CS 766 Midterm Report

Yun-Shiuan Chuang
ychuang26@wisc.edu

Varun Sreenivasan
vsreenivasan@wisc.edu

Jacob Lorenz
jlorenz2@wisc.edu

March 2021

1 Summary of our original proposal

Our original plan involved implementing the Mask R-CNN [3] as in the original paper and replicating the results on the **COCO Dataset** [5] and then implementing the model on autonomous vehicle datasets such as the **Cityscapes Dataset** [1] and the **Indian Driving Dataset** [2]. We also intended to spend time on enhancing model performance by making modifications such as tweaking the model architecture (e.g., the R-CNN backbones), and exploring different training techniques (e.g., multi-scale train/test, horizontal flip test).

2 Progress on the original proposal

After attempting to execute the original plan (e.g., training the models with different datasets), we realized that it was impractical given the time and computer resource available to us and did not have confidence we would be able to make significant contributions in the time we had. We will summarize what we have done and the challenges we faced below.

Following the initial proposal, we spent time reading and understanding scripts that can be used to train the Mask R-CNN model. We came across two relevant packages that are widely used for training Mask R-CNN: Detectron2 from Facebook Research [1] and Matteport's Mask R-CNN [2].

We have tried using both packages to train the Mask R-CNN model with the full COCO Dataset on Google Colaboratory ("Colab") and encountered three major problems. First, the dataset must be uploaded to Google drive in order for Colab to read it. This was a significant bottleneck because uploading large dataset onto Google drive is slow - with the bandwidth of 100Mbps download speed and 40Mbps upload speed, uploading 1GB of files to Google drive took us 5 hours. Although we were able to find an existing COCO dataset on a publicly accessible Google drive [3], other datasets we were considering (e.g., Cityscapes Dataset, Indian Driving Dataset) were not publicly available on Google drive. This constraint limited our flexibility to train the Mask R-CNN model with large-scale datasets.

The second problem was related to the limitation of the Colab environment. Even when we were able to mount the public Google drive that contains the COCO dataset, Colab crashed upon loading the large-scale dataset (2017 COCO training set is of size 19GB). After days of research and trial and error, we eventually managed a workaround using *symbolic links* [4], which ensures that Colab wouldn't crash due to being inundated by many images. This, however, requires the dataset to be present on the Google drive in a specific format. Currently, we were only able to configure the 2017 COCO dataset into such formats, but not for other datasets.

The third problem was the most serious one. Although we were able to start training the Mask-RCNN model with the COCO dataset using the aforementioned workarounds, we soon realized that Colab was under-powered to train the Mask R-CNN model from scratch with such a large dataset. The free version

[1]Detectron2: `https://github.com/facebookresearch/detectron2/`

[2]Matteport's Mask R-CNN: `https://github.com/matterport/Mask_RCNN/`

[3]This public Google drive contains the 2017 COCO dataset: `https://github.com/sawyermade/detectron2_pkgs`

[4]For our training attempt with the workaround using symbolic links, see our Colab notebook at: `https://colab.research.google.com/drive/16SrN_1wgG82gey5UiRxsTs0I6iWiyvlP?usp=sharing`. Note that it requires a UW-Madison G Suite account to access it.

of Colab only provides 12.8 GB of RAM, one single GPU (with 11.4MB of memory), and each session only allows 12 hours of training. More annoyingly, if the browser gets idle, the Colab session would be terminated automatically, which means the the client computer can't be turned off or used for other purposes while training. The authors of [3] said that it took them 32 hours to train the Mask R-CNN model from scratch with COCO dataset using a 8-GPU machine. With an single-GPU machine, the 12-hour session limit, and the termination mechanism, we deemed Colab not appropriate for such kind of heavy training task (see Figure 2 for a desperate training attempt using Colab).

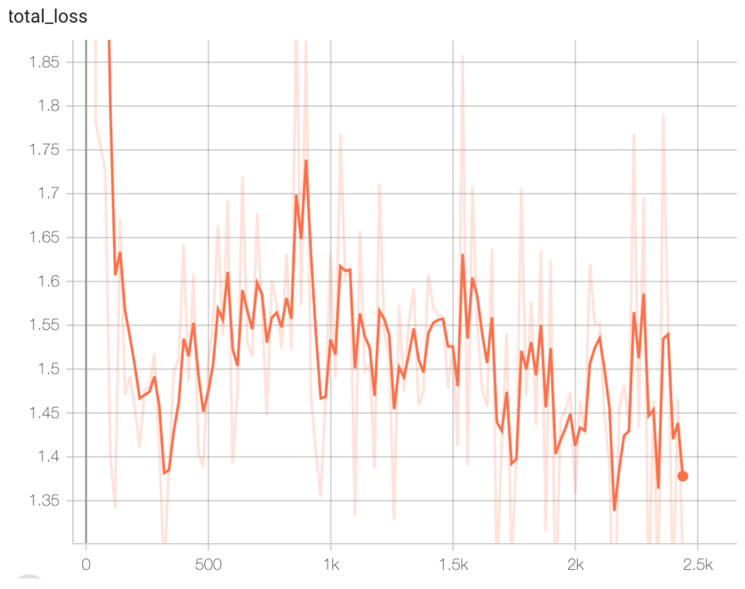

Figure 1: An unsuccessful attempt of training the Mask-RCNN model with COCO 2017 training dataset on Colab. The training got aborted because the browser got idle after training for about 90 minutes. The x axis is the number of iterations, and the y axis is the total loss on the training set. The total loss of the Mask-RCNN is defined in [3]. We also followed the exact same hyper-parameters as in [3] for the sake of replication. For the notebook of the training setup, see footnote 4.

Given these problems, plus the feedback for our proposal (see Section 3), we have decided to switch our direction and make use of pre-trained models with fine-tuning, rather than training from scratch (see Section 4).

3 Reflection on the proposal feedback

In addition to the problems we outlined in Section 2 and based on the professor and TA's feedback on our proposal, we realized that our original plan lacked concreteness and entailed spending too much time training neural network models. In light of this, we have decided to change the direction and emphasis of our project to deliver an end result that is both tangible and achievable.

4 New Proposal

After discussing with the professor and the TA, we have decided to switch gears to building an interactive app using pre-trained Mask R-CNN models. We are intending for the interactive app to have several different capabilities as outlined below.

The most tangible part of the app will be the front end web-app component, which will first and foremost allow the user to upload their own images and have them processed with the instance segmentation model and returned as an annotated image. The annotated image would include the annotations for each of the instances detected in the image. Each annotated instance includes the bounding box, the class label, and the binary mask that segments the instance from the background (see Figure 2 for examples). The annotations would be based on the predictions of Mask R-CNN pretrained models. For the pretrained models, we plan to use the ones with the widest coverage (i.e., include as many generic classes as possible; see Section 5 for details). This would annotate the generic classes in the images. For the sake of app usability, we will be focusing on optimizing performance for speed over accuracy and looking for any places we can improve the speed of the segmentation.

Apart from this generic class pretrained-annotator, we are also considering doing transfer learning on specific domains, i.e., fine-tuning the pretrained model using other data sets in specific domains (e.g., litter[5]). The motivation is for the app to contain several "modes", with a "generic object mode" (using the pretrained model for COCO dataset), as well as other more specific modes like "litter mode"(using a fine-tuned model specifically for segmenting different types of trashes). The users can choose the mode that fits best with their interest. These types of data sets for fine-tuning tend to be smaller and should be able be trained even with our our under-powered machines.

In addition to the simple annotation functionality, we also aim to include other interactive features. For example, if the users are not satisfied with the instance segmentation results (e.g., the instance is not detected, or the predicted segmentation mask is off), the app would provide them with a tool to draw a rough boundaries around the instances they want segmented, and a segmentation algorithm would run using this prior knowledge (e.g., lazy snapping [4], grab cut [6]).

We have drafted a plan for building such an interactive app, including the frontend and the backend setups. The first priority will be to get the basic functionality of annotating an user's images working first and then adding in more features and customization as time allows. Our aim is to first build an app that can be cloned and run locally (which requires installing the required packages). If time permits, we aim to host the online (e.g., with AWS) after the prototype is stable. Please see Figure 3 for a low-fidelity prototype of our first draft of the app's frontend layout.

In addition to the web-app portion, we are also intending to offer the image segmentation capabilities in the form of HTTP API calls. The web-app will be a fun and interactive way to see your original image and the annotated one together; however, if you are using the application for the purpose of segmenting a large quantity of images, you might not want to do each one manually using the UI. In a case such as this, it would be nice to be able to call an API and pass along a batch of images and get back the annotated ones. One potential liability with implementing this successfully is the practicality of sending images via HTTP requests. Our first priority is to get the web-app portion working; however, having an API available is part of our plan.

5 Progress on the new plan

In this section, we will describe the progress relevant to the new proposal.

5.1 Annotate images using pretrained Mask R-CNN model

Since the goal of our app is to have a model that can detect a wide variety of objects, we would ideally like to have it trained on a large dataset like the Open Images Dataset that has 300 classes of common objects[6]. We found a pre-trained model for this dataset. However, there were compatibility issues when the model was trying to produce predictions.[7]

Given the issue with the Open Images Dataset, we decided to use the model pretrained on the COCO dataset (with 80 classes of common objects). The pretrained model is included in the Matterport package. Note that Matterport is only natively supported in Linux and MacOS, but we want to ensure the app we build would run on Windows PC as well. After exploring workarounds, we managed to have it work on

[5] `http://tacodataset.org/`

[6] `https://storage.googleapis.com/openimages/web/index.html`

[7] This pre-trained model requires using a deprecated Keras Mask R-CNN (`https://github.com/fizyr/keras-maskrcnn/`) package and we experienced problems while trying to load the model using this. We realized that a lot of people had posted about such issues in relation to this package and it seemed like the issues haven't been solved yet.

Windows PC. We were now able to annotate images with the pre-trained Mask R-CNN on COCO dataset with the Matterport package (Fig. 2).

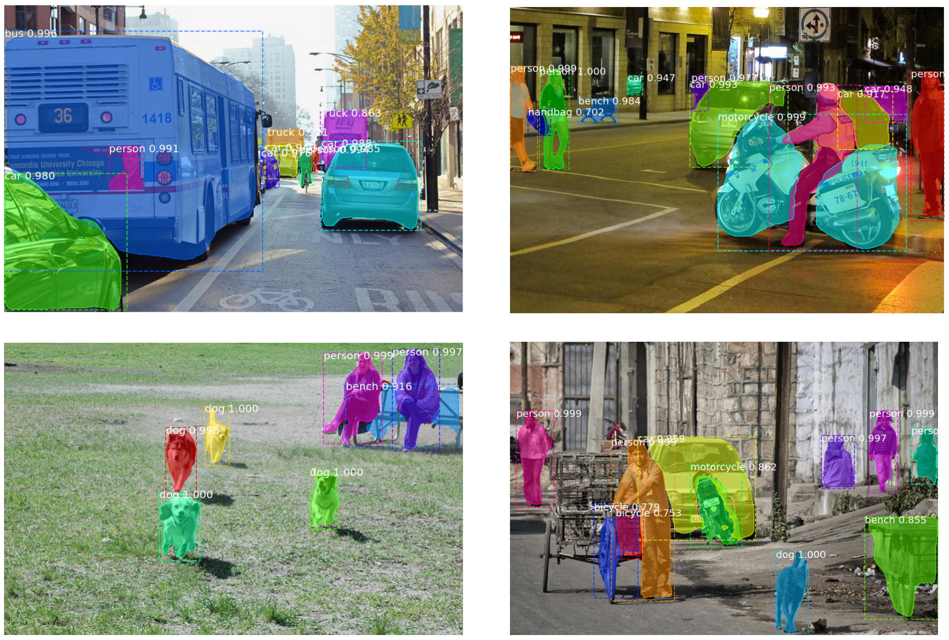

Figure 2: Annotations on input images with the pre-trained Mask R-CNN model on COCO dataset. The annotated image includes 1) the bounding boxes, 2) the class labels (along with the predicted probability), and 3) the binary masks that segments the instances from the background.

5.2 The interactive app

Below is a low-fidelity prototype design of the general image segmentation tab of our web-app. The user interface will be adapted and updated as more progress is made depending on feedback from our test trials. Currently we are working on getting the web server and web app setup and a structured interface, and then we will get into integrating the functionality for the instance segmentation capabilities. The app will initially be built and run locally and will be made available to clone from github at a minimum. If there is enough time, we will look into hosting the app on AWS so that users do not need to go through the process of installing all the dependencies and building the app themselves.

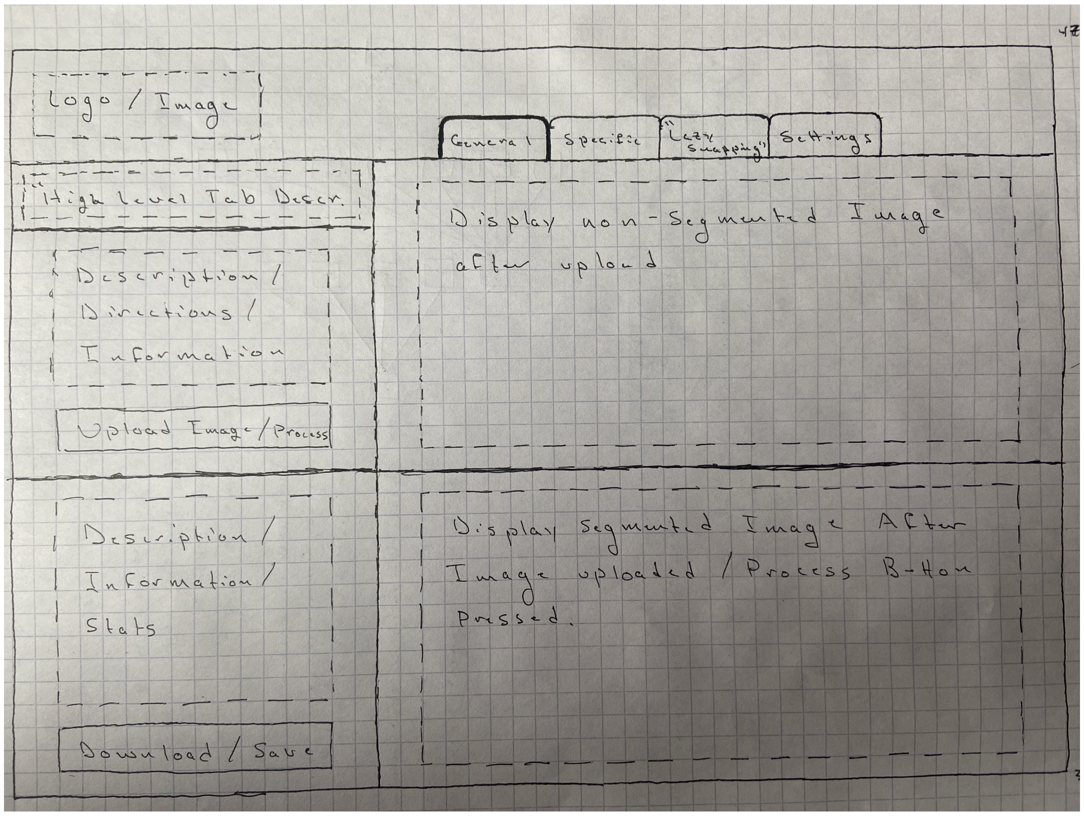
Logo / Image
General
Specific
"Lazy snapping"
Settings
"High level Tab Descr."
Display non-Segmented Image after upload
Description / Directions / Information
Upload Image / Process
Description / Information / Stats
Display Segmented Image After Image uploaded / Process B-tton Pressed.
Download / Save

Figure 3: A low-fidelity prototype of the general annotation front end portion of the instance segmentation web-app.

6 Time table

See Table 1 for a tentative time table.

Date	Task	Milestone
03/22	• Work on the mid-term report • Test the annotation function with pre-trained models • Find datasets to fine-tune on • Prototype the app • Explore the idea of Lazy Snapping and Grab Cut	Project mid-term report due (03/24)
03/29	• Fine-tune the models on chosen datasets. • Experiment with Lazy Snapping and Grab Cut • Begin working on UI of app.	
04/05	• Continue working on UI of app. • Work on the back-end of app. • Experiment with Lazy Snapping and Grab Cut	
04/12	• Make enhancements to the app. • Integrate Lazy Snapping and Grab Cut to the app	
04/19	• Try deploy the app on AWS. • Prepare the project presentation	
04/26	• Work on the project webpage and finalize the app	Project Presentation (04/28, 10:20-10:30am)
05/03	• Work on the project webpage and finalize the app	Project webpage due (05/05)

Table 1: The Time Table

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[2] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. V. Jawahar. IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1743–1751, 7.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[4] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics (ToG)*, 23(3):303–308, 2004.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[6] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.