COCOONO: Crowdsourced Image dataset to test and train computer vision Frameworks more effectively

Mateus Zitelli, Stephen Gould
Australian National University
Canberra, ACT, Australia
Email: {u5767001, stephen.gould}@anu.edu.au

May 28, 2016

Abstract

Despite the current advancements in image captioning libraries, the existent solutions still fail in simple identification tasks. The captioning libraries normally generate captions based on the scene as a whole, not identifying individual objects, failing on scenes slight different from the scenes used for training. This report documents and evaluates the creation of an alternative dataset to address this problem. Our dataset was generated based on the COCO dataset from Microsoft. We generated alternative versions of many images from COCO by removing specific objects from the images using a graph-based inpainting algorithm from Kaiming and Jian’s [1]. However, automatic inpainting, as any automatic edition methods, could generate erratic results. To address this issue an web interface for crowdsourced image evaluation and edition was developed to guarantee the final dataset quality. With a preliminary data acquisition, we showed that our dataset could be useful in the evaluation of computer vision frameworks. We also analyzed the impact that the size of the removed area have in the quality of the final inpaint and how it affects image captioning frameworks. Our results show that automatically generated datasets could be a feasible solution for computer vision frameworks validation.
Acknowledgement

First of all, I would like to thank my supervisor Stephen Gould for the great opportunity to work on this project and all the provided guidance during the last months. Also huge thanks to Max Wang, who was my learning partner during the development of this research, helping with all the support and unstoppable dedication to the project. Finally, I want to thanks, Weifa Liang for all the guidance during this semester.
# Contents

1 Introduction .............................. 4

2 Properties of COCOONO .................. 5

3 COCOONO Construction .................. 5
   3.1 Base data-set .......................... 5
   3.2 Data generation ....................... 5
   3.3 Evaluation and edition platform ..... 6

4 Implementation details ................... 7
   4.1 Technology choices .................... 7
   4.2 Web platform architecture .......... 8
      4.2.1 Back-end .......................... 9
      4.2.2 Front-end ....................... 9

5 Results ................................ 10
   5.1 Effects of inpainted area on image quality 10
   5.2 COCOONO as a validation data-set ..... 11
   5.3 Analysis of Results .................. 12

6 Concluding Remarks & Lessons Learned 13
1 Introduction

In the last years, the easy access to data and advancements in machine learning are propelling great progress in the computer vision field. However, the training data is determinant to define how the machine learning models are going to behave in a validation situation. Therefore, there are many image datasets with different purposes. For instance, ImageNet [2] is a big dataset of iconic photos of different objects classified in 21841 synsets (set of synonyms). Another example is the Microsoft COCO data-set [3], that in contrast to ImageNet, presents many objects in each image, normally in a non-iconic perspective, classified in 80 categories and all pixel labeled.

The variety of datasets is crucial for machine learning because the training data defines how the models are going to perform with real world situations. Then, for computer vision applications, those data-sets should be diverse enough to allow the models learn how to generalize the detection in complex scenarios.

One problem of the available datasets for computer vision is the absence of unitary permutations between images. For instance, Microsoft COCO brings pictures where objects are in complex real-world contexts, however, the dataset lacks the diversity to provide examples where one or two objects are missing in a similar situation than another image. Such lack of diversity generates problems in models trained with MS COCO, such as the lack of generalization displayed on the figure 1. We aim to aid this problem creating an alternative dataset based on Microsoft COCO named COCOONO, with permutations of the original images on Microsoft’s dataset.

Figure 1: Example of failure in image captioning: in the right we have the original image from the Microsoft COCO data-set and in the left the result after the giraffes are removed by an automatic inpainting method. NeuralTalk identify giraffes in both images.

In the COCOONO dataset, we are distributing many versions of images available on COCO dataset with objects removed by an automatic inpainting algorithm. To ensure the quality of the inpainted results we created a crowdsourced rating website where humans can evaluate the quality of the generated images. We also integrated an online image editor which allow the users to fix possible problems in the inpainted images. Finally after evaluation and possible manual edition, just the results with the bests grades are added to our dataset.

In Section 5 we present the analysis of our methods for image inpainting and its results quality, we also use our dataset for validation of NeuralTalk.

This report explores the following opportunities:
The possibility of using image inpainting as a technique to diversify existent images
data-sets.

Use of a methodology to achieve high-quality datasets based on automated image
editing using crowdsourced information.

Presents a analysis of the acquired dataset being validated in NeuralTalk.

2 Properties of COCOONO

COCOONO is based on the Microsoft COCO dataset. The COCO dataset have more
than 300,000 images tagged in more than 2 million instances, grouped in 80 categories.
At the moment that this paper is written, we have a total of 399,430 images generated
through inpainting which 2471 where graded or edited by humans. From the evaluated
images 101 of them were edited and 532 presents maximum grading. Therefore, in
total, we have 633 images with objects removed which their edition is unperceivable for
humans.

3 COCOONO Construction

The project of COCOONO aims to achieve a rich data-set of alternatives images to most
of the COCO dataset, for both validation and training of image recognition frameworks.
Here we describe the methods that we used to construct the COCOONO data-set.

3.1 Base data-set

The choice of COCO as the base dataset was made because it offers images where the
objects are in non-iconic images, with diverse contexts and non-canonical perspectives.
Such factors were fundamental in a dataset to be applicable to our approach because
our graph-based inpainting method requires extra information beyond the region to be
removed. Furthermore, our dataset has as motivation to be used as the base for general,
real world, image identification, following the principles of the Microsoft’s data-set.

3.2 Data generation

With the base dataset in hands, we decided to generate as many alternatives images as
possible for further rating and human editing. Therefore, to achieve this we developed
a database with entries related to COCO images and a Node.js server which received
object removal tasks. Such server was responsible for image inpainting and database
recording of the results, doing its processing in multiple CPU cores.

For each object removed in the image, we generate 7 alternative inpaintings. To achieve
those alternative results we expand the polygon structure of the mask defined originally
in the COCO data-set. We expand the mask in 2, 4, 5, 6, 7, 9 and 12 pixels, each
expansion resulting in a mask and then in an inpaint.
Figure 2: The original image (a) have the region defined by the mask (b) defined as a bird removed, resulting in the image (c). This result is stored with a relationship to the original image and will be sent for users evaluation and editing.

3.3 Evaluation and edition platform

After acquiring the inpainted results our goal was to select the best images. Seeing that we had more than 300,000 images we decided to use crowdsource image evaluations and edition to achieve our goal. So we created an online platform as shown in figure 3 where users could register and evaluate images on a scale from 1 to 5, where 1 represents really bad results, unfixable by image editing, and 5 was reserved for images that were perfectly acceptable as real photos. Also, we created a profile page, showed on figure 4, where the user can check its evaluation statistics, reevaluate and edit the images. For editing, we integrated the web image editor Pixlr from Autodesk in our web page, as shown in figure 5.

Figure 3: COCOONO evaluation interface.
4 Implementation details

4.1 Technology choices

Since the beginning of the project, our idea was to develop tools that could be used in large scale to generate our dataset by crowdsourcing. Therefore, a natural choice was to use the web as a framework to achieve such scalability. So, we decided to develop our code base in JavaScript because it can be executed on browsers and can be easily ported
to desktop with Node.js.

In the choice of a database for our project, we required a technology that could be easily integrated with JavaScript, flexible for changes in the data scheme during the evolution of the application development, and also a database that make the access to data easy. To fit those necessities we chose RethinkDB, a new noSQL database which offers a web interface for accessing the data and total flexibility in the data scheme structure.

The source code development was conducted mostly by me and Max Wang, to manage the pair development we adopted the usage of Git on GitHub for code versioning. Initially, we had just a single repository with the previous implementation of the inpaint algorithm made by Max, which we used as a base for the web platform development. However, after identifying a clear separation between inpaint and platform code we decided to divide into two repositories. Such separation allowed us to make the inpaint as an NPM package, which showed to be really handy.

4.2 Web platform architecture

When developing the platform we decided to separate back-end from front-end by making the former as a REST API server and the latter a web app. Such model allows that all the logic of the user interface and experience be encapsulated in static files. Therefore, after the user access the page, the browsers downloads those files and all the further interactions do not need any page reloading. Any extra data required is requested from the REST API, that returns a JSON, which is them processed and displayed by the
front-end application.

4.2.1 Back-end

Our back-end consists of a Node.js application that serves a REST API using Express.js to handle the routing and management of HTTP requests. We also use Node clusters to allow multi-core support, making our server scale through many CPU cores as needed. For handling user authentication we used Passport.js with user verification by cookies.

4.2.2 Front-end

For the implementation of the user interface and experience, we used React.js with Redux. Such approach follows a more functional approach of developing user interfaces, making them much more predictable and modular.

React.js plays a role of rendering the HTML. In its core, it uses a virtual DOM representation of the web page and it handles the synchronization of this virtual representation and the real DOM on the browser. All the development of React.js is made creating UI components that can be combined, it is possible to combine buttons into a menu, and then with each composed component create more and more complex interfaces, and even an entire web page. Every component is a pure function that returns virtual DOM components based on received arguments and its internal state. So, it is expected that components will always generate the same resulting HTML given the same arguments and internal state. In those components, it is also possible to handle user events and program actions based on this.

Redux in a library that implements the Flux architecture, which replaces the model-view-controller when developing web applications. Such architecture uses an unidirectional data flow (see Figure 6) and the characteristics of React.js to make web applications more predictable, declarative and functional.
5 Results

5.1 Effects of inpainted area on image quality

As we discussed before, automated methods of image editing could generate erratic results. Therefore, we expected to see a relation between the size of the edited area and the final quality of the image. To achieve this we extracted data from the 2471 evaluations and plotted them in relation to the respective inpainted areas.

Our hypothesis was that the increase in the inpainted area would result in the reduction of inpaint quality.
As we can see in 7 there is a correlation between grades and inpainted area. Varying from grades around 4.5 when the square root of the inpainted area is near zero to grades around 3 when it gets to $200\sqrt{\text{px}}$. It is important to notice that just the edited area was considered here. However, the inpaint method that we use require extra information beyond the removed area, therefore, we could find a stronger correlation to the grades checking the relative inpainted area to the total area of the image.

5.2 COCOONO as a validation data-set

With some images in hands, we decided to validate some computer vision framework. The framework that we chose was NeuralTalk, mostly because is was trained on COCO and show good results on captioning for this dataset. With that, we could find any lack of generalization in scene understanding, to spot which objects were harder to identify and in which conditions. To check the accuracy of the NeuralTalk identifications we created a script that verified if any word related to the objects on the image were in the generated caption.

We expected to find some misclassification of not very iconic objects on the photos and
possibly a low error rate, one moment that our dataset is based on the original dataset used for training of NeuralTalk.

The results that we found showed that NeuralTalk still was identifying the removed objects in 15% of the images. A total of 93 identifications in 610 images. In Figure 8 we can identify the most commonly mistaken objects. It is possible to identify that the most commonly mistaken objects are surfboards and skateboard, we think that the reason for this is because those objects normally are in very specific contexts, making the learning of NeuralTalk biased. However, cases like bears and giraffes do not fit very well in this explanation, one moment that those animals can be in a variety of contexts. Therefore, we think that NeuralTalk is failing in the task of generalizing objects.

![Figure 8: Relation between object and misclassifications.](image)

We also expected that NeuralTalk would fail more when small objects were removed. However, as we can see on Figure 9 that the bigger the removed object more probable is to it to be misidentified. It is hard to understand this result and more analysis is needed on this topic.

### 5.3 Analysis of Results

With those result it is possible to identify that bigger editions generate worst results, however, presents more value for validation of computer vision frameworks, one moment that they are mostly misidentified. We also showed that objects very associated with the overall context of the image, like a surfboard or a sink, are more prone to be misclassified than more general objects, however, many objects present in general contexts still
have high classification error when removed. This shows that COCOONO could be an opportunity not just for further empirical studied of crowdsourced image datasets and automated edition methods, but also showed value as a validation dataset for computer vision frameworks.

6 Concluding Remarks & Lessons Learned

On this report, we provided some details about how we managed to develop a crowdsourced web platform for the construction of an alternative and diverse image dataset based on Microsoft COCO using image inpainting and manual image edition. Moreover, showing our technological decisions to achieve a user-friendly interface that could lead to good quality data. As a result, we found that data-sets made by permutations of other datasets could be useful for validation of image frameworks.

We chose to use a web interface as the platform to get the data required to construct our dataset, and to do this we developed a really modular and scalable web service, that could be applicable for different images datasets and evaluation tasks. Also, with a preliminary run for a limited set of users we were able to gather interesting data and learn the following lessons:

1. Computer vision frameworks are failing to generalize concepts on images and require more diverse datasets for training and evaluation.

2. Object highly associated with a given context are more prone to have their identification biased as we say on section 5.2.
3. Bigger subjects on images are more prone to misclassification.

4. Inpainted results get worse with the increase in the inpainted area.

References


INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UniID: u5767001
SURNAME: Zitelli Dantas
FIRST NAMES: Mateus
PROJECT SUPERVISOR (may be external): Stephen Gould
COURSE SUPERVISOR (a RSCS academic): Stephen Gould
COURSE CODE, TITLE AND UNIT: COMP3710

SEMESTER ☑ S1 □ S2 YEAR: 2016

PROJECT TITLE: COCO ONO: Crowdsourced Image Editing to gather a dataset to test and train computer vision frameworks more effectively.

LEARNING OBJECTIVES:
After successfully completing this subject, the student should be able to:
1. Develop skills relevant to real world research in computer science, such as managing a research project of significant size and scope and presenting research.
2. Gain insight into the conference publication process, while hopefully successfully publishing a paper in a high impact conference.
3. Demonstrate an ability to develop a database connected dynamic web application to collect, manage and verify crowdsourced data.

PROJECT DESCRIPTION:
The project involves the planning, development and distribution of a web application to crowdsource data of considerable scale. The web app will be written in javascript, based on node.js, and will integrate with Mechanical Turk or another crowdsourcing framework to automatically distribute and assign “tasks”.

Upon successful completion of this project, the student will gain insight into the process and difficulties of developing a substantial web application as well as experience with managing a NoSQL database indexing hundreds of thousands of images. The application will be developed with a rapid agile styled approach.
ASSESSMENT (as per course’s project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
</table>
| Report: name style: Research Report  
(e.g. research report, software description...) | 60 | Basura Fernando |
| Artefact: name kind: Web crowdsourcing platform  
(e.g. software, user interface, robot...) | 30 | Stephen Gould |
| Presentation: | 10 | Stephen Gould |

MEETING DATES (IF KNOWN):
Weekly meetings at Wednesday 11am, ad hoc meetings as required (daily up to conference deadline).

STUDENT DECLARATION: I agree to fulfil the above defined contract:

![Signature]

18/02/2016

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student’s academic record and believe this student can complete the project.

![Signature]

22 Feb 2016

REQUIRED DEPARTMENT RESOURCES:
High-end server for pre-processing and webserver for front-end application delivery will be supplied by the supervisor.

SECTION C (Course coordinator approval)

![Signature]

Date

Research School of Computer Science

Form updated Jun-12