Editing Semantic Model Predictions for Extending Datasets

Eric Bianchi

Virginia Polytechnic and State University

ABSTRACT

Semantic segmentation continues to be a researched computer vision topic, with many applications. As such, there are numerous methods and models which offer a means to semantically segment image data. While one may be successful at accessing trained models, it is far less likely to access the annotated training data itself – especially if the training data is protected for security or privacy reasons. Even if the researcher had access to the dataset, and wanted to continue building upon the dataset, they would have to begin annotating their new images from scratch. This project uses purely computer vision methods to take any semantic segmentation model prediction masks and transforms the prediction masks into an organized JSON file compatible with open-source annotation software. The model predictions are easily editable to enable the extension of datasets for any data domain. A conducted user-study confirmed that both speed and accuracy improved by using the model predicted annotations when compared with annotating images from scratch.

Keywords: semantic segmentation, annotation, dataset

1. INTRODUCTION

Deep learning models have data-heavy training requirements. Specifically, supervised semantic segmentation models require a diverse and numerous sample size of consistently annotated training data. The training annotations for this type of model consists of polygon masks drawn over each semantic class of interest. As one can imagine, this way of preparing data for training can be quite time-consuming. The specific data domain that motivated this research was structural bridge inspections. For inspectors it would be useful to have semantic-level predictions for detecting and quantifying typical damage. The author trained several models for detecting damage like steel corrosion and cracking, as well as material segmentation which can improve the detection and classification of material-specific damage.[1] The images for these models derived from bridge inspection reports provided by the Virginia Department of Transportation (VDOT). The author supervised a team of undergraduates who annotated each of the datasets.

Like many other data domains, structural bridge images can be difficult to obtain due to security reasons, privacy concerns, or inaccessibility problems due to poor dataset distribution or upkeep. This research was partially driven by the challenge to obtain new data and also due to the restrictions the author faced from granting access to a 4000+ image structural bridge dataset the author created. A major concern was that the data providers, VDOT would not grant public access to the dataset, and only let the trained model be publicly accessible. This would mean that if other researchers wanted to build their own dataset, with only the model available, they would have to start annotations from scratch. Effectively, the 180 hours that went into the creation of the 4000+ image dataset would be far less useful to the field as just a trained model.

This is why the author has proposed a method to mitigate the losses from not being able to access datasets by granting the ability to make editable, first-pass, model prediction annotations to speed up the process of extending an existing dataset or creating a new dataset. The first-pass annotations are achieved through an effective contouring algorithm which transforms semantic model predictions into editable vertices. Thus, even if researchers were unable to obtain the semantic segmentation data, they would still be able to leverage the trained model's predictive annotations to enable editing and growth of the same data domain.

Further author information: (Send correspondence to Eric Bianchi)
Eric Bianchi: E-mail: beric7@vt.edu

2. RELATED WORK

To the best of the author's knowledge, recent state of the art research on the topic of contouring segmentation objects focuses on contouring the objects for the purpose of instance segmentation.[2] ContourRend is a model trained using deep learning techniques to input an image and output a contour of the detected objects. However, this method and others like it, do not handle internal holes well. Very often, the contours for steel or concrete have holes within the body of the contour. Additionally, much of the research in literature does not focus on contouring segmentation for the purposes of refining the annotations. The method proposed here, by the author of this research, resulted in the same outcome as the implementation procedure from ContourRend.[2] However, it was able to handle any holes, and it did not require any training to operate. It is also universal to any segmentation output, making it extremely well-generalized and versitile.

3. METHODOLOGY

The methodology for this process begins with two core computer vision tasks to achieve editable segmentation masks. The first task was to obtain semantic class pixel labels (class masks), the second task was to contour edges of the class masks. Then the vertices along the class mask contours were ordered to efficiently re-create the polygon. Once each of the contours and their vertices orderings were found, the information was compiled into a meta-data text file. The meta-data text file was formatted in such a way that made it easy to read and transform into a JSON file format that the annotation software (labelme2016) could read.

3.1 Semantic Prediction Mask

Instead of a raw image input, like what was used in ContourRend,[2] the input to this process was the output prediction of a pre-trained semantic segmentation model. The expected format for the prediction worked for RGB format or a one-hot-encoded-vector 2D matrix. One-hot-encoded-vectors means that the corresponding class number is mapped to each of the pixels in the image. The segmentation model shown in this experiment was trained using +4000 structural bridge images,[3] focused on identifying structural material, using the DeeplabV3+[4] architecture. The model weights may be accessed through `https://github.com/beric7/structural_inspection_main`. One-hot-encoded-vectors may be easily obtained from any semantic segmentation model prediction, as this is the typical format they are in before being decoded into a corresponding RGB value. However, should one begin with a RGB mask, the author provided a script to convert RGB masks into one-hot-encoded-vectors.

3.2 Contouring

To begin the contouring process, each of the predicted classes in the RGB segmentation image, were transformed into one-hot-encoded vectors. Each material class (concrete, steel, metal decking) within the one-hot-encoded-vector image was evaluated for contours as their own separate image. When the class material was isolated it was converted into a binary image format. The image was given a 5 pixel padding so that the edges of the image would be contoured properly. Then the contours were traced, for the class material binary image, using the OpenCV contouring method based on the algorithm proposed by Suzuki.[5] This algorithm uses a binary border-following procedure to track parent and hole (child) contours. After the contours and their respective vertices were found, the number of vertices were reduced using the vertices reduction method based on the discrete curve evolution (DCE) algorithm.[6] In the DCE, vertices are ranked and ordered based on a function of the adjacent line lengths and 180 degrees minus their resultant angle formed at the vertices. The vertices with the least straight angles and the longest lengths are weighted higher over those near 180 degree resultant angles and those with shorter line lengths. The author used the equations presented in the DCE paper[6] to accurately re-create the algorithm. A visual flow chart representation of the entire described process, from prediction mask to editable contours, is shown in Figure 1.

The detected contours were arranged into contour hierarchies which tracked internal and external contours figure 2. Using these hierarchies were important when forming the overall path of points for the JSON file. The path with which the contour points follow matter since each point is connected to the previous position. For example, the path formula used by the human annotators to handle holes is described in 2. This same route-finding was automatically implemented in the construction of contour paths. The automatic implementation

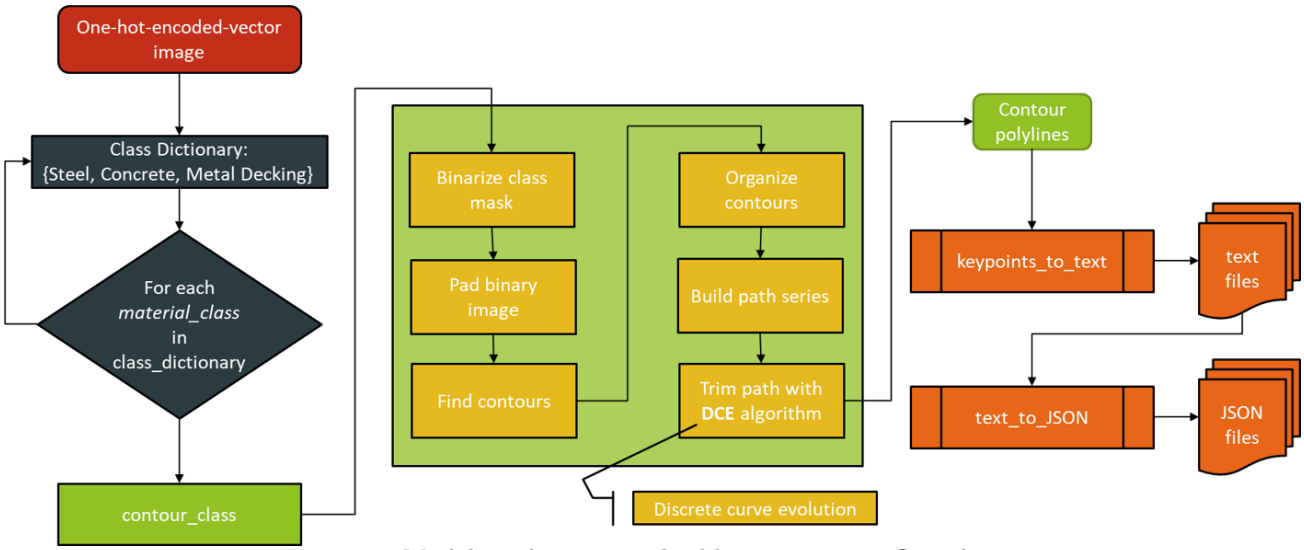


Figure 1: Model prediction to editable annotation - flow chart.

consisted of finding optimal entry and exit points for handling the holes as shown in Figure 3. The entry and exit points were chosen based on its proximity to the internal contour's centroid and its proximity to the external contour's edge.

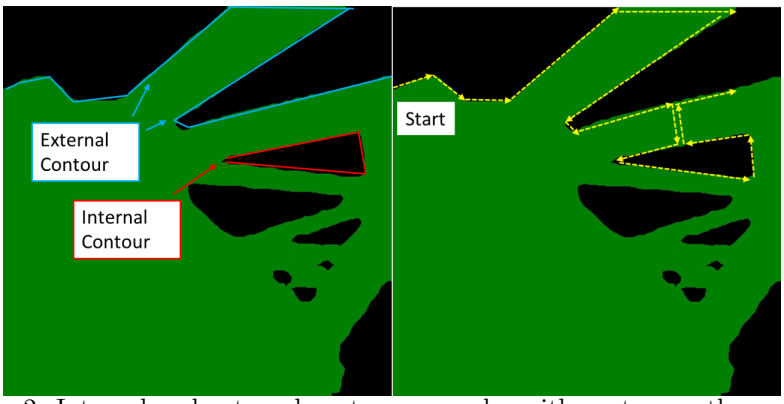


Figure 2: Internal and external contour examples with contour path example.

Contour identification and reduction is a basic computer vision problem. The reduction of complex geometry into simpler shapes can boost rendering speeds or reduce storage for those files. While the storage for one complex shapefile may be trivial, the compounding effect of many shapes and many files is significant. The management of this problem is found on any well-designed dynamic map with numerous complex shapes, like Google Maps. This concept of contour management is illustrated when zooming out of Google Maps; the detail of shapes decreases to account for the increased number of shapes in the frame of view. This same concept of contour reduction was applied to convert the segmentation predictions into more simplified shapes with fewer vertices. This design choice was necessary for drastically reducing the file size, while maintaining the original shape of the predicted segmentation masks. Additionally, having fewer vertices meant that making edits to annotations were easier to manage since there were less points to potentially fine-tune.

The author chose (2) DCE passes to reduce the number of contour vertices present in the prediction and to make image edits more manageable. Figure 4 contrasts the difference in editable vertices before and after the contour reduction. Figure 5 highlights this design decision by using image (4) of the sample image set used in the annotation study. The top series of images in Figure 5a show the how each DCE pass (0 to 4, from left to right) affects the prediction masks. With each pass the number of vertices are reduced by half, therefore with

Figure 3: Contour entry and exit point example.

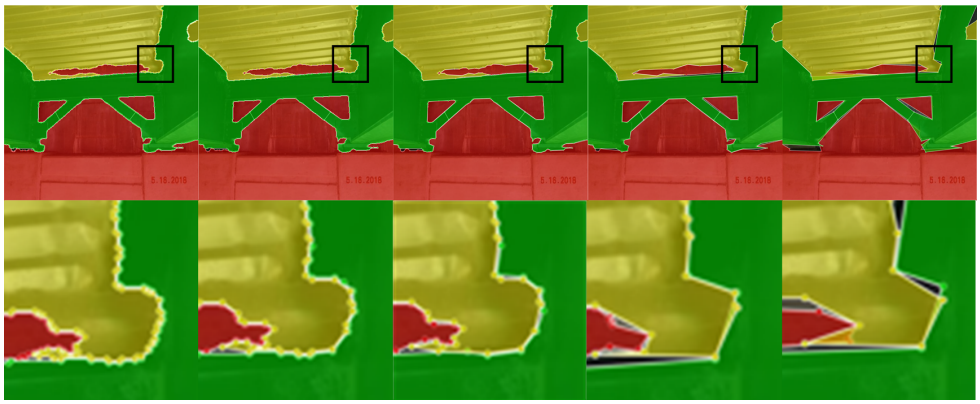
(a) (b)

Figure 4: Before (a) and after (b) DCE contour reduction. The polylines and points in green indicate the material class 'steel' and the red indicate 'concrete'.

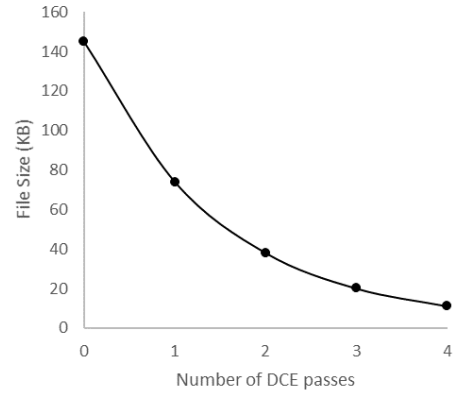
(2) passes, the author reduced the number of original vertices by 75%. This reduction is directly related to the file size as shown in Figure 5b. The goal was to push the number of DCE passes as far as it could go without sacrificing prediction performance. Ultimately, the number of DCE passes was chosen based on its impact on the original prediction's accuracy and precision (Figure 5c). It was found that after 3 or 4 passes the overall prediction accuracy and precision started to drop quickly. Beyond the reasoning for storage and accuracy, it was almost just as important to consider the editing process. As mentioned before, it would be beneficial for annotation editors to work with less potential fine-tuning points. Less points means less time spent on editing the predictions, at the cost of starting at a more coarse or inaccurate shape. Although it was not tested in this study, it would beneficial to evaluate the speed-accuracy trade-off for the number of DCE-passes on the editor's ability to quickly and accurately edit an image.

3.3 Data Transform

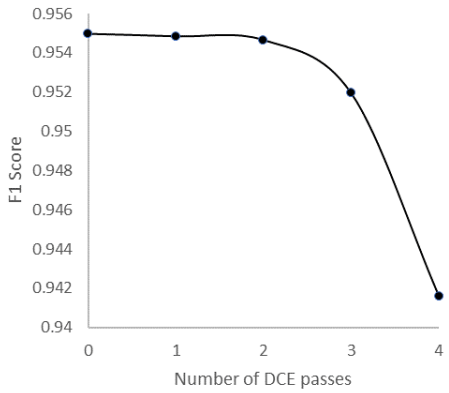
The path that was found by the contouring algorithm considered each internal set of contours and followed the same formula for the subsequent vertices. Once the vertices list for each segmentation instance was gathered, it was sent to a meta-data text file. This meta-data text file was a recorded version of the vertices list of each of the object instances. Thus, this meta-data text file was a convenient way to store the raw detection data. The raw data was transformed from the meta-data files to the labeling format which was needed. In this case, the raw data was transformed into the JSON format to match the annotation tool of choice, labelme2016.[7] A summary of the data transformation process is described in Figure 10 of the Appendix.



(a)



(b)



(c)

Figure 5: Diminishing returns of contour reduction: (a) the prediction mask with progressing DCE passes from 0-4, (b) file size as a function of DCE passes, (c) f1-score as a function of DCE passes.

4. EXPERIMENT

The material annotation study was the first part of a two-part study. The second part of the annotation study evaluated the consistency of following annotation guidelines for subjective corrosion condition state annotations. This second part of the study was not included, since it was not relevant to this paper. The material segmentation annotation study evaluated the effectiveness of the pre-annotated images generated from the model's predictions compared to the *blank*, *non-annotated* images. The success metrics were the speed and accuracy of the subsequent semantic annotations. Accuracy was determined based on the comparison to the ground truth labels provided by the author, and speed was determined on completion time of the annotation. The study was conducted completely virtually, and was granted institutional review board (IRB) exemption-status from Virgina Tech's IRB. The experiment consisted of 24 participants paid $25 each for approximately 1.5 hours of their time.

Participants joined through zoom meetings in groups which ranged from 2-6 depending on the schedule. Groups were kept small so that any technical issues could be fixed and questions could be answered during the on-boarding process. In the beginning of each of the sessions the author first checked if the necessary software (Anaconda with python 3.6, and labelme2016) had been downloaded. Once the software had been checked or set-up, then the participants were asked if they had the necessary data for the study. After both the software and data was checked, the participants were given a short 5-minute lecture on the motivation and the use-cases of semantic segmentation in bridge inspections and civil engineering. Following this presentation, the participants were shown the annotation guidelines for the material segmentation and followed along to a brief on-boarding of how to use the annotation software. Once the participants were comfortable with the software, they were allowed

to begin annotating the images, and self-recording their finish-times. Once all the images had been annotated, they submitted their data and recorded times through a Google form.

Each participant annotated 4 images. The ordering of the presentation of images were in what is called a Latin Square. Figure 6 shows the order of how the images were presented for each participant group. The design of the Latin Square reduces any systematic bias by shuffling the order of the presented material so that it appears exactly once in each column and row of the square matrix. Because there was not enough time for all participants to go though the eight image variations, the Latin Square was staggered. Another step the author took to reduce experimental bias was by scripting the on-boarding presentation so that the same examples were shared in the same way for each participant group.

Participants	Annotation Order			
1-3	1*	1	2	2*
4-6	1	1*	2*	2
7-9	2*	2	3	3*
10-12	2	2*	3*	3
13-15	3*	3	4	4*
16-18	3	3*	4*	4
19-21	4*	4	1	1*
22-24	4	4*	1*	1

Figure 6: Staggered Latin Square of image annotation ordering. The star indicates that the image contained first-pass annotations.

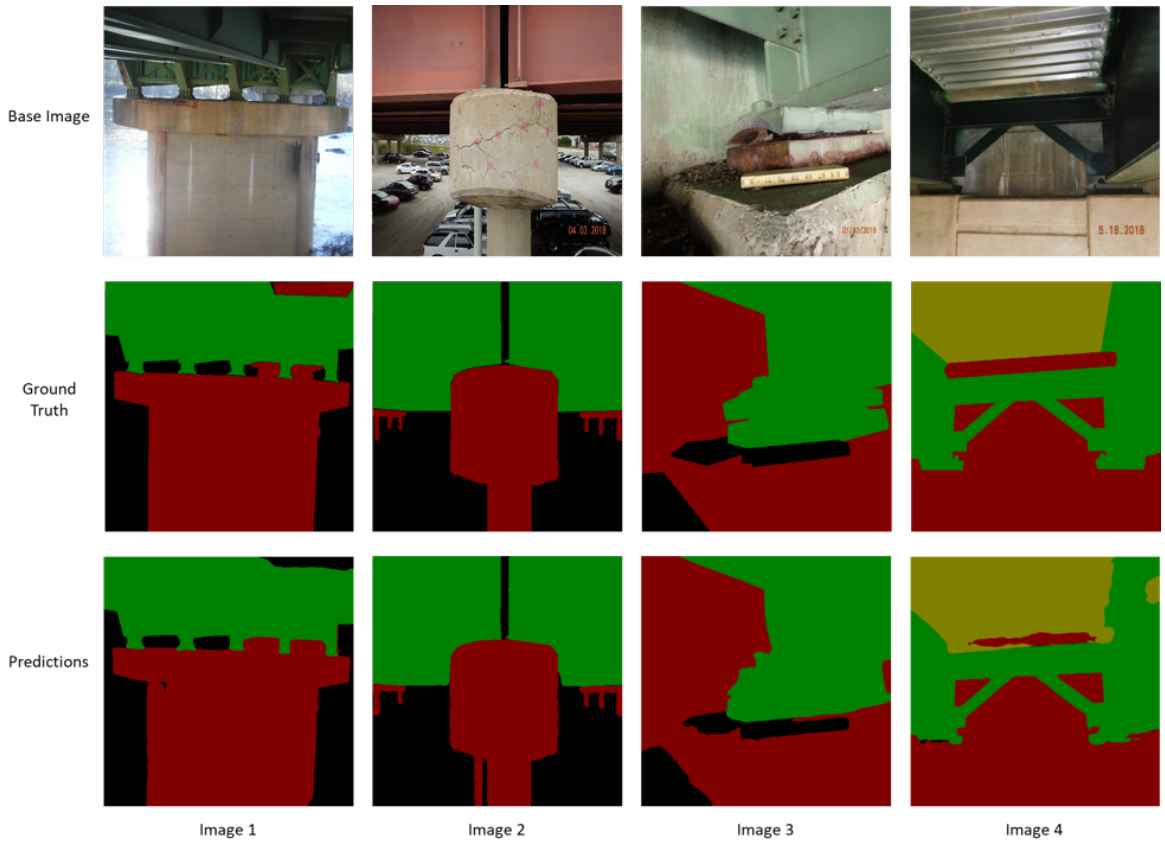
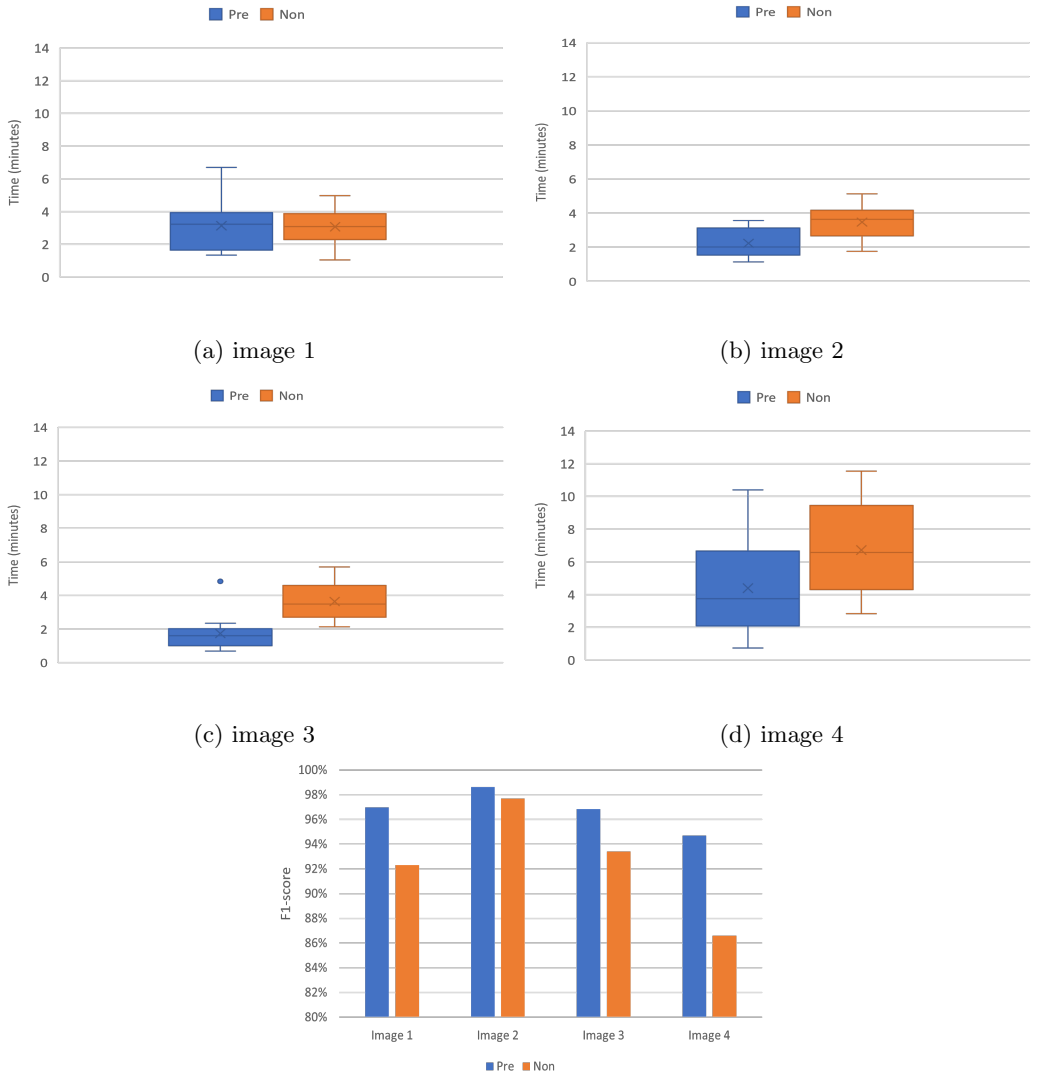


Figure 7: Images used in annotation study. Top row are the base images, the middle row are the ground truth labels, and the bottom row are the model predictions.

5. RESULTS

Completion times and F1-scores from the four images in the annotation study, listed in Figure 7 were compiled. An F1-score is a typical metric used when evaluating semantic segmentation predictions, which equally weights accuracy and precision. Each box and whisker plot had twelve data points for each the with and without first-pass model prediction annotations. The median of each of the plots are indicated by the solid line within the box, and the mean is the $\mathcal{X}$ within the box. Outliers in the data, indicated by a single point on the box and whicker plots, were identified but not included in the plot's creation (Figure 8c).



(a) image 1

(b) image 2

(c) image 3

(d) image 4

(e)

Figure 8: Time to completion for images 1 through 4 (a through d, respectively), and annotation F1-scores compared to the ground truth annotations.

6. DISCUSSION

The results from the study indicated that providing a first-pass of model predictions enhances the speed and accuracy of annotations. The author managed the confounds, of human subject participation by eliminating the order of the image annotation presentation. Still, there were some undesirable observations which were partially due to the restrictions by COVID-19, and due to the design of the experiment. There were five notable problems observed given the virtual format and the design of the study: remote teleconference, attention to detail and effort, time constraint, comprehension, and in-session engagement.

Since this study was conducted during the COVID-19 pandemic all interactions were virtual, which presented several unique challenges and exacerbated others. Being virtual meant that it put the burden of downloading software and data on the participants, and it required that they use the command prompt, which was unfamiliar for most of them. If the study had been conducted in a computer lab, it would have been much easier to moderate the downloading process, either before the participants arrived or by simply being in the same room as them. For example, there were occasions where the author had to remote control their computers, which slowed down the progression of the study for everyone on the zoom call.

The assumption was that participants would be giving their best effort, however, there were varying levels of attention to detail and effort between the sessions. Upon submission, it was noted that some participants performed significantly better than others, especially when it came to editing the first-pass annotations. Figure 9 shows how participant editors missed material or were lazy with their annotations. In some cases, it appears that little to nothing was altered.

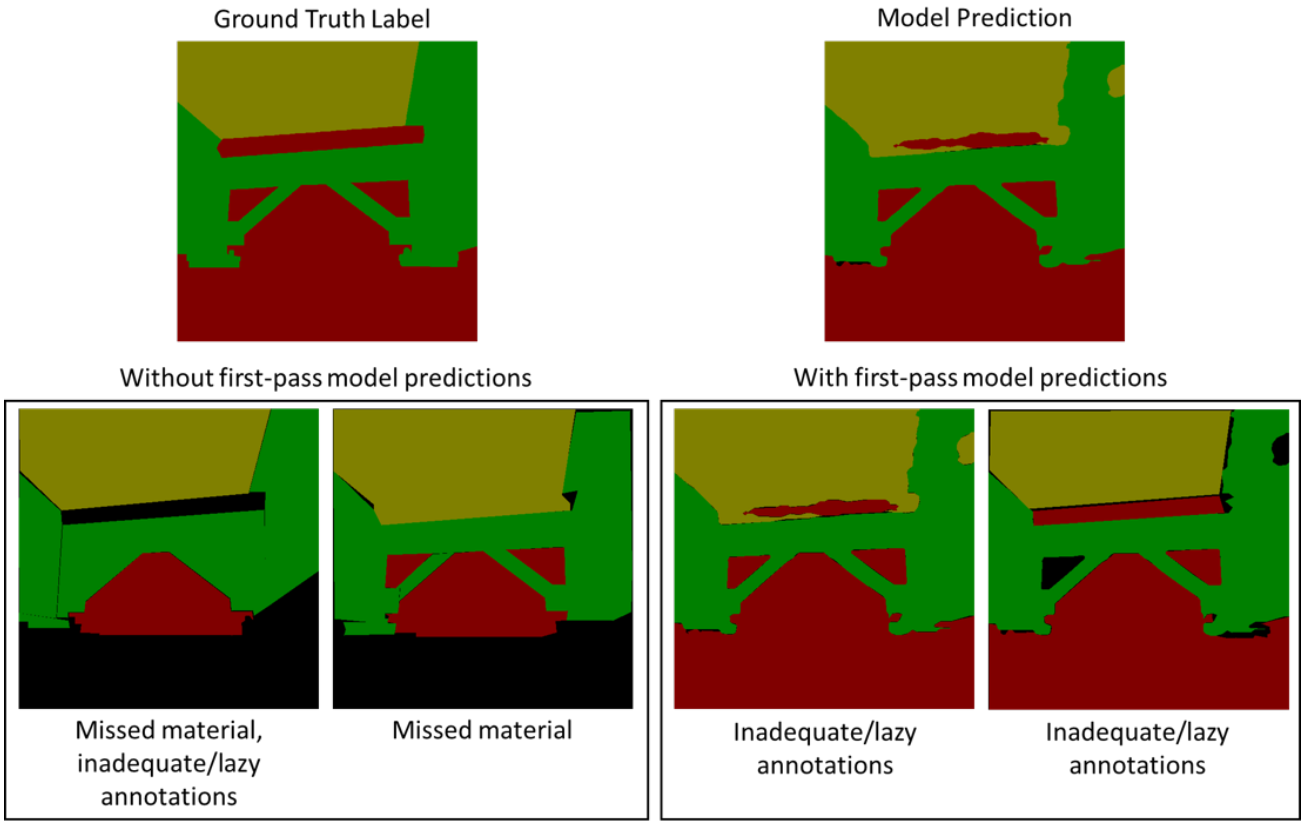


Figure 9: Annotation inconsistencies.

The author did not validate participant submissions, which was an intentional decision. The underlying reason was to verify the comprehensive effectiveness of following a how-to-annotate-material annotation guideline document provided by the author. While this format allows the researcher to see the accuracy and participant's

perceived completion time between the two methods, it does fall short of capturing the completion time for a perfect annotation. A more direct format for measuring the speed, while ignoring accuracy, would be to provide a template to the user to follow for the given image with the annotations already provided. In this format, participants would be required to match the template, by author verification, in order to move onto the next image. This experiment format would solve the participant effort, annotation guideline comprehension, and attention to detail problem.

The last observation was that there were varying levels of engagement during the sessions. For example, some sessions would ask more questions when practicing with the annotation tool before they began annotating their assigned images. These questions may have benefited the entire group, boosting their performance, while other groups were more quiet. For a more normalized experience, it would have been better to have only one or two sessions instead of eight.

7. CONCLUSION

The key conclusion of this report is that model-prediction first-pass annotations gave time and accuracy advantages when compared to images without baseline annotations. The model-prediction first-pass annotations were provided by a script written by the authors to take semantic segmentation predictions and convert them into an editable version of itself. This script is especially impactful in hard-to-access data domains like structural bridge inspection images. This method will enable researchers to quicken their own dataset creations when either the original data is not provided, or when they want to extend the sparse datasets available to them.

Acknowledgment

I would like to thank my advisor, Dr. Abbott, for the constructive feedback, sharing opportunities and relevant papers with me. I also would like to give a huge thanks to the support and encouragement from my wife and family.

REFERENCES

[1] Hoskere, V., Narazaki, Y., Hoang, T. A., and Spencer, B. F., "Madnet: multi-task semantic segmentation of multiple types of structural materials and damage in images of civil infrastructure," *Journal of Civil Structural Health Monitoring* , 1–17 (2020).

[2] Chen, J., Lu, Y., Chen, Y., Zhao, D., and Pang, Z., "Contourrend: A segmentation method for improving contours by rendering," (2020).

[3] Bianchi, E., "Structural inspection datasets: a review and contribution," *Journal of Civil Structural Health Monitoring* (2020).

[4] Chen, L., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H., "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR* **abs/1802.02611** (2018).

[5] Suzuki, S. and be, K., "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing* **30**(1), 32–46 (1985).

[6] Barkowsky, T., Latecki, L. J., Richter, K.-F., and florian Richter, K., "Schematizing maps: Simplification of geographic shape by discrete curve evolution," in [*Spatial Cognition II*], 41–53, Springer (2000).

[7] Wada, K., "labelme: Image Polygonal Annotation with Python." https://github.com/wkentaro/labelme (2016).

APPENDIX A.

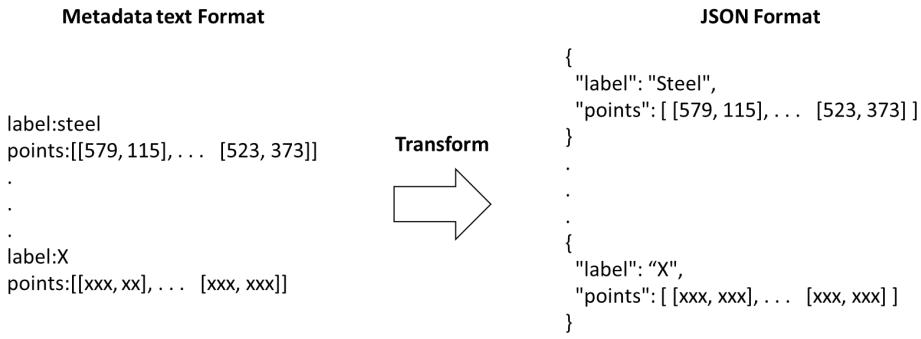


Figure 10: ...