Improving Natural Disaster Relief with Machine Learning

Gregory Lull

Motivation

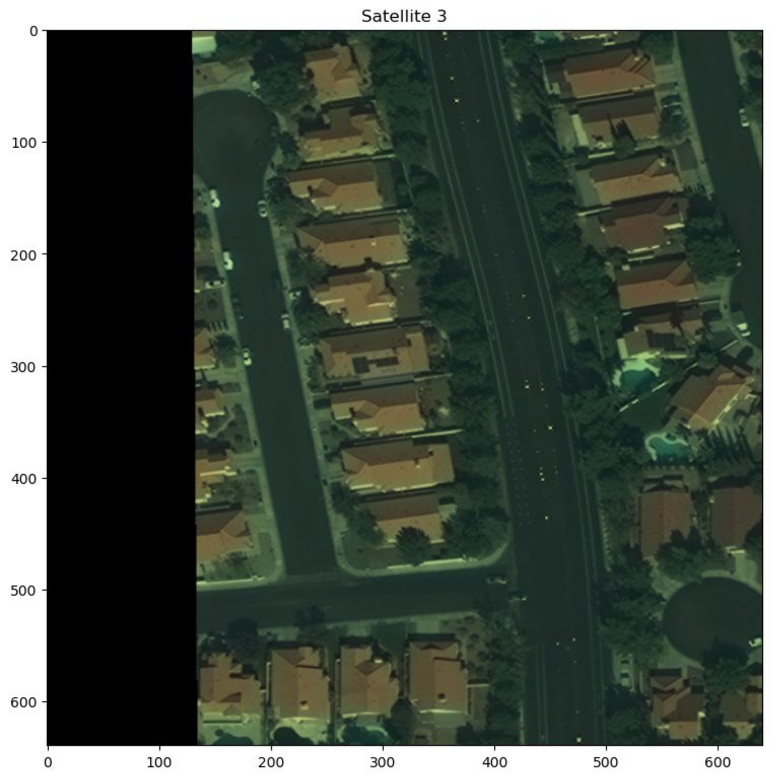
- Use cases: Navigation, rideshare apps, updating urban developments, disaster relief
- Modern cartography uses satellite imagery, GPS traces, location analytics
- This process requires lots of human input and could be error prone

Data and Methodology

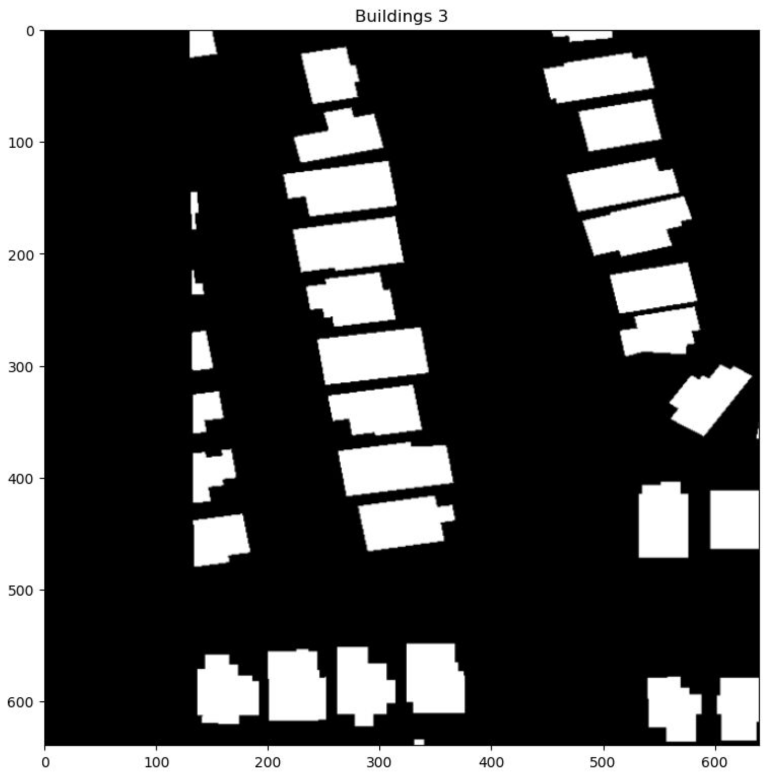
- SpaceNet: 3500 satellite images with geo-coordinates for buildings
- Modeling: U-Net convolutional network for image segmentation
- Tech stack: keras, tensorflow, solaris
- Hardware: Nvidia GPU reducing train time by 90%

Data: Inputs and Outputs

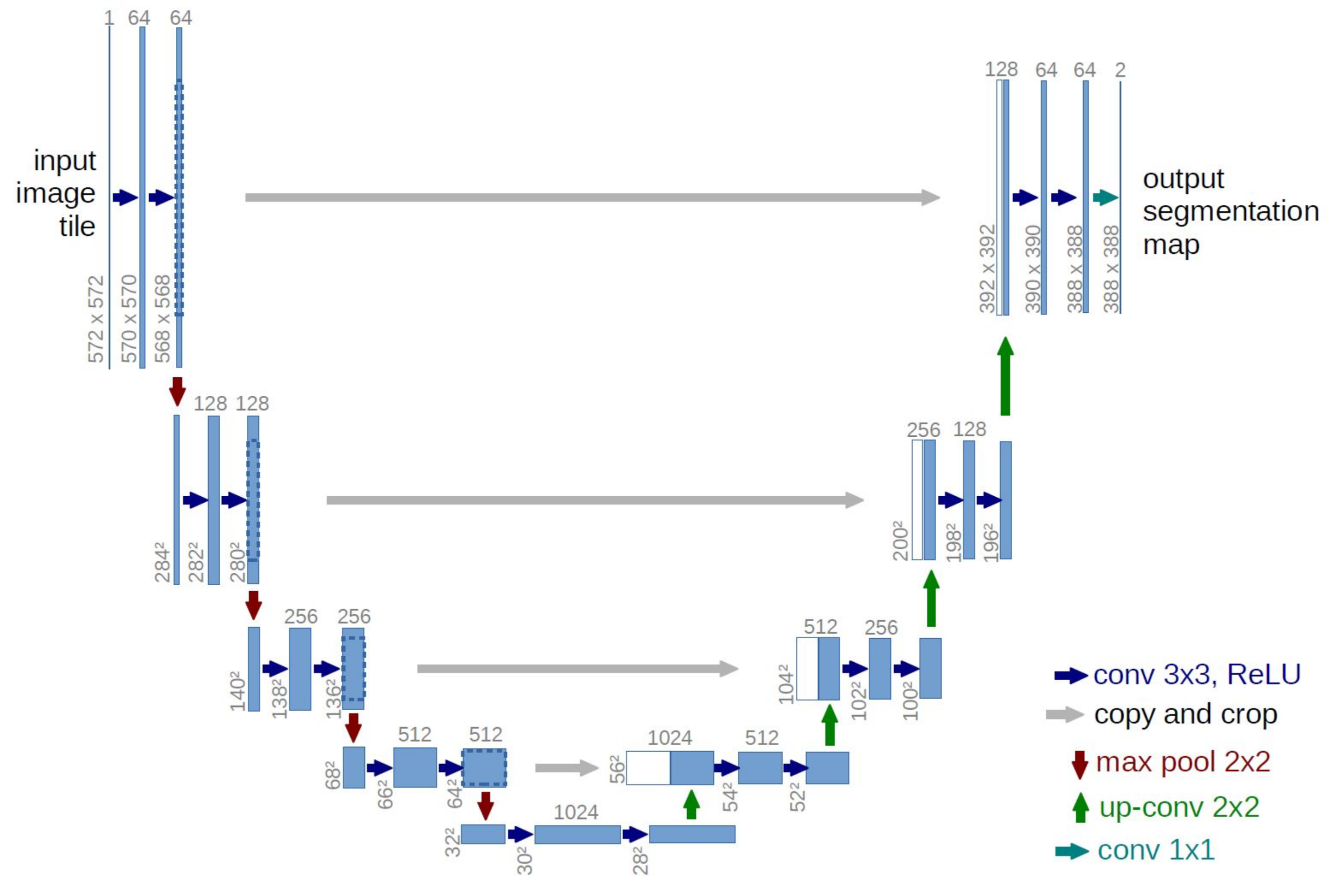
Satellite Image



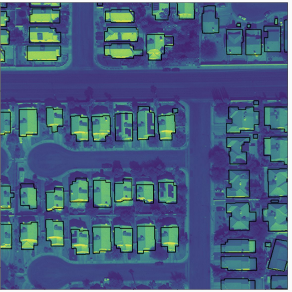
Geo coordinate masks

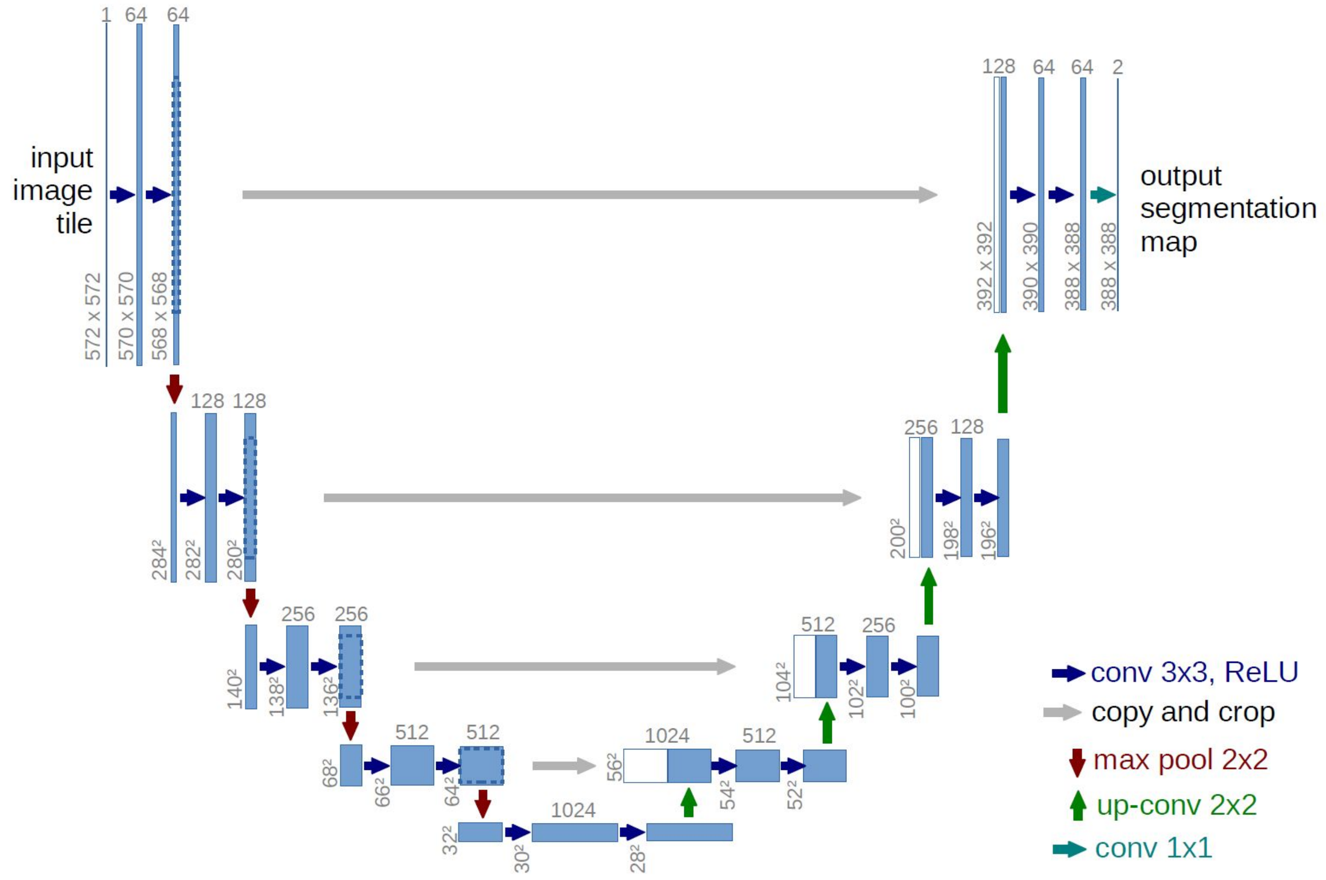


Modeling: U-Net Convolutional Network

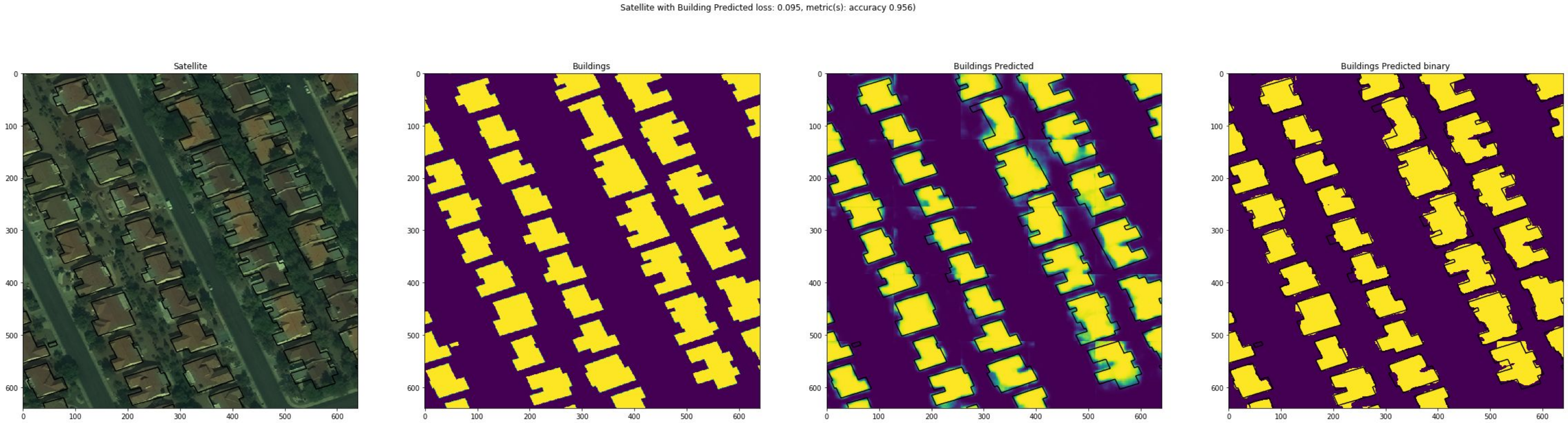


U-Net: Convolutional Network





Results


Satellite with Building Predicted loss: 0.095, metric(s): accuracy 0.956)
Satellite
Buildings
Buildings Predicted
Buildings Predicted binary
0
100
200
300
400
500
600

Results

Conclusion

- With an IoU of 75% it is probably not as good as human examining each satellite image, but the savings in time and human errors will add up.
- Examining and prioritizing the city of Seattle would take about 20 minutes.

Future Work

- In addition to buildings there are other structures and terrain to map out.
- Looking at before and after images for areas with rapid change: city expansion, disaster relief

Thank you

Gregory Lull

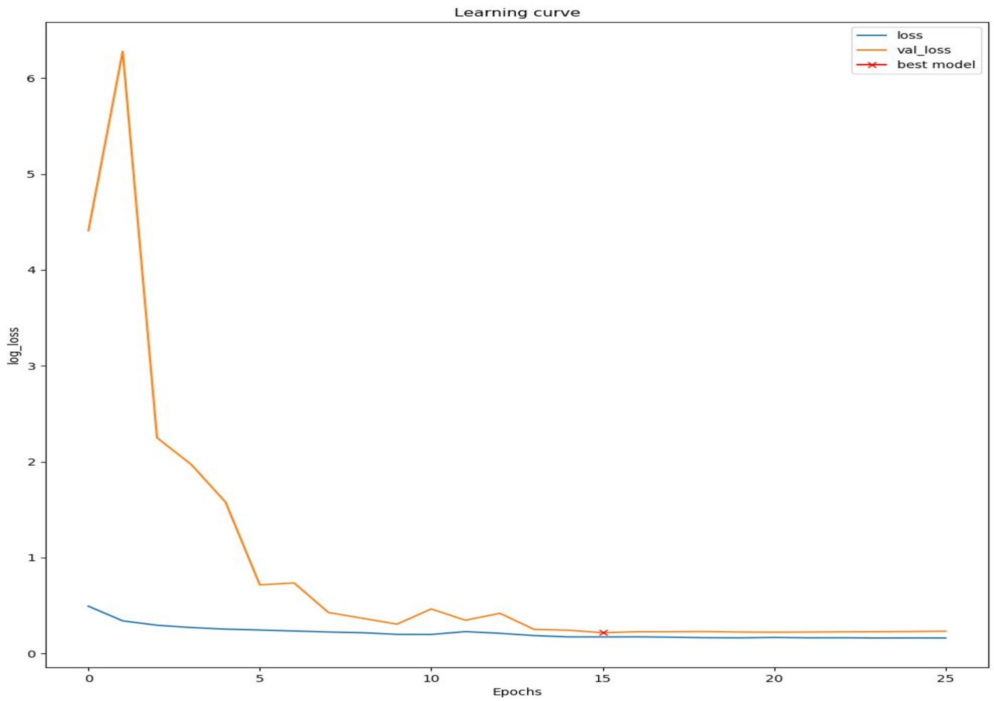
gregorylull@gmail.com

linkedin.com/in/gregorylull

Appendix

Total params: 1,179,121
Trainable params: 1,177,649
Non-trainable params: 1,472

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
img (InputLayer)                 (None, 128, 128, 1)   0
____________________________________________________________________________________________________
conv2d_2 (Conv2D)                (None, 128, 128, 16)  160         img[0][0]
____________________________________________________________________________________________________
batch_normalization_2 (BatchNor  (None, 128, 128, 16)  64          conv2d_2[0][0]
____________________________________________________________________________________________________
activation_2 (Activation)        (None, 128, 128, 16)  0           batch_normalization_2[0][0]
____________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)   (None, 64, 64, 16)    0           activation_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 64, 64, 16)    0           max_pooling2d_1[0][0]
____________________________________________________________________________________________________
conv2d_4 (Conv2D)                (None, 64, 64, 32)    4640        dropout_1[0][0]
____________________________________________________________________________________________________
batch_normalization_4 (BatchNor  (None, 64, 64, 32)    128         conv2d_4[0][0]
____________________________________________________________________________________________________
activation_4 (Activation)        (None, 64, 64, 32)    0           batch_normalization_4[0][0]
____________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)   (None, 32, 32, 32)    0           activation_4[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 32, 32, 32)    0           max_pooling2d_2[0][0]
```



Learning curve
loss
val_loss
best model
log_loss
Epochs
0
1
2
3
4
5
6
0
5
10
15
20
25