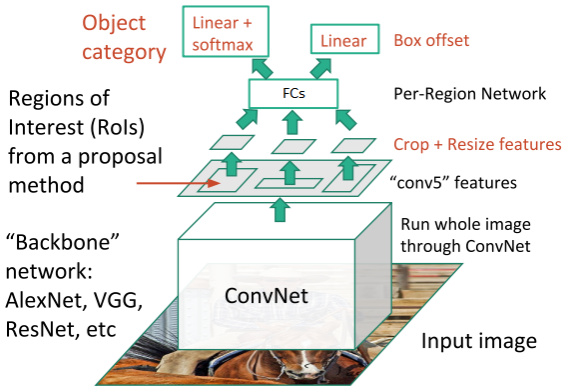


Figure 2.3: Fast R-CNN.

The model is represented in figure, as can be seen, now as first step the whole image is passed through the ConvNet that produce a convolution feature map. Note that since this ConvNet has just convolutional and pooling layers, the size of the input image is not constrained. As in the R-CNN model, a region proposal algorithm is used to define the candidate regions, but in this case, the regions get projected to the feature map. But, since the next layer is a fully connected layer, this mean that the feature vector relative to each region must have a fixed size (e.g., h = w = 7 for VGG16). To do so, a layer called region of interest (RoI) pooling layer has been described. This layer will divide the projected region into h x w grid and selecting the highest value in each cell, producing a feature vector of dimensions C x h x w. Each feature vector is then fed into a sequence of fully connected layers that finally branch into two output layers: one that produces softmax

probability over K object classes plus "background" class and another layer that outputs four real-valued numbers for each of the K object classes.

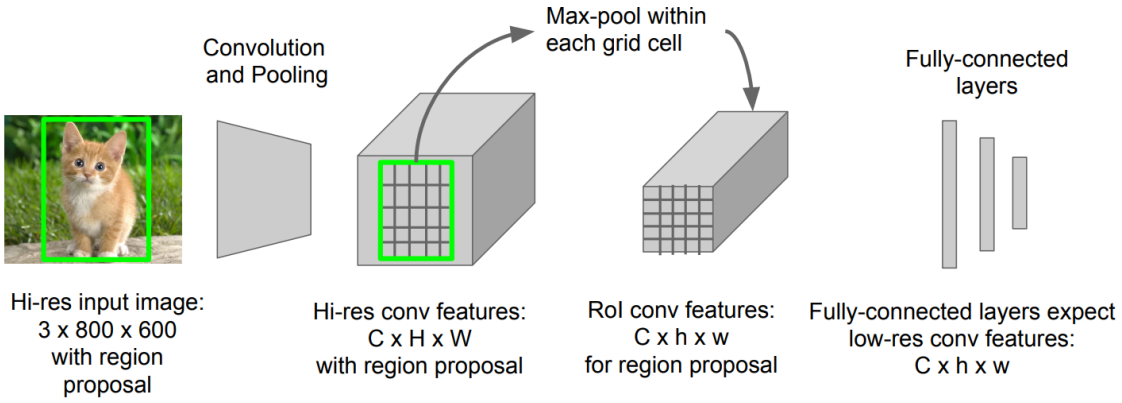


Figure 2.4: Roi pooling layer.

2.4 Faster R-CNN

Faster R-CNN has been developed with the goal to reduce the time of the region proposal stage, since it was the bottleneck for the Fast R-CNN system. In Fast R-CNN the region proposals are generated by a segmentation algorithm called selective search. In Faster R-CNN has been inserted a region proposal network (RPN) to substitute the external proposal algorithm used in the previously versions. The RPN predicts regions from feature map. Then is the same as Fast R-CNN.

The test time speed comparisons of the systems are shown in Fig. 2.6

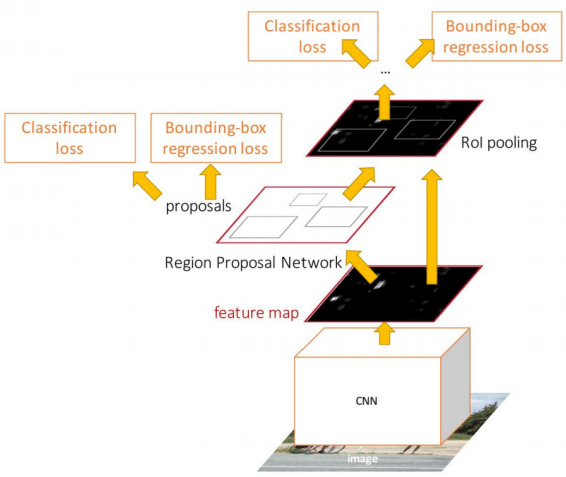


Figure 2.5: Faster R-CNN.

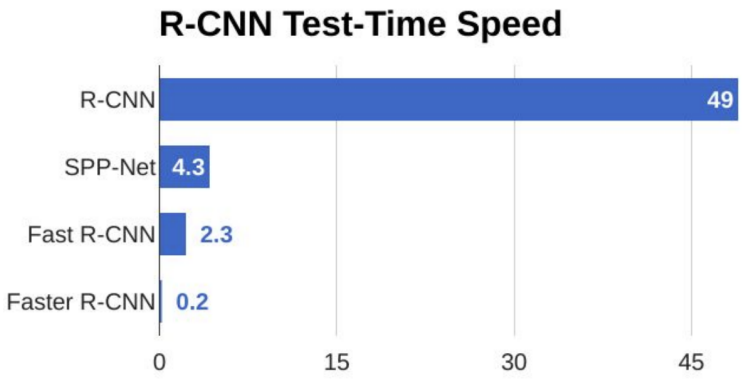


Figure 2.6: Test time speed comparison.

Chapter 3

Development

For my project, I decided to use the most flexible and fastest R-CNN solution, i.e. the Faster R-CNN model. Since, all the three R-CNN proposals are already implemented in MATLAB, I didn't had to worry about it, so the next step requires to work on the extraction of the ground-truth bounding boxes and the corresponding labels of all the individual symbols present in the training formulas.

3.1 Symbol extraction

To do the symbol extraction I used online data informations provided for each image, to reconstruct all the bounding boxes corresponding to each symbol.

For this purpose I created a function "extract_info.m" that do the parsing of every InkML file, and extract these regions. The main obstacle is related to the fact that, as you can note in Fig. 1.1, the coordinates values are not absolute. Since the poor documentation provided by the competition I had to do some reverse engineering on the scripts they provided, to understand the underlying process on how they reconstruct images from online data.

An example of the final result of this procedure is shown below in Fig. 3.1.

3.2 Training

The training stage was made using ResNet-50 [3] as pre-trained convolution neural network. An execution of the trained detector on a test image is shown in Fig. 3.2. Since time constraints I couldn't do an in depth analays to test the real accuracy of the detector, this will be leaved for future works.

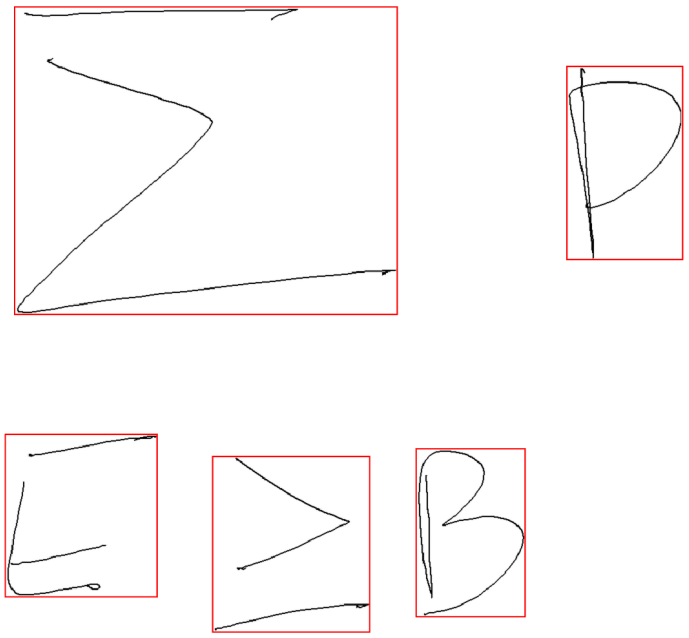


Figure 3.1: Ground-truth bounding boxes.

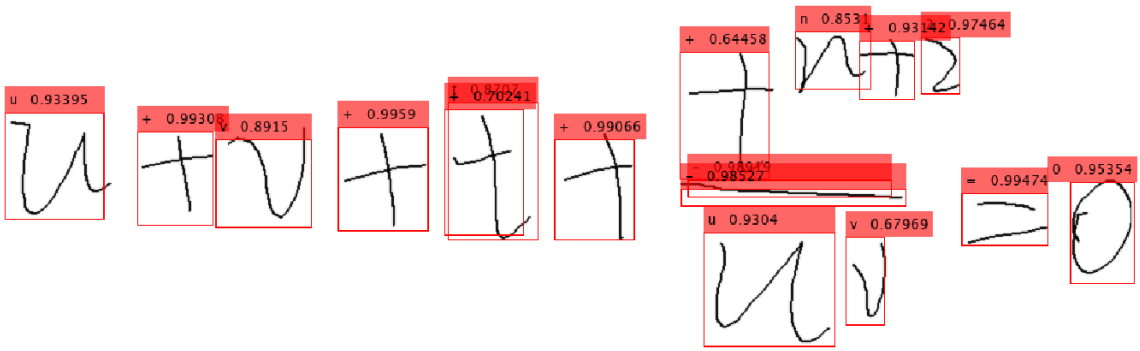


Figure 3.2: Detector on a test image.

3.3 LaTeX translation

The last stage of this project consists to do the translation of the symbols detected. As a first approach I decided to develop an heuristic method based on a recursion algorithm. The method consists to parse the symbols detected from left to right, giving precedence to the other edge relationships (above, below, subscript, superscript, inside).

The recursive algorithm stops when there are no others right symbols to parse, this condition is limited by a fixed distance between the symbols when the symbol in question is not at level 0 of the expression. Other than this, when a fraction is found, the field of view of the calling symbol if restricted to "see" just up a to the level of the fraction symbol if it is below for how concerns the y axis, or vice versa if the symbol is above. The field of view can also be fixed for the x axis, in this case the restriction is based on the length of the fraction symbol. Analog consideration has been made for the square root symbol.

An example of this translation algorithm for the image in Fig. 3.3 is shown below:

$$q^2 = \sum_{a=}^{n} (x^a)^2$$

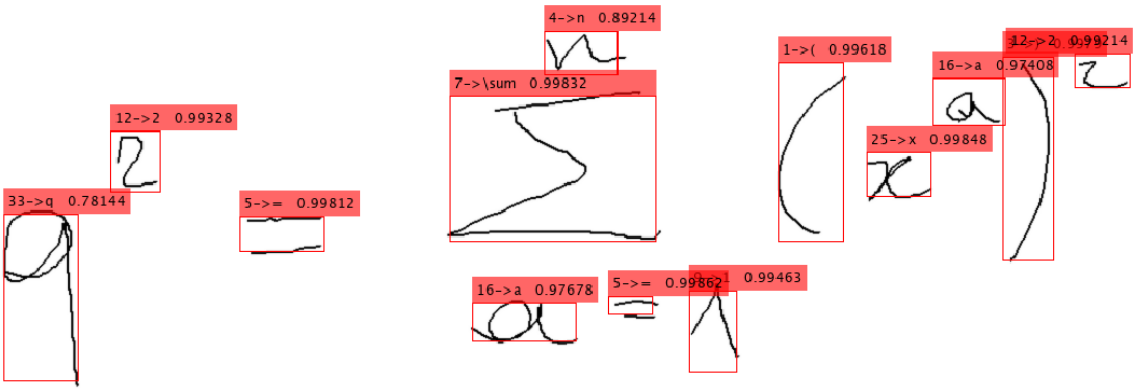


Figure 3.3: Detector test 2.

Chapter 4

Conclusion

Thanks to the big progress in the machine learning field in the last few years, I was able to build an automatic system to recognize handwritten mathematical expressions and visualize them in LaTeX . The efficacy of the system is still under review, and a further in depth analysis should be done. Minor bugs has to be fixed in the translation process, and major improvements can be made. For future works more complex and flexible model could be adopted, for example working with the deep learning model called visual Attention to get a more precise translation of the detected symbols.

Bibliography

[1] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[2] Ross B. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[3] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[4] H. Mouchère et al. "ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions". In: (2016), pp. 607–612. ISSN: 2167-6445. DOI: 10.1109/ICFHR.2016.0116.

[5] Ali Sharif Razavian et al. "CNN Features off-the-shelf: an Astounding Baseline for Recognition". In: *CoRR* abs/1403.6382 (2014). arXiv: 1403.6382. URL: http://arxiv.org/abs/1403.6382.

[6] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[7] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *International Conference on Learning Representations (ICLR) (Banff)* (Dec. 2013). URL: https://arxiv.org/abs/1312.6229.

[8] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014). URL: https://arxiv.org/abs/1409.1556.

[9] J. R. R. Uijlings et al. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171. ISSN: 1573-1405. DOI: 10.1007/s11263-013-0620-5. URL: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[10] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: (2014). Ed. by Z. Ghahramani et al., pp. 3320–3328. URL: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

[11] Jianshu Zhang et al. "Watch, Attend and Parse: An End-to-end Neural Network Based Approach to Handwritten Mathematical Expression Recognition". In: *Pattern Recognition* 71 (June 2017). DOI: 10.1016/j.patcog.2017.06.017.

[12] Ting Zhang. "New Architectures for Handwritten Mathematical Expressions Recognition". Theses. Université de nantes, Oct. 2017. URL: https://hal.archives-ouvertes.fr/tel-01754478.