

Scuba Steve - Anjan Patnaik, Art Jiang, Chris Labbe

1. Background and Significance of Project	2
2. Related Work (e.g., papers and other GitHub repos) - Art	3
2.1 Mask-RCNN	3
2.2 DeepMask	4
2.3 PixelLib Github Repo	5
2.4 Dash Plotly Repo	5
2.5 Docker	5
2.6 Google Cloud and Cloud Run	6
3. Explanation of dataset(s)	6
3.1 ADE20k	6
3.2 COCO	6
3.3 Pixellib Nature Dataset	7
4. Explanation of processes (methods)	7
4.1 Instance Segmentation on Image	8
4.2 Semantics Segmentation on Bounding Boxes	8
4.3 Combining multiple Region Proposals	8
4.4 Custom Training on User Annotated Images	9
5. Explanation of outcomes (results)	10
5.1 Mean Average Precision @ IOU thresholds	10
5.2 Annotation Time Improvements	13
6. Ethical considerations	14
7. Summary and future directions	14

1. Background and Significance of Project

Image object recognition technology today is amazing. Computer systems can pick out and identify many different types of objects at very high accuracy of properly identifying what many of the items in an image are.

This technology has only been able to happen because of a huge set of images that have been labeled and segmented by humans to use as training sets for the machine learning models in use today. One estimate says that it is a minimum of 300 samples of an object for the latest models to learn what that thing looks like and then start to be able to identify it in unlabeled images.

Our estimate of a human drawing an outline of an object in an image is 45 seconds. Even if the person doing the labeling is much faster after a lot of practice, 30 seconds is likely a reasonable estimate for the outlining and processing of the image and relevant information.

Translating that to a famous dataset that has 27000 labeled images, each of which has many outlined and labeled objects, real people have been required to outline over 100,000 objects to make that dataset work properly. Using the 30 second estimate suggests this is over 800 hours of non-stop work. And this dataset only understands 150 generic object types. For there to be a day where computers can process like humans and identification moves from “that is a bird” to “that is a mocking jay” will require many millions of labeled images at the cost of hundreds of thousands of labor hours. In 2020 this market for labeling data was \$1.3B, and it is expected to grow with a CAGR of 25%.

The goal of this project is to significantly reduce the effort required for this process through the use of generalized object identification partnered with human interaction. The hypothesis is that a person simply performs a “click-drag” action to create a box to tell the system what part of the image to process and indicates through a drop-down list of options what the target object is. If this works as expected, the time to find and create a training sample will be only a few seconds. The resulting benefit in the image labeling world could be a reduction of cost or an acceleration of available training objects by a factor of 5 - 10x.

2. Related Work (e.g., papers and other GitHub repos) - Art

2.1 Mask-RCNN

Mask RCNN is a new kind of deep neural network that tackles instance segmentation problem. It learns and translates an image into classes and masks. It has a two-stage process that generates proposals about regions where an object might be given an image input. It then predicts the class of the object and generates a mask in pixel level. Mask-RCNN has backbone that is composed of three main components: a bottom-up pathway, a top-bottom pathway and lateral connections.

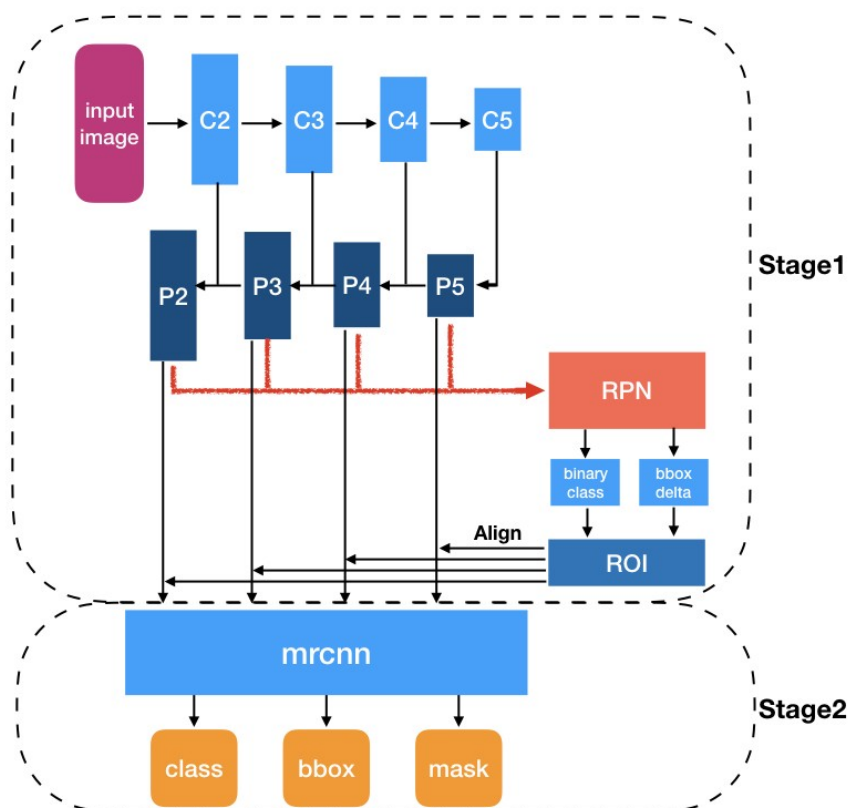


Figure 1: Mask-RCNN Structure. Zhang, X. (2021, November 17). *Simple Understanding of Mask RCNN* - Xiang Zhang. Medium.

In the figure above, the backbone, Feature Pyramid Networks(FPN) has bottom-up structure that extracts features from raw images and top-bottom pathway generates feature pyramid map; FPN also uses lateral connections to combine the top-down and bottom-up pathway for inferences.

A sub-neural network, RPN scans the top-bottom pathway of FPN to find regions that contain objects. While doing so, it binds the features to its raw image location, and determines the locations of the objects and the sizes of the bounding boxes.

The second stage of Mask-RCNN uses proposed regions to identify the areas of a feature map. It then scans these areas and generates object classes and masks. Similar to RPN, stage-two uses an algorithm called ROIAlign to find the relevant areas of the feature map.

2.2 DeepMask

In this project, we use **DeepMask** to increase the accuracy of semantic segmentation. **DeepMask** is a generalized object discovery model that does not aim to predict the class of the object. It has good recall on objects not present in the training set and proven to be a useful model for proposing regions for previously unseen objects specified by a bounding box.

Traditional object detection systems rely on the two critical steps: first, an estimate of the set of object proposals, and second, the proposed object classification. **DeepMask** is a method that combines the two critical steps and is first proposed in a research by *Chen et al.(2018)* [1].

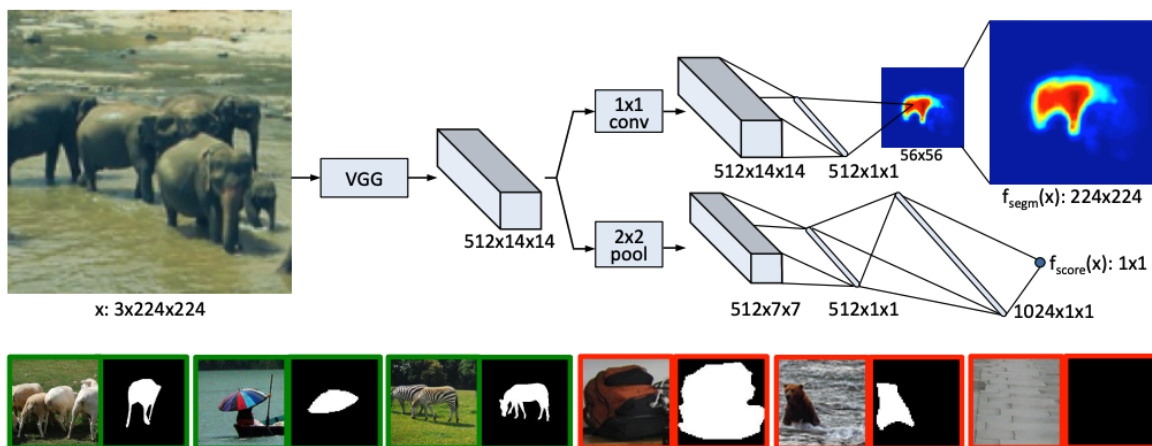


Figure 2: Model architecture: the network is split into two branches after the shared feature extraction layers. The top branch predicts a segmentation mask for the the object located at the center while the bottom branch predicts an object score for the input patch.

Deep Mask model is trained to identify the class-agnostic segmentation mask of an image patch, and to generate a set of segmentation masks for the object that it's centered on. It does so by estimating the likelihood of the patch being focused on a full object.

DeepMask achieves significant improvements over existing object proposal algorithms and is the reason why we chose this model in our toolset.

2.3 PixelLib Github Repo

In the implementation of our project, we use an open-source library, *PixelLib* [2] for performing segmentation of objects in images and videos. It supports the two major types of image segmentation: Semantic segmentation and Instance segmentation. The pre-trained coco model used in our PixelLib implementation is able to detect 80 classes of objects. We are also able to filter out unused detections and detect the classes the model has not seen before thanks to the Mask-RCNN model used.

2.4 Dash Plotly Repo

Our front-end work relies on the open-source framework, Plotly Dash. Written on top of React.js and Plotly.js, Dash is an SDK that simplifies the work of developers by allowing them to create interactive data apps with predefined user interfaces.

Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build a full-stack web app with interactive data visualization. Rendered in the web browser, we built our Dash app in Docker container and then deployed them on Google Cloud platform through Cloud Run. Since Dash apps are viewed in the web browser, Dash is inherently cross-platform and mobile ready.

2.5 Docker

Docker is an open-source containerization platform that enables developers to package their apps into containers. It simplifies the deployment of distributed apps. For deployment and simplified development, we set up a docker image running on Linux environments.

Docker containers rely on Linux kernel capabilities, such as process isolation and virtualizing to run smoothly. With Docker, various applications share the same resources across a host operating system. Unlike VMs, containers don't carry the entire OS' payload. Instead, they only contain the necessary steps and dependencies to execute the code.

We are able to run multiple copies of our app on the same hardware, and reduce the cloud spending.

2.6 Google Cloud and Cloud Run

Cloud Run is a managed compute platform that enables our team to run containers that are invocable via requests or events. Cloud Run is serverless: it abstracts away all infrastructure management, and the Scuba-Steve team can focus on building the application instead of deployment details.

3. Explanation of dataset(s)

3.1 ADE20k

ADE20k is a semantically segmented dataset that has over 27K outdoor and indoor images. Labeled data comes in json format with classes specified in a list format as well as x and y points corresponding to the image segments. There are training and test splits between the data and it contains over 180 classes of labeled images. Additionally the training data set includes the image as a jpg, a segmented version of the image and a json enumerating the segments.

Training on this dataset is difficult since labeled images are often in the background and sometimes very small. One of the challenges of using this dataset to train a new version of instance segmentation was that the new class we wanted to train had such high variance in its location in the image, size and quality. This made it difficult to train a model using the Mask-RCNN framework to find one class in a particularly large image. In this case the model tended to have many mistriggers and confuse the class with other similar concepts (like exposed brick confused for a door).

As such the ADE20k dataset was not a focus for this project as it was difficult to establish baselines on retraining our instance segmentation models (section 4.4). Future works should focus on generalizing our approach to hard to classify images and also introduce safeguards to prevent over-triggers on other classes.

3.2 COCO

COCO is a Microsoft dataset of >200k labeled images that has 80 classes of labeled images with both the raw image and a json that stores the image class and polygon data. The dataset comes with two APIs that help store annotation data efficiently in Run Length Encoding and visualize and explore the labeled data. In general images are more focused with the main image clearly in the foreground and taking up a size-able amount of the image.

Our instance segmentation model from the pixellib library is trained on this dataset and represents the base model that we try to extend from by adding new classes.

3.3 Pixellib Nature Dataset

A set of 800 images provided in the pixellib library that has either butterflies or squirrel objects. These are classes that are not in the COCO dataset and therefore not something the base model can infer. Since these images have data in the LabelMe annotation tool JSON format it makes establishing baselines a bit easier. Also most of the images have our target class alone as the focus of the image which allows us to draw larger bounding boxes. Because of the uniformity of these images this was a good dataset to test our DeepMask assisted annotation process.

4. Explanation of processes (methods)

Computer vision has been widely used in various industries to improve the efficiency of their operations. It is also used to identify and classify objects. Modern computer vision has created new markets and economies. In order for an AI system to accurately predict what objects will appear to it, it must first learn from thousands of annotated images. Our goal is to develop a deep learning tool-set that can help speed up human annotation.

Image segmentation is a process that involves partitioning a digital image into several segments. The goal of this process is to make the representation of an image easier to understand. Image segmentation is a process used in digital image processing to divide a digital image into various segments. A segmentation is a set of segments or regions that collectively cover an entire image. The pixel sizes in a region are relative to some of the properties of the image.

We include two powerful – yet distinct – techniques in Image Segmentation that help human annotation: Semantic Segmentation and Instance Segmentation.

4.1 Instance Segmentation on Image

An instance segmentation allows us to identify objects within a defined category. For example, the image on the lower right is the instance segmented output using a pretrained model.



Figure 1.1: Original Image and instance segmented output using pretrained model.

While instance segmentation labeling is expensive, its robustness and ability to identify multiple objects in an image make it a powerful tool for image analysis.

Our tool passes the user uploaded images through an instance segmentation pipeline where the model was pretrained on CoCo dataset with PixelLib using Mask R-CNN model.

4.2 Semantics Segmentation on Bounding Boxes

Semantic Segmentation works by grouping objects into specific categories. For instance, in a street scene, you could group all the vehicles into a certain area. Semantic segmentation will be used for grouping the pixels of an object. In our application specifically, if the user draws a bounding box over two cars and asks the model for suggestions on contours, semantic segmentation will give out the same label for all the pixels of the two cars enclosed inside the bounding box.

4.3 Combining multiple Region Proposals

During our development process of semantic segmentation, we noticed that feeding a fixed number of contours to the model for prediction does not provide an optimal solution every time. We implemented the capability for human annotators to change the number of polygons (inside the bounding box) that the model takes into consideration when proposing contours. Such capability further improved the performance of our model and can be illustrated with Figure 3.

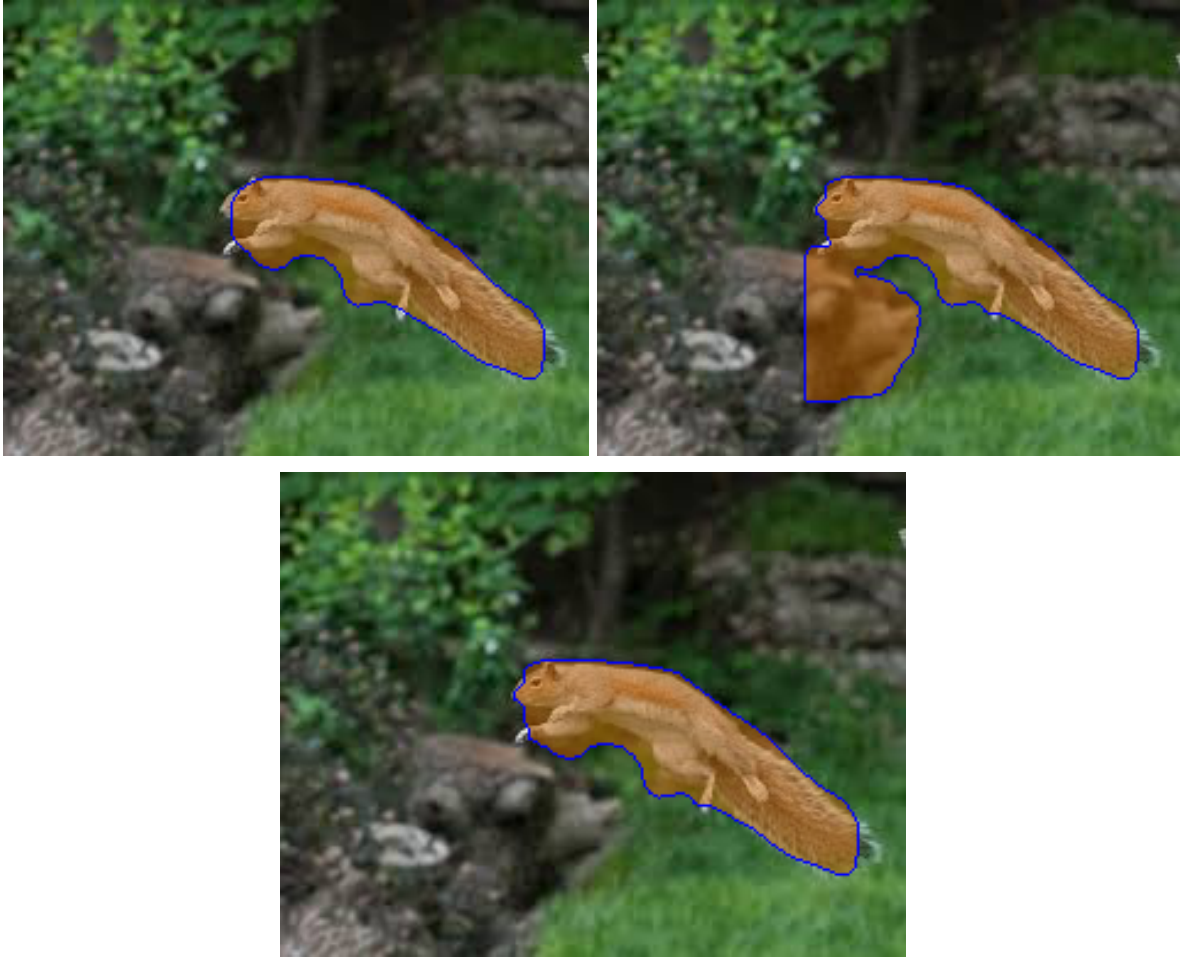


Figure 3: Contour proposal by our application with different number of polygons to be considered for the same image. **Top Left:** One polygon used for consideration. **Top Right:** Six Polygons taken into consideration. **Bottom:** Three Polygons taken into consideration and got the best contour proposal result.

4.4 Custom Training on User Annotated Images

Our application has three capabilities regarding annotating objects outside of pre-trained classes:

1. Instance Segmentation on New Classes: The Mask-RCNN model used for instance segmentation will be able to detect objects that it has not seen using the features extraction it learned from the training process.
2. Semantic Segmentation and Contour Proposal on New Classes: The DeepMask is able to provide inference on the contour coordinate information on objects not present in the training set. Provided with a bounding box, the DeepMask model has good recall for new objects it has never seen before.

3. Custom Training on Annotated Images: PixelLib open-source library allows us to train on new image object classes using Mask RCNN model trained on COCO dataset as backbone.

5. Explanation of Outcomes

5.1 Mean Average Precision @ IOU thresholds

We trained an instance segmentation model in pixellib to be able to detect butterflies using >200 images of butterflies labeled by DeepMask annotation. As described in previous sections the workflow was like this:

- 1.) Extract images of 150 - 200 butterflies with original ground truth data and train a custom instance segmentation model on them for 100 epochs. Generally, we did not tune the hyperparameters and found an odd relationship between validation loss and our metric of choice AP@IOU, which will be discussed later.
- 2.) Create DeepMask assisted ground truth data by using our tool to propose regions and not modifying those proposals at all on just the training set of data. This data is extracted and converted into LabelMe format for training. Here we train the model again against the training set while leaving the validation or test set unchanged (in terms of which ground truth we are using). Our resulting evaluation of the model seeks to understand how well our tool generalizes to out of sample hand labeled data.

During training our metrics of interest are training loss and validation loss. Both losses are the combination of multi-objective loss functions for each process. It is a weighted sum of different losses listed below:

- RPN Class Loss - loss assigned to the improper classification of anchor boxes
- RPN Bbox Loss - loss assigned to the localization accuracy of the region proposal network.
- MRCNN Class Loss - loss corresponding to improper classification of the object in the region proposal. Should be correlated to RPN.
- MRCNN BBox Loss - loss corresponding to localization of bounding box on identified class
- MRCNN Mask Loss - loss corresponding to mask on a pixel level on identified objects.

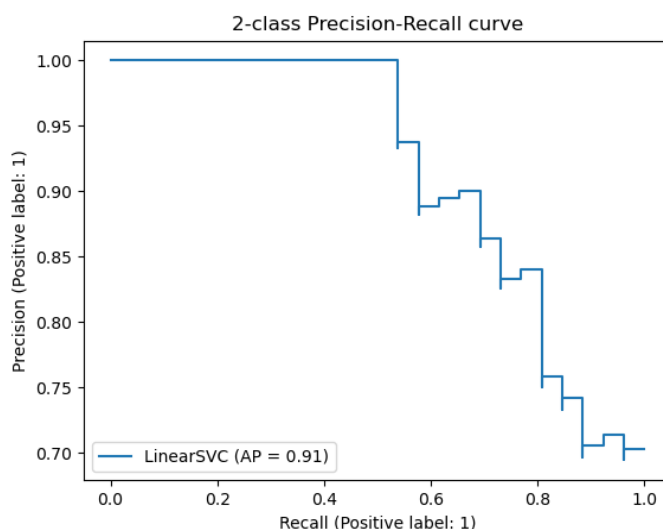
These metrics have interesting potential tuning opportunities but are more operational metrics to make decisions on when to end training. For this task we looked at the overall combination of these losses in validation set as our target operational metric.

Our outcome metric of interest is Mean Average Precision @ IOU. This metric essentially takes the mean area under the curve of the precision and recall line across all classes and can be valued at different Intersection over Union thresholds.

Breaking this down further we can define a few terms by starting with key concepts.

- True Positives (TP) = # of times the mask adequately overlaps with the ground truth and is the right class.
- False Positive (FP) = # of times the mask predicts a class that isn't there.
- False Negative (FN) = # of times the mask misses (because of a lack of overlap or lack of detection) a class that IS there.

With these metrics we can get both precision ($TP/TP + FP$) and recall ($TP/TP + FN$) metrics. We can get multiple measurements of both metrics. by simply varying the threshold at which we consider an inference a true prediction. Doing so gives us the familiar P/R curve:

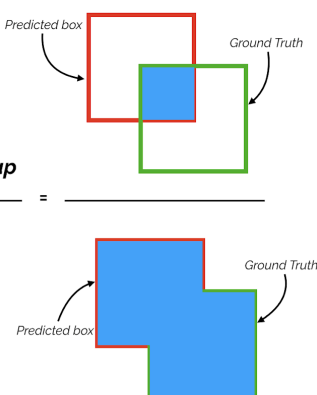


Generally precision and recall tradeoff with each other, which makes intuitive sense since if we cast a wider net we will get more salmon but also more kelp. Conversely a smaller net gets us only salmon but not as much salmon as we would like.

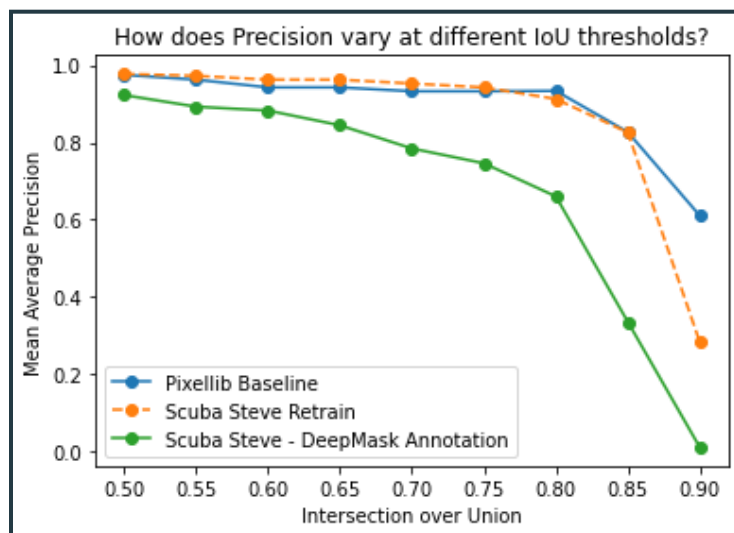
Confusingly for the purposes of this metric the Average Precision is defined as the area under this curve. So if a model is really good there's less of a tradeoff between precision and recall and the area converges towards 1.

There is also another parameter that can vary. In our definition of TP, FP and FN we have to take into account not only the class prediction but also how well our predicted mask overlaps with the ground truth mask. This metric is called Intersection over Union and affects the Average Precision measurement at different thresholds.

IOU is defined as the following:

$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


So when we say Average Precision @ IOU = 0.5 this means we are looking for Average Precision when the ground truth and mask overlap is at least 50% of their union. Any overlap that is below our threshold will be considered an FN. If we get stricter about what IOU threshold we use, Average Precision will naturally decline.



Results from training on butterflies data with original training labeled data and augmented training label data (using DeepMask are shown above).

We are able to generally replicate the pixellib baseline using the original labeled data. Even when we use the DeepMask labeled data we get comparable results at IOU thresholds up to 0.7. However there is a steep decline at higher IOU thresholds. This makes sense because our labeled data might not always approximate boundaries well which penalizes us at stricter IOU requirements. This can potentially be remedied by tuning the weights of our loss function to weight the mask loss higher and therefore ensure pixel precise boundaries at training time (this can give us a better approximation on how noisy we are making the ground truth labels with our approach).

Another observation of training this model was that validation loss tended to fluctuate suggesting that either our batch size was too low or our learning rate was too high. Future work will explore tuning both of these parameters. Addressing batch size may require some innovation in image processing as we typically run into OOM errors using larger batch sizes.

5.2 Annotation Time Improvements

One of the other key areas we were evaluating was the actual time saved by utilizing DeepMask assisted annotation. By timing how long it takes to manually annotate images (45 second on average) we were able to see a 4.5x gain in annotation time. The implication for this is that we can generate 4.5x labels in the same amount of time.

The natural extension of this finding is to see how the imprecision in ML assisted labelling trades off with the increased number of images we have. Specifically we'd want to know how many imprecisely (but quickly) labeled images it takes to converge in performance to more precisely (but slow) labeled images. If the ratio between the two is less than 4.5x we have a winning candidate.

Additionally there are several methods we can employ in the future to reduce annotation time with our ML assisted solution. We can pre-compute our proposed regions by having the annotator quickly do a first pass of bounding boxes, compute the region proposal in the background and then a second pass of accept/reject of the proposed regions. Furthermore, we can improve the latency of region selection by optimizing the math used to calculate it.

6. Ethical considerations

In the final version of the project leading up to the capstone presentation, the features originally planned for that would have potential ethical concerns were not implemented. The project as submitted uses open source projects, models, code and images each with their own adoption of public use agreements.

However, this project will continue in development and the eventual product will have some areas that need to be better understood from an ethical standpoint. Specifically the project intends to allow for users to upload their own pictures for segmentation or inference and this opens up the potential for using the software to create training models on any type of object in any set of images.

The three biggest areas of concern are:

- Use of the system to process images of specific people
- Uploading non-public images that constitute IP infringement
- Retrieving or uploading images of an illegal nature

It will be necessary to implement some additional machine learning structure to the final product that makes an assessment of any uploaded images and attempts to recognize these behaviors before accepting and processing the image. Or eliminate certain object types from being allowed in the masking and training processes.

7. Summary and future directions

The early results from this project have shown that the hypothesis was correct. It is possible to significantly change the effort required to make annotated objects for training models to detect currently unknown objects. The metrics show that with older models and early development efforts a new object (butterfly) was successfully added to the popular MaskRCNN instance segmentation model.

For the future of this project, there are many areas that need to be explored, developed, and integrated.

Currently, the system uses the DeepMask model developed by Facebook in 2015. There is an update to this technology released in 2016 called SharpMask that has been shown to create much more accurate contours of generalized objects at the sacrifice of time for the search and processing of images.

Additionally, the latest version of the application includes preliminary work to improve on the detection of the target object through a search structure nicknamed “middle-out” that begins the search with an assumption of the target object being in the middle of the box

drawn by the user. Combined with the concept of using multiple region proposals to increase the total space involved in the target, it is very likely the model performance will improve significantly.

The new techniques mentioned above will push the time to outline up to 15 seconds from the 5-second performance demonstrated. As a result, there is a need to refactor the search and contouring to avoid repeated calculations and improve performance.

All testing and development of the masking process have been done with older versions of Tensorflow and Torch on CPU. Significant improvements in contouring performance will be seen with an effort to modernize the Pixellib structure and get it working correctly on GPU systems.

In the app, there is an extensive list of user features and enhancements to help with user selection of the best mask to save for training, recommend a label for the selected object, enable model training on user request and utilize the newly created models for inference for the example pane.

References

1. Chen, T., Lin, L., Wu, X., Xiao, N., & Luo, X. (2018). Learning to Segment Object Candidates via Recursive Neural Networks. *IEEE Transactions on Image Processing*, 27(12), 5827–5839. <https://doi.org/10.1109/tip.2018.2859025>

2. Ayoola Olafenwa. (2020). *GitHub - ayoolaolafenwa/PixelLib: PixelLib*, <https://pixellib.readthedocs.io/en/latest/>. GitHub. <https://github.com/ayoolaolafenwa/PixelLib>

3. @inproceedings{DeepMask,
 title = {Learning to Segment Object Candidates},
 author = {Pedro O. Pinheiro and Ronan Collobert and Piotr Dollár},
 booktitle = {NIPS},
 year = {2015}
}

@inproceedings{SharpMask,
 title = {Learning to Refine Object Segments},
 author = {Pedro O. Pinheiro and Tsung-Yi Lin and Ronan Collobert and Piotr Dollár},
 booktitle = {ECCV},
 year = {2016}