Detecting hill country erosion from aerial imagery

Semantic Segmentation

Classifying each pixel in an image from a predefined set of classes.
In our case, 1 = erosion, 0 = everything else (background)

segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

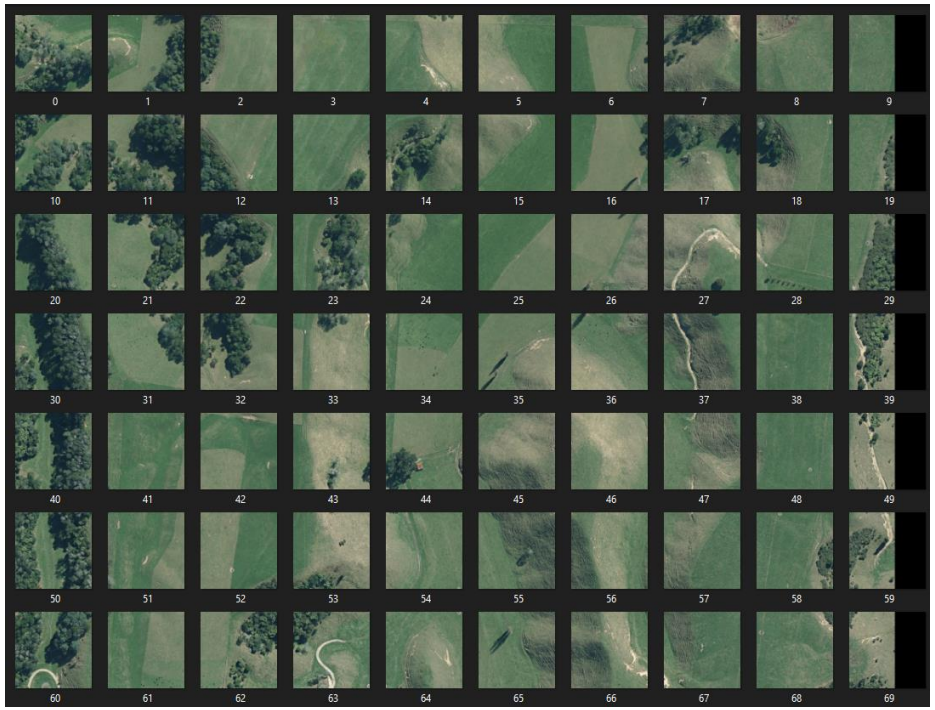
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Input

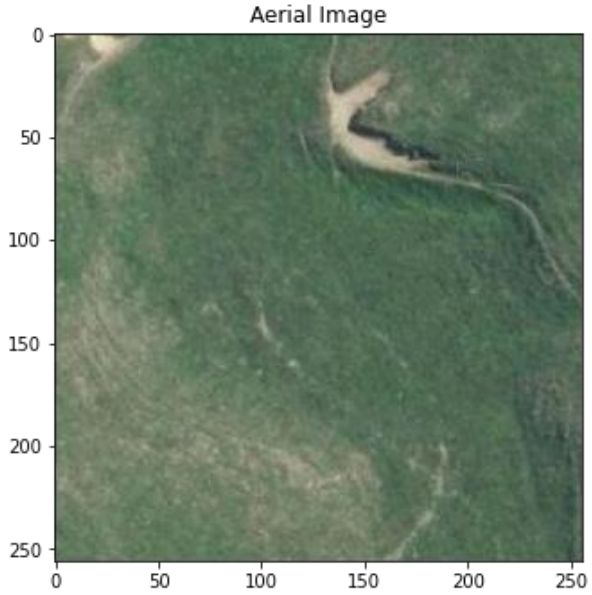
Semantic Labels

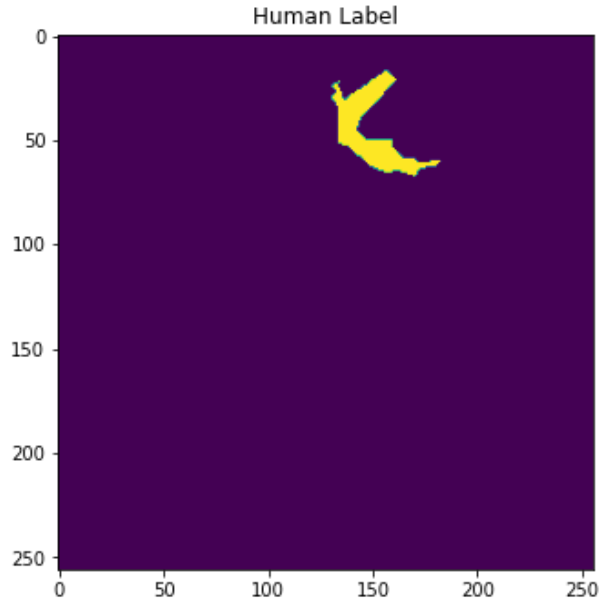
Creating the Dataset:

- ML model can only take in one size of image
- 256 x 256 sized image chunks



691 Images + Labels





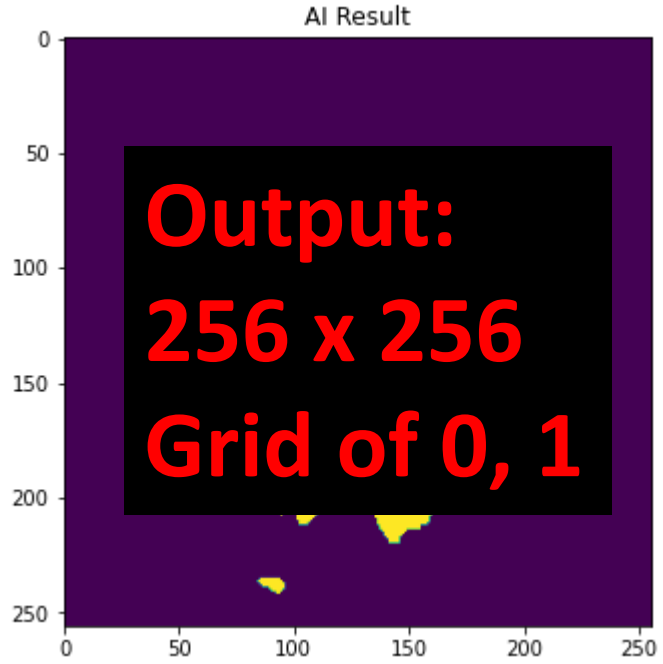
train

ML Model

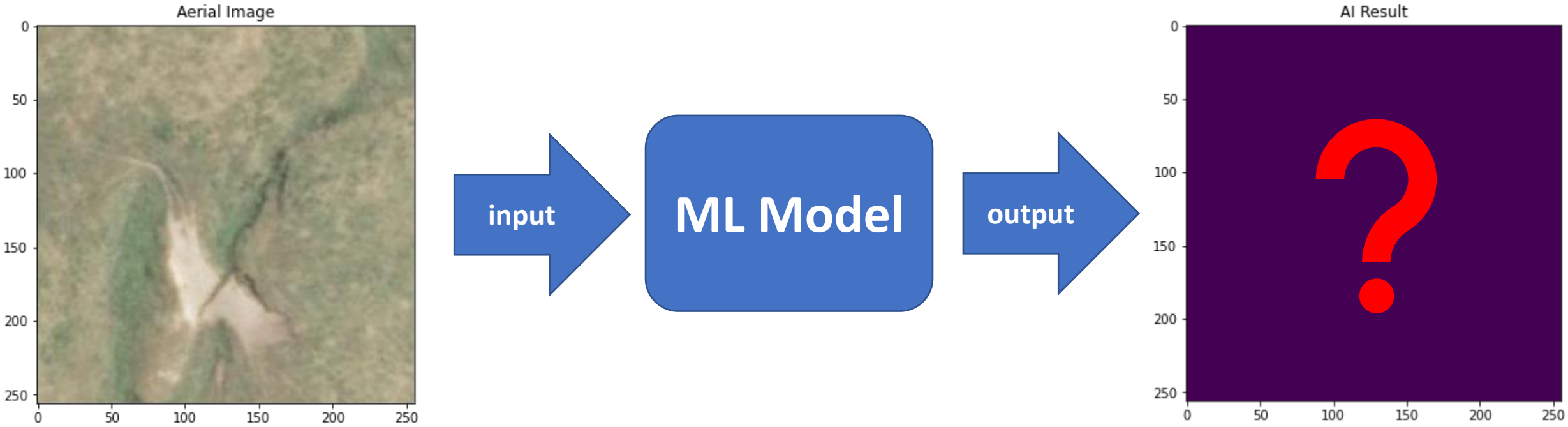
Previously
Unseen Image

test

output



After training the model, the accuracy was 95%!!!

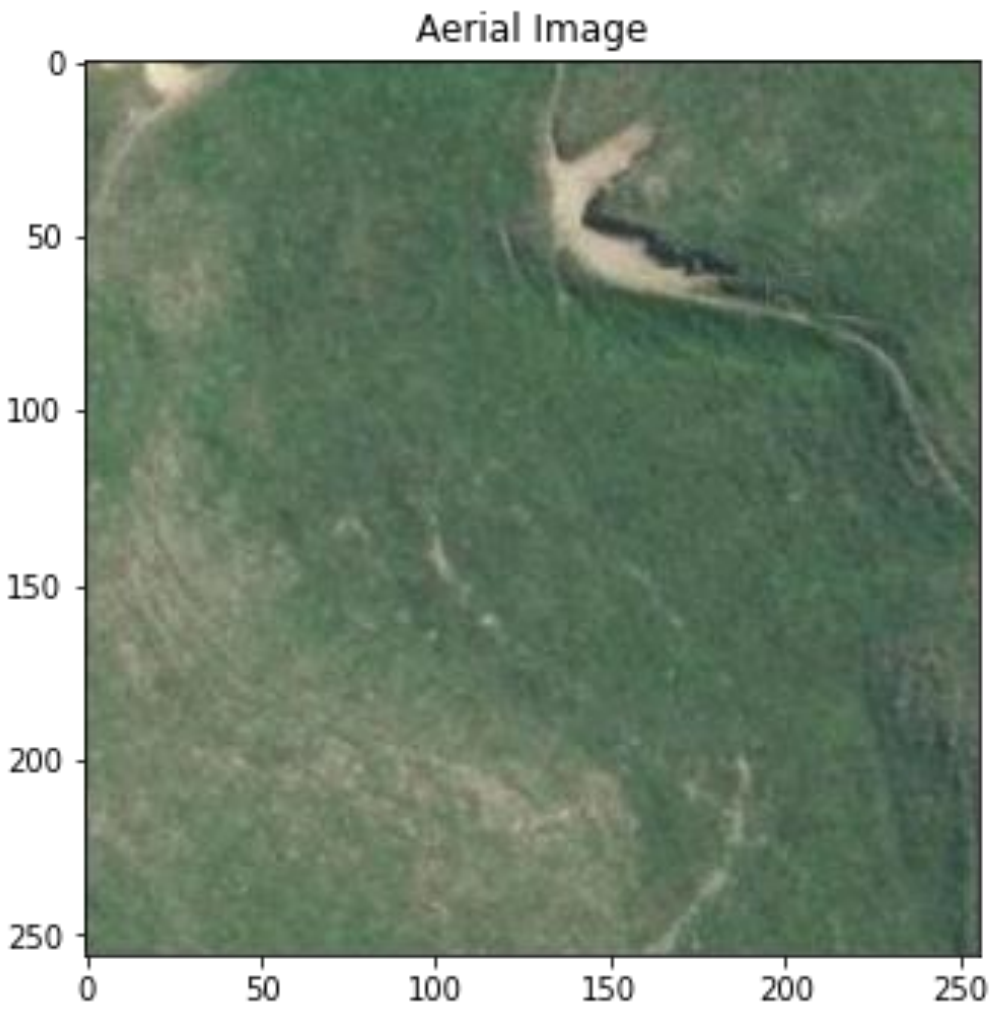


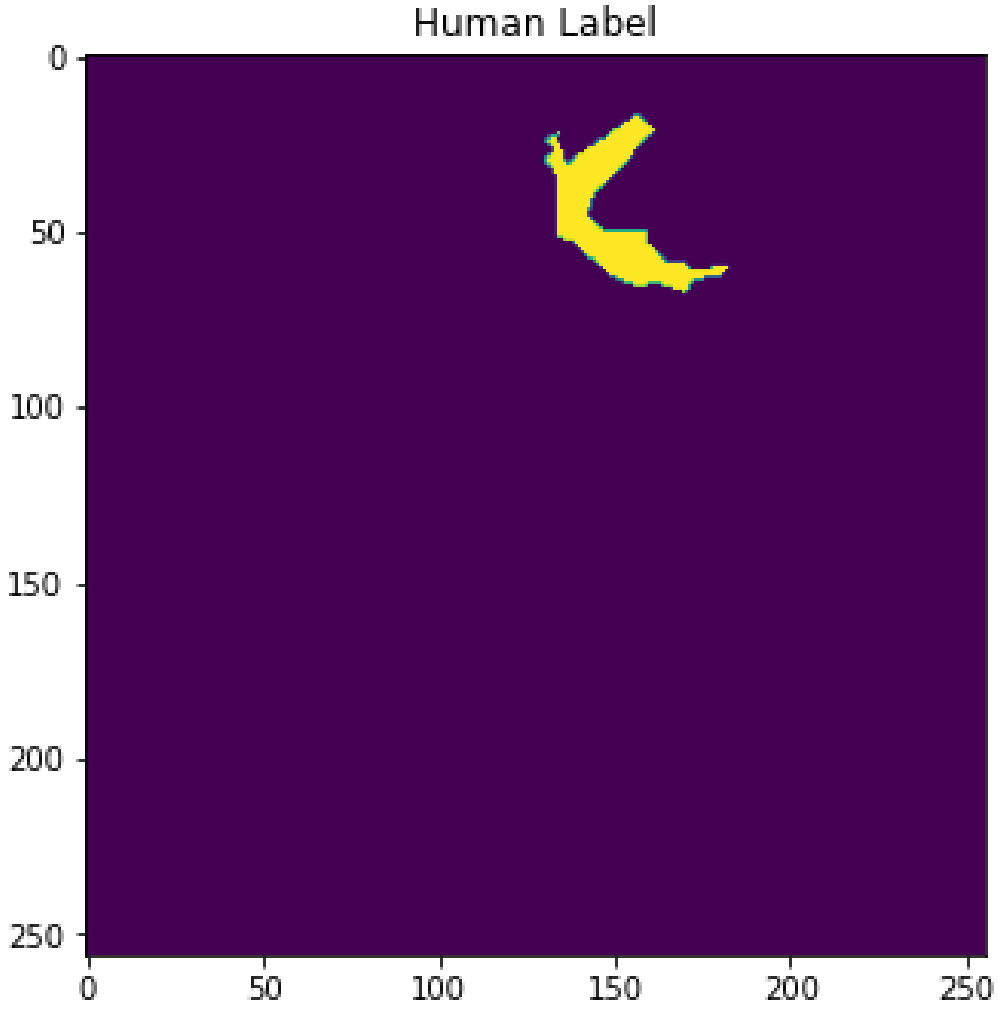
The model thinks every pixel inside the input image belongs to background class (i.e. 0) and none of them belongs to erosion (i.e. 1)

Question: Why does my accuracy not match up with the end result?

Imbalance of Classes

Purple: background, Yellow: erosion

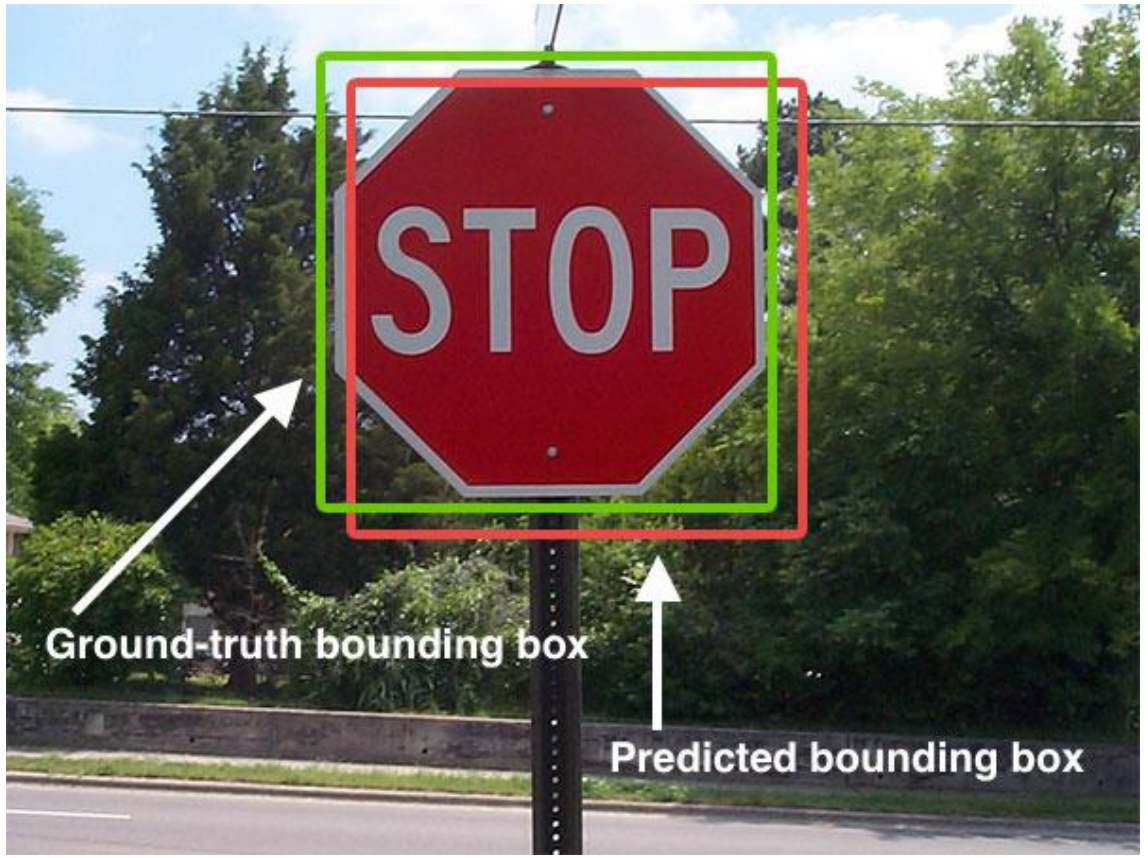




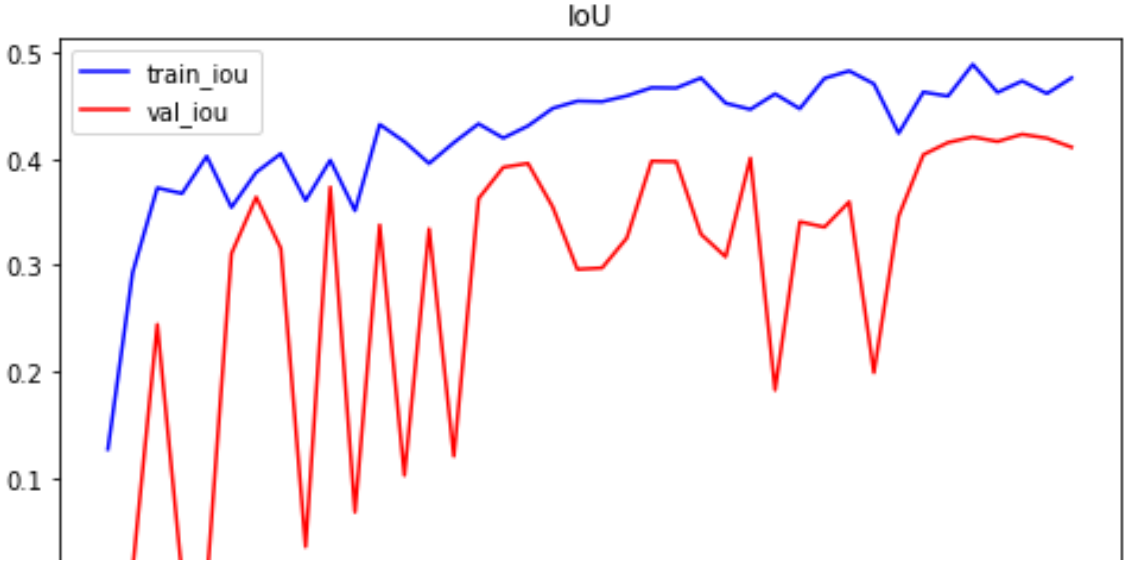
Problem: Background is ~95% of all the images, Erosion is only ~5%

Solution: Intersection over Union (IoU)

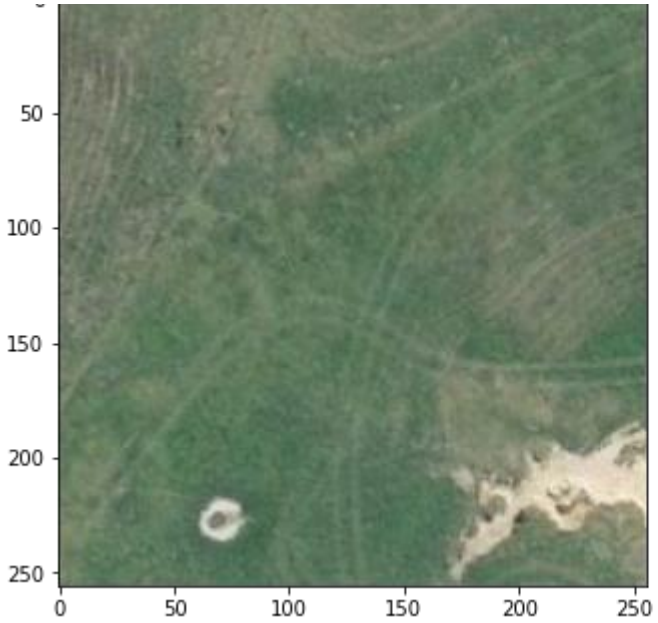
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

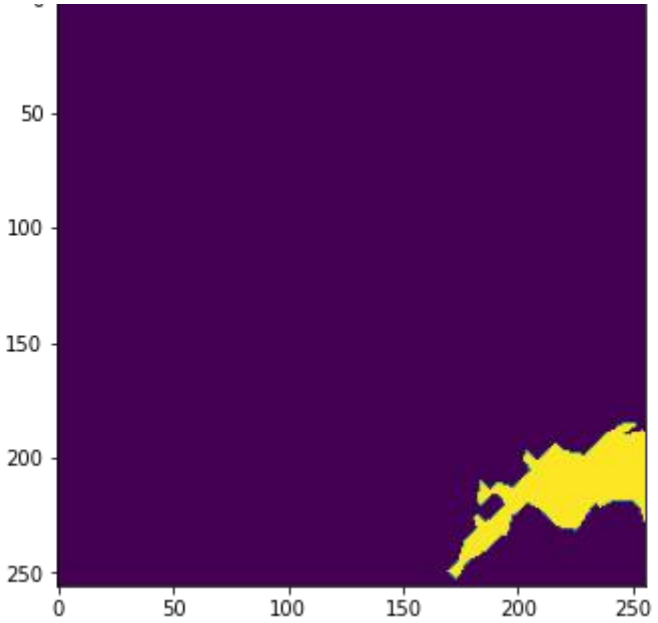


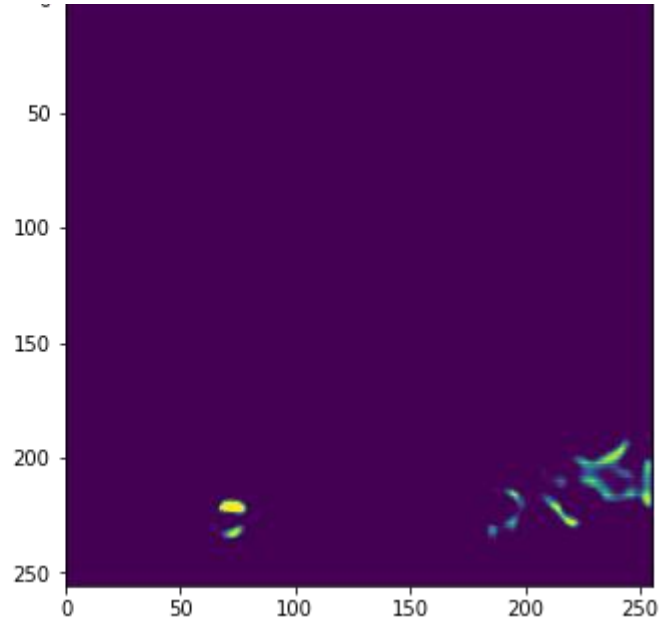
Results After Implementation:



Can we do any better than a score of 50%?

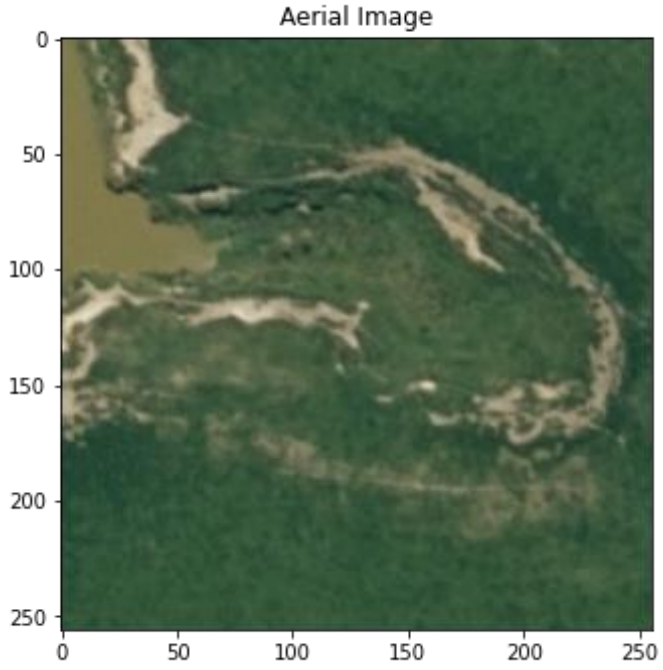


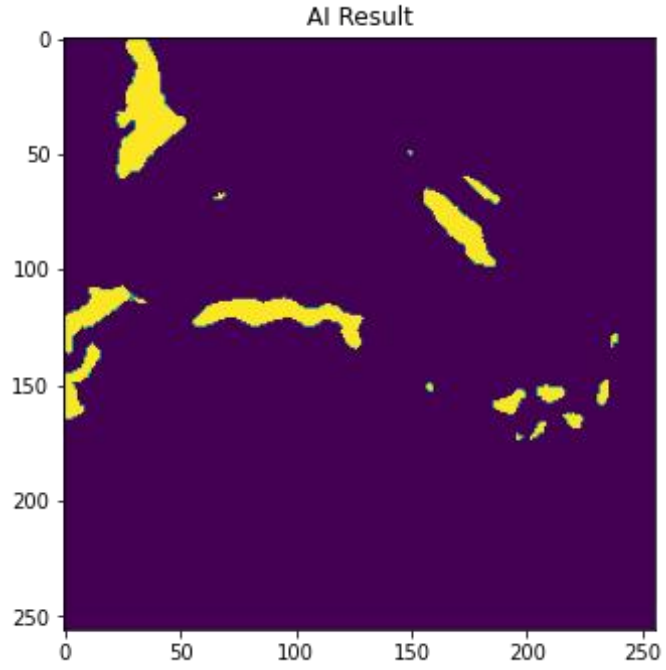


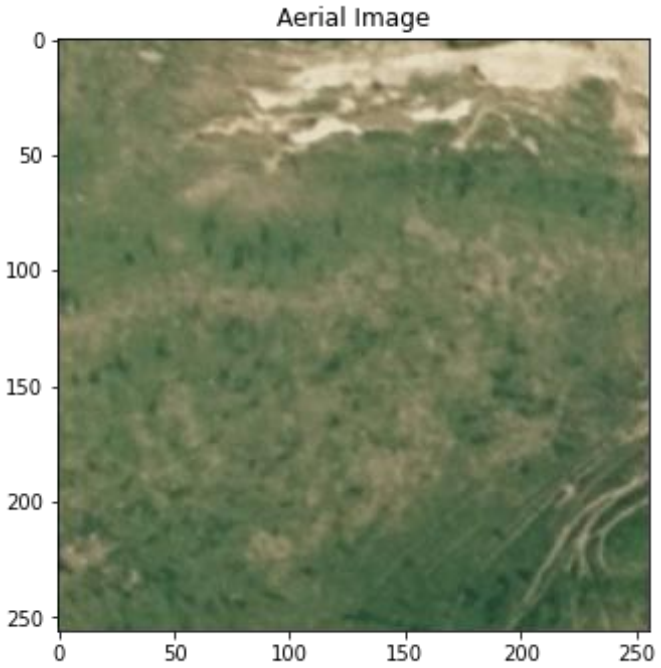


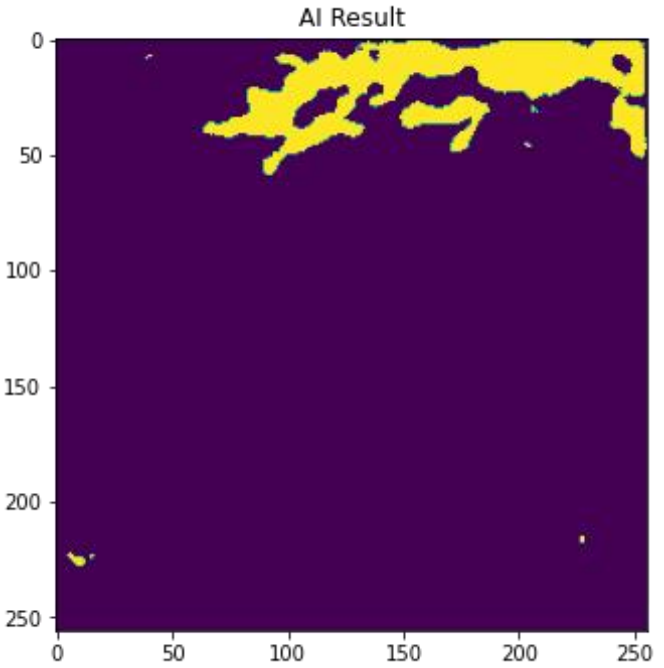
Increasing the Dataset

- Rotated the images and labels
- 90, 180, 270 degrees
- Now 691 x 4 = 2764 images to train the model

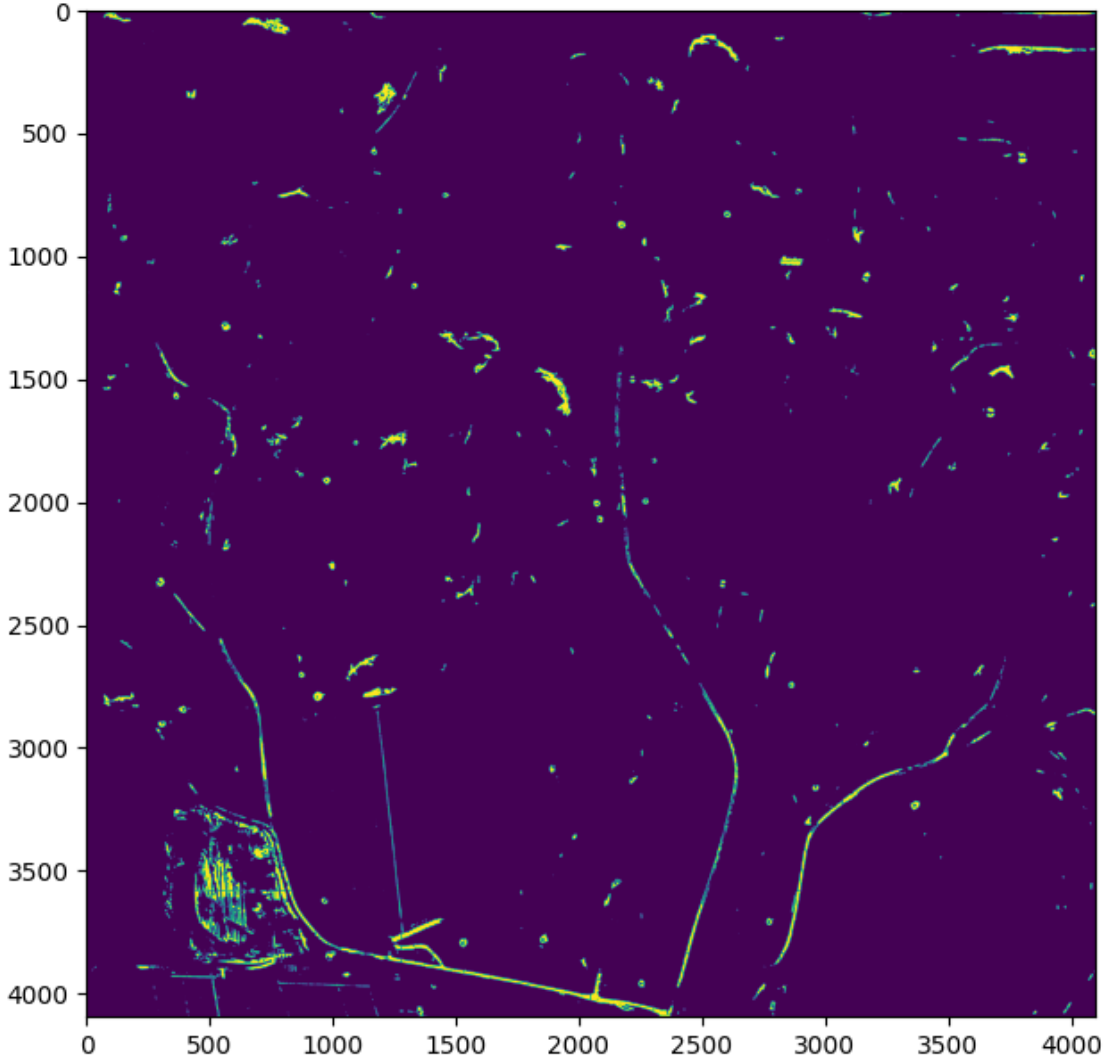




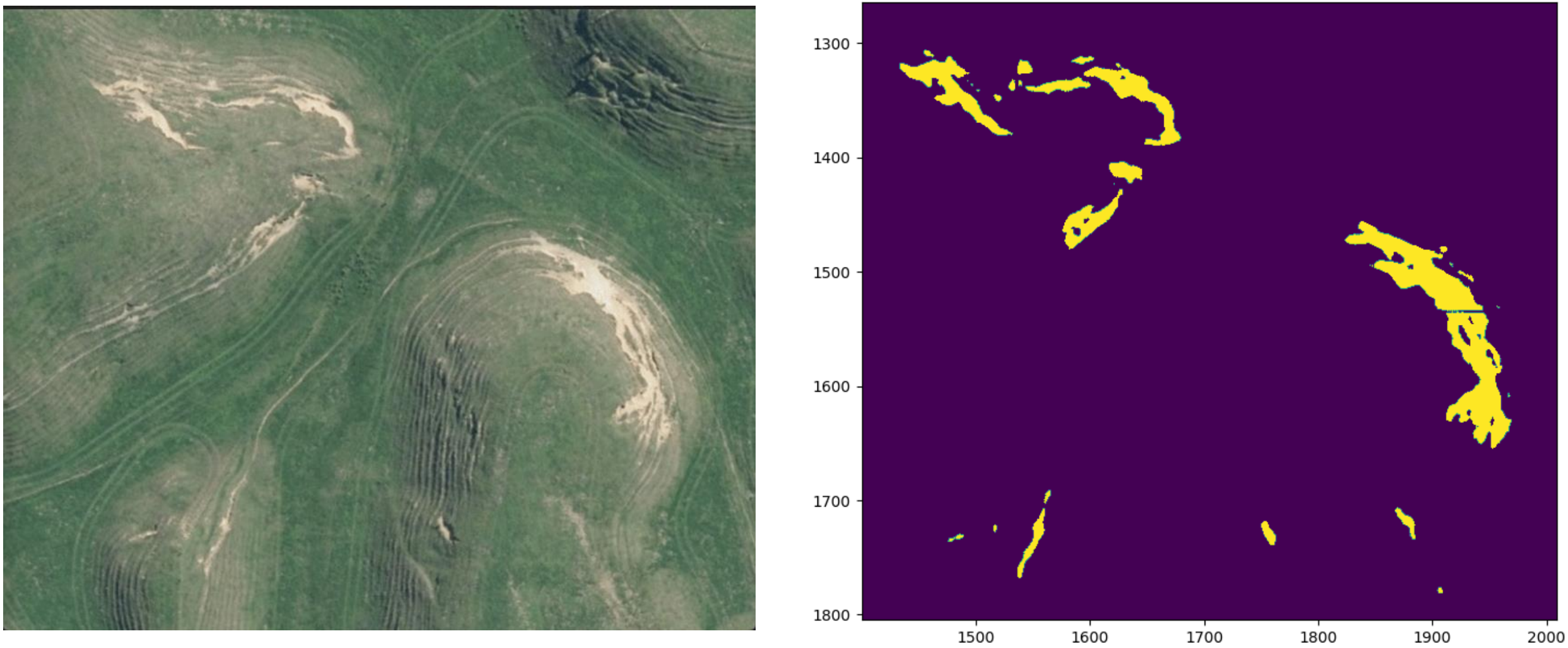




Putting everything back together:



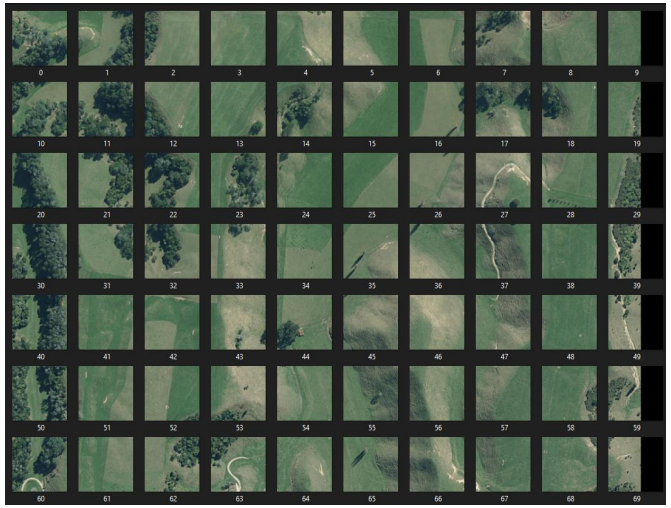
Input image (left) and the output segmentation map (right)



Input image (left) and the output segmentation map (right) zoomed in

1000m x 1000m
4096 x 4096 pixels
Image from ArcGIS

chunking

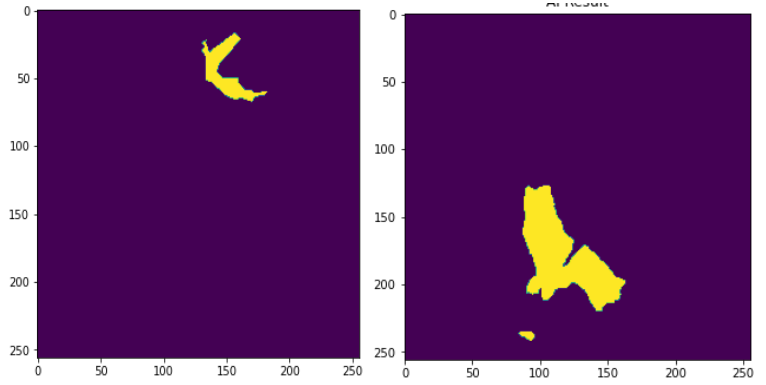


256 x 256 pixel images

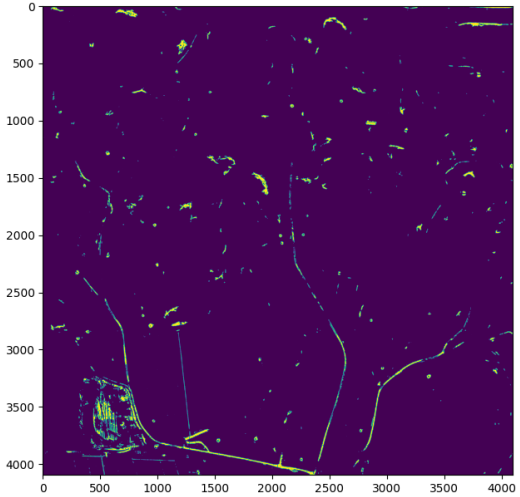
Load into model

ML Model

Get results



Combine



Turn it into features

GeoJSON containing property boundary information

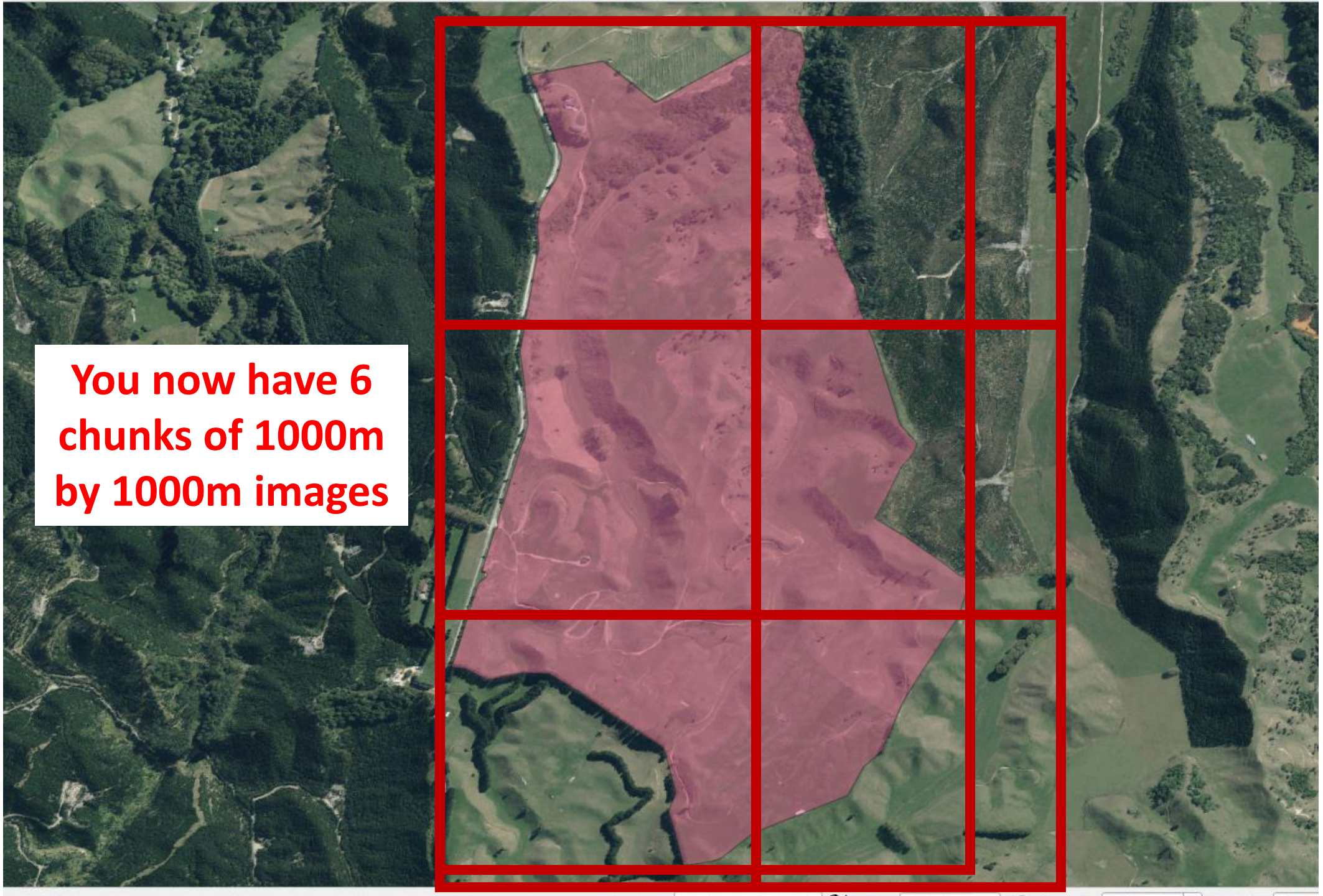

```
{
  "type": "Feature",
  "properties": {
    "objectId": 70,
    "propId": "pZ0FVo5xI",
    "title": "Hilo Farm",
    "officialName": "Brad and Rachel Strange ",
    "address": "155 Ridge Road Pongakawa",
    "city": null,
    "postcode": null,
    "state": null,
    "country": null,
    "active": null,
    "customData": null,
    "createdUser": null,
    "createdDate": null,
    "lastEditUser": null,
    "lastEditDate": null,
    "deletedDate": null,
    "primaryContactEmail": null,
    "primaryContactName": null,
    "secondaryContactEmail": null,
    "secondaryContactName": null
  },
  "geometry": {
    "type": "MultiPolygon",
    "coordinates": [
      [
        [
          [176.426346813897425, -37.907716539054867],
          [176.42621398292539, -37.9074657521181],
          [176.426451660388381, -37.906797131753102],
```


1000m / 4096 pixels

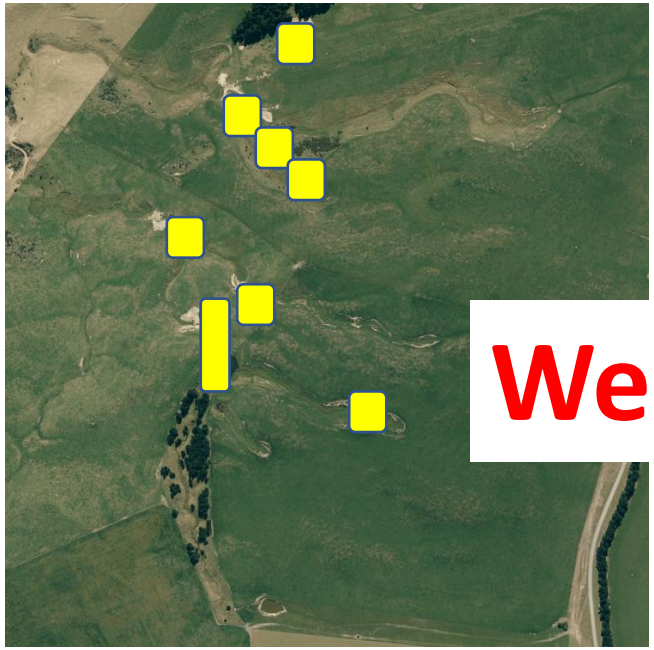
1000m / 4096 pixels

Question:

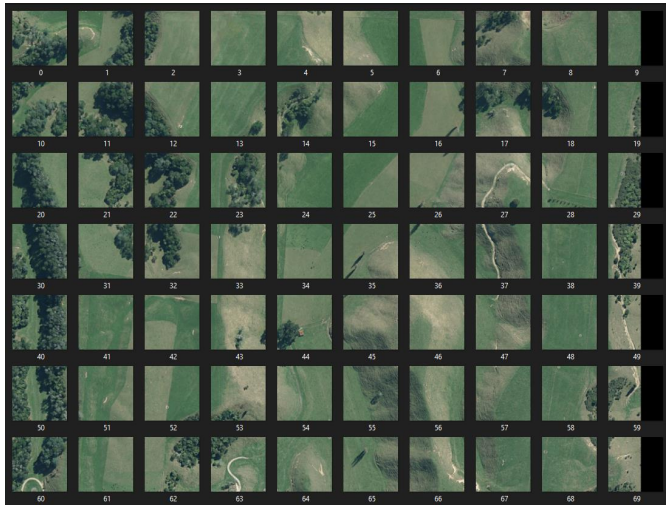
How can we process a large enough image that encompasses the entire property?


You now have 6 chunks of 1000m by 1000m images

1000m x 1000m
4096 x 4096 pixels
Image from ArcGIS



chunking



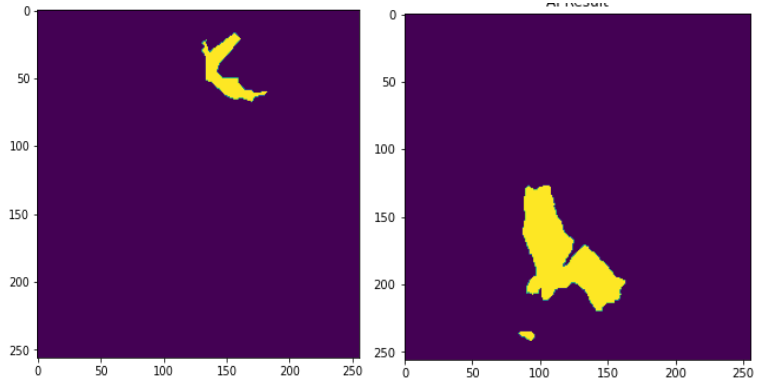
256 x 256 pixel images

Load into model

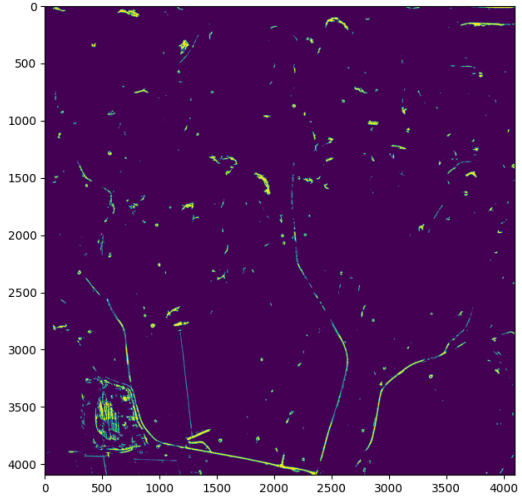
ML Model

We now need to do this 6 times!

Get results

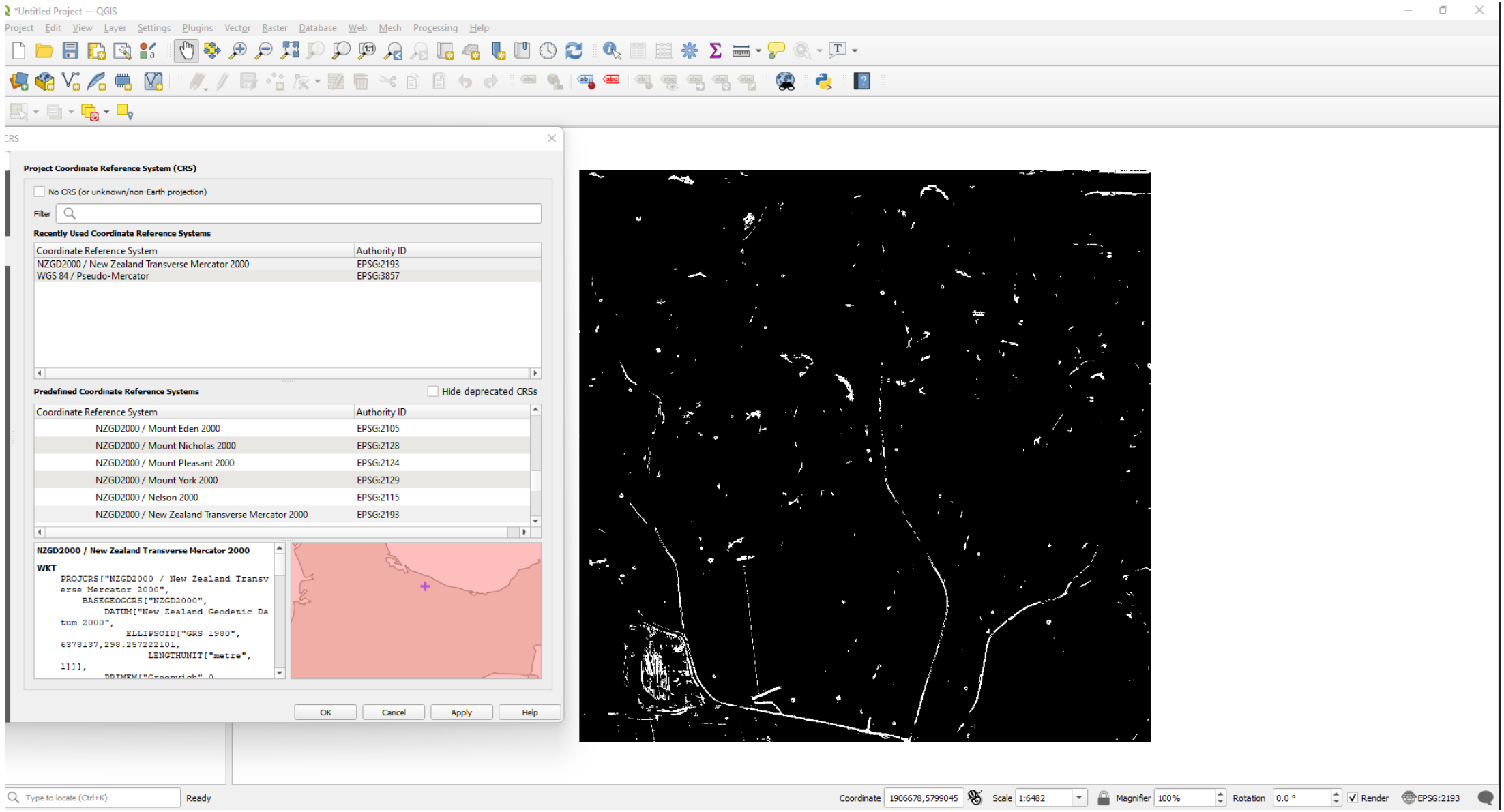


Combine

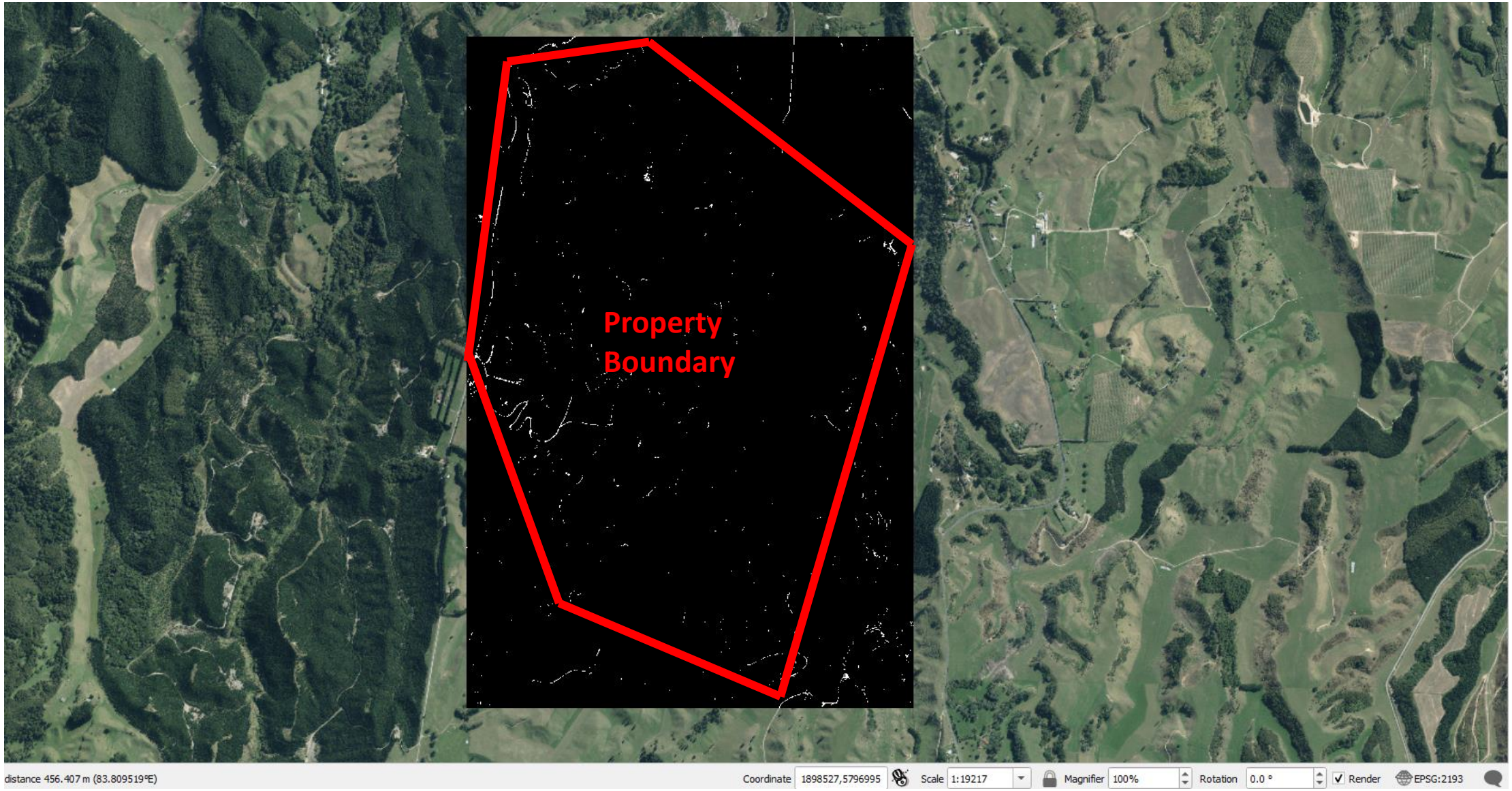


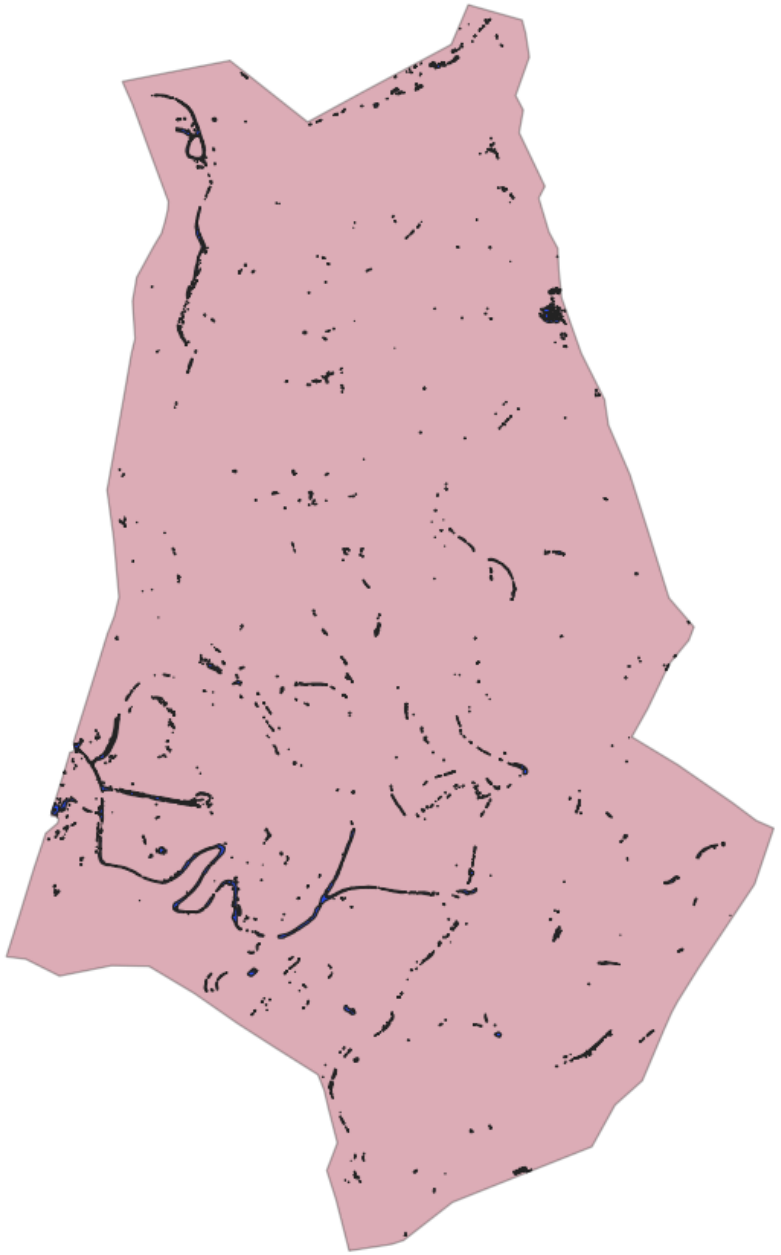
Turn it into features

Convert each 1000m by 1000m chunk into GeoTiff



Combine the 6 GeoTiff into one giant GeoTiff



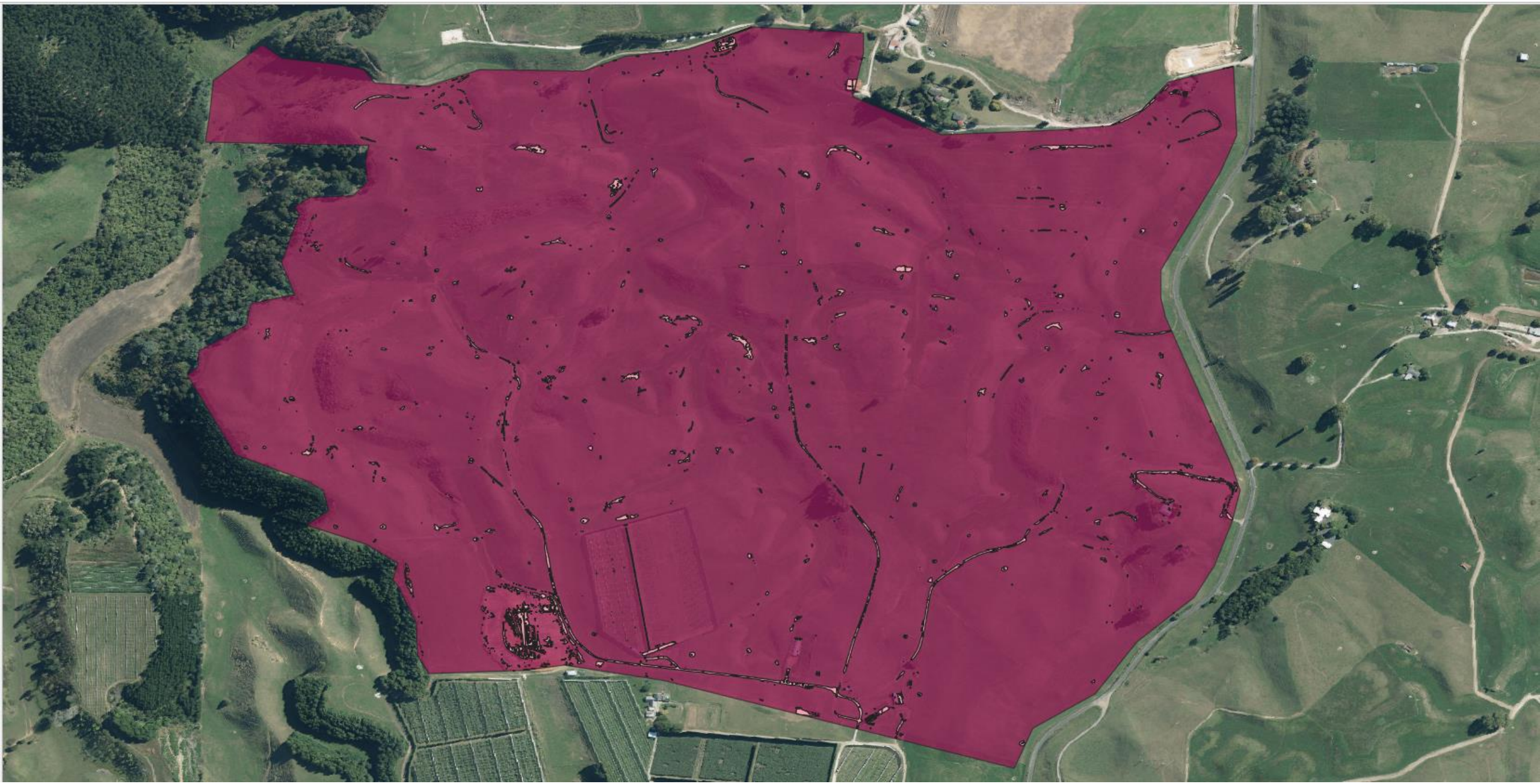


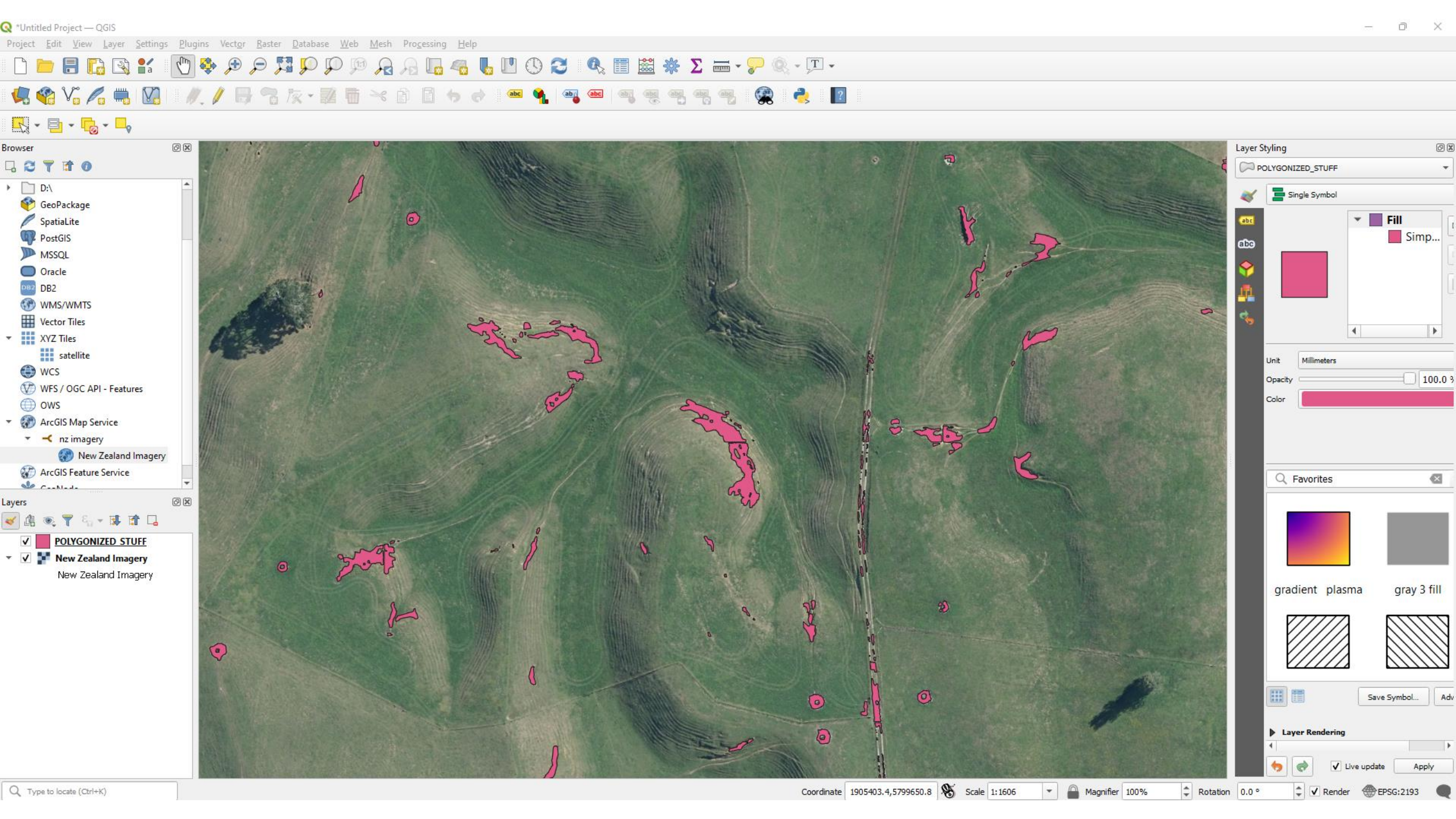
WGS84
(EPSG4

NZTM
(EPSG2193)


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [176.425772, -37.906135],
            [176.425775, -37.906135],
            [176.425775, -37.906138],
            [176.425779, -37.906138],
            [176.425779, -37.90614],
            [176.425777, -37.90614],
            [176.425777, -37.906142],
            [176.425775, -37.906142],
            [176.425775, -37.906145],
            [176.425772, -37.906145],
            [176.425772, -37.906147],
            [176.42577, -37.906147],
            [176.42577, -37.90615],
            [176.425767, -37.90615],
```

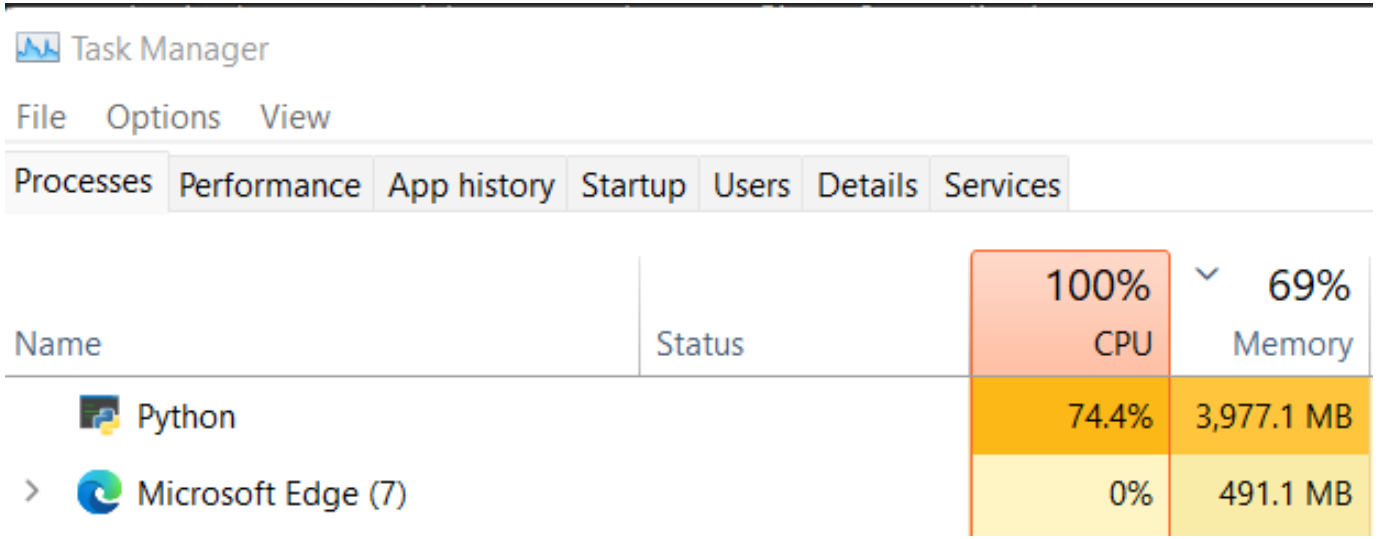

Output: Feature Polygons
(GeoJSON)




*Untitled Project — QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help
Browser
D:\
GeoPackage
SpatiaLite
PostGIS
MSSQL
Oracle
DB2
WMS/WMTS
Vector Tiles
XYZ Tiles
satellite
WCS
WFS / OGC API - Features
OWS
ArcGIS Map Service
nz imagery
New Zealand Imagery
ArcGIS Feature Service
Layers
POLYGONIZED_STUFF
New Zealand Imagery
New Zealand Imagery
Layer Styling
POLYGONIZED_STUFF
Single Symbol
Fill
Simp...
Unit
Millimeters
Opacity
100.0 %
Color
Favorites
gradient plasma
gray 3 fill
Save Symbol...
Layer Rendering
Live update
Apply
Type to locate (Ctrl+K)
Coordinate
1905403.4,5799650.8
Scale
1:1606
Magnifier
100%
Rotation
0.0 °
Render
EPSG:2193

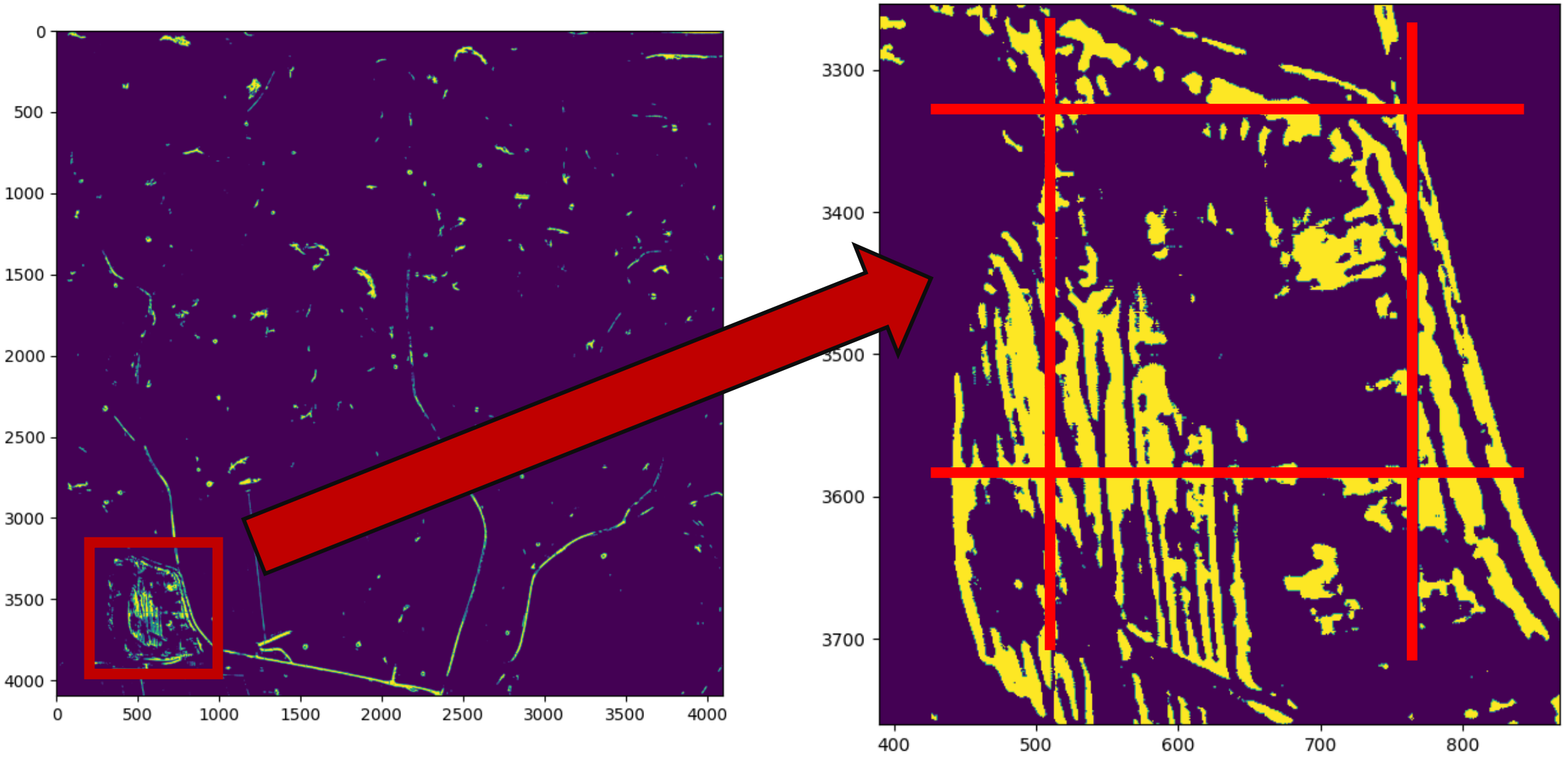
Computationally Expensive

```
C:/Users/skwe9/Desktop/1.tif
got image no: 2
C:/Users/skwe9/Desktop/2.tif
got image no: 3
C:/Users/skwe9/Desktop/3.tif
got image no: 4
C:/Users/skwe9/Desktop/4.tif
got image no: 5
C:/Users/skwe9/Desktop/5.tif
Finished in 646.2466 seconds
PS C:\Users\skwe9>
```



Grid Lines

Can see clearly where the image chunks were joined together



A Known Issue:

Make smooth predictions by blending image patches, such as for image segmentation

One challenge of using a U-Net for image segmentation is to have smooth predictions, especially if the receptive field of the neural network is a small amount of pixels.