Satellite Image Building Detection using U-Net Convolutional Neural Network

Liam Coulter, Teague Hall, Luis Guzman, Isaac Kasahara

***Abstract*—Convolutional Neural Networks (CNNs) are at the forefront of current image processing work in segmentation and object detection. We apply a U-Net CNN architecture to a satellite imagery data set to detect building instances in birds-eye view image patches, since the U-Net has been shown to achieve high accuracy with low complexity and training time. We present our results in the context of the SpaceNet Building Detection v1 Challenge, and compare scores against the winning algorithms. Results from our implementation demonstrate the U-Net architecture's ability to successfully detect buildings from satellite images. We discuss the challenges and limitations of using CNNs and the U-Net in particular for image segmentation, despite the success of our implementation.**

I. Introduction

THE availability of high-resolution satellite imagery has motivated attempts at automatically generating civil feature map systems. These maps should contain information about man-made structures such as buildings and roadways which can then be used for applications in civil planning, humanitarian response efforts, and more. An automated approach for generating these feature maps is a cheaper, faster, and potentially more accurate alternative to manual map creation. In addition, an automated approach reduces the cost and effort to update feature maps, as old manual map systems become outdated. This could be very valuable for geographic regions experiencing fast population growth, or for updating maps after natural disasters.

SpaceNet is an open innovation project hosting freely available image data with training labels for machine learning and computer vision researchers. The aim of SpaceNet is to aid advancements in the field by offering high quality labeled training and testing data, and corresponding challenges. Historically, satellite image data was privately managed or government controlled, making it difficult or impossible for researchers to use for algorithm development. SpaceNet has compiled extensive image sets along with labeled training data which is made free to the public in the hope of advancing automated algorithms. The majority of activity in these algorithms has been dominated by machine learning approaches. In addition to providing free data sets, SpaceNet organizes targeted competitions that look at specific sub-problems within satellite image mapping. These competitions intend to motivate interest and advancements in the topic.

The first SpaceNet challenge ("Building Detection v1") focuses on building detection from optical and hyperspectral images. This challenge provides 3-band and 8-band satellite images along with bounding polygon vertices denoting the presence and location of buildings. The objective is to automatically generate a set of building polygons that most closely approximates the manually generated polygons. In order to assess the quality of automatically generated building polygons or masks, SpaceNet uses several metrics including the Intersection over Union (IoU), precision, recall, and F1 scores described in the next section. Subsequent SpaceNet challenges focused on extracting other features from satellite imagery such as roadways (challenges 3 and 5), generating building detections from other types of data including Synthetic Aperture Radar/SAR (challenges 4 and 6), and making high-level inferences about route transit time or urban development (challenges 5 and 7). We focus only on building detection here.

Our project solves the SpaceNet Building Detection v1 challenge by using a U-Net Convolution Neural Network (CNN) architecture for semantic segmentation. The U-Net architecture was designed for biomedical image segmentation and shows a high degree of accuracy with low complexity and low training time. In addition, the U-Net employs "skip connections" instead of fully connected layers, which allows it to preserve finer details in the final segmentation map. We discuss these more in detail later.

The field of image segmentation has been active in recent years, aided by advancements in machine learning methods. Application areas range from biomedical imaging (for which the U-Net was developed) to autonomous vehicles. Semantic segmentation differs from instance segmentation in that we do not care about separating instances of the same class in the final segmentation map, but rather assigning a class to each pixel in the image. For example, a semantic segmentation task would be to classify every pixel in an image as either a building or not a building; an instance segmentation task would be to distinguish separate occurrences of buildings in an image, and represent that in the final segmentation map [2]. The current state-of-the-art in semantic segmentation consists mainly of machine learning methods since the development of AlexNet [3], specifically CNNs and U-Net variants, since fully connected CNNs tend to have trouble with fine details. Two recent U-Net variants confirm this difficulty, underscoring the importance of using skip connections instead of fully connected layers [4], [5].

This report is organized as follows. Section II provides a description of the SpaceNet Building Detection v1 challenge, with discussion of data, labels, solution requirements, and evaluation metrics, as well as a detailed description of the U-Net architecture and our specific implementation including points of difficulty in our implementation and training. We

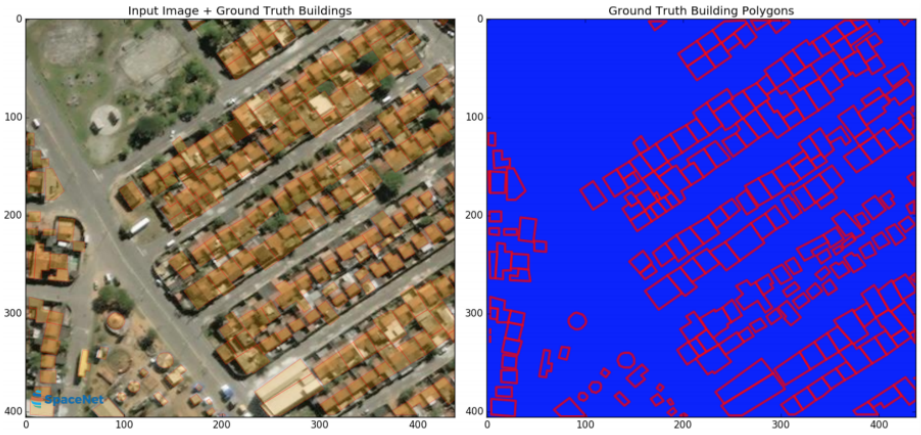


Fig. 1: Rio de Janeiro image tile with building polygons superimposed (left) and associated building label polygons (right) [1]. Building polygons have varied shapes.

also describe our implemented evaluation metrics, and how they may differ from those used by actual contestants in the Building Detection v1 challenge. In section III we present the results from our U-Net implementation, and compare them to the winning results from the SpaceNet Building Detection v1 challenge. Section IV contains a discussion of the strengths and weaknesses of our approach, and a discussion of the limitations of the U-Net architecture. Finally, we draw conclusions from our experimentation and analysis in section V.

II. Methods and Theory

A. The SpaceNet Building Detection v1 Data

The SpaceNet Building Detection challenge v1 data was collected from the WorldView-2 satellite and covers a 2544 square kilometer region centered over Rio De Janerio with 50cm resolution [1]. The dataset is broken into individual images with size 200m×200m. These image "tiles" are between 438 and 439 pixels wide and between 406 and 407 pixels tall, and the 3-band RGB images have been pan-sharpened. The training data set consists of approximately 6000 such 3-band RGB image tiles. Along with the processed images, SpaceNet provides building labels in the form of sets of polygon vertices denoting the corners of buildings in each image. These building label polygons were generated using rough automated techniques and then detailed by hand. This labor intensive process resulted in approximately 380,000 buildings for the entire Rio de Janeiro geographic area, over the 6000+ image tiles. The building labels polygon data are provided in geoJSON files which correspond to each image tile. In the original SpaceNet Building Detection v1 challenge, participants used this entire labeled data set for training and validation, and submitted geoJSON files with proposed building detections from images in a separate unlabeled testing data set. However, since we do not have access to labels for the testing set and thus are unable to determine our performance from this set, we use the labeled training data set for both training and validation. We employ a split of 80% for training and 20% for validation, and report our accuracy numbers from the validation set, which was not used for training. Figure 1 shows a single image tile along with the corresponding building label polygons.

B. SpaceNet Evaluation Metrics

The SpaceNet challenges use several metrics to assess building prediction quality: Intersection over Union (IoU), precision, recall, and F1 score. The precision, recall, and F1 metrics are based on the IoU for individual buildings in an image, and can be computed for an individual image or across several images as a weighted average. IoU is a measure of how well predicted building label polygons or masks overlap with ground truth polygons, and is commonly used to assess semantic segmentation results. Let A and B be two sets of points; in our application, A represents the set of pixel locations contained within the bounds of the ground truth building label polygon and B represents the set of pixel locations contained within the bounds of a predicted building location polygon or mask. The IoU then, is

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

where $|\cdot|$ represents set cardinality. It is apparent that IoU is shift-invariant, and independent of the size of given sets A and B, since the union normalizes the IoU result to within the interval [0, 1]. A perfect IoU score of 1 happens when a predicted building label polygon exactly coincides with the ground truth label polygon, and an IoU score of 0 occurs when there is no overlap at all. The IoU metric can be limiting for small objects where low pixel resolution can greatly influence the result. Further, the IoU doesn't account for the shape of predictions relative to ground truth polygons; as long as the size of the intersection and union are the same, we could have any number of (incorrect) predicted building shapes, and any sort of overlap, and the IoU would give the same score. Finally, the IoU doesn't account for building instances where there is a complete mismatch between the predicted and ground truth polygons; the predicted polygon could be very close (but

not intersecting) the ground truth polygon, or the predicted polygon could not exist. In either case though, the IoU would give a score of zero. So, the IoU by itself does not give a complete picture of the quality of prediction results.

To generate an overall evaluation metric for an entire image or group of images, it is necessary to combine the results of each IoU. This is achieved by first establishing an IoU detection threshold of 0.5; any individual IoU result greater than this threshold is considered a successful detection. For example, if there is one building in an image and the IoU score is 0.4, we do not report a building detection since the score is below the threshold. In the case where there are several buildings in an image, we search over all predicted building polygons for each ground truth polygon and compute the IoU score between the current ground truth and all predicted polygons. The largest IoU is chosen as the correct match for the current ground truth building, and we check against the threshold to see if the current prediction constitutes a detection. If we have a detection, we say we have found a true positive, and if not we say we have found a false positive. We then remove the current predicted and ground truth polygons from the list, and move on to the next ground truth polygon (cycling through all remaining predicted polygons again). This process is shown in figure 2.

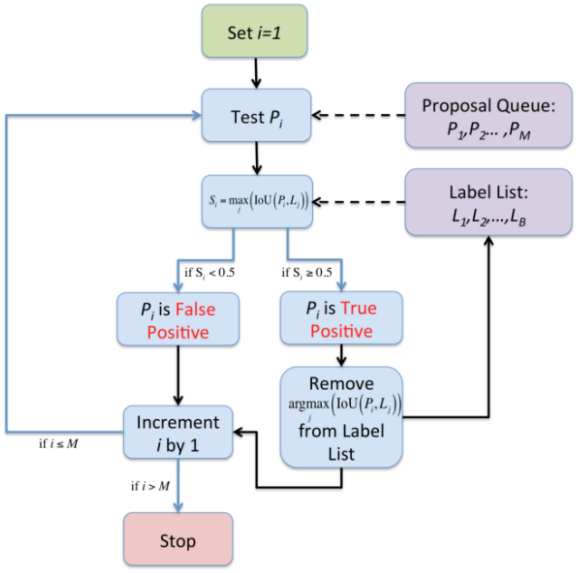


Fig. 2: Assessing predicted building polygons using IoU [6].

Once we have determined the number of true positives, false positives, and false negatives (where we have a ground truth building but no predicted building polygon), we define the *precision* and *recall*. *Precision* is defined as the number of true positive predicted building detections divided by the total number of predicted buildings. That is,

$$\text{Precision} = \frac{|\text{True Positives}|}{|\text{Total Predicted Buildings}|} \tag{2}$$

This metric conveys information on the algorithm's ability to detect buildings while avoiding false detections; a higher precision means more detections are correct, and fewer are incorrect. Precision alone is not sufficient to assess prediction quality because it is unaffected when the algorithm fails to detect buildings. *Recall*, then, is the fraction of true building polygons that are detected by the algorithm. That is,

$$\text{Recall} = \frac{|\text{True Positives}|}{|\text{True Positives}| + |\text{False Negatives}|} \tag{3}$$

$$= \frac{|\text{True Positives}|}{|\text{Ground Truth Polygons}|} \tag{4}$$

Recall conveys information about the algorithms ability to detect all objects in the image and thus gives a lower score when objects are missed.

To give an overall picture of the quality of a given prediction, we combine the precision and recall into the F1 score:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{(\text{Precision} + \text{Recall})} \tag{5}$$

Since precision and recall range from 0 to 1, so does F1; the 2 in the numerator serves as a normalizing factor.

Figure 3 shows an example image with predictions and ground truth polygons. The ground truth polygons are shown in blue, and predictions are either green or yellow, signifying an IoU score above or below 0.5, respectively.

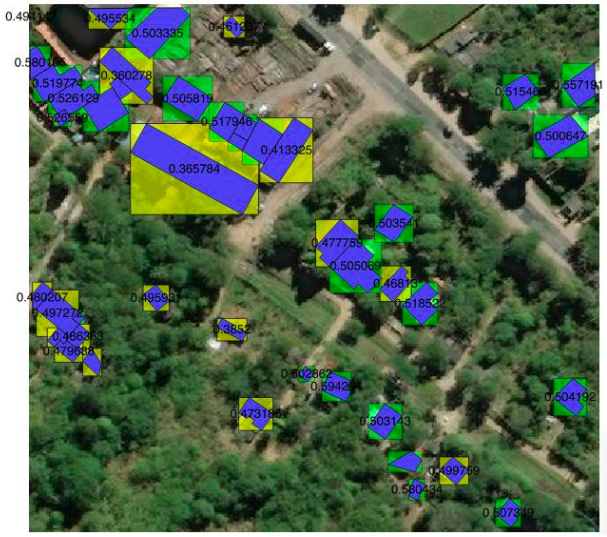
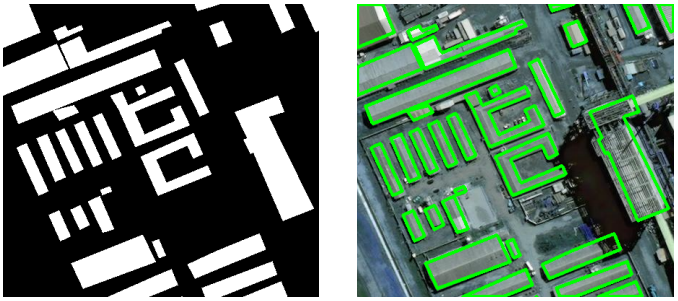


Fig. 3: Performance evaluation of image tile using IoU [6].

C. Image Pre-Processing

Since SpaceNet data was in the form of TIFF images and corresponding geoJSON building polygon labels, we had to convert these to a format that was easier to work with in the context of training a CNN. We made use of the publicly-available SpaceNet utilities [7] to convert the building polygons in the geoJSON files from latitude and longitude to image coordinates in pixels, and save the result as a binary image mask with 0 denoting that there is no building in that pixel location and 1 denoting the existence of a building.

In order to make use of the IoU metric, we had to be able to identify how many building mask polygons existed in both the ground truth and predicted building masks. For this we



(a) Building Mask (b) Building Contours

Fig. 4: Building mask and resulting polygons from `cv2.findContours`.

used the openCV function `findContours` on the image masks, which returns a list of objects with polygon corners in pixel coordinates for each polygon in the image. Figure 7 shows the building mask and result of `cv2.findContours`. One possible discrepancy between this and the polygons in the geoJSON files is that collections of buildings which are very close together may appear in the building mask as one long building. Since we used the same image masks for training and this occurrence was not very frequent, we decided this was an issue we did not need to address.

Finally, we normalized the RGB images to [0, 1] for numerical stability when training our U-Net. We also standardized the size of the images by removing a column and/or row where necessary so that all images had size 406×438, and then resized the images to 286×286.

D. U-Net CNN Architecture

Our building detector is based on the U-Net CNN architecture. This architecture was originally developed for biomedical image segmentation, to distinguish cells in an image [8]. Since then, this architecture has been applied many times to image segmentation problems in a variety of applications, and shows robustness and accuracy without long training times or high complexity. Thus, it is an appropriate method for our building detection problem.

The U-Net architecture gets its name from the contraction and subsequent expansion of feature maps, conceptually shown as a U shape as in figure 5. The contraction or encoding path compacts the input image signal to a map of feature channels, and the expansion or decoding path then generates a segmentation mask through up-convolution, which reduces the number of feature channels at each step. This expansion path operates in a feed-forward manner.

Figure 5 shows the U-Net architecture for a 512×512 input image. The contraction path resembles a typical CNN architecture found in many machine learning applications where the image is compacted to a feature map representation with additional channels, termed feature channels. At each level of the contracting path, the signal undergoes a set of two 3×3 convolution operations which reduce the signal size if padding is not used. Each convolution operation is followed by a rectified linear unit (ReLU) activation function, defined by

$$\text{ReLU}(x) = \max\{0, x\} \tag{6}$$

to give an activation map. Then, a 2×2 max pooling operation is applied with a stride length of 2 in order to downsample the signal to a still smaller size. The 2×2 max pooling operation involves taking the max of each 2×2 patch of the signal, as a way of downsampling the signal while keeping the elements with the largest activation result. At each level of the contraction path, the feature channels are doubled. At the base of the U we have our most contracted signal feature map, with dimension $1024 \times 30 \times 30$, meaning we have 1024 feature channels. The intuition behind feature channel doubling is to allow the network to build more complex features in subsequent layers. Since the feature channels of the previous layers act as primitives for the construction of new features, doubling the number of channels allows for a balance between quantity and semantic meaning of the learned features.

The right side of the U-net is an expansive path where each level involves up-convolving the feature channels followed by two 3×3 convolutions and ReLU. Up-convolution is an upsampling technique where a learned convolution mask is multiplied by each cell of the spatially lower-resolution signal in order to obtain a higher-resolution signal. This differs from traditional up-sampling where an unlearned mapping is used to increase signal resolution. Additionally, each level of the expansion path is concatenated with its corresponding contraction level feature map, which is cropped to match the size of the expansion feature vector. This allows the U-net to maintain spatially localized information that would otherwise be lost during contraction. Each level of the expansion path halves the number of feature channels while increasing the spatial resolution of the signal.

The channels of the output signal will correspond to the desired number of segmentation classes, in our case one. This is achieved by a final 1×1 convolution layer that maps the output to the desired number of classes. The architecture illustrated in figure 5 corresponds to a two class segmentation problem, and hence has an output depth of two.

The U-net architecture is trained using the satellite images and associated segmentation maps (binary building masks), using a cross-entropy loss function. The original method proposed in [8] features a pixel-wise softmax over the output feature map combined with a cross entropy loss function. The softmax is defined pixel-wise as

$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{k'=1}^{K} e^{a_{k'}(x)}} \tag{7}$$

where $a_k(x)$ is the activation in feature channel k at pixel location x, $p_k(x)$ is the new value of the pixel in feature channel k at location x, and K is the total number of feature channels. We then combine this soft-max function with a cross-entropy penalty defined over the pixel region Ω as

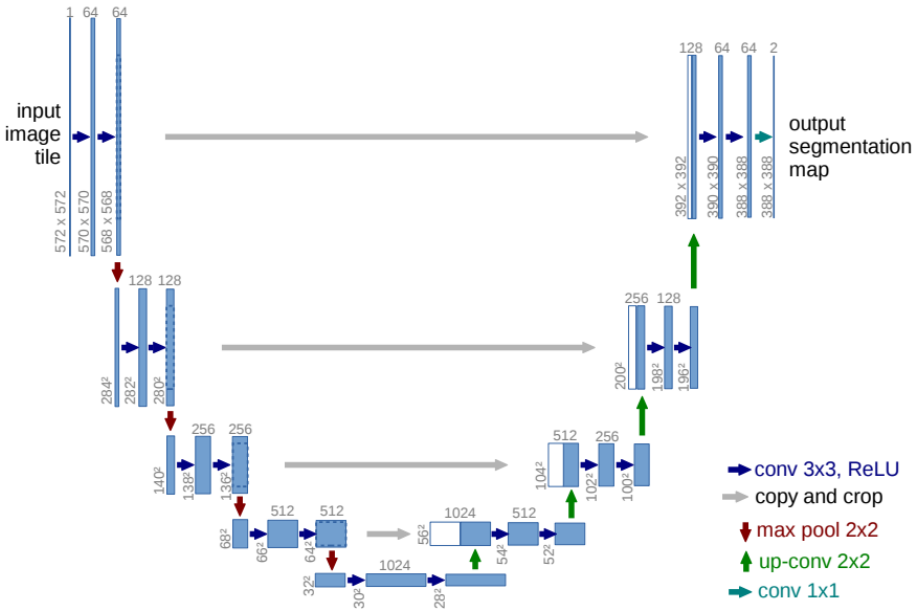


Fig. 5: U-Net CNN architecture for 512×512 input images [8].

$$E = -\sum_{x \in \Omega} p_{l(x)}(x) \log\left(p(x)\right) \tag{8}$$

where $l(x)$ is the true label of the pixel at location x, $p_{l(x)}(x)$ is the true probability distribution of each class, and $p(x)$ is the estimated distribution.

The original U-Net paper used a slightly different loss function aimed at improving performance of border pixels, defined as

$$E_{\text{weighted}} = \sum_{x \in \Omega} w(x) \log\left(p_{l(x)}(x)\right) \tag{9}$$

with

$$w(x) = w_c(x) + w_0 \exp\left\{-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}\right\} \tag{10}$$

where w_c is a pre-computed weight map intended to balance segmentation class frequencies, $d_1(x)$ is the distance from x to the neared object border, and $d_2(x)$ is the distance from x to the border of the second closest object. The parameters w_0 and σ can be chosen manually.

The weight function $w(x)$ is predefined by the ground truth segmentation mask, and is used to train the network to differentiate between instantiations of the same class. Defining the weight function this way ensures separate instances of the same object do not morph into a single instance.

We chose to use the simpler cross-entropy loss due to its ubiquitous use in the training of segmentation CNNs and its seamless integration with PyTorch, and thus we did not use E_{weighted} from equation (9).

As a final note, it is important to initialize the U-Net weights appropriately to achieve maximum occupancy of the network. Unbalanced weights can cause certain parts of the network to dominate network operation. An ideal approach would be to choose initial weights where each feature map has unit variance. For the U-Net architecture then, weights should be drawn from a zero-mean normal distribution with standard deviation $\sigma = \sqrt{2/N}$ where N indicates the number of input nodes for each neuron.

E. U-Net Implementation

We implemented our U-Net using PyTorch. Following the architecture just discussed, our U-Net had 5 encoding levels and 4 decoding levels (plus the final 1×1 convolution), and output a final binary segmentation map.

In order to load the images and building masks, we implemented a custom data loader using PyTorch that would standardize the image and mask sizes by removing a row or column where necessary, and reduced image sizes to 286×286, half that used in [8] as previously discussed. This size was chosen to speed up the training process and to avoid stability issues we encountered when training on larger images. We also normalized the RGB images to $[0, 1]$. As mentioned above, we used the SpaceNet Building Detection v1 training data for both training and validation/testing. In our data loader we used 80% of the data for training, and 20% for validation/testing.

Our implementation differs slightly from the original paper. We found that cropping the encoder feature vectors before concatenation leads to unstable gradient updates. This would lead to our network predicting `nan` at unpredictable points during the training process. We attribute this to exploding/vanishing gradients from the pixels that are deleted during the cropping process. To solve this issue, we instead chose to pad the decoding feature vector so that we are not deleting any information inside of the network. Additionally, we chose to set padding to one for each convolution layer whereas the original paper sets padding to zero. This step is particularly important for our input image size of 286×286, since without padding,

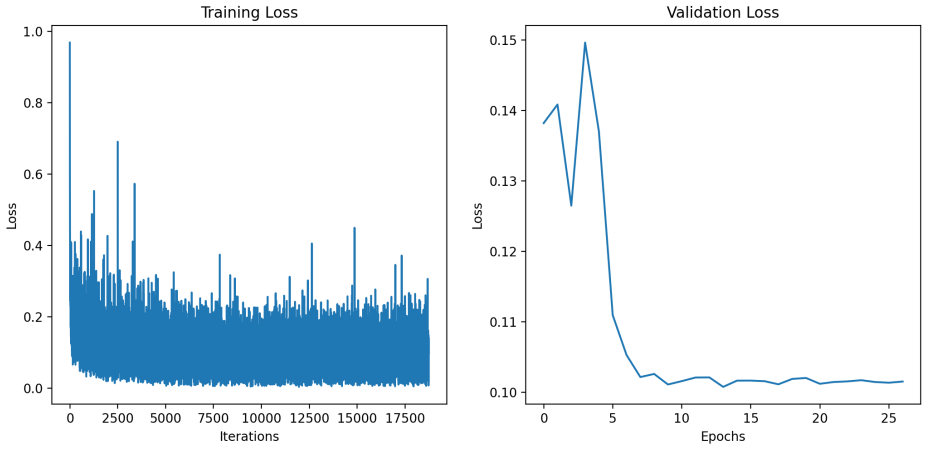


Fig. 6: Training and Validation loss for 25 epochs.

we lose significant image resolution at each convolution step. We observed this modification to also result in a more stable training process. Both these modifications to the padding result in output image sizes that are 286×286, which is conveniently the same size as our input image. We verified that our all tensor shapes in our network matched the original paper before making these modifications.

For training, we used the Adam optimizer with a learning rate of 1×10^{-4}, weight decay of 1×10^{-8} and momentum of 0.9. We used a batch size of 16, which saturated the 12GB of video memory we had available for training. We also included a scheduler, which would decrease the learning rate whenever a two-epoch plateau in the validation loss is observed. Decreasing the learning rate in this way can result in additional performance by avoiding overshoot of the network weight updates during later epochs. The update gradients are also capped at 0.1 to avoid too large of updates at each step.

We trained the U-Net for 25 epochs with the previously described softmax cross-entropy loss. In PyTorch, this loss function is implemented as `BCEWithLogitsLoss` for binary classification. Figure 6 shows the training and validation loss for the 25 training epochs. In each epoch we trained the net on all the training images, then evaluated the U-Net predictions and calculated the loss. We noted a plateau in our training and validation loss after 15 epochs, so we chose these network weights as our final model to avoid overfitting.

Figure 7 shows training images, their predicted masks, and ground truth masks, for two different training images. To generate a final binary mask, we threshold the output of our network at 0.5. All pixel values below 0.5 correspond to the label "not building" and everything greater than 0.5 correspond to the "building" label. This is unrelated to the IoU thresholding performed for assessment.

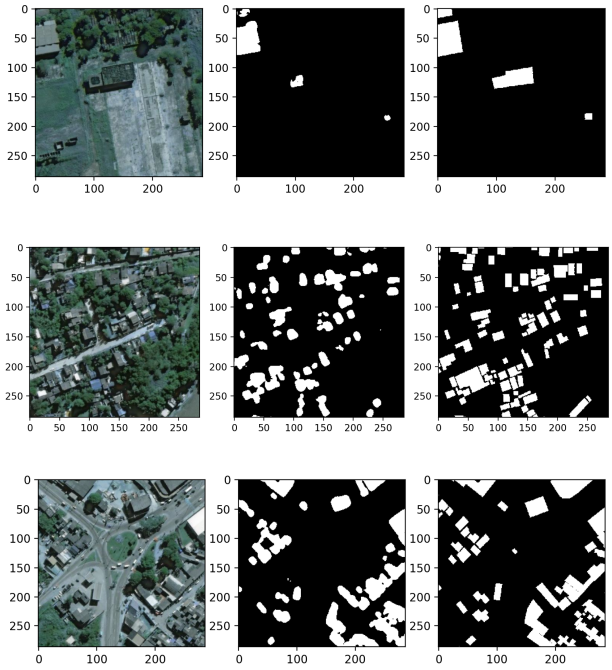


Fig. 7: Top to bottom: rural, sub-urban, urban scene. Left to right: training images, predicted building masks, and ground truth masks.

III. Results

A. U-Net Segmentation Results

From visual inspection, we observe that the image masks of our U-Net segmentation network match remarkably well with the ground truth. We qualitatively evaluated three types of scenes: rural, sub-urban, and urban (pictured with ground truth and predicted labels in figure 7), and found that our network succeeded in classifying buildings at each density. In dense urban environments, the network excludes roads and courtyards, and in rural areas, false positives are low. We noted that the largest difference between our masks and the ground truth was near building edges and when multiple buildings are classifies as one.

In order to calculate our U-Net results, we used the openCV function `findContours` to find the boundaries of predicted and ground truth building footprints, and calculated the IoU as described in section II, following the SpaceNet convention of a 0.5 threshold to determine whether an IoU constitutes a detection or not. Table I shows the true positive, true negative, and false negative results from our validation set. For our

problem, the concept of a true negative is not well defined: a true negative is where there is no building in the ground truth mask or the predicted mask, and so there is no way to "count" the locations where we have a true negative.

From table I it is apparent that our algorithm misses many buildings, having a somewhat high rate of false negatives. The number of false negatives is likely explained by the situation where many nearby buildings are detected as one. In this case, only one of the buildings would be considered a true positive, and the rest are false negatives, even though the image masks match well. Similarly, our false positive rate could also be skewed by the situation where a single building is detected as two separate masks due to occlusion. We found this effect to not be as prevalent as the many-to-one scenario. Overall our network detected 7,333 out of 33,541 ground truth buildings. Even considering these false positive and false negative results, our network performed similarly to the winning submissions.

TABLE I: U-Net Building Detection Results.

	Actual Positive	Actual Negative
Predicted Positive	7,333	17,192
Predicted Negative	26,208	N/A

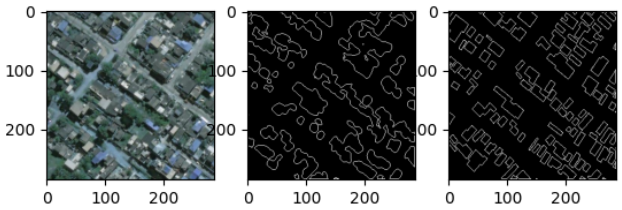
Figure 8 shows two validation images with the corresponding predicted and ground truth building polygons and masks. We see that the U-Net does a good job of finding buildings in the image along with general shapes and orientations, but sometimes fails to separate buildings which are close together. In addition, the U-Net predicts general shapes which tend to be more rounded instead of having sharper corners, as the ground truth masks do.

We calculated the precision, recall, and F1 as described above from the total true positive, true negative, and false positive building detections in table I, and computed the overall F1 score from these precision and recall numbers. We also calculated a simple average IoU score over all the validation images, by taking the IoU between all predicted building masks and all ground truth building masks for each image, and averaging the result over the number of validation images. Note that this uses a different approach from that used in taking the pairwise IoU to determine detections, and thus is not related to the precision, recall, and F1 scores. The average IoU is an indication of how well our masks match the ground-truth overall and does not consider building instance detection. Table II shows all four computed metrics for our validation set.

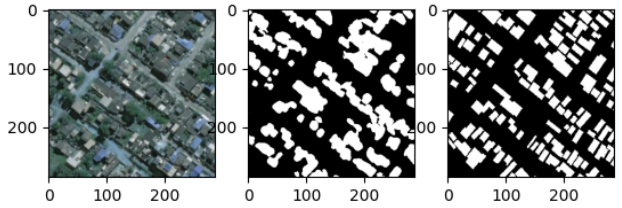
TABLE II: U-Net Building Detection Results.

Average IoU	Precision	Recall	F1 Score
0.507564	0.299001	0.218628	0.252575

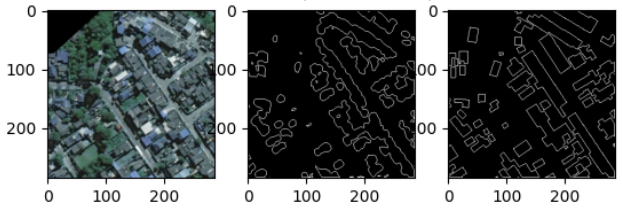
Note that average IoU score is significantly higher than the other metrics because it is not affected by the same false negative and false positive errors described earlier.



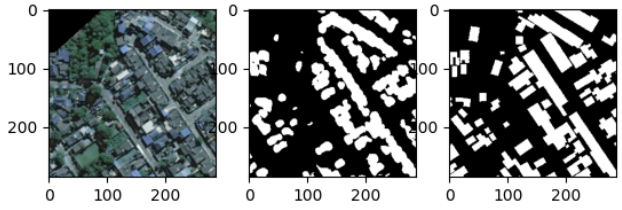
(a) Validation image 1: building polygons



(b) Validation image 1: building masks



(c) Validation image 2: building polygons



(d) Validation image 2: building masks

Fig. 8: Left to right: validation images, predicted building polygons & masks, and ground truth polygons & masks.

B. SpaceNet v1 Challenge Winners

To provide context on the effectiveness of our U-Net building detection implementation, we present the top 3 winning scores from the SpaceNet Building Detection v1 Challenge. The three top scores were submitted by users *wleite*, *Mark.cygan*, and *ginhaifan*.

The first place algorithm first put each pixel into one of three classes: border, inside building, or outside building. Then *wleite* used a random-forest based classificaton method (this method did not use a deep neural network) to classify buildings, and another random forest classifier to determine building polygon footprints. The second place algorithm from *Mark.cygan* used the same 3-category classification (border, inside building, outside building) but used a typical CNN classification approach. The CNN output was a heat map that was then converted into a polygon footprint mask. The third place implementation from *ginhaifan* used a multitask network cascade approach which is a deep convolution, instance aware segmentation network.

The challenge winner's F1 scores are shown in table III, along with the F1 score from our U-Net implementation.

Our F1 score appears to compete with the winning implementations; our score puts us between the 1st and 2nd place

TABLE III: SpaceNet Building Detection V1 challenge winners and our U-Net results.

	1st Place	2nd Place	3rd Place	Our U-Net
F1 Score	0.255292	0.245420	0.227852	**0.252575**

entries. However, there are several reasons why comparing our F1 score to those reported by the winners may not be exactly fair.

First, we evaluated our method on a different set of images than the winners, since we did not have access to building masks for the official testing image set. Second, instead of comparing geoJSON truth polygons to our building footprints, we evaluated using `cv2.findContours`, and so some buildings that were close together may have been considered to be single buildings (in both the ground truth images as well as our predictions), which could impact the IoU score as well as the total number of ground truth buildings, thus increasing precision and recall. Third, our approach used an essentially unmodified U-Net coupled with a simple contour detection method to find building polygons. In many of the winning algorithms however, they employed complicated multi-step approaches for both building detection and polygon generation. Finally, we used our own implementations of the IoU, precision, recall, and F1 scores, which may or may not be slightly different from the official implementations.

IV. Discussion

Our U-Net shows very good performance relative to the winners of the SpaceNet Building Detection v1 challenge, indicating that it is a powerful method for image segmentation. The relative ease of implementation, and its conceptual simplicity make it an attractive option for semantic segmentation tasks. However, the method has several drawbacks.

First, there were some implementation quirks. Initially we had implemented the U-Net without padding, instead just cropping the input signal from one step of the contracting path to the next. This resulted in a final segmentation mask that eventually became full of not-a-number (NaN) values. After adjusting the algorithm and using padding in the contracting and expansion paths, this was no longer an issue.

Second, our network occasionally struggles with differentiating buildings when they are too close together. The original U-Net paper attempts to improve this performance by including a term in the loss function that specifically weights border pixels. We believe that implementing this loss function could improve our performance, but we leave this fine-tuning to future work.

Finally, the U-Net sometimes shows difficulty in predicting the building sizes in the images we used. For example, in the bottom image of figure 7, there is a ground truth building near the middle of the image that is much larger than its predicted counterpart. We observed that this most often happens for long, skinny buildings, which could be a byproduct of not having enough resolution for the convolution operations to build reliable features. Another downside of our approach is the absence of sharp edges and corners in our predicted mask. This is another result of the convolution down-sampling, and we likely lose out on some IoU value due to the rounded corners of the predicted buildings. Some more advanced segmentation networks aim to fix this by including additional skip layers that preserve fine details in the mask.

For these reasons, the U-Net seems to be an effective semantic segmentation method, with some limitations.

V. Conclusion

The amount of recent activity in the development of machine learning methods coupled with an influx of satellite image data has motivated efforts in automatically generating civil system maps which before recently has been impossible or very difficult. Organizations like SpaceNet play an important role in these efforts by providing free satellite image data sets and targeted competitions aiming to solve common problems associated with satellite image mapping. Our project focused on the first SpaceNet challenge associated with building detection, and shows promising results.

Our solution leverages the U-net CNN architecture, a powerful method for semantic segmentation. This network deviates from the fully connected CNN in that its contraction and expansion paths use only the valid portion of the signal from the previous level, but makes gains in training simplicity and accuracy as a result. The contraction path builds a low-resolution feature-deep representation that maintains feature context beyond just the desired number of output classes. The expansion path then combines this feature-deep representation with the localized features associated with the previous contraction levels. This expansion process reduces the number of features while increasing image resolution until a high-resolution image is obtained containing the desired number of output segmentation classes. For our application, the U-Net produces a segmentation mask representing either a building or non-building.

Our results suggest that the U-Net is an effective method for semantic segmentation, potentially competing with the SpaceNet Building Detection v1 challenge winners from the original challenge. Although our U-Net misses buildings at a moderately high rate, our overall IoU, precision, recall, and F1 scores demonstrate the algorithm's effectiveness for the building detection segmentation task. The U-Net has several limitations which could prove to be significant drawbacks depending on the application, but our implementation showed strong results for the task of building detection.

References

[1] A. V. Etten, D. Lindenbaum, and T. Bacastow, "Spacenet: A remote sensing dataset and challenge series." ArXiv, 2018.

[2] J. Jordan, "An overview of semantic image segmentation," Online, 2018. [Online]. Available: https://www.jeremyjordan.me/semantic-segmentation/

[3] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäine, "Deep learning for generic object detection: A survey," *ArXiv*, 9 2018.

[4] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *ArXiv*, 10 2017.

[5] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," *ArXiv*, 9 2016.
[6] P. Hagerty, "The spacenet metric," Online, 2016. [Online]. Available: https://medium.com/the-downlinq/the-spacenet-metric-612183cc2ddb
[7] jshermeyer, T. Stavish, dlindenbaum, N. Weir, lncohn, and W. Maddox, "Spacenet utilities," Online, 2017. [Online]. Available: https://github.com/SpaceNetChallenge/utilities
[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *ArXiv*, 5 2015.