CenterNet applied to multi-digit detection based on the MNIST Detection dataset

Moustapha Diop
moustapha.diop@epita.fr

Mathieu Rivier
mathieu.rivier@epita.fr

Martin Poulard
martin.poulard@epita.fr

Gregoire Gally
gregoire.gally@epita.fr

January 2023

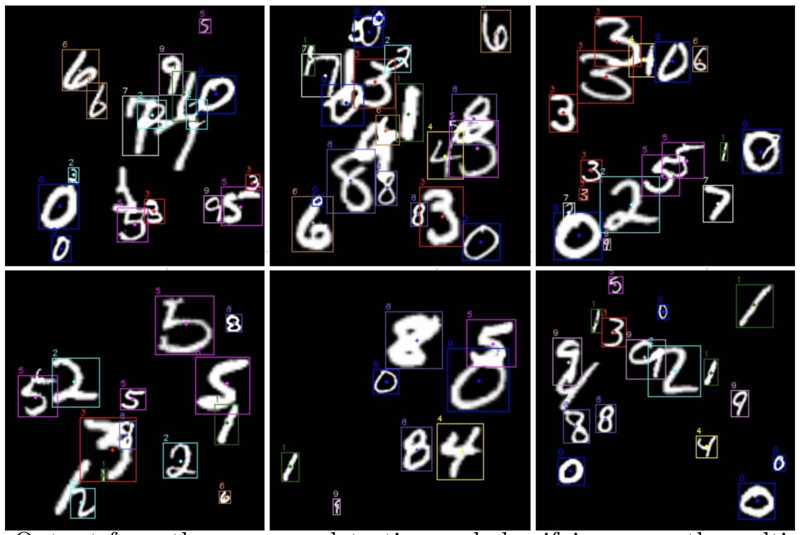

Figure 1: Output from the program detecting and classifying correctly multiple digits.

Abstract

This paper is a re-implementatin of the popular CenterNet paper based itself on CornerNet. It focuses more specifically on applying CenterNet to the MNIST multi-digit dataset. This research has enabled to provide great results with average accuracy reaching 85% on test sets. The code is available at: https://github.com/yxyfer/DNN-centerNet

Contents

0.1 Introduction 1

1 CenterNet for multi-digit detection 1

1.1 What is CenterNet 1

2 Re-implementing CenterNet 1

2.1 Issues, Problems & solutions 1

2.2 Model Performance 2

2.2.1 Implemented Metrics 2

2.2.2 Results 3

3 Understanding and showcasing CenterNet's shortcomings 3

3.1 Creating datasets that are complex for CenterNet 3

3.2 Exposing CenterNet's limitations with regards to partly concealed digits 4

0.1 Introduction

1 CenterNet for multi-digit detection

1.1 What is CenterNet

CenterNet is an improvement over CornerNet which aims at detecting objects as a triplet of key point rather than a pair like in CornerNet. CenterNet detects object by locating the top-left and bottom-right points but further detects the object's central point. This methods enables a better localisation of objects within an image and better detection precision in comparison to CornerNet. Centernet is a one staged detector, just like CornerNet and YOLO. It means they detect and classify objects all at once.

CenterNet's architecture is based on a backbone: generally a ResNet or Hourglass network. Although, Hourglass implementations seem to consistently yield better results in terms of detection precision. Two model specific architectures have been introduced: Cascade Corner Pooling and Center Pooling. The Cascade Corner Pooling comes in first enabling the computation of heatmaps, embeddings and offsets for the top-left and bottom-right corners. The Center Pooling then enables to compute the heatmap and the offset for the center point. More specifically the heatmaps represent the location of keypoints, the embeddings enable checking whether two corners belong to the same object and the offsets enable to remap the heatmaps keypoints to the entry image. By combining the information from the corner keypoints and center keypoint, CenterNet can accurately detect and classify objects.

To detect objects, CenterNet initially uses a method that consists in creating K-pairs corners. Those pairs are then compared to center points in order to assess their relevance in object detection. If a center point corresponds to the center of a bounding box, it signifies that an object is contained at this place and the bounding box is kept. Otherwise, it is deleted. Using a center point to validate corner pairs enables CenterNet to perform considerably better in detection precision than CornerNet

CenterNet has been used as base in this paper, although, modifications have been added to adapt it to the Mnist detection dataset. To do this, 300x300 images have been used and 75x75 feature maps. Hourglass was purposely not used as backbone just like it isn't in the original version of CenterNet, a smaller pre-trained Residual Network, trained on Mnist data was rather used.

2 Re-implementing CenterNet

This sections is composed of two main parts. The first section will be structured as a back and forth between problems that have encountered and the solution implemented to solve them. The following section will focus on the metrics and obtained results with the model.

2.1 Issues, Problems & solutions

As in with every re-implementation task, the first step is to make the smallest possible working program. To do this the first step is to read the paper multiple times, and get familiar with the original code base. This was mostly a straightforward process, except for a few key terms had different names between the paper and implementation, which slowed down the initial understanding. Such key terms include (Paper–Implementation): embeddings — tags ; regrs — offset... Another hurdle that slowed down the initial process is the lack of documentation/docstrings. This is why this implementation is explained in depth and documented. Furthermore, some functions are questionable! Simple examples include functions that have unused parameters, or different functions that simply instanciate the same class multiple times. Such examples can be found at the following links: example.1, example.2. The main problem is that those re-directions make it harder to understand the link and effect of different parts of the program. Making it ultimately harder to fully grasp.

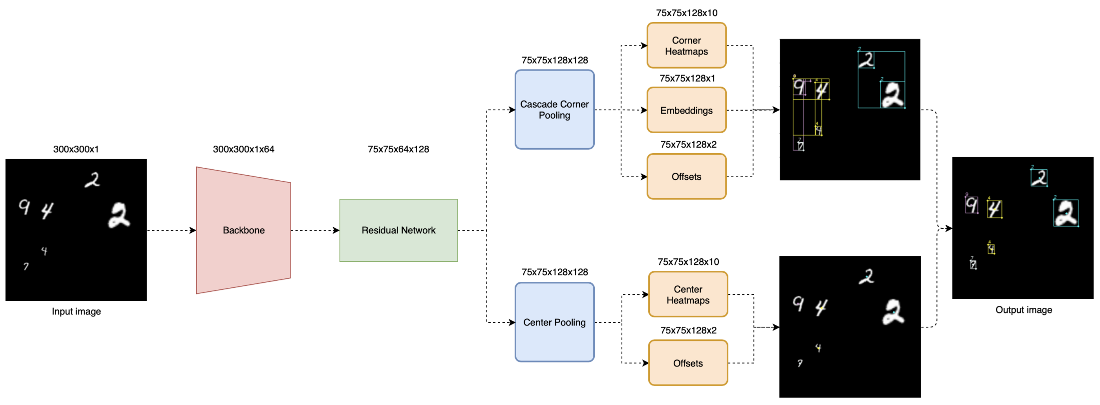

Figure 2: Final Implementation Architecture

Although, those remain rather small problems, some of the other issues that have been encountered made the initial implementation performance lacking. In fact, the paper's architecture Figure 5 in the initial paper, details a straightforward Center Pooling Module followed by a Cascade Top Corner Pooling Module, although the results with first implementations where very limited and did not fit with the expectations of a CenterNet implementation. More attention was thus given to the implementation of the architecture in the paper's code, it has revealed that the actual implementation two Center Poolings that are then merged and two Cascade Top Corner Poolings that are also then merged. This architecture has yielded much more convincing results and is the one implemented in this paper. The final architecture implemented in this paper is the one in figure 2.

Another time consuming issue in reimplementing CenterNet was the Loss function. Indeed while the paper defines the loss, it is not explained at all, in order to understand it reading the CornerNet paper is necessary. Furthermore, the Loss in the CenterNet paper is not exactly the one mentionned in CornerNet nor is it exactly the one implemented.

2.2 Model Performance

This section is divided into two subsections: the first one explaining which metrics were used to gather performance of the model and the second to study the normal case performance of the final model.

2.2.1 Implemented Metrics

In order to understand the results that were achieved in multi-digit detection with this model, it is first interesting to understand how the performance was registered.

Throughout the implementation of this project, the performance of this implementation of CenterNet was performed using multiple metrics. The first metric is the Average Accuracy (refered to as AA), which measures the average digit-detection accuracy. Four different AA measures are used in this paper: AA5, AA50, AA75 and AA95, which corresponds to only keeping the detections that have IoUs (Intersection over Union) that are superior to the corresponding threshold.

The second metric used in this paper is False Discovery (refered to as FD), The implementation in this paper is slightly different to the one used in the original paper in which $FD = 1-AP$. In this paper it enables the measure of the percentage of bounding boxes (bbox) that are incorrectly detected. This allows for a better estimation of the total number of wrongly generated bboxes by the model.

The average IoU (refered to as mIoU) is the final metric used in this paper. It provides an estimation of the proximity of the bounding boxes generated by the model compared to the bounding boxes provided by the ground truth (labels).

2.2.2 Results

This section explores the results that were obtained using the model on training and testing data. Those two dataset were well sized with seven-hundred labeled images and two-hundred labeled images respectively.

While a direct comparison with the initial model is hard since the metrics that are implemented in this paper are quite different from the initial paper, it is possible to make observations on the general quality of the model.

	Train	**Test**
mIoU	0.9129	0.8568
AA05	99.16%	85.09%
AA50	98.75%	82.34%
AA75	96.79%	78.56%
AA95	31.29%	19.63%

Table 1: Average IoU (mIoU) and Average Accuracy (AA) table on both training and testing sets

As can be seen from table 1, the results from the model are very satisfactory. An average IoU at 90% translates a very good detection of the bounding boxes. It is important to further note that this score includes the non-detected bounding boxes. The average Accuracy with a confidence of 5% is around 85% on the test set which lies right in as a great model. Tests on other datasets has confirmed that precision. This attests to the quality and usability of the model.

	Train	**Test**
FD05	0.48%	4.88%
FD50	1.71%	8.17%
FD75	3.76%	12.65%
FD95	68.90%	78.10%

Table 2: False Detection (FD) table on both training and testing sets

The False Detection rate in table 2 is also remarkable. With about 5% false detection rate at the 5% threshold, it is reasonable to say that the model has a very low error rate.

These results amount to say that the model is very usable and has high enough results to make it a releavant method for detecting multi-digits.

3 Understanding and showcasing CenterNet's shortcomings

This section is divided into two distinct parts. The exploration and creation of complex datasets. And the results that prove initial hypothesis. Our main hypothesis is that CenterNet performs worse when encountering occultation. Two datasets where created, The idea being that the one with more digits within the same size frame, would lead to more overlap and thus lower results. This Hypothesis held true in the testing performed on the two datasets. We will explain in more details those result throughout this section.

3.1 Creating datasets that are complex for CenterNet

In order to understand CenterNet's limitations specific datasets containing drawbacks had to be created. Ideas as to what might cause CenterNet came to mind and where tested. Such options include: having many digits in the same image, turning them sideways, or even strange writings. Although, the number one problem CenterNet encountered was concealing even small parts of a digit. Two datasets where created, one with 15 digits max per image, and another one with 30 digits max per image. The idea being that the one with more digits within the same size frame, would lead to more overlap and thus lower results.

This Hypothesis held true in the testing performed on the two datasets.

3.2 Exposing CenterNet's limitations with regards to partly concealed digits

The original hypothesis was that with more partly concealed digits, which would happen naturally with more digits in the same frame, the scores would fall.

	15 images	**30 images**
mIoU	0.856	0.828
$AA_{0.05}$	0.825	0.797
$\mathrm{FD}_{0.05}$	0.038	0.045
$\mathrm{AA}_{0.5}$	0.813	0.762
$\mathrm{FD}_{0.5}$	0.055	0.086
$\mathrm{AA}_{0.75}$	0.78	0.704
$\mathrm{FD}_{0.75}$	0.091	0.157
$\mathrm{AA}_{0.95}$	0.186	0.14
$\mathrm{FD}_{0.95}$	0.782	0.834

Table 3: Average Accuracy and False Detection for 15 and 30 image datasets studying overlap

In table 3 this hypothesis holds true, as lower considerably lower scores can be observed in all metrics that is AA, FD and mIoU. The fact that both dataset were catered towards favouring concealed digits proves the initial hypothesis.

such images look like figure 3 for 30 digit max images:

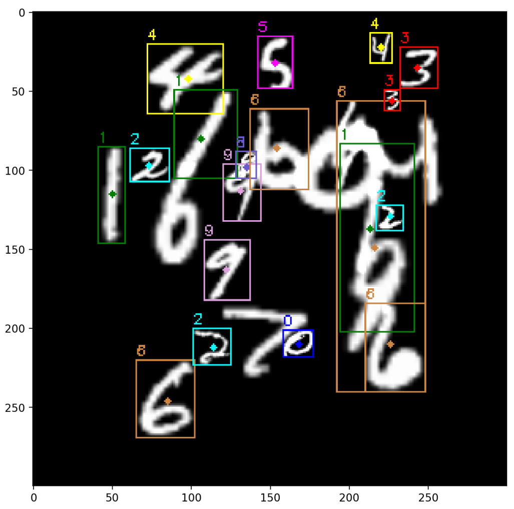

Figure 3: Output from the program detecting and classifying multi-digits overlap.

It is clear from figure 3 that some digits have not only not being correctly classified but not even detected. Those include amongst other: 6, 7, 1, 0. All the digits that have uncorectly been classified or not detected are at least partly concealed by other digits. In our experience, even slight conceal of a digit may in some cases be enough for a digit not to be detected.