Detecting Human Settlements Lacking Electricity with Remote Sensing

by Palak Agarwal and Max Masuda-Farkas

Problem:

Many people in the developing world do not have access to reliable electricity.

Governments in these countries are often hamstrung in their ability to provide electricity particularly among informal human settlements.

How can we identify these settlements to know where electricity is lacking and accordingly where resources should be allocated?

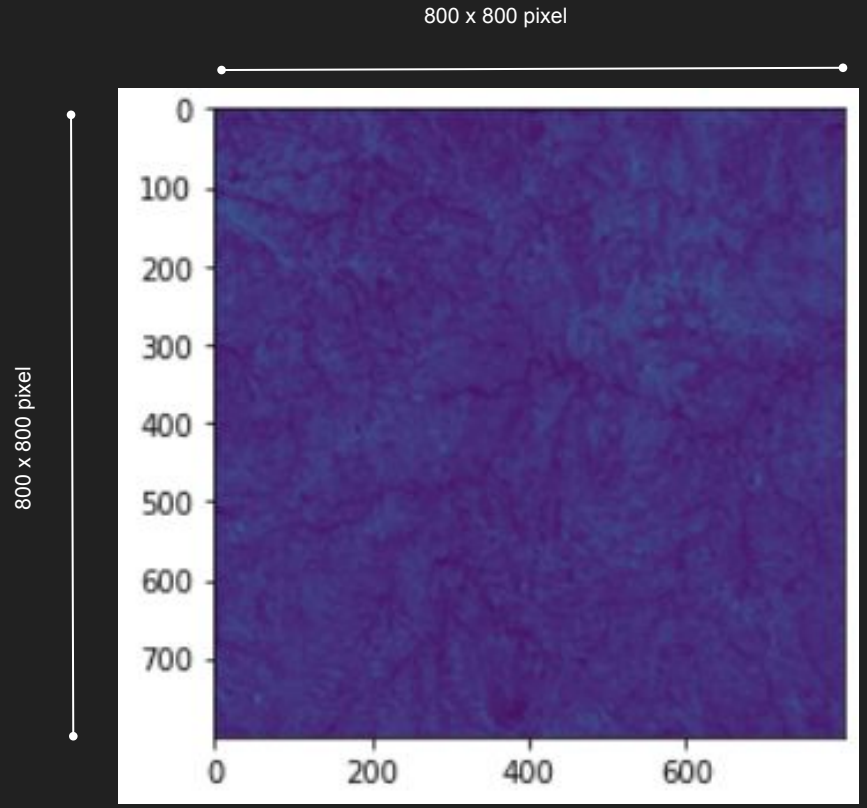
Solution:

Automate the detection of human settlements that lack electricity using machine learning and satellite imagery.

Data

- 60 tiles
- 800x800 pixels per tile
- 12 bands
- 16x16 groundTruth image containing 4 class labels:
  1. Human settlements without electricity (Region of Interest)
  2. No human settlements without electricity
  3. Human settlements with electricity
  4. No human settlements with electricity

Satellite Image

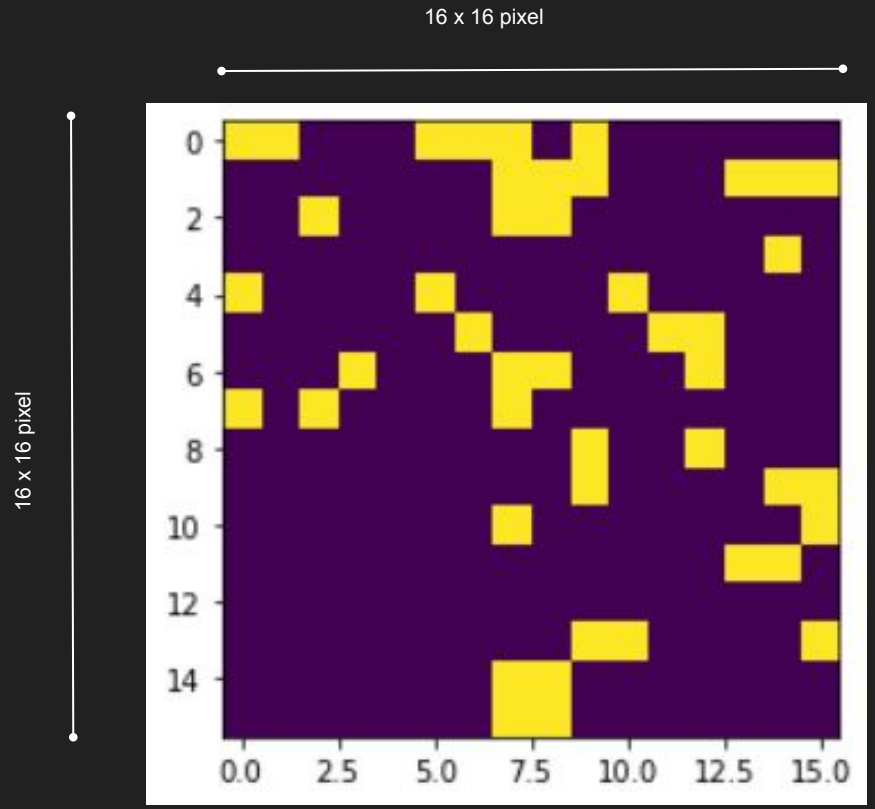


- Resolution - 800 x 800 pixel
- Each pixel - 10 m by 10 m
- Entire image - 64 sq km

Sentinel - 2 :

- Number of bands : 12

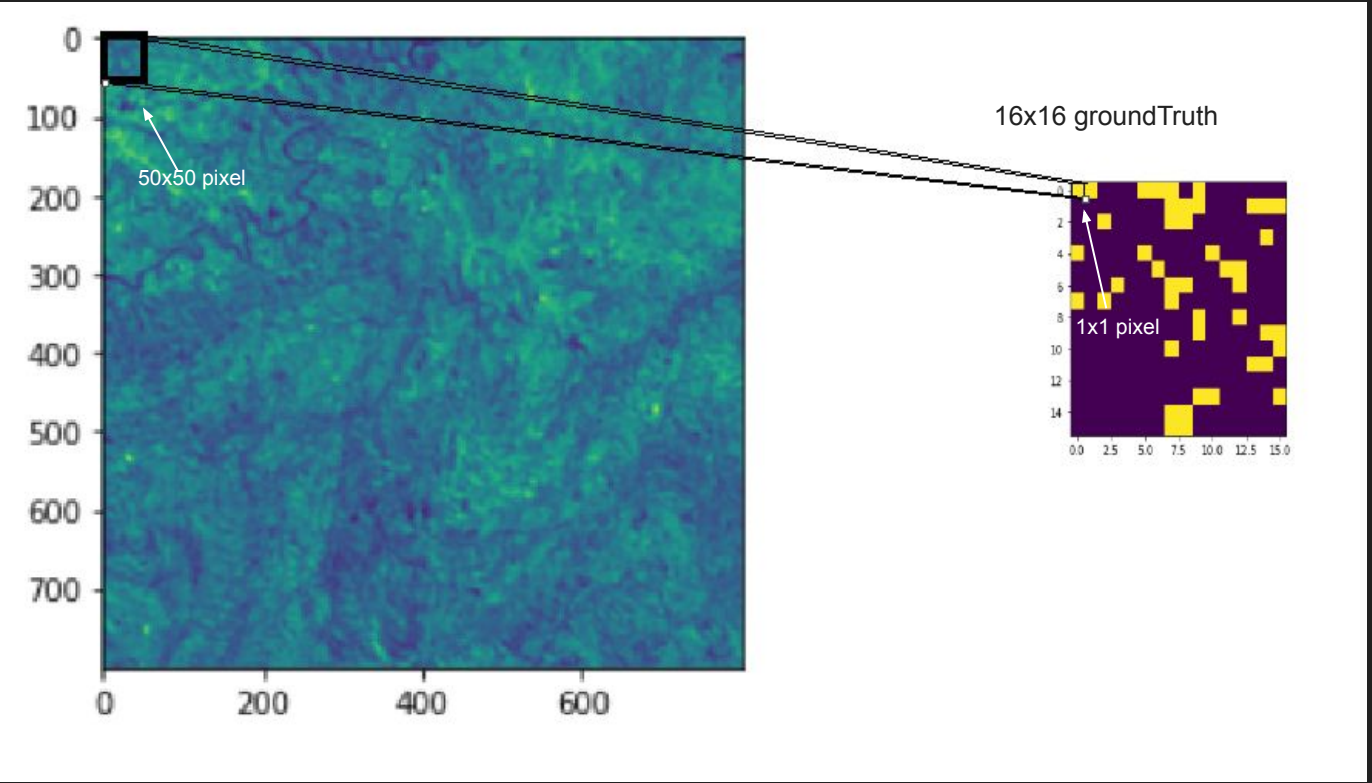
Ground Truth (labels)



- Resolution - 16 x 16 pixel
- Each pixel - 500 m by 500 m
- Entire image - 64 sq km

Labels :

- 1 : Human settlements without electricity (Region of Interest)
- 2 : No human settlements without electricity
- 3 : Human settlements with electricity
- 4 : No human settlements with electricity


0
100
200
300
400
500
600
700
0
200
400
600
50x50 pixel
16x16 groundTruth
1x1 pixel
0
2
4
6
8
10
12
14
0.0
2.5
5.0
7.5
10.0
12.5
15.0

Data Wrangling

Because each label corresponded to a 50x50 pixel area of each tile, we reshaped the data into a four dimensional array of the following shape:

(2500, 256, 12, 60)

Respectively, these values represent: number of pixels per label after flattening (50x50), number of labels per tile (16x16), number of bands, and number of tiles.

Method 1: Statistical Learning

After being reshaped into a two-dimensional array, the data was split into training and testing sets and used to train a **SVM, Random Forest, and k-NN** model.

```
# train SVM model using training sets
svm_classifier.fit(X_train, y_train)
```

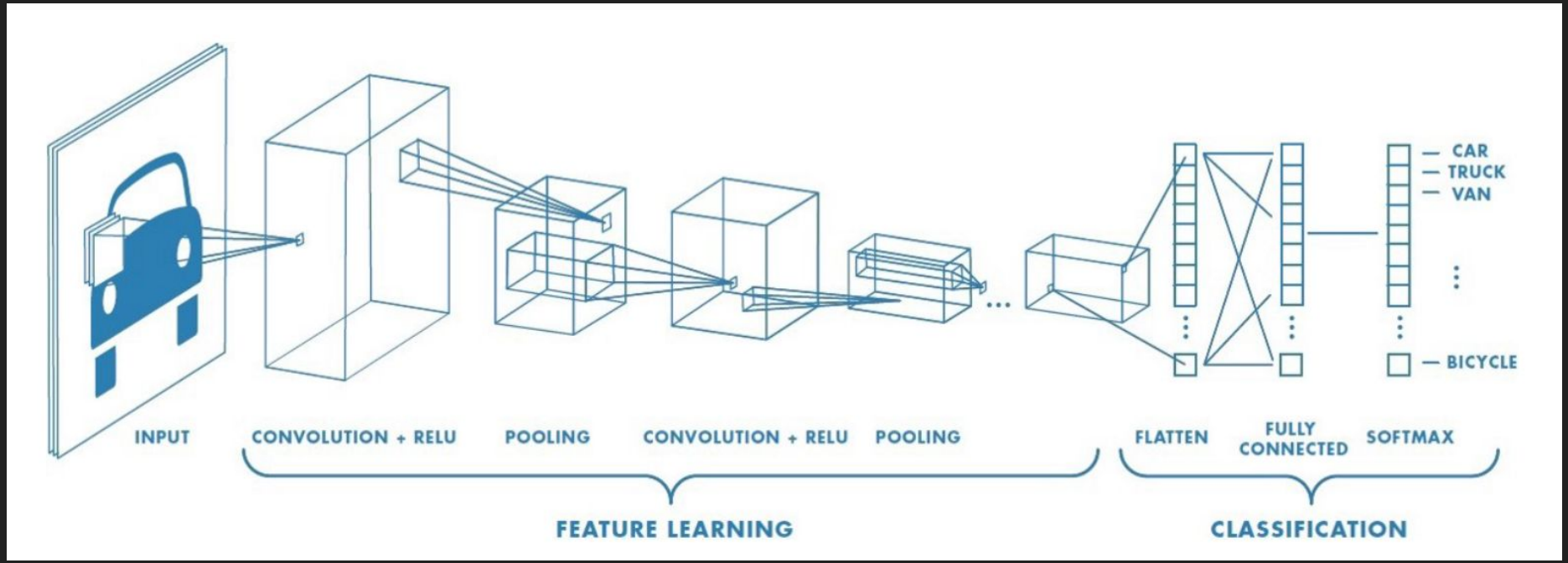
```
# train RF model using training sets
rf_classifier.fit(X_train, y_train)
```

```
# fit k-NN model to training data
knn_classifier_5.fit(X_train,y_train)
```

Note: Because the statistical learning models needed the data in two-dimensional form, the models could only handle a 20-tile subset of the total 60 tiles given the enormous size of the flattened data.

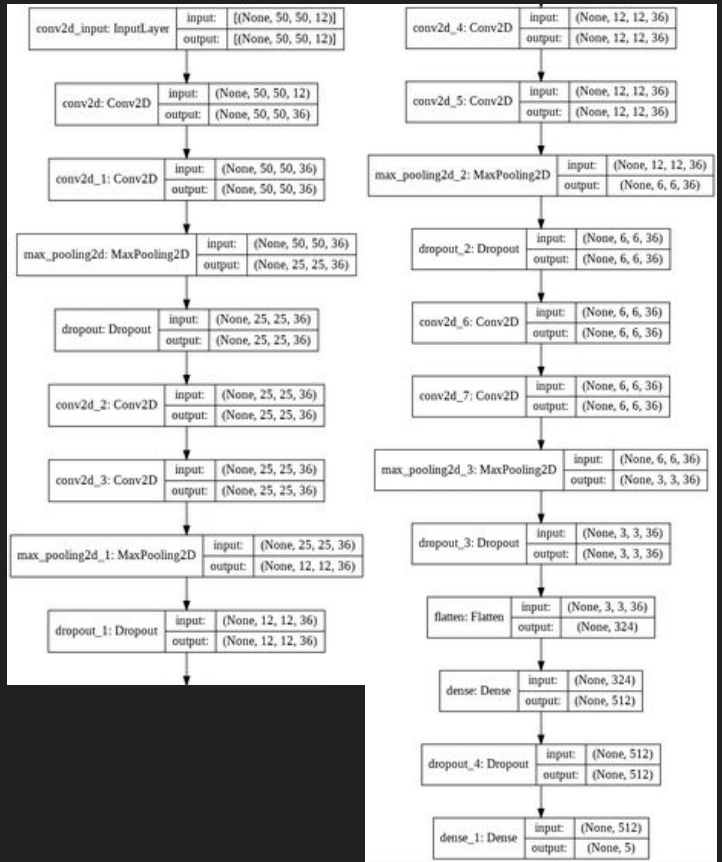
Method 2: Deep Learning

CNN network



Method 2: Deep Learning

- The model was optimized by changing the size of filters and maxpooling size.
- We ran the convolution network twice before maxpooling and reducing the dimension.
- We played around with the final dense layer in terms of features.
- We used different epochs and batch sizes firstly to wrangle the large dataset but also to increase testing accuracy while reducing loss



Results: Statistical Learning

Even after parameters were optimized, all of the statistical learning models returned unfavorable accuracy scores hovering around **0.49**.

```
# SVM testing accuracy
y_test_predict = svm_classifier.predict(X_test)
print("Testing accuracy: " + str(accuracy_score(y_test, y_test_predict)))
```

```
Testing accuracy: 0.486328125
```

```
# RF testing accuracy
y_test_predict = rf_classifier.predict(X_test)
print("Testing accuracy: " + str(accuracy_score(y_test, y_test_predict)))
```

```
Testing accuracy: 0.4850260416666667
```

```
# k-NN testing accuracy
y_pred_5 = knn_classifier_5.predict(X_test)
accuracy_score(y_test,y_pred_5)
```

```
0.4850260416666667
```

Results: Deep Learning

After optimization, the deep learning model returned a robust **0.7172** accuracy.

```
# training
model.fit(
    X_train,
    y_train,
    batch_size=36,
    epochs=10,
    validation_data=(X_test,y_test))
```

```
Epoch 1/10
356/356 [==============================] - 211s 590ms/step - loss: 1.0051 - accuracy: 0.5129 - val_loss: 0.9406 - val_accuracy: 0.4902
Epoch 2/10
356/356 [==============================] - 211s 593ms/step - loss: 0.9088 - accuracy: 0.5336 - val_loss: 0.9207 - val_accuracy: 0.4902
Epoch 3/10
356/356 [==============================] - 215s 605ms/step - loss: 0.8792 - accuracy: 0.5408 - val_loss: 0.9761 - val_accuracy: 0.4902
Epoch 4/10
356/356 [==============================] - 217s 611ms/step - loss: 0.8759 - accuracy: 0.5550 - val_loss: 0.8904 - val_accuracy: 0.5777
Epoch 5/10
356/356 [==============================] - 215s 603ms/step - loss: 0.7924 - accuracy: 0.6327 - val_loss: 0.9087 - val_accuracy: 0.5664
Epoch 6/10
356/356 [==============================] - 207s 581ms/step - loss: 0.7522 - accuracy: 0.6516 - val_loss: 0.7923 - val_accuracy: 0.6359
Epoch 7/10
356/356 [==============================] - 200s 562ms/step - loss: 0.7325 - accuracy: 0.6661 - val_loss: 0.7647 - val_accuracy: 0.6605
Epoch 8/10
356/356 [==============================] - 201s 564ms/step - loss: 0.6933 - accuracy: 0.7007 - val_loss: 0.8247 - val_accuracy: 0.6422
Epoch 9/10
356/356 [==============================] - 202s 568ms/step - loss: 0.6887 - accuracy: 0.7009 - val_loss: 0.7290 - val_accuracy: 0.6922
Epoch 10/10
356/356 [==============================] - 201s 564ms/step - loss: 0.6732 - accuracy: 0.7147 - val_loss: 0.6939 - val_accuracy: 0.7172
<tensorflow.python.keras.callbacks.History at 0x7f4ed052c6d0>
```

Discussion

Statistical Learning	Deep Learning
● Unreliable, largely unusable ● Could not handle >20 tiles ● Far slower run time	● Strong accuracy score ● Could easily handle all 60 tiles ● Blazing fast

Next Steps

To refine these models further, two main options still remain:

1. Use available images taken from **satellites other than Sentinel-2**.

2. Reduce the dataset from four labels to a **binary class** of (1) human settlements lacking electricity and (2) everything else.

Code links

1. Statistical learning model:
    a. https://drive.google.com/file/d/1zL6M9BxGVBS1Mut0REusrz4e7Fe7EbLy/view?usp=sharing
2. Deep learning model:
    a. https://drive.google.com/file/d/1z939ZfD_NyeAGO7dFDdtMIfN7Rb7y7SS/view?usp=sharing