WHITE PAPER

Intel & American Red Cross Southern Uganda Bridge Identification

Authors

Alexei A Bastidas, Matt Beale, Sean McPherson, Hanlin Tang
Intel AI Lab

Kris Sankaran, Yasser Salah, Eddine Bouchareb
Montreal Institute for Learning Algorithms (MILA)

Jigar Doshi
CrowdAI

Table of Contents

Executive Summary

Intel AI Lab, MILA, and CrowdAI have conducted a joint data-science collaboration to develop deep-learning image-segmentation models that can be utilized to detect bridges in remote areas using satellite imagery. Working with the American Red Cross, we identified that it would be valuable to base this work on imagery of territory in Uganda, given the country's historical suffering from both community epidemics and seasonal weather events.

In order to develop models for this region, we created a custom training dataset with CrowdAI, utilizing four band (RGB plus near-infrared) high-resolution satellite imagery as our inputs. We trained multiple deep-learning models to segment images into road, waterway, bridge, and background classes.

Utilizing a custom evaluation method that is appropriate for the sparse bridge detection problem, we selected a top-performing model with which to run inference and identify bridges across locations in Southern Uganda previously unseen to the model. Through our pipeline, we were able to identify 70 new bridges.

Problem Statement

Preparing and responding to humanitarian disasters requires accurate and timely mapping of affected regions. Current approaches satisfy the goals of mapping affected regions in a short amount of time, but they also require resource-intensive manual labeling from teams of human volunteers, such as the Humanitarian OpenStreetMap Team (HOT).

The Red Cross has played a major role responding to past outbreaks of diseases including Ebola in Africa. Based on their experience, the Red Cross identified an important learning opportunity in Uganda to support community epidemic and disaster planning work, with the intention to accelerate remote mappers by assisting them using machine learning.

The described need is to geolocate areas of potential bridges in Uganda so that mappers can plot them, and then for the Ugandan National Society to use the data to facilitate planning of evacuation and aid-delivery routes to avoid delays during major disasters. To facilitate this work, Intel, MILA, and CrowdAI created a training dataset across Northern Uganda and used this data to develop deep-learning-based models to automate bridge identification.

Data Science & Analysis

To train models for bridge detection in Uganda, we created a new training dataset, in conjunction with CrowdAI, featuring bridge, waterway, and roadway vector annotations for 339 unique bridge locations in Northern Uganda, visualized in Figure 1. In order to capture variance across time and sensors, the training dataset contains multiple views of the same bridges to enable the models to learn invariance to seasonal and nadir-angle changes.

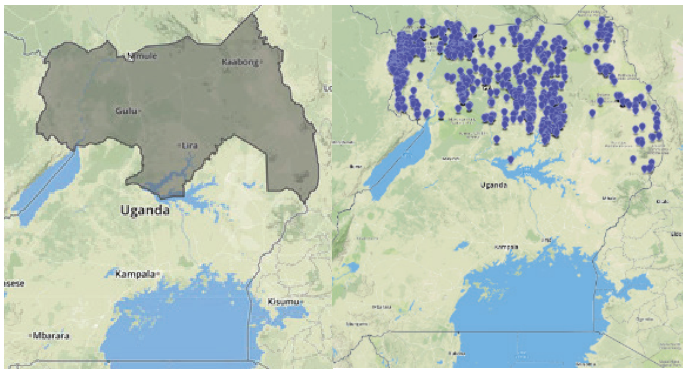


Figure 1. Bridge locations in Uganda used for training dataset.

Training Dataset Creation

To create high quality training data for machine learning models, CrowdAI uses its custom-built annotation interface. The annotation process takes place across three steps:

- **Phase I (Initial Annotation)**. A trained annotator uses CrowdAI's web-based tool to label an image or video for the target features. These may require a mix of polygons, polylines, or boxes, depending on the feature and desired training data type. CrowdAI will sometimes "pre-annotate" an image by having a machine-learning model predict on that image, creating a "first pass" mask that annotators need only edit and correct to ground truth.
- **Phase II (Review)**. A second trained annotator will review the results of the Phase I annotation to check for accuracy and conformity to ontology for the task.
- **Phase III (Final Review)**. The last review is performed by CrowdAI's highly-trained workforce of distributed Final Reviewers. These geospatial experts use CrowdAI's annotation interface to perform final checks for label accuracy, consistency, and conformity to ontology for that particular feature.

CrowdAI's internal data operations team then performs final spot checks on the data before moving the dataset to production for model development.

For this project, we utilized the OSM labeling nomenclature to determine what was labeled as a particular class. In particular, through discussions with the American Red Cross, the team determined to use the labeling schema shown in Table 1 to follow OSM conventions.

Table 1. OSM-compliant labeling schema.

Class Label	OSM Tags Used as Labeling Guidelines
Bridges	"bridge=yes"
Roads	"highway = primary", "highway=secondary", "highway=tertiary"
Waterways	"natural=water", "water=*"

For roadways, the cut-off was "highway=tertiary," which is defined by the OSM-Africa Wiki as follows:

> "Major transportation routes connecting towns and larger villages. Collector function in urban areas.
>
> Passable by vehicles with 4 or more wheels, motorcycles, bicycles, or foot and animal traffic.
>
> Indicative info only - can vary. Width: 3 to 7 meters; may be paved."

Thus, we did not label footpaths or agrarian walkways; these road types are common in rural areas, but they are relatively small and not part of the primary road network, which is the higher priority for route planning in the context of emergency response.

Training Dataset Characterization

In order to prepare the dataset for training, we bin images by spatial location, so that all images of the same location are grouped together in either the training or validation sets. We consider an epoch to be a single pass across all spatial locations, where for each spatial location, we randomly sample one image out of the stack available at each location. Figure 2 shows these different views of a single bridge.

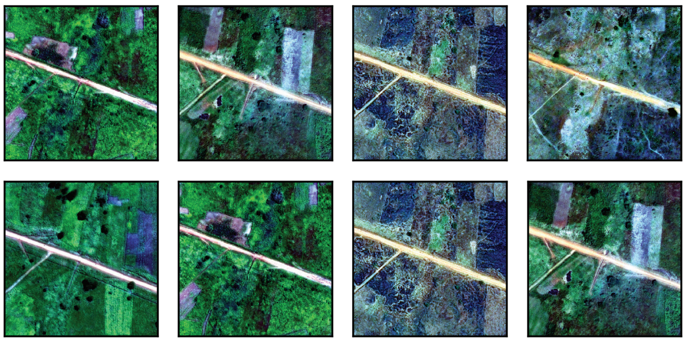


Figure 2. Multiple views of a single bridge.

Binning into unique spatial locations ensures that the model is trained and validated on distinct bridges. If the training and validation datasets were constructed by splitting across the entire duplicate set of images without binning, the model would have inevitably validated on a different image from a location seen in the training samples, leading to a noisy and elevated performance metric due to validation of images the model had already seen.

The feature and image counts for both the unique-spatial set (UG-U), and the entirety of the data (UG-IID) are shown in Table 2.

Table 2. Feature and image counts.

	Images	# Bridges	# Culverts	# Roads	# Water
UG-IID	4718	2251	3687	8284	5114
UG-U	751	339	483	1288	755

To display the range covered by the dataset, we visualize sample images drawn from the training set shown in Figure 3.

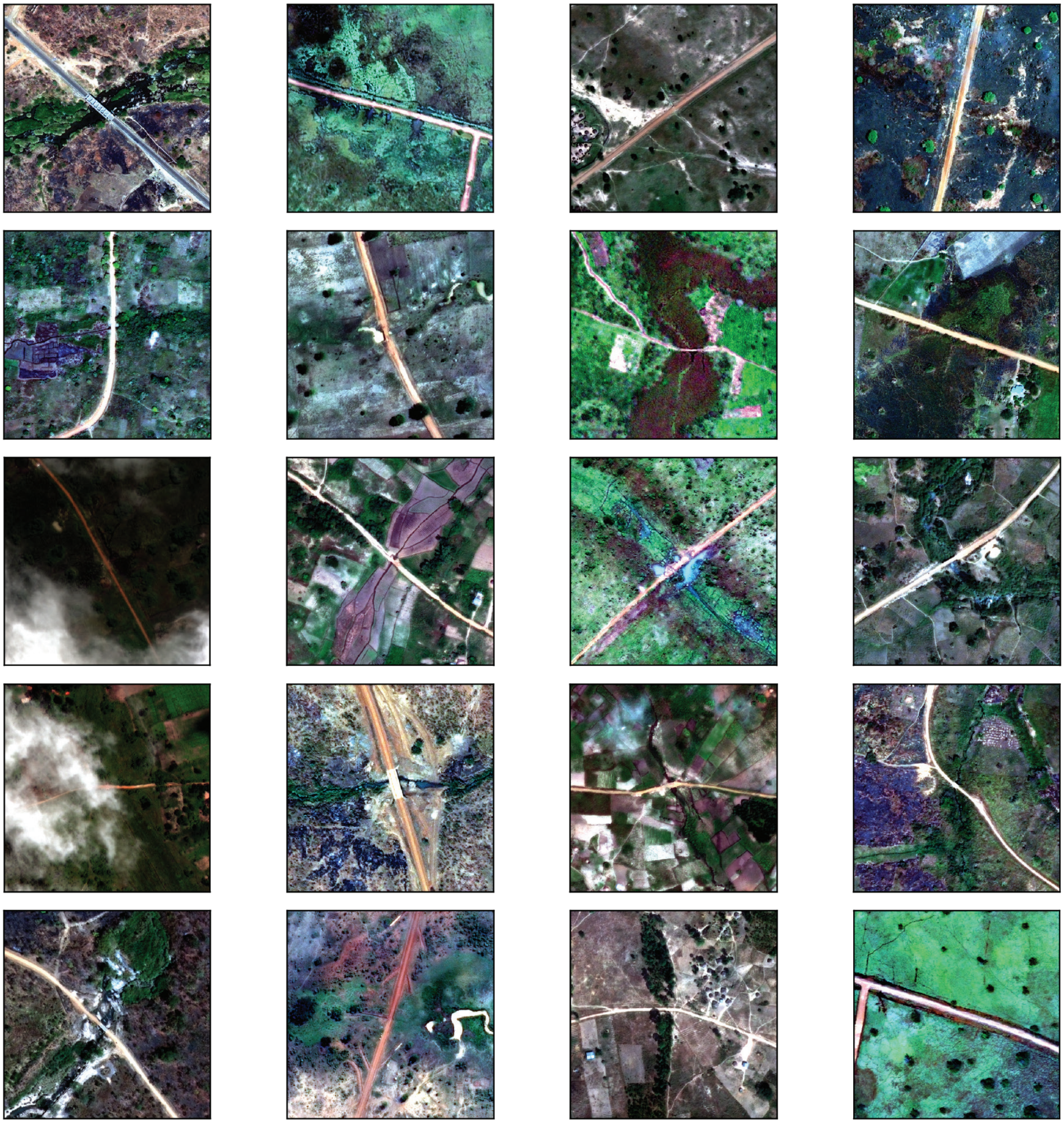


Figure 3. Sample images showing the diversity of the dataset.

Due to the small cardinality of the dataset (n = 751, with only n = 339 bridges), as opposed to splitting into a traditional 70 percent train, 20 percent validation, 10 percent test, we trained and evaluated our models on an 80 percent train, 20 percent validation split and utilized K=5 Cross-Fold Validation to determine our final validation performance metrics. Moreover, due to the small cardinality of the dataset, we opted to train on all available nadir angles within the training data. The plot in Figure 4 shows the distribution of nadir angles across the dataset.

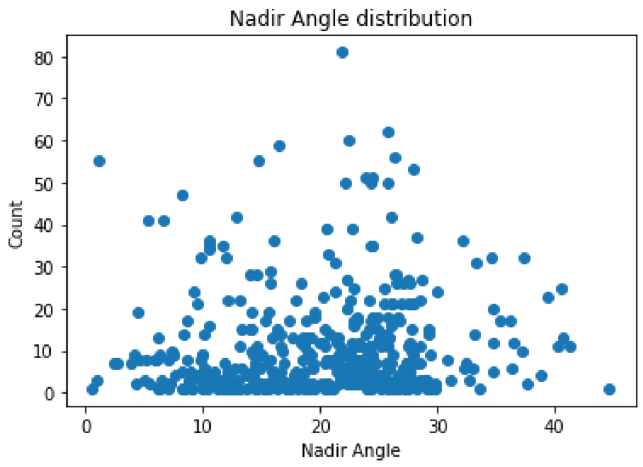


Figure 4. Distribution of nadir angles across the dataset.

To fit the images onboard an accelerator for training and inference, we chipped the image tiles to 256mx256m in order to generate 512x512 pixel images that retain the source spatial resolution (i.e., we do not down-sample the image tiles).

Model Topology Selection

Given the problem statement, we had two ways to frame the problem as a deep learning task: object detection and segmentation. In object detection, classes of objects (e.g., people, vehicles, etc.) are detected within images along with their locations in the image, typically demarcated with bounding boxes. For segmentation, each pixel in an image is classified based on whether it portrays, for example, a road, waterway, bridge, or background.

Object Detection

For the object detection task, we discard the waterway and road labels and try to train a model solely to detect bridges, by generating a bounding box encasing the bridge, as shown in Figure 5.

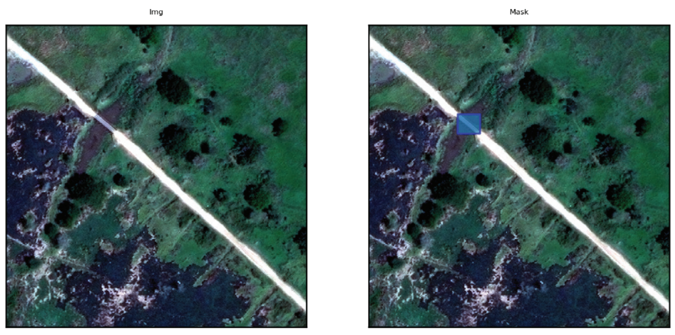


Figure 5. Generation of a bounding box around a bridge.

For this particular framing of the problem, we tested Mask-RCNN[1] and Fast-RCNN[2] models, with poor results. In the future, we could try revisiting object detection, but try using models that are not based on Region Proposal Networks. It is our belief that RCNN models struggle with this type of sparse object detection because of requiring proposals and anchors to be generated from a fixed set, meaning that if the anchors didn't cover the object, the model will not detect it.

Given that we are looking for at most two to three objects in the scene, we have to utilize a large number of proposals to guarantee coverage, but this comes at a computational and performance cost as the model needs to process and combine more AOIs than it needs.

In the future, we would like to try single shot learning variants such as YOLOv3[3]. Single shot methods are faster, do not suffer from region proposal problems and can match RCNN performance even on dense tasks, given a good enough feature extractor[4].

Segmentation

The alternative framing of the problem, and the method used for the final model, was to treat bridge detection as a semantic segmentation problem. For this task, the model generates a prediction at each pixel, classifying whether the pixel belongs to a particular class.

We treated the task as segmenting across four classes: road, water, bridge, and background. An image that follows this segmentation schema is shown in Figure 6. Since we also procured annotations for culverts, we tested model performance when segmenting road, water, and background, as well as a combined "road crossing" class including bridges and culverts.

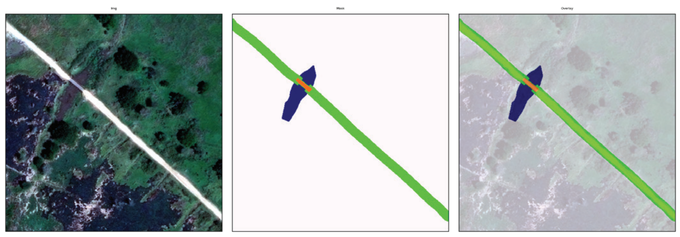


Figure 6. Semantic segmentation of pixels across classes.

For the model architecture, we tested two variants:

- **U-Net**[5], an older model that shows high performance on semantic segmentation of data with high spatial resolution and a low number of classes, as seen in medical imaging and other satellite image tasks.
- **A Fully Convolutional Network**[6] **with a Resnet-50 backbone**, a more modern approach that has achieved high performance on similar tasks.

The models were tuned to fit the specific task, and pre-training was not used because unlike standard image datasets consisting of images with three color channels (e.g., RGB), satellite imagery typically contains additional spectral bands. In this work, the satellite imagery featured four spectral bands, including RGB and near-infrared.

Model Performance

Polygon Extraction

While we trained our models for semantic segmentation, ultimately the task is bridge identification, which requires converting from a segmentation mask into a latitude-longitude (lat-long) point or polygon that localizes the bridge. To conduct the bridge identification, we extrapolate polygons from the pixels in our produced segmentation masks by checking for pixel connectivity, as diagrammed in Figure 7.

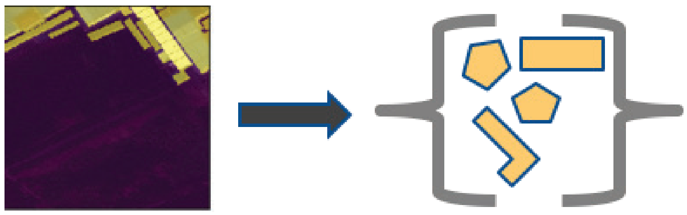


Figure 7. Transforming semantic segmentation into polygons.

These polygons, extracted from pixel coordinates (x-y), are then converted into lat-long coordinates by utilizing the source image's affine transform, a piece of metadata that provides the mapping between lat-long and x-y pixels for a specific image. Given the variances in sensors, nadir angles, cloud cover, and time, there is no off-the-shelf mapping between pixels and lat-long; depending on the image, a pixel can represent different measures, which is why we always use the source image's transform.

Loss & Optimization

During training, we optimized the models' weights relative to the Categorical Cross Entropy loss. We utilized the Adam[7] optimizer and utilized a base learning rate of 1e-4 with three different linear learning rate schedules to train the model. Two of the schedules utilized the same gamma coefficient of 0.5 but mutated the learning rate at steps of 25, 50. The third variant was trained without mutating the learning rate.

Metrics

During training, we monitor the cross-entropy loss, as well as pixel-wise DICE coefficient, and the polygon F1. DICE values are computed at the image level and averaged across the dataset. This metric is helpful to monitor but is not a perfect fit for the object extraction task, as it only measures individual pixel performance and is not tied to the bridge identification.

In order to measure actual model performance, we utilized the SpaceNet F1[8] metric in order to assess the bridge identification. We chose to utilize this metric as it evaluates the performance of the actual extracted polygons, in contrast to traditional semantic segmentation metrics that only measure the performance of pixel-wise classification at the image level, without regard for instances of objects extracted.

In order to compute the F1 metric, we take the generated segmentation masks, polygonize them, and evaluate the performance of the generated polygons with the ground-truth bridge polygons by utilizing the algorithm depicted in Figure 8. Note that this metric evaluates over the entire dataset, and not just an average across images.

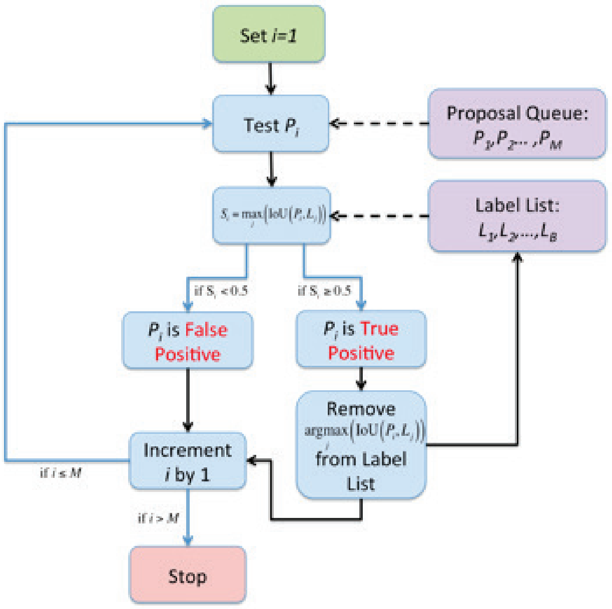

Set i=1
Test P_i
Proposal Queue: $P_1, P_2, \ldots, P_M$
$S_i = \max_j(\mathrm{IoU}(P_i, L_j))$
Label List: $L_1, L_2, \ldots, L_B$
if $S_i < 0.5$
if $S_i \geq 0.5$
P_i is False Positive
P_i is True Positive
Remove $\operatorname{argmax}_j(\mathrm{IoU}(P_i, L_j))$ from Label List
Increment i by 1
if $i \leq M$
if $i > M$
Stop

Figure 8. Evaluation workflow.

We report two variants of the F1 metric. The *IoU* variant is the same as the documented SpaceNet F1 metric referenced above—we compute Intersection Over Union values between ground truth and proposed bridges, using a 0.5 IoU as a threshold for a True Positive. In contrast, the *inclusion* variant is meant to be more reflective of the workflow intended for these results.

Per our discussions with the American Red Cross, rather than delivering polygons, we will deliver points. In turn, rather than evaluating the polygons, the *Inclusion* F1 tallies a True Positive if the centroid of a generated polygon falls within a ground truth polygon. This inclusion metric allows us to focus less on perfectly fitting polygons, and more on localizing them.

Training Results

Table 3 shows the model performance on the K=5 Cross Validation set. For each row, five different models were trained with five different random seeds for a total of 200 epochs. Each model was trained on an 80 percent subset of training data, and validated on the remaining 20 percent. All model inputs were normalized utilizing pre-computed mean and variance for the dataset.

We also utilized random rotations, randomly selected from 15-degree intervals from 0-365 and random vertical and horizontal flips to capture the rotational invariance required for working with satellite imagery. For the values reported below, each row is the average performance for the listed model architecture and training regime across the five folds.

Table 3. Model performance on the K=5 Cross Validation set.

	Bridge F1 - Inclusion	Bridge F1 - IoU	Bridge DICE	Road DICE	Water DICE
UNet - S=0	**0.4130**	0.1150	0.4150	0.7959	0.3955
UNet - S=25	**0.3367**	0.1097	0.3326	0.8017	0.4034
UNet - S=50	**0.3719**	0.1034	0.4022	0.8084	0.3905
FCN - S=0	**0.4487**	0.1574	0.4634	0.8180	0.3943
FCN - S=25	***0.4605***	***0.1852***	***0.4914***	***0.8296***	***0.4269***
FCN - S=50	**0.4585**	0.1850	0.4999	0.8281	0.4440

Analysis & Model Selection

Based on the quantitative results shown above, we opted to select the Fully Convolutional Network with a ResNet 50 backbone and retrained it on the entirety of the training data, utilizing the step = 25 training regimen and all the same transforms.

Predictions on Southern Uganda

Imagery Selection

In order to run our trained models on Southern Uganda, we had to first create a new imagery dataset of candidate locations. Given that the Ugandan government and OpenStreetMap (OSM) have many of the existing bridges labeled, we first set out to generate spatial latitude-longitude candidate points where we think there might be a previously unknown/unlabeled bridge location.

To generate candidate locations, we began with a bounding box around Southern Uganda (two bounding boxes in practice, for southwest and southeast separately, as illustrated in Figure 9).

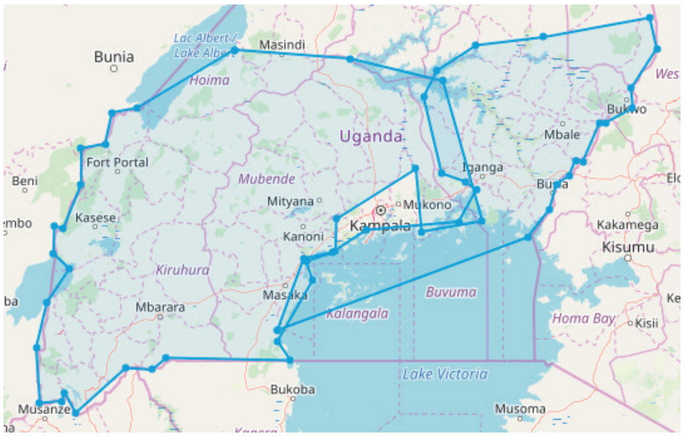


Figure 9. Two polygon regions used to search for candidate bridge locations.

To reduce computational complexity, the bounding box was divided into smaller areas. For each area, a Python API was used to query Overpass for all highways and waterways contained within that area, as well as any labeled bridges in the area, as illustrated in Figure 10.

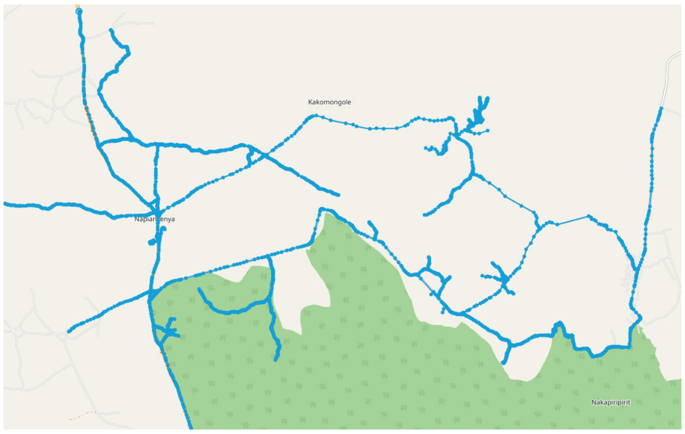


Figure 10. Example bounding box query result showing highways and waterways.

Each waterway returned was compared to each highway returned, and any intersection points (latitude/longitude) of the two feature types were collected as potential candidate points.

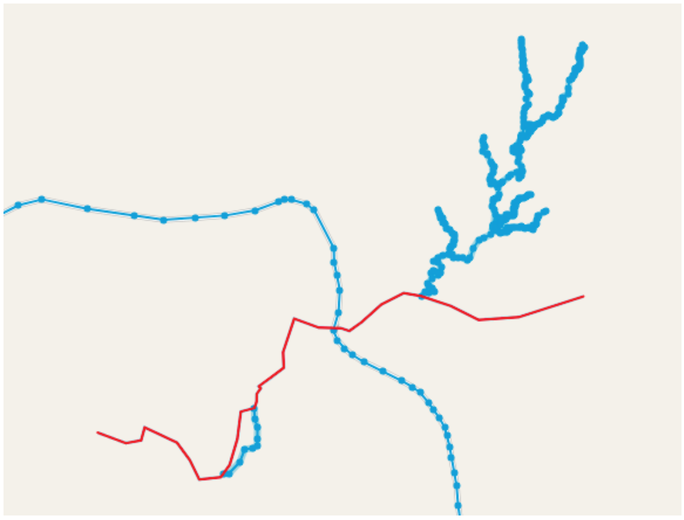


Figure 11. Identified intersection of highway (blue) and waterway (red).

Candidate points were tested to see if they were located within ~30 meters of an already known bridge location, as shown in Figure 12. Known bridge locations were derived from a Uganda government dataset and OSM features returned with property 'bridge=yes'. Any candidate points within 30 meters of known bridge locations were discarded.

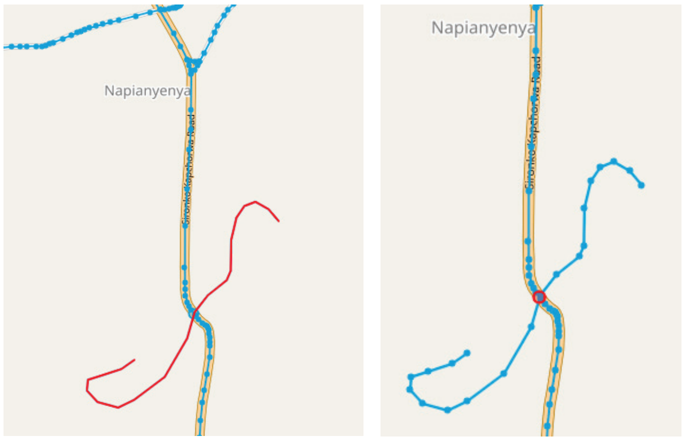


Figure 12. Left: identified highway/waterway intersection, Right: existing bridge location shown in red, intersection filtered from candidate locations.

In this methodology, highway and waterway types were not used to filter the candidate points (aside from highways labeled 'bridge=yes'); therefore, minor highway and waterway categories were included in the candidate points. Employing additional filters on highways and waterways would improve the likelihood of candidate points containing verifiable bridges.

Once we determined the candidate points, we created 256mx256m bounding boxes about these points and queried Digital Globe's GDBX Spatial Data platform to pull images that contained the anchor boxes. As part of the query to GBDX, we filtered out images with more than 35% cloud cover, an off-nadir angle larger than 30 degrees, and restricted to images captured since January 2017.

From this query, we downloaded a total of 1227 unique images. We ran the model through and extracted a total of 70 unique bridges, shown in Figure 13.

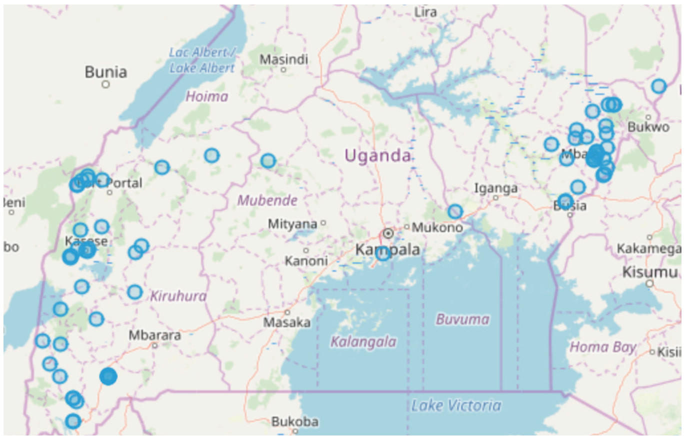


Figure 13. Bridges extracted by the model.

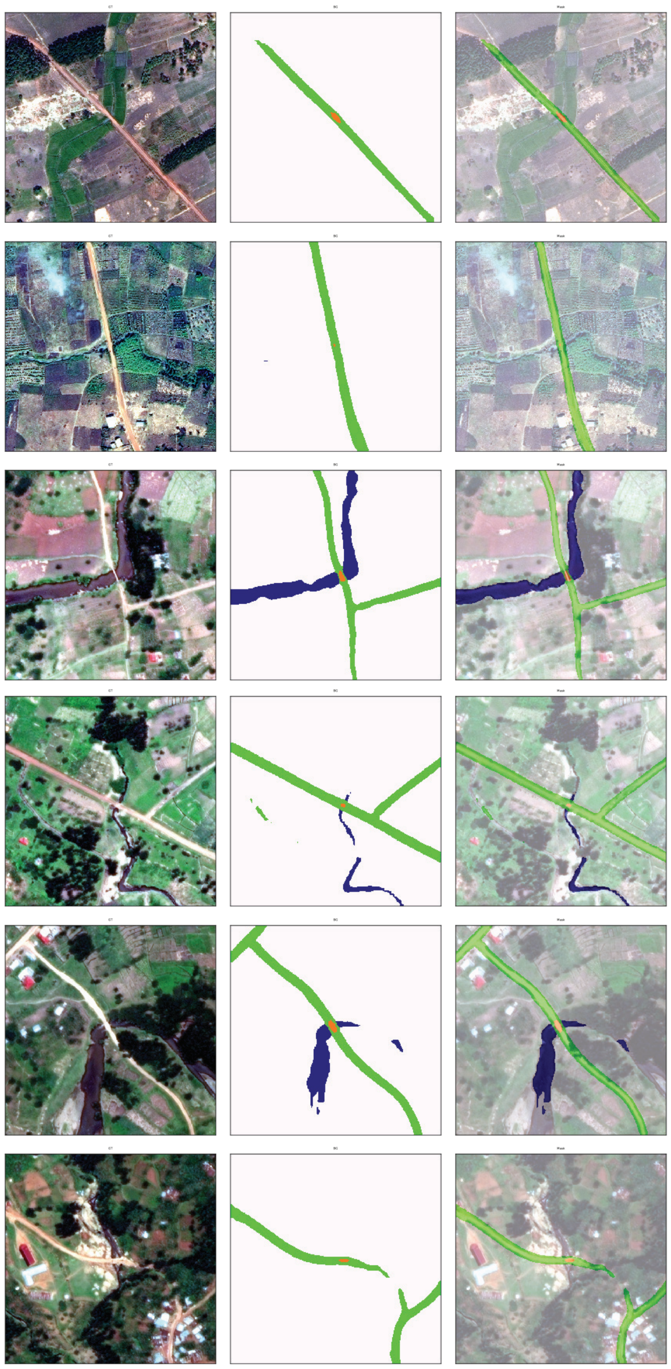
Sample Predictions

Figure 14 shows the raw segmentation prediction results of the model. For all the images, we display the input satellite image on the left, the predicted mask in the middle, and the mask overlaid on the image on the right.

The color map is as follows:

- **Background**: white
- **Road:** green
- **Water:** blue
- **Bridge:** orange

Successes



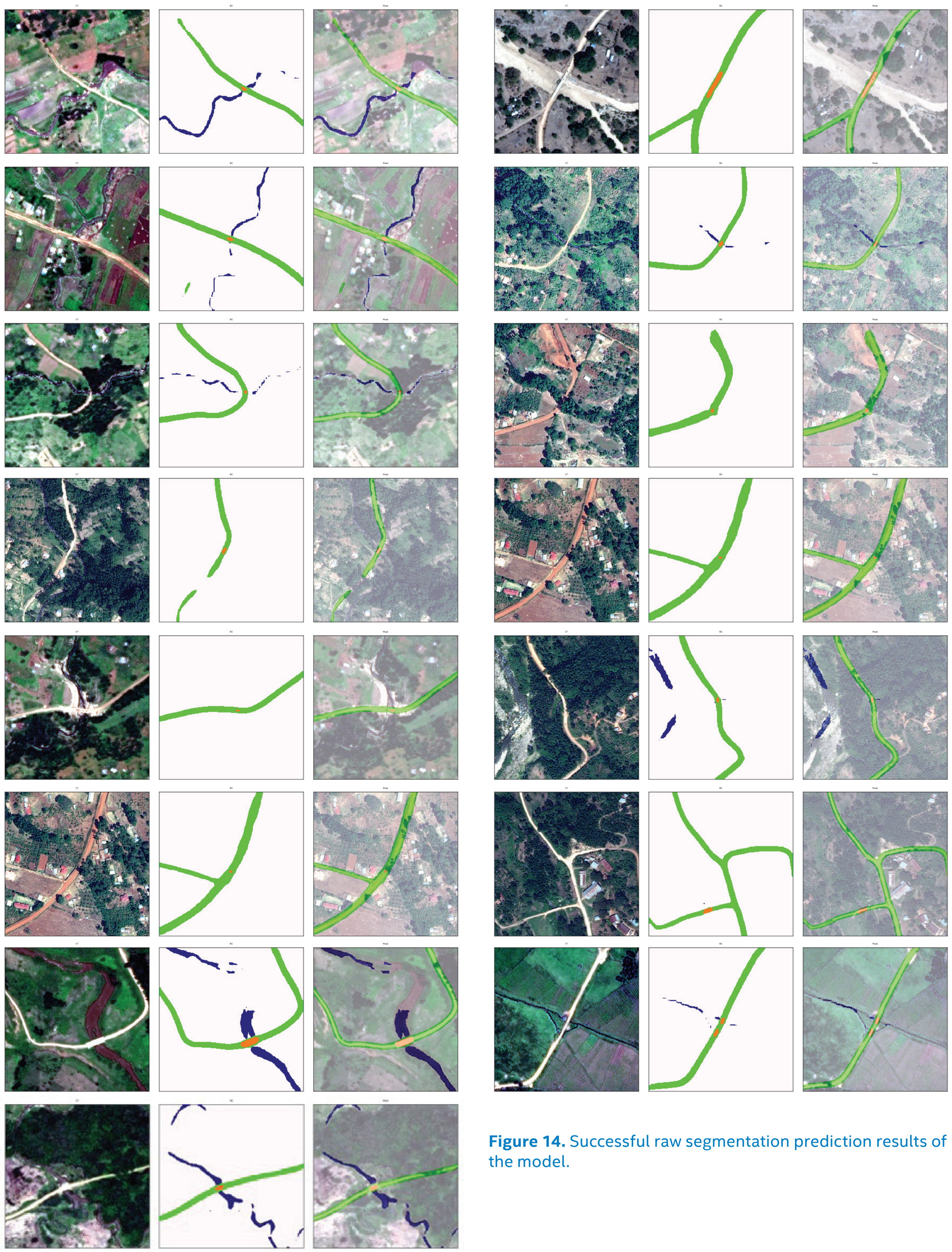


Figure 14. Successful raw segmentation prediction results of the model.

Failures

While the above predictions successfully extract the bridges in the scene, the model is far from perfect. As shown in Figure 15, the model struggles in particular with road types not found in the training data, as well as in situations where clouds or atmospheric noise hinder the signal.

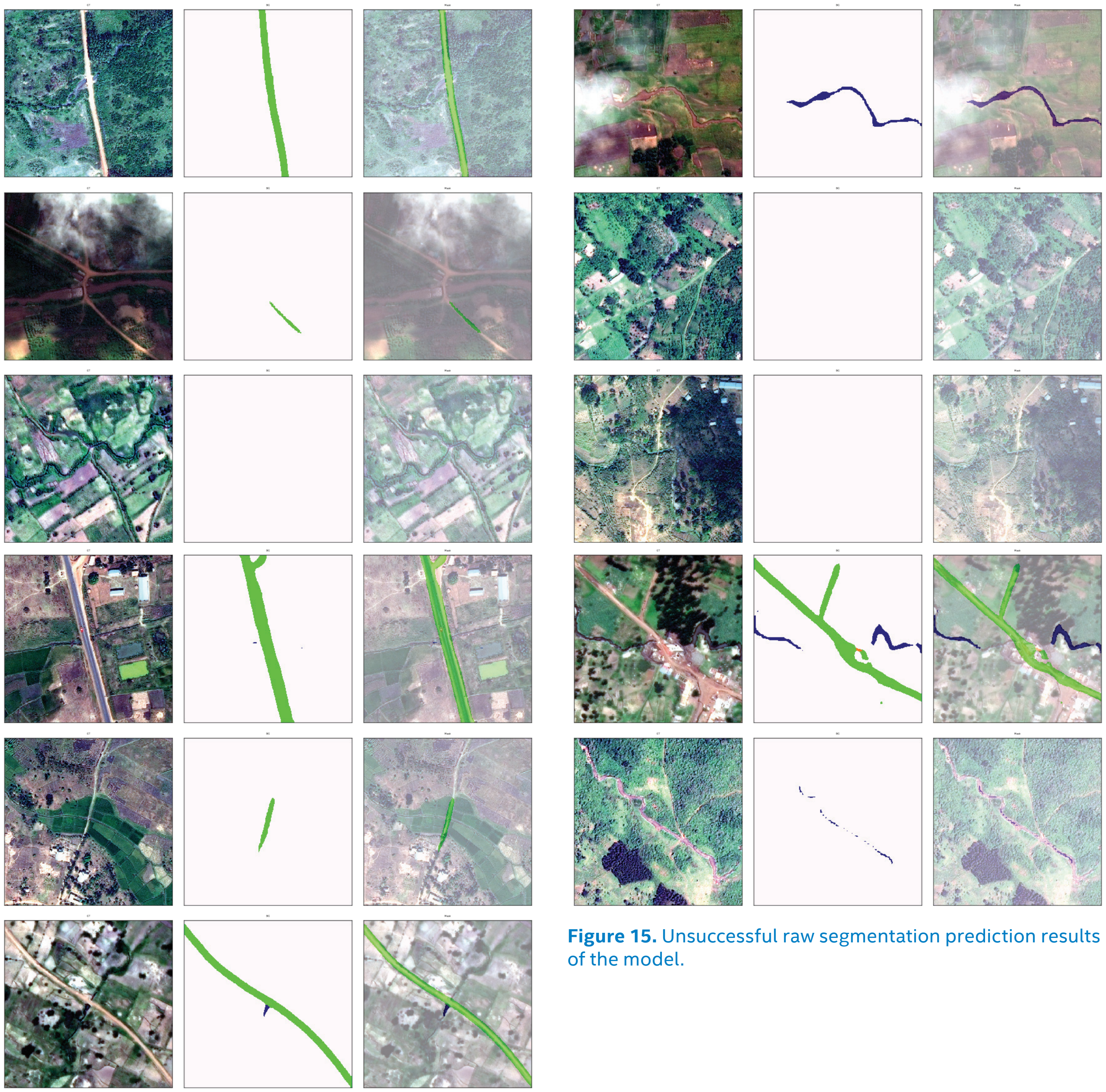


Figure 15. Unsuccessful raw segmentation prediction results of the model.

Note that the majority of the failures are the result of misalignment with the training data. In particular, the missed roads fall below the threshold of what was labeled as a road for this project. The small dirt and grass paths were not fed to the model as road samples, so it is not surprising it fails to segment them. In the images above, there are also cases of unsegmented bridges, with segmented roads. It is unclear whether the non-segmented bridges are bridges or culverts, which we did not train to detect.

[1] He, K., Gkioxari, G., & Dollár, R. (2017). Mask R-CNN. *arXiv e-prints*, arXiv:1703.06870.

[2] Girshick, R. (2015). Fast R-CNN. *arXiv e-prints*, arXiv:1504.08083.

[3] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv e-prints*, arXiv:1804.02767.

[4] Huang, J., Rathod, V., Sun, M., Korattikara, A., Fathi, I., Wojna, Z., Song, S., & Murphy, K. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv e-prints*, arXiv:1611.10012.

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, arXiv:1505.04597.

[6] Evan Shelhamer and Jonathan Long and Trevor Darrell (2016). Fully Convolutional Networks for Semantic SegmentationCoRR. *arXiv e-prints*, arXiv:1605.06211.

[7] Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, arXiv:1412.6980.

[8] Hagerty, P. (2016, Nov 10). The SpaceNet Metric. Retrieved Oct 10, 2019. https://medium.com/the-downlinq/the-spacenet-metric-612183cc2ddb.