Automatic detection of residential PV arrays and its features from satellite imagery

Kaushik Sadashiva Reddy, Khalid Almubarak, Byoungdoo Kong, Raghavan Venkataraman
Department of Electrical and Computer Engineering
University of Massachusetts Amherst

***Abstract* — Renewable energy, in particular, solar power is more accessible, affordable and prevalent today than in the past. This has allowed small residential installations to grow at a rapid rate. Solar energy can provide several benefits including economic growth, climate change mitigation and national electricity security if they can be integrated into the electric grid. However, it is difficult to obtain precise information on distributed PV's using existing manual methods like surveys as it is expensive and time consuming. Also, the full extent of solar PV is not well understood as the data for residential and commercial solar installations is typically reported only at the level of the state. In this paper, we present an automated method to detect the PV arrays and its features (size, orientation) using aerial imagery as input to machine learning algorithms. This information can be useful for government agencies and utilities to understand the distributed PV installations at a state or national level and successfully integrate them into the electric grid. Utilities, in particular, face many challenges integrating such a high amount of distributed generation and therefore will benefit greatly from understanding the magnitude and distribution of solar PV.**

***Keywords* — solar energy, image processing, satellite imagery, photovoltaic, machine learning, deep learning**

I. INTRODUCTION

Solar power in the United States includes utility-scale solar power plants as well as local distributed generation, mostly from rooftop photovoltaics. In 2016, 39% of all new electricity generation capacity in the country came from solar, more than any other source and ahead of natural gas (29%). By 2015, solar employment had overtaken oil and gas as well as coal employment in the United States. This is mainly because the quantity of solar photovoltaic (PV) arrays has grown rapidly in the United States in recent years, with a substantial proportion of this growth due to small-scale, or distributed, PV arrays as shown in Figure 1.

These small-scale installations are often found on the roofs of commercial structures, or residential homes and therefore are often referred to as rooftop PV.

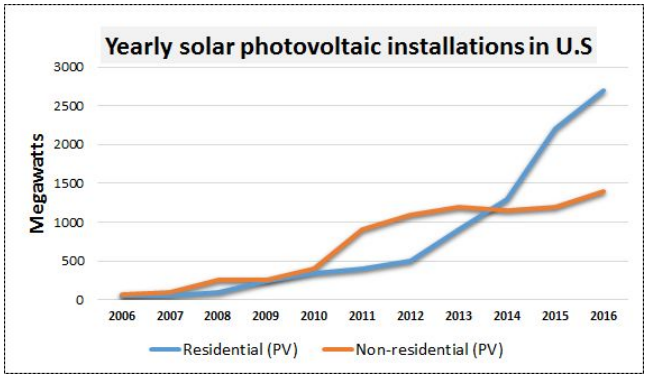


Figure 1: US Solar Market Forecast*(Source: GTM Research, 2017)*

Although prices of solar PV differ widely depending on the size and the complexity of the system, the increase in the residential PV installations is largely influenced by the reduced solar PV prices as shown in Figure 2.

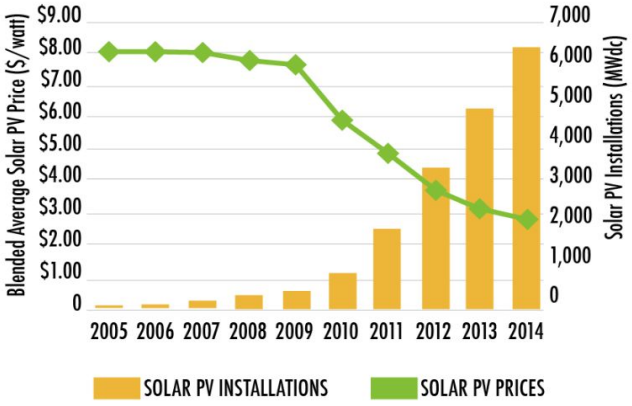


Figure 2:Price changes influence PV installations*(Source: GTM, 2015)*

Distributed PV technology offers significant benefits. Solar energy is a renewable resource that is available everywhere in the world. Solar PV technologies are small and highly modular and can be used virtually anywhere, unlike many

other electricity generation technologies. Unlike traditional power plants, solar PV has no fuel costs and low operation and maintenance costs.[1] However, integrating solar PV into existing power grids is challenging in terms of grid reliability. The grid features such as voltage regulation and primary frequency regulation can be affected by solar PV. For purposes of grid planning, it is valuable to know the size, orientation, and distribution of solar arrays in a given region. To understand and evaluate the factors driving distributed PV, and to aid in its integration, there is growing interest among government agencies, utilities, and third-party decision makers in detailed information about distributed PV; including the locations, power capacity, and energy production of existing arrays. In particular, utilities have to monitor renewable energy such as wind and solar power for grid reliability. As the amount of distributed generation in distribution systems increases, neighboring distribution systems, and even transmission can be affected by the distributed generation. Therefore, utilities need a new system such as distributed PV monitoring system to analyze the power system. However, installing the infrastructures and facilities is very costly. Also, the availability of a nationwide map of solar-powered rooftops can spur further adoption of solar power.

Although the available information on distributed PV is expanding, it is nonetheless difficult to obtain. Existing methods of obtaining this information, such as surveys and utility interconnection filings are costly and time-consuming. They are also typically limited in spatial resolution to the state or national level. On the other hand, identifying solar panel arrays in high-resolution satellite imagery is a fast, scalable, and inexpensive method to obtain solar PV estimates. [2] describes an approach for collecting distributed PV information that relies on using computer algorithms to automatically identify PV arrays in high-resolution color aerial imagery. Our goal was to reproduce the results in [2] and potentially build upon their ideas to extract other features of PV arrays like the size and orientation. Orientation is important since it influences the power-producing ability of a solar panel as a function of time-of-day. For example, west-facing arrays could produce more power in the afternoon and less in the morning.

This work can help in the collection of distributed PV information at a very high geo-spatial resolution. Also, because the approach is automated, it is relatively inexpensive to apply and to do so repeatedly as new imagery becomes available.

Ⅱ. CHALLENGES

The primary challenge was to develop an algorithm that can reliably identify PV installations from satellite imagery. After the identification, we try to predict the size and orientation of PV arrays.

[2] proposes a supervised algorithm to address the challenge of detecting PV arrays in aerial imagery. The main component of the algorithm described in [2] is a supervised Random Forest (RF) classifier [3], which assigns a "confidence" to each pixel in an image indicating its likelihood of corresponding to a PV array. We were set out to validate the results of this approach, while also trying to use other methods like a convolutional neural network (CNN) to explore the possibilities of a weakly supervised approach.

[2] does not deal with finding the size and orientation of the PV arrays. The key challenge is to build a new dataset that contains features pertaining to solar panels and labelling them accordingly and using this to train an efficient classifier that can reliably detect solar PV arrays from these high resolution images. Post that, we need to fit a model that can effectively use the feature set to predict the PV array size and orientation. Since the dataset we use has no orientation information, we manually find the orientation for a portion of the PV arrays as described in Section V and use this information to train a model.

Ⅲ. DATA SET DESCRIPTION

For this analysis, we use two different sets of data. The first data set has human annotations of the exact location of the PV arrays in the image which is used as input to a supervised learning algorithm. The second data set consists of a large number of unannotated images, which is used as input to an unsupervised learning algorithm.

The first dataset is created by the Energy Data Analytics Lab at Duke University that is publicly available for download at [5]. All the imagery was collected in the same month in 2013, using aerial photography. The imagery has a spatial resolution of 0.3 meters per pixel, and all the imagery has been ortho-rectified. [5] contains the geospatial coordinates and border vertices for 19,433 solar

panels across 601 high resolution images from four cities (Fresno, Stockton, Oxnard, and Modesto) in California in two CSV files. Images are each 5000 by 5000 pixels, but with varying resolution. Over 100 images of Fresno are available in [5], from which [2] randomly sampled 60 of the available images for their analysis. We use all the images from Modesto, Oxnard, Stockton and a portion of the images in Fresno totalling 187 images for our analysis.

City	No of Images	No of PV arrays
Fresno	34	2082
Modesto	18	582
Oxnard	50	1595
Stockton	85	2546

In order to develop an effective computer vision algorithm, as well as accurately assess its performance, it is necessary to have the precise locations where PV installations appear in the aerial imagery. In order to obtain this information, human observers have visually scanned the imagery and annotated all the (visible) PV arrays. For improved quality, two annotators scanned each part of the imagery, and their annotations were combined by taking a union of each observer's annotations. There is a total of 19,433 individual solar PV regions in the imagery after the merging process.

The second data set we use was created as part of a Stanford project known as DeepSolar. This dataset is constructed by collecting over 320,000 satellite images using Google Static Maps API. These images were collected over 50 different cities in the United States. In addition, the satellite imagery has a spatial resolution less than 30 cm. This dataset consists of positive and negative images as global annotation corresponding to images with solar panels and images without solar panels respectively. For this analysis, we use only a portion of the large dataset that is available at our disposal due to computational limitations. The training set consists of 30,000 images, the validation set comprises 5,000 images and we use 12,000 images as our test set.

Ⅳ. BACKGROUND & RELATED WORK

The scope of the algorithms in [2] falls broadly under the category of supervised machine learning. The authors propose an approach to use four processing steps to achieve the detection of rooftop PV as shown in figure 3.

The first step is feature extraction in which the algorithm extracts image statistics or features to map 3-channel RGB image into an M-channel image. The M represents the number of features extracted around each pixel.

Then, they used a machine learning classifier to assign a probability or confidence to each pixel in the image to determine whether it corresponds to a PV array or not. The output of this step is a spatial map of where PV arrays are likely to be. They used RF as the classifier in the second step which is a supervised statistical classification that uses ensemble decision Trees. This is followed by the post processing step to improve the accuracy of the confidence map. This step is identifying the individual pixels with high confidence and growing regions of the pixels around them. The confidence values of the pixels outside these regions are set to zero. Finally, the last step in this process is to do object detection. The identified contiguous pixels with high confidence correspond to a single PV array. Each one of these identified contiguous pixels is returned as the detected object.

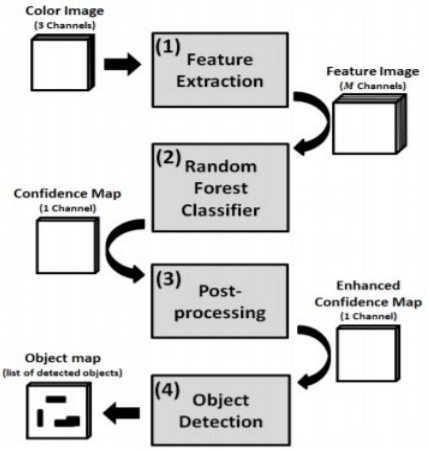


Figure 3: An overview of the PV detection algorithm used in [2]

We do not follow the exact approach as described above due to its complexity and time limitations. However, we use existing libraries in Python for image processing and feature extraction which is simpler to use and reasonably efficient. More details are described in Section V.

We also try a weakly supervised end to end machine learning approach that does not need PV array annotated images. A deep learning method is an example of end to end learning that has been widely researched and successfully employed for various image classification and object detection tasks. Convolution neural network (CNN) is an example of such a deep learning method that can achieve better feature extraction compared to state of the

art hand crafted feature detectors. CNN as shown in Figure 4 consists of one or more convolutional layers that uses sub-sampling steps that work as feature extraction [4]. This is followed by one or more fully connected layers that are equivalent to classification or object detection. The convolution layer is designed to deal with images as input by looking at local connections that are tied by "weights". The weights work as filters of the image to extract better features. The features extracted from the convolutional layers can be directly fed to the fully connected layers, but this becomes computationally intensive as the output from the layers usually have large dimensions (~ 7000). So, this is followed by a subsampling layer called the pooling layer which reduces the dimensions of the features by aggregating results at various regions of the image. This is followed by a fully connected layer that usually consists of three multilayer neural networks and a Softmax classifier. The softmax classifier assigns probability for each region or cell in the image representation that comes from the previous layer. The region that has a valid probability is then surrounded by a bounding box.

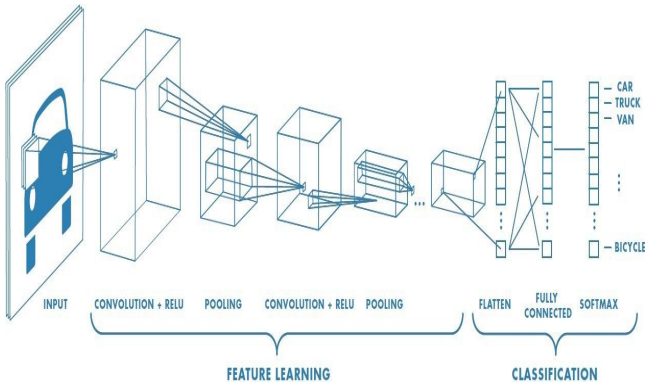


Figure 4: Convolutional neural network architecture*(Source: towardsdatascience.com)*

We also tried an alternative method of using a pre-trained model. This refers to initializing the CNN weights with the weights of another network that is already trained. For example, [6] describes how the weights of a CNN were trained with ImageNet dataset. Then, the pre-trained CNN can be fine tuned to identify PV arrays from aerial satellite imagery.

V. IMPLEMENTATION

As mentioned in the previous sections, we are exploring broadly two different methods of learning - supervised and weakly supervised. In the supervised case, we use the imagery dataset [5] that has human annotations of PV arrays in each image. The weakly supervised method uses the images available at [12] which only have a global annotation i.e if the images are positive (has solar panels) or negative (no solar panels).

1) Supervised Learning

The implementation can be broadly divided into the following steps:

i) Data Labelling
ii) Data Munging
iii) Image Segmentation
iv) Feature Extraction
v) Classification
vi) Regression

i) Data Labelling

As far as the size of the PV array is concerned, the data set has a feature called 'area_pixels' which correspond to the actual size of the PV arrays in square meters. We verified the correctness of this data using a tool on Google Earth as shown in Figure 5, as well as checking the actual ground truth by visiting the company website that installed the solar panels.

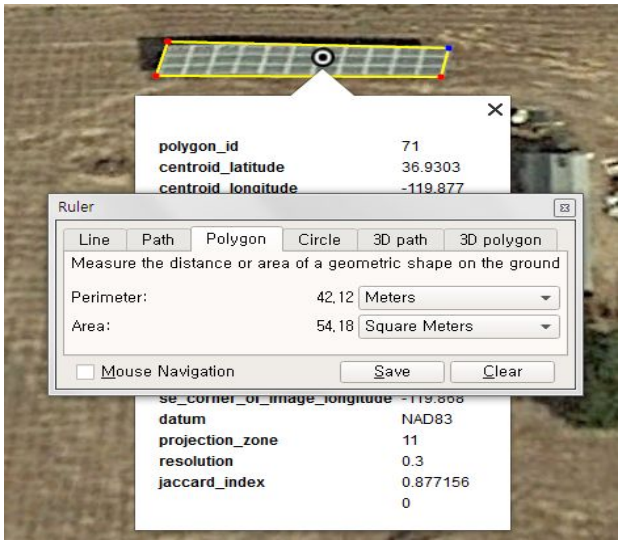
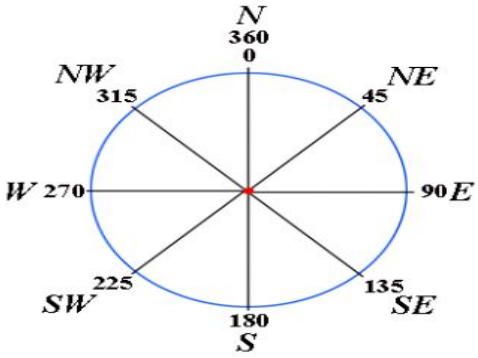


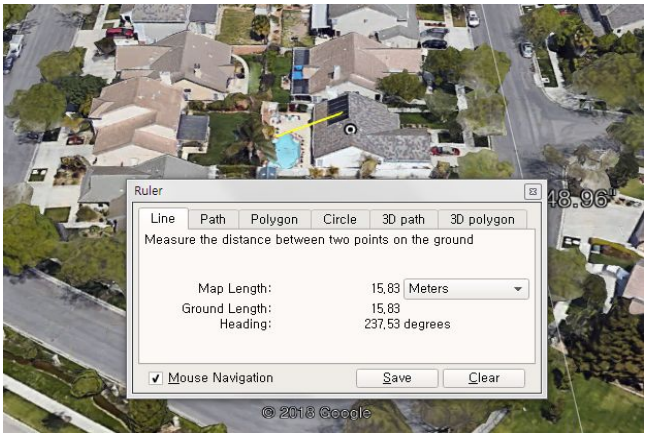
Figure 5: Size of PV array using Google Earth

However, the dataset did not provide any orientation information, so we manually added the labels for about 8000 solar panels using Google Earth.We use a tool to draw a line along the panel edge, depending on the tilt of

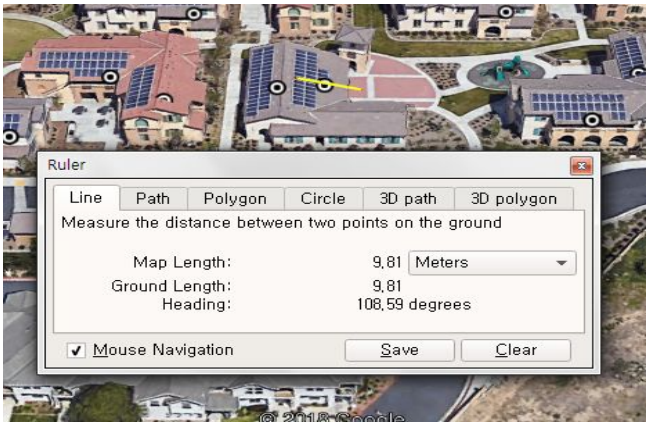
the roof and this gives us an angle of orientation based on the compass below.



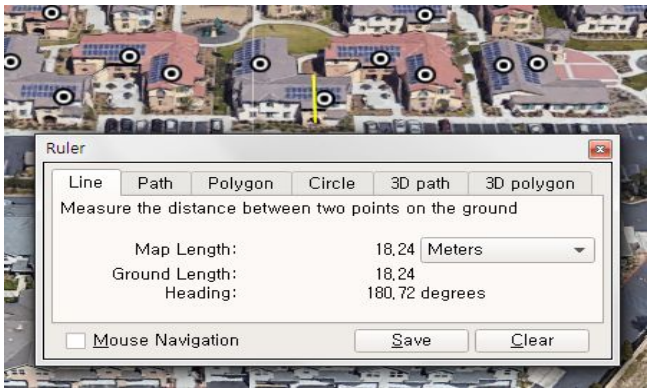
The images below show panels that are oriented to the West, East, South and North directions respectively and their corresponding orientation angles as given by the tool.



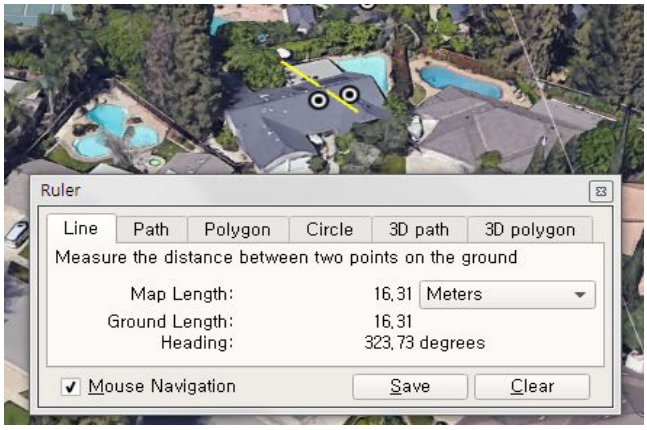
West Facing



East Facing



South Facing



North facing

We haven't verified the correctness of the orientation angle as we had no access to actual ground truth data.

ii) Data Munging:

The dataset gives two CSV files; 'polygonDataExceptVertices.csv' contains the polygon ids for each PV array, the city it belongs to, it's area and pixel coordinates of the centroids which can be used to plot in any given image. 'polygonVertices_PixelCoordinates.csv' contains the number of vertices each polygon has and also the pixel coordinates for each vertex. The main cleaning operation is on the this data. The table contains an ID ('dataPolygonsPixels1') for each solar panel as well as the corresponding vertices. Each x, y point in the vertices is broken out into a seperate column. To make the analysis easier, we converted these into a single coordinates list and merged the two CSV files to create a new dataframe as shown in Figure 6.

	polygon_id	centroid_latitude_pixels	centroid_longitude_pixels	city
0	1.0	107.618458	3286.151487	Fresno
1	2.0	45.977659	3286.352946	Fresno
2	3.0	22.280851	3303.465657	Fresno
3	4.0	2048.362567	2547.366116	Fresno
4	5.0	2060.014890	2529.504997	Fresno
5	6.0	2058.955498	2501.533607	Fresno
6	7.0	2059.194862	2474.746434	Fresno

image_name	polygon_vertices	area_pixels	orientation
11ska460890	[(132, 3360), (88, 3249), (74, 3214), (82, 321...	136.192872	200.51
11ska460890	[(70, 3361), (13, 3218), (13, 3217), (12, 3216...	155.511714	200.51
11ska460890	[(48, 3358), (48, 3358), (42, 3361), (39, 3362...	111.796591	200.51
11ska460890	[(2068, 2572), (2067, 2571), (2067, 2571), (20...	62.004008	226.75
11ska460890	[(2091, 2564), (2022, 2504), (2022, 2504), (20...	95.480150	226.75
11ska460890	[(2090, 2537), (2058, 2508), (2021, 2477), (20...	95.072936	226.75
11ska460890	[(2091, 2508), (2090, 2508), (2090, 2509), (20...	96.092942	226.75

Figure 6: Merged Data Frame

We used this dataframe to explore the annotations available and was also used to create the image segments corresponding to the solar panels as described in the next section.

iii) Image Segmentation:

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). It consists of grouping pixels together based on likeness (e.g. color value, proximity). Relevant information can then be extracted and aggregated across each image segment, instead of from the individual pixels themselves. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Additionally, segment-based classification can often achieve better accuracy than a pixel-based method.

For this, we use scikit-image, a collection of algorithms for image processing in Python. We tried three different algorithms namely Felzenszwalb's efficient graph based segmentation, Quickshift image segmentation and Quickshift image segmentation. Quickshift worked best for the images we had. Quickshift is a 2D image segmentation algorithm, based on an approximation of kernelized mean-shift. Therefore, it belongs to the family of local mode-seeking algorithms and is applied to the 5D space consisting of color information and image location. One of the benefits of quickshift is that it actually computes a hierarchical segmentation on multiple scales simultaneously. Quickshift has two main parameters: *sigma* controls the scale of the local density approximation, *max_dist* selects a level in the hierarchical segmentation that is produced. There is also a trade-off between distance in color-space and distance in image-space, given by *ratio*. Figure 7 shows an example segmentation using quickshift algorithm applied to a satellite image of the Fine Arts Center at UMass Amherst containing a solar panel. As you can see, the image has been partitioned into contiguous regions with similar pixel values.

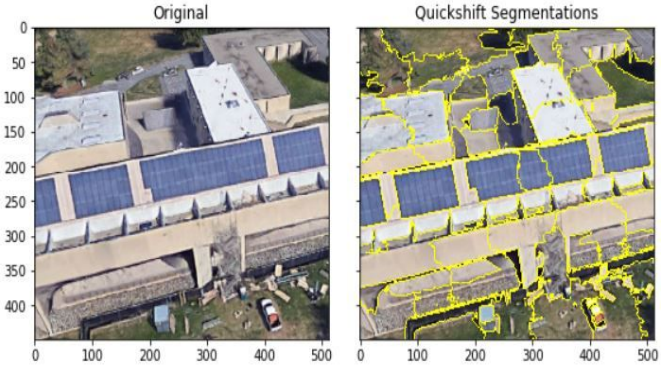


Figure 7: Example Segmentation using Quickshift

The satellite image above is only 400x500 pixels. So, we modify the segmentation process for the larger 5000x5000 pixel images. In particular, we use the annotations data to "segment" the solar panel arrays. In the larger images, solar panels make up a tiny fraction of the total area (< 1.0%). Therefore, to generate the segments that do not contain solar panels, we sample sections of the image and run the segmentation algorithm on these

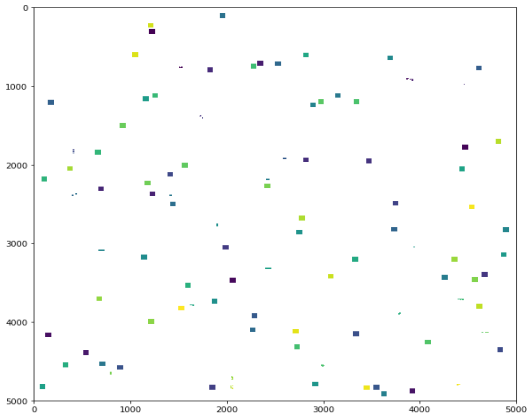


Figure 8: Panel and Non Panel segments

sections. In Figure 8, the colored squares are the random windows that contain the non-panel areas of the image.

Here, there are 75 windows and each window is of size 60*60. To ensure they do not overlap with any of the solar panels, panel regions are labelled by its polygon id while the non panel regions are labelled by numbers starting from zero. Rest of the image is labelled as 'NaN' and will not be

used. The next step is to extract features from these segments.

iv) Feature Extraction

Feature extraction is a key component of image analysis and object detection. It is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set. Here, we aim to classify segments or superpixels and will therefore calculate features based on these regions of pixels, rather than the pixels themselves. Once a feature set is established, we will then be able to train a classifier to identify regions that are best representative of solar panels and non solar panels.
Features are mainly spatial and color related. It contains summary statistics for each color channel (e.g. mean, min, max, variance) but will also include other descriptive values such as area, perimeter and circleness. We use scipy.stats and skimage.measure.regionprops to calculate these values. We then combine these features into a new dataframe and label the positive features as 1 and negative features as 0 corresponding to solar panels segments and non solar panel segments respectively. The feature columns are : image id, segment id, perimeter, area, circleness, centroid, r_min, r_max, r_mean, r_variance, r_skewness, b_kurtosis, panel_class, area_pixels, orientation. Here area_pixels = area in meter square of the PV array. The shape of this training data frame is (78642, 28). We will use this to train a classifier.

v) Classification:

We tried two different types of classifiers namely Logistic Regression and Random Forest.

Logistic Regression :
We choose logistic regression because of its simplicity. The training data is split into 70% training and 30% testing. The Null/Baseline Accuracy =max(np.mean(y), 1 - np.mean(y)) = 91.5% , which means even a dumb classifier that predicts 0 (no panel) for everything, will get it right 91.5% of the time. The training was done with the following parameters to GridSearchCV : l1 and l2 regularization ; liblinear solver and value of C (inverse of regularization) from 0.0001 to 500 with 50 divisions. So, after a total of 500 fits, best estimator was LogisticRegression(C=20.40825918367347, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, warm_start=False) with accuracy of 96.53%. Cross Validation score on the test set was 96.73% which is 61% better than the baseline accuracy. The training and test accuracy is close which means we're not overfitting or underfitting the model.

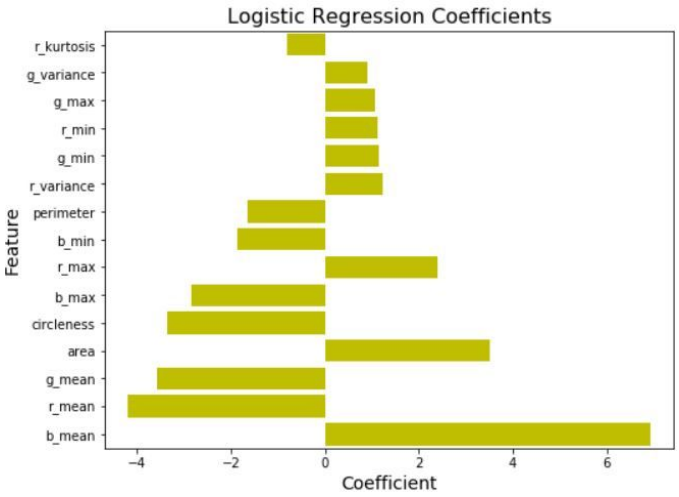
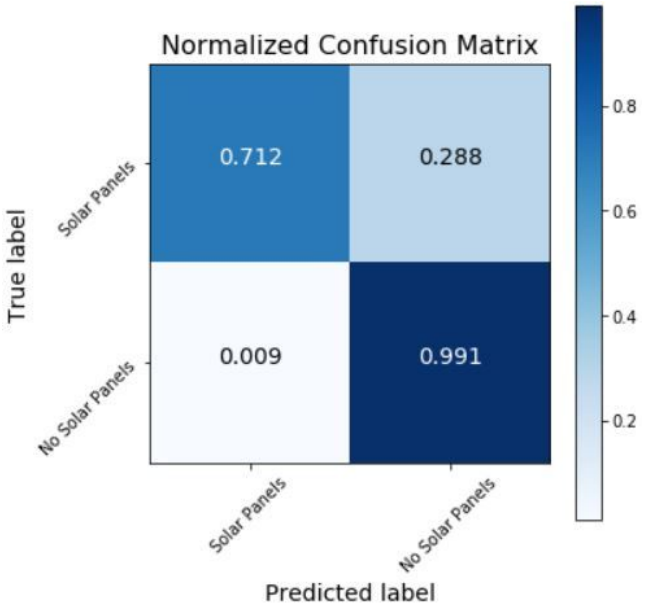


Figure 9: Feature Importance

Figure 9 shows the weights/coefficients of each feature. The features that are most meaningful are the mean color values, area, and circleness.
A confusion matrix analysis gives a better understanding of the performance of the model.



As we can clearly see, 71.2% of the solar panels are being correctly classified as solar panels. However, 28.8% of the solar panels are being misclassified as shown in Figure 11. 99% of the non solar panel segments are being correctly classified which is very good.

Random Forest:

Random forests (RF) or random decision forests are an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. Decision Tree models consists of series of decisions, where an element of the data is tested. These tests are derived by evaluating how effective they are at dividing the data into homogenous groups or splits. Random decision forests correct for decision trees' habit of overfitting to their training set. They are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance with a small increase in bias[11]. Also, it is efficient to learn complicated, non-linear relationships. Hence, for those reasons we choose an RF classifier.

Training data was split again into 70% training and 30% testing. After about 245 fits using GridSearchCV, the best estimator was RandomForestClassifier (bootstrap=True, criterion='gini', max_depth=None, max_features=5, min_samples_leaf=1, min_samples_split=2, n_estimators =50, n_jobs=1, oob_score=False) with an accuracy score of 98.04%. The score on the test set was 98.30% which is 79% better than the null accuracy of 91%.

The training and test accuracy is close which means we're not overfitting or underfitting the model.

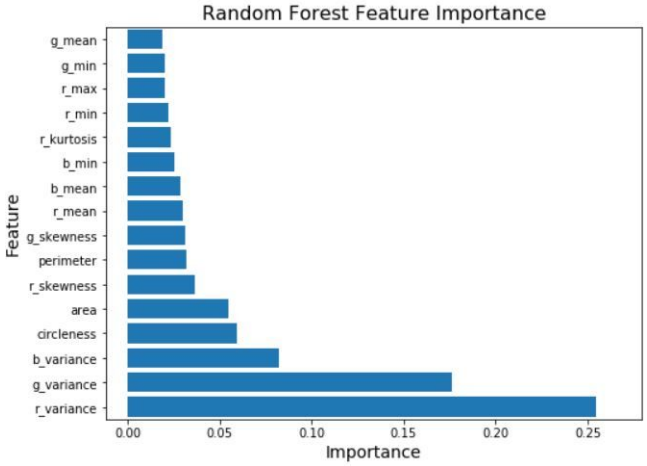
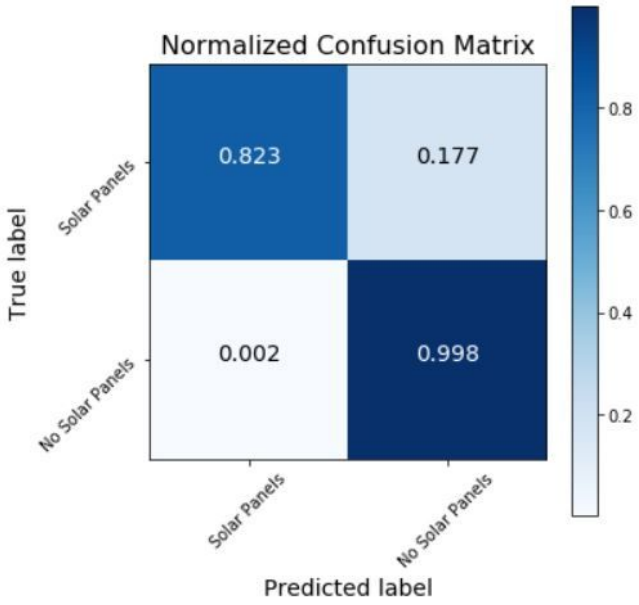


Figure 10: Feature Importance

From Figure 10, it is clear that just like Logistic Regression, Random Forest placed high importance on the area and circleness features. However, the most important features were the color variances.

A confusion matrix analysis gives a better understanding of the performance of the model.



As we can clearly see, 82.3% of the solar panels are being correctly classified as solar panels, so it performs better than Logistic Regression model. 99.8% of the non solar panel segments are being correctly classified which is very good. However, still 17.7% of the solar panels are being misclassified as shown in Figure 11.

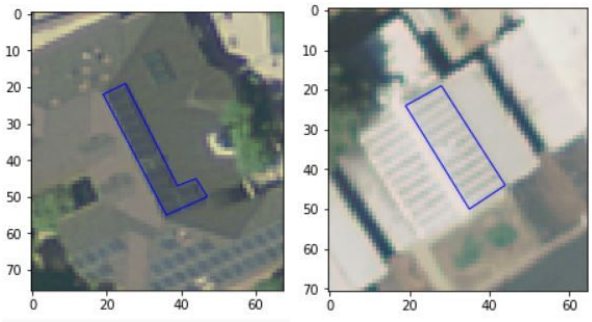


Figure 11: Misclassified Panels

Figure 11 shows PV arrays that had the least probability of being classified as a solar panel and hence were misclassified by the model. These are mostly panels that appear to have the same color as the roof and tend to blend in with the roof of the building.

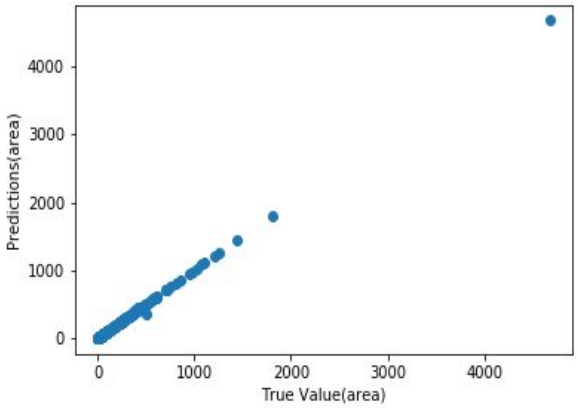
vi) Regression:

Now that we have identified the solar panel segments from the images, we can feed these segments to a regression model to predict the features like area and orientation of the solar panels.

Area :

70-30 train-test split : X_train : (4559, 21) ; Y_train: (1955, 21) X_test: (4559, 1); Y_test: (1955, 1).

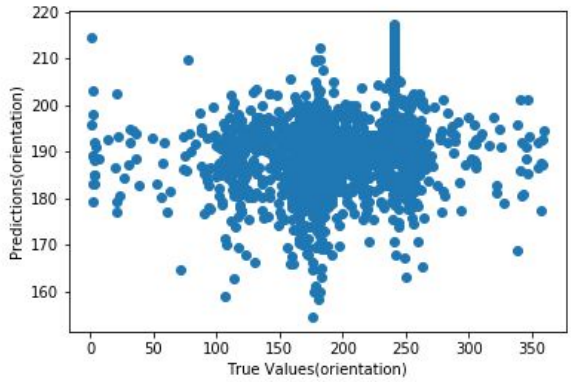
After fitting about 50 models, the best model gave an accuracy score of 99.97% which is extremely good.
Model with lowest estimated out of sample error (Mean Absolute Error) has a Ridge penalty with Alpha = 1 gives an error of 0.93 metres square.



As we see from the graph above, most of the predicted values almost completely overlap with the true values with a very small error. So this method is very efficient to predict the area of solar panel arrays in aerial imagery.

Orientation :
70-30 train-test split : X_train : (4559, 21) ; Y_train: (1955, 21) X_test: (4559, 1); Y_test: (1955, 1).

After fitting about 50 models, model with lowest estimated out of sample error (Mean Absolute Error) has a Ridge penalty with Alpha = 10 gives an error of 32.85 degrees.



The above graph shows the distribution of the true orientation values versus the predicted values. It can be seen that the model predicts that most of the panels are south facing. This is happening because most of the panels(80%) are south facing, so probably the panels that face north, east and west are considered as outliers.
Also, the grid lines or panel edges, which we used to manually obtain the orientation angle could be confusing the model. Also, since the images in this dataset are orthorectified(tilt features removed), it could be a potential reason for the errors in predictions.

One possible solution to this problem is to create segments closer to the panel (roof,ground) and extract those features to have it as part of the training set for each panel. Then the model could potentially learn from the tilt of the roofs to accurately predict the orientation of the PV arrays.

Limitations :

The above analysis was done on annotated images, where we could segment the image in such a way that the entirety of the solar panel was included in its own region. But when we feed unannotated images and run the Quickshift algorithm on it, the segmentation is sometimes inaccurate, especially due to the grid lines. This results in multiple segments for a particular PV array. So, it requires fine tuning of segmentation algorithm parameters and this will change as the resolution of the input images change which can be a very tedious process. In order to counter this problem, we chose to try a weakly supervised model that does not require PV array annotations. This approach is described in detail in the next section.

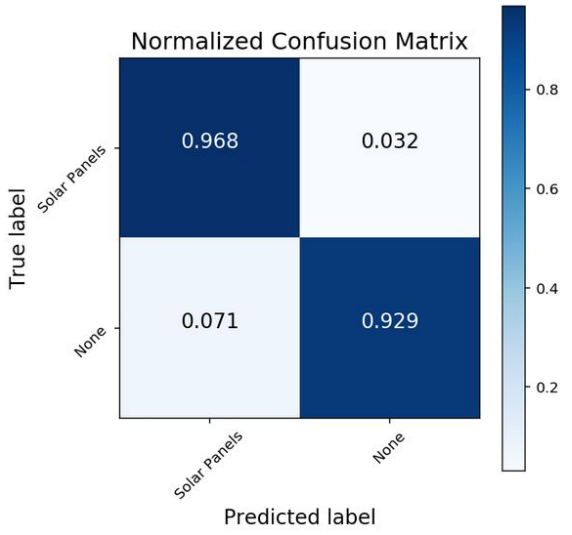
2) Weakly Supervised Learning

Convolution Neural Network

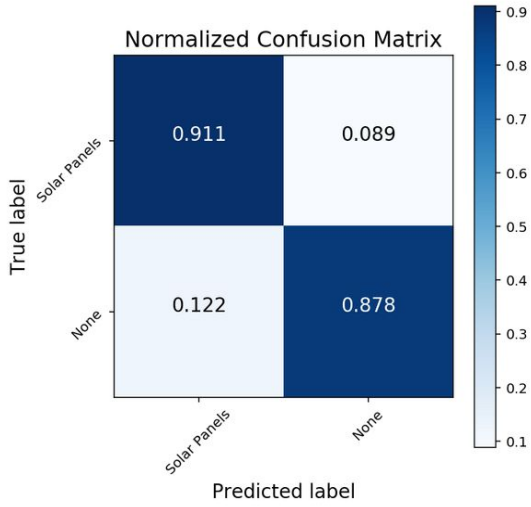
In the convolution neural network implementation, we first try to understand the capability of the convolution neural network to distinguish the images that contain solar panels from the one that does not. We tried a pretrained convolutional neural network and an untrained convolution neural network.

We used the residual neural network architecture because as the neural network becomes deeper in terms of layers, the convergence starts to have a degradation problem. The degradation causes the training accuracy to decrease as the number of layers increase. Therefore, [8] proposed to use residual neural network framework to solve this issue. The

residual neural network combines both the information extracted



Pretrained Initialization



Random Initialization

Figure 12: Confusion Matrix

from the previous layer and the results from the earlier layers to preserve the information from earlier layers as the network becomes deeper. This showed rapid improvement in terms of training the deep neural network.

In addition, the loss function that is used in the convolution neural network was a cross entropy loss function to measure the performance of the classification of both pre trained and random initialization model whose output is a probability value between 0 and 1 for each of the class. Also, another important hyperparameter that we use in our CNN is Relu which is used as an activation function in deep learning models. The Relu function returns 0 if it receives any negative input. However, it returns the value back if it is positive. The function can be written mathematically as follow:

$$f(x) = max(0, x)$$

Further, we trained the images on both settings through 15 epochs. The epochs here refer to one complete presentation of the training set to the deep learning model. The optimizer that we chose is Adam. It is an optimization algorithm that is used to update network weights after each batch. Adam has many benefits over other optimizers such as dealing with non-convex optimization problems. The batch size of the training is 16 instances. The batch size refers to the number of training examples used in one iteration. The approach of the batch size in the implementation falls under mini-batch mode where the number of batch size is greater than one but less than the total number of the dataset size.

The results in figure 12 indicate that convolution neural network can recognize if the images contain a solar panel or not regardless if we start with pretrained weights or random weights. However, the pretrained model achieved high accuracy in a small number of epochs compared to the random initialization as shown in figure 12. This is due to the efficiency of transfer learning. It is usually rare to train the entire convolution neural network from scratch with random initialization because it is difficult to obtain a dataset which is sufficiently large enough to learn an efficient model. Therefore, it is better to use some pretrained model that has a very good feature extractor that is trained on massive dataset such as Imagenet to fine tune it to our specific task. In addition, Figure 13 visualize the weights of the early convolution layers in Residual neural network. This illustrates that early convolution layers learn some common patterns in all of the images such as edges, lines, and color. Therefore, this pattern can be transferred easily to other tasks without any need to initialize from scratch. This is further discussed in [9].

After that, we tackle the problem of an unlabeled dataset by looking into a new learning paradigm known as weakly supervised learning. This method relies only on global labels. In addition, this approach is only possible due to the explicitly characteristic of a convolution neural network.

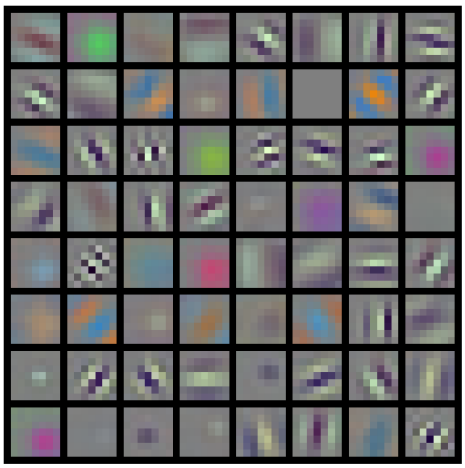


Figure 13 : Visualizing the weight of convolution layers

The authors of [10] proposed a method to leverage global average pooling layers in CNN to localize the objects of an image even though it is only being trained on image-level labels.

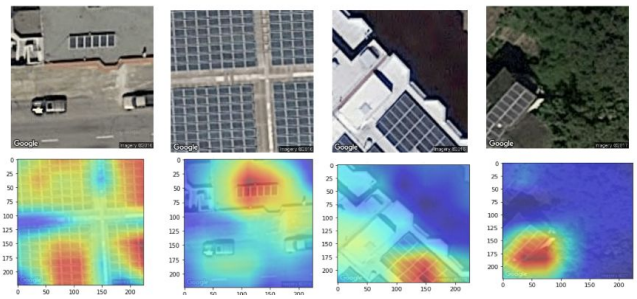


Figure 14 : Heat map of solar panel detection

We tried in the implementation to produce class activation map using global pooling in CNNs. The class activation map illustrates the discriminative image regions that used by CNN to identify the global labels such as found solar panels or not in our case. We use our residual neural network that was trained to recognize the availability of solar panel in an image or not to generate the class activation map. The main idea is to use the global average pooling operation over the feature maps of the image that we are trying to object identified. Then, we produce the class of the image either there is a solar panel or not. After that, we extract the class activation map that determine this class by projecting the weights of the average pooling output on the convolutional feature maps to produce a class activation map that maps to the image using heat map.

Figure 14 shows some of the results that our convolution neural network produced. Even though the results show promising improvement in dealing with dataset without labels, the deep neural network has some limitation and challenges. The first limitation with deep learning model is the problem of interpretability. Many, machine learning experts argue that deep learning is like black box models that we did not exactly how it works that good. This raises concerns about the possibility of a deep learning model to fail in critical tasks. This is why many security issues have been rise such as adversarial attacks that can fool deep learning models to force it to mistakenly classified images. Another concern that even limit the scope of this project is the computation expensive. The convolution neural network needs roughly one day and a half to train through all the constructed dataset. That is a long time to train a model to recognize solar panels. In addition, we were limited to using only 50 layers instead of 150 layers neural network because of limitations in the hardware. Large models that can achieve good results in image classification are difficult to run with small GPUs.

VI. INDIVIDUAL CONTRIBUTIONS

Name	**Contribution**
Kaushik	Literature Review, Coding, Supervised Classification, Final Report
Khalid	Literature Review, Coding, Weakly Supervised Classification, Final Report
Byoungdoo	Literature Review, Orientation Labelling
Raghavan	Orientation Labelling

VII. REFERENCES

[1] Singh GK. Solar power generation by PV (photovoltaic) technology: A review. Energy 2013;53:1–13. doi:10.1016/j.energy.2013.02.057.

[2] Jordan M. Malof , Kyle Bradbury , Leslie M. Collins and Richard G. Newell, "Automatic solar photovoltaic panel detection in satellite imagery," in *International Conference on Renewable Energy Research and Applications (ICRERA)*,2015,pp. 1428-1431.

[3] Breiman L. Random forests. Mach Learn 2001;45:5–32. doi:10.1023/A:1010933404324.

[4] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[5] Bradbury K, Saboo R, Malof J, Johnson T, Devarajan A, Zhang W, et al. Distributed Solar Photovoltaic Array Location and Extent Data Set for Remote Sensing Object Identification. Figshare 2018. https://dx.doi.org/10.6084/m9.figshare.3385780.v3

[6] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database.

[7] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp.2278-2324.

[8] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[9] Pan, S.J. and Yang, Q., 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), pp.1345-1359.

[10] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. and Torralba, A., 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921-2929).

[11] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome(2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.

[12] Yu, J., Wang, Z., Majumdar, A. and Rajagopal, R., 2018. DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States. Joule, 2(12), pp.2605-2617.