Detecting Solar Arrays in Aerial Imagery Data With Supervised Learning

Azu Morales, Viggy Kumaresan, Sicong Zhao, Yifei Wang, Anna Berman

Duke University

Abstract

As the solar industry grows in size, it is becoming increasingly important to monitor consumer solar adoption. Specifically, differences in energy consumption between consumers with solar panels compared to those without is critical to informing future business and policy decisions. The first step in understanding consumer energy consumption is to first delineate homes with and without solar panels. To address this first step, the following analysis evaluates several modeling approaches to identifying solar PV in high resolution aerial imagery data. Our findings suggest a convolutional neural network approach yields high rates of accuracy of solar PV detection from satellite aerial imagery.

Introduction

Solar power currently accounts for 1% of the world's electricity generation. In fact, estimates of solar energy production predict a potential 65-fold growth by 2050, eventually making solar power one of the largest sources of energy across the globe (Energy Transition Outlook, 2018). Solar photovoltaic (PV) power installed on top of rooftops, or solar PV, is estimated to make up thirty percent of this energy generation. In recent years, solar PV power has already begun playing an increasingly large role in US electricity generation. From 2008 through 2017, there was a 39-fold growth in annual solar generation representing an increase of 75,123 GWh (Environment America, 2018).

As consumer solar PV adoption increases, the consumption habits of solar adopters are of vital interest to a variety of stakeholders (Solar Industry Research Data, 2019). Policymakers, for example, are reliant on accurate measures of both the saturation of solar PV as well as on the energy consumption and production rates associated with these installations. This information is integral the structuring of tax incentives and credits for installation and use of solar PV (Matasci, 2019; Solangi et al. 2011). In another example, solar manufacturers also rely on detailed solar PV market insights. In order to acquire new customers, solar manufacturers depend on this information to target specific customer segments and to create customized marketing and sales strategies (Janes, 2014; Rai et al. 2016).

Traditional data sources such as consumer surveys and market research, are costly and time-consuming to collect, and ultimately can often only give a partial or biased view of the market. Satellite imagery, on the other hand, allows us to see overhead views of households all over the country and does not rely on self-reported data. The utilization of these images to detect consumer solar PV has the potential to result in a more consistent and cost-effective view of solar adoption in the US.

The following analysis examines the possibility of developing a model that can reliably identify solar PV installations in satellite aerial images. The use logistic regression, k-nearest neighbors (KNN), gaussian naive bayes, and convolutional neural networks (CNN) are explored. Our findings suggest that a CNN approach yields the highest rates of accuracy of detection.

Background

Previous studies have explored the use of machine learning for satellite image classification. For example, studies have shown that it is possible to automatically detect roads in aerial imagery using neural networks (Mnih et al., 2010) as well as to automatically detect oil spills using supervised clustering techniques (Kubat et al., 1998). These, as well as other applications of automatic detection have been shown to be successful with high rates of accuracy of detection with a low rate of incorrect classification.

Most relevantly, machine learning techniques have shown to be successful at detecting solar PV specifically. A random forest classification approach has been shown to collect distributed PV information over large areas using aerial or satellite imagery. Random forest classification displayed an ability to not only detect solar PV, but to also accurately measure both the shape and size installations (Malof et al., 2016). Moreover, convolutional neural networks have been shown to be effective at identifying solar PV in low resolution satellite imagery with high accuracy of detection and low rates of incorrect classification (Golovko et al., 2017). CNN approaches have not only been shown to accurately detect the existence solar PV, but have also been shown to accurately predict of capacity and the amount of energy installations produce (Malof et al., 2017).

Importantly, in all of the techniques discussed, a supervised learning approach was taken. As with any supervised learning algorithm, a training set of data with labelled classes is required in order to both model the data as well as evaluate performance. Unfortunately, the number of available labelled datasets containing satellite imagery of solar PV is limited. An example of such a dataset is the Distributed Solar Photovoltaic Array Location and Extent Dataset containing geospatial coordinates and border vertices for over 19,000 solar panels across 601 high-resolution images from four cities in California (Bradbury et al., 2016).

Data

The dataset used for our analysis very closely resembles the data published by Bradbury et al. (2016). Our dataset consists of 1,500 satellite images in TIFF format, each image contains a 101 pixels x 101 pixels size and has three color channels (Red, Green, Blue). In total, each image is represented as 101 x 101 x 3 array, which is a total of 30,603 numerical entries with values ranging from 0 and 255.

As previously discussed, in order to take a supervised learning approach, labels for each image is required for both model creation and model evaluation. Each image in our dataset has been labeled as one of two classes, either containing a solar PV, or not containing a solar PV (examples of both classes are shown in Figures 1 and 2). The labels for this dataset were created by human observers who visually scanned the imagery and and annotated all of the visible solar PV. There are 555 images that contains a solar PV and 995 images without a solar PV, which means that the classes in our dataset were not entirely equally balanced.

Fig 1. Examples of images with solar panels. Solar PV instances in this dataset vary in size and color, but share similar shapes.

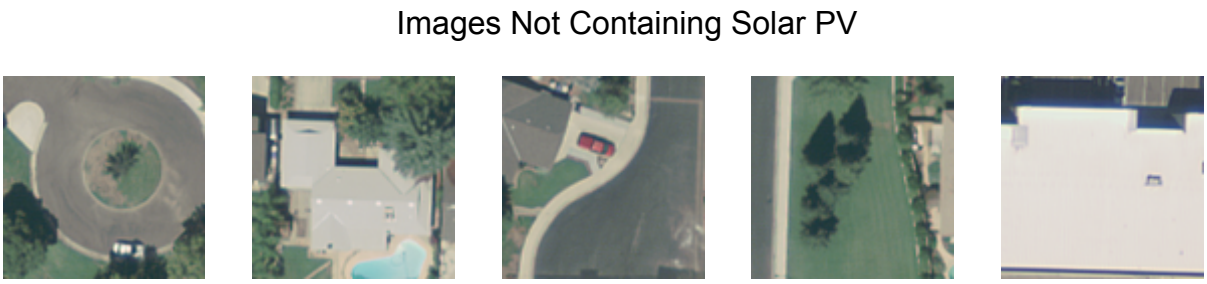

Fig 2. Examples of images without solar panels.

As seen in the Figure 1, the solar PV included in our dataset do not necessarily share standard form. In general, solar PV instances in this dataset vary in size and color and are not always at the center of the image. However, many solar panels tend to be rectangle with sharp angles and edges. Additionally, the surrounding landscape in images of both classes is also rather diverse. Examples of rooftops, pavement, grass, and residential pools can be seen in both classes. In order for our model to be successful, this large amount of diversity needs to be addressed in our model. Additionally, our model should predict the a consistent class regardless of orientation of each image. Therefore, rotation and scaling of images should be tolerated in our final model.

In addition to the data described above, an additional 558 number of images and labels were housed on Kaggle.com. These images were used as a test set during our model selection process.

Methods

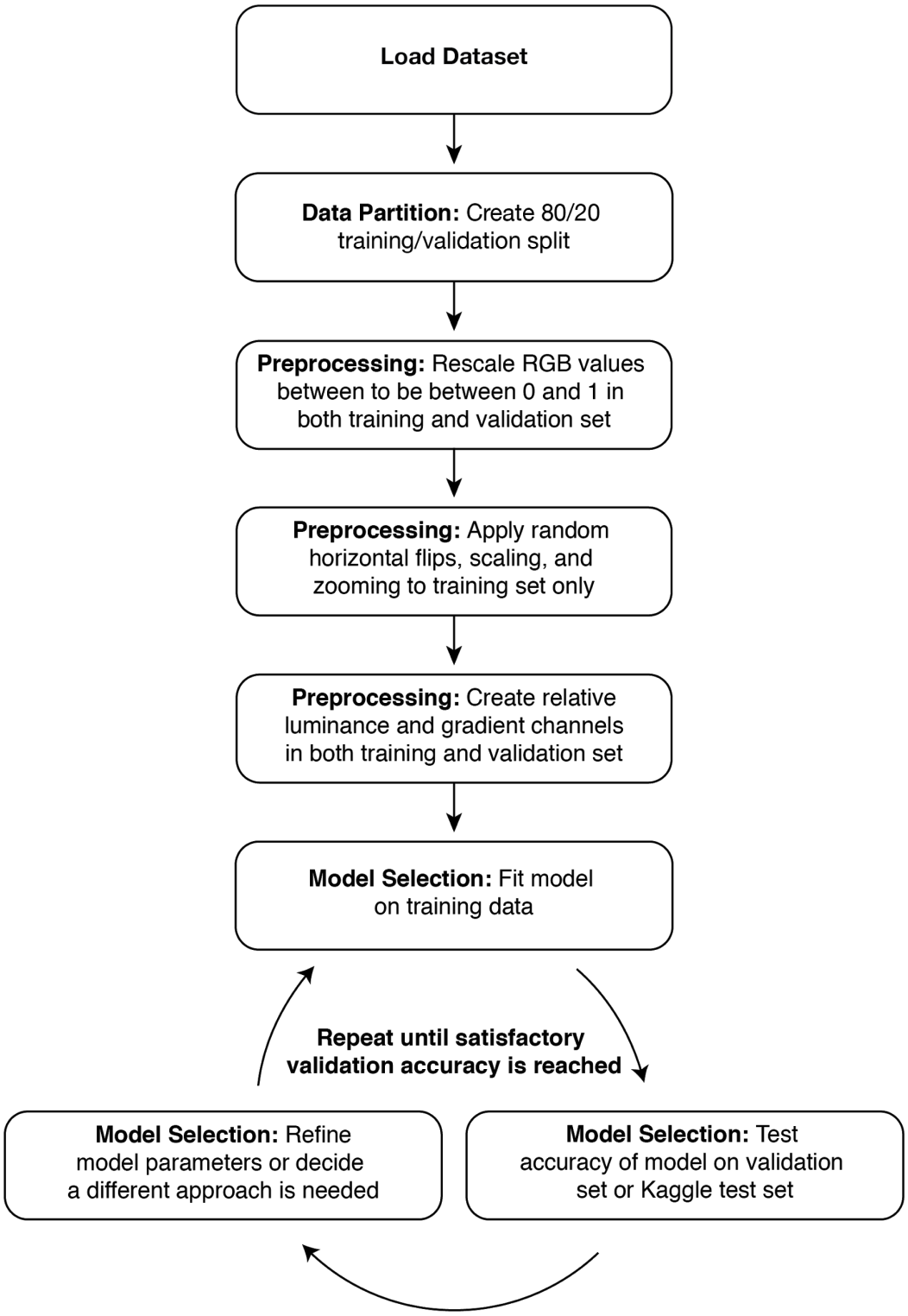

Fig 3. Flowchart of analysis. Each box represents a major processing step.

Data partition

In order to both create as well as evaluate the performance of our model, we separated our original dataset into a training and validation set. Out of the original 1,500 images, we split the dataset into 1,200 images for training and 300 for validation, i.e. the training set contains 80% of the images and the validation set contains 20%. We chose to allocate the majority of the the images in the training set because each of our approaches required a significant number of training images to result in accurate validation performance. When partitioning the data, we ensured a ratio of classes proportional to the original dataset in both our training and validation sets. Figure 4 shows the details of this partition. As mentioned previous, an additional 558 number of images and labels were housed on Kaggle.com. These images were used as a test set during our model selection process.

Data Partition	**Images Without Solar PV**	**Images With Solar PV**
Training Set	800	400
Validation Set	105	195
Kaggle Test Set	558 Unknown	

Fig 4. Number of classified images in each data partition. When partitioning the data, we ensured a ratio of classes proportional to the original dataset in both our training and validation sets.

Data Preprocessing

RGB Rescaling

When preprocessing our data, we prepared our images in three main steps. In the first step of preprocessing, we rescaled the RGB values of our images. As previously discussed, each pixel in our dataset is associated with a three number tuple of RGB values ranging from 0 to 255. We rescaled these values in both our training and our validation set to be in the range of 0 and 1 in order to avoid having high values compared to a typical learning rate (Chollet, 2016).

Random Image Modifications

During the second step of preprocessing, we applied a series of random modifications to our data in order to prevent overfitting of our model. Specifically, we applied horizontal flips, image shears ranging from 0.0 to 0.2 and a random zoom ranging from 0.0 to 0.2. The specific values of these transformations were selected after looking through code examples of CNNs and Keras blog (Chollet, 2016; Venkatesh, 2017; Lewinson 2018). These transformations and in general all the model, were created using the keras package for image preprocessing, models and layers (Chollet, 2016). The transformed versions of our images essentially replaced our original images therefore our training sample size remained the same. Theoretically, these transformations should not affect the predicted classification our model assigns to each image. By increasing the diversity of images in our training set we hoped to reduce overfit and ultimately to increase predictive accuracy outside our current dataset. Examples of these random transformations are shown in Figures 5.

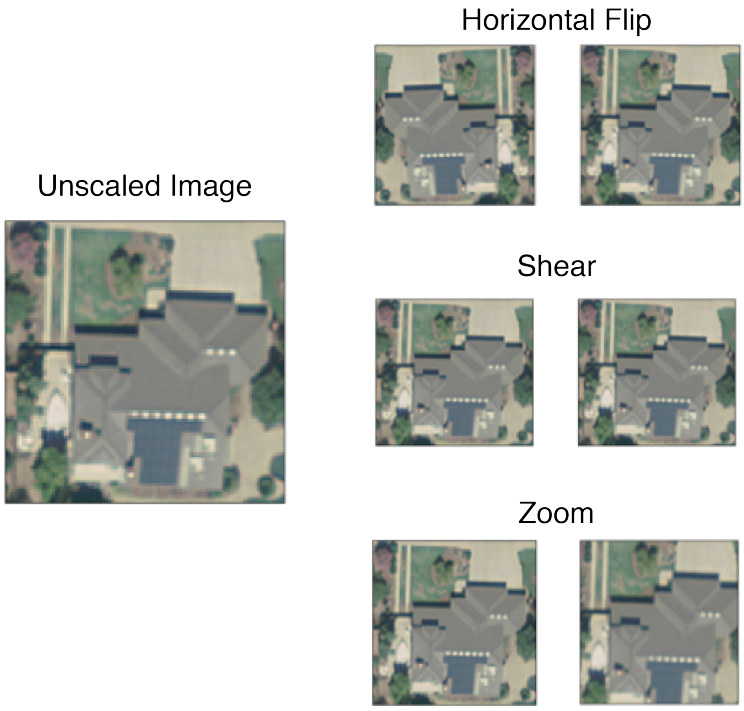

Fig 5. Example of image preprocessing. We applied a series of random modifications to our data in order to prevent overfitting of our model. We applied horizontal flips, image shears ranging from 0.0 to 0.2 and a random zoom ranging from 0.0 to 0.2. In some cases, it is difficult to see these effects with the naked eye. If we were to increase the value of these transformations

the differences would be more pronounced. The specific values of these transformations were selected after looking through code examples of CNNs and Keras blog (Chollet, 2016; Venkatesh, 2017; Lewinson 2018).

Relative Luminance and Gradient Channels

During the final step of preprocessing, we added lumiance and gradient channels to each of our 1,500 images in the hopes of creating a clearer delineation between our two image classes (Stokes et al., 1996). We based our decision to add these two channels on two qualities of solar PV. First, it is the case that solar PV generate electricity by absorbing sun rays. Because of this, we first hypothesized that images of solar PV may have relatively lower luminosity compared to other objects which may reflect more light. Secondly, the shapes of our aerial images of solar PV are typically relatively similar. This shape is usually rectangular with well-defined, sharp edges. Therefore we next hypothesized that we may be able to detect solar PV by detecting sharp gradients in our images. To address these two qualities of solar PV, we added both luminance and gradient channels respectively to our data in the hopes of generating features that would be more predictive of the presence of solar PV in our images.

Relative luminance is calculated pixel-wise from the RGB channels. The following formula was used to calculate the relative luminance from linear RGB components:

$$\boldsymbol{Y} = \square.\square\square\square\square\boldsymbol{R} + \square.\square\square\square\square\boldsymbol{G} + \square.\square\square\square\boldsymbol{B}$$

where *R*, *G* and *B* denote the linear value for red channel, green channel and blue channel, respectively. We simply added this additional channel value for each pixel in our dataset creating an array size $101 \times 101 \times 4 = 40804$ features for each image.

Gradient channel values are calculated such that grayscale images are computed using the magnitude of the derivative for each image (Jonsso, 2008) (For additional information about how to calculate gradient channels, please see Appendix 1). In line with our hypothesis that sharp gradients may help detect solar PV, we used $\sigma = 0.5$ in order to minimize the smoothing effect on our images. When computing the gradient channel, we calculated the gradient over one channel (i.e., red, green, blue, or relative luminance). The two most effective channels for computing gradient appeared to be the red and relative luminance channels as depicted in Figure 6. Similar to our relative luminance channel, we simply added the gradient channel value for

each pixel in our dataset creating an array size $101 \times 101 \times 5 = 51005$ features for each image. Sample results both relative luminance and gradient channels are shown in Figure 7.

During our analysis, relative luminance and gradient channels channels were utilized for several models. Most notably, use of these additional channels improved performance of logistic regression and random forest approaches. Detailed results are discussed in the Results section.

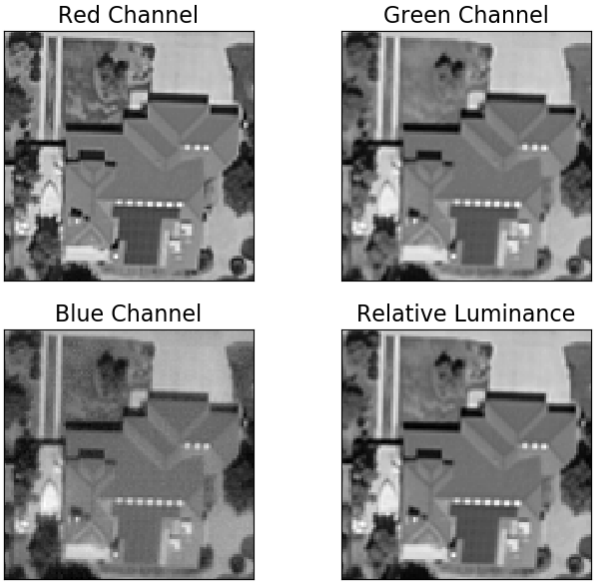

Fig 6. The effect of grayscale on RGB channels. The red channel appears to create a better distinction between solar PV and its surroundings (creating a darker solar PV and lighter rooftop). Relative luminance is able to achieve a similar contrast. Blue appears to perform the worst in this context generating a very similar shade of grey for both the solar PV and the rooftop.

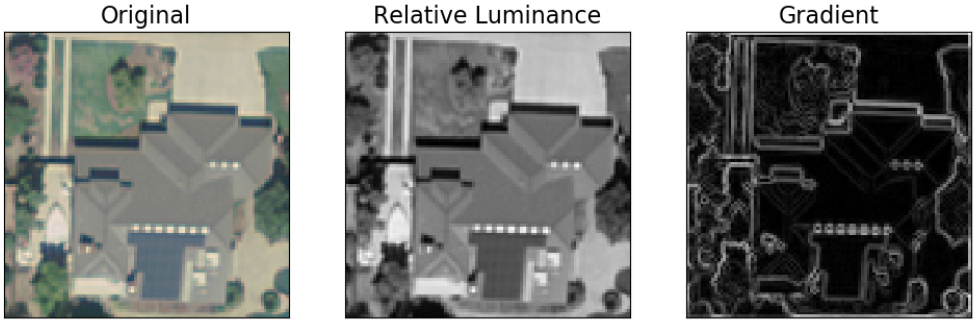

Fig 7. Example effect of relative luminance and gradient channels. We use relative luminance formula and normalized 2-dimensional truncated Gaussian function ($\sigma = 0.1$) to calculate the relative luminance and gradient, respectively. By added these features we hope to better delineate between images with and without solar PV by making the differences between classes more salient.

Metrics

During our model selection process, we attempted several modeling approaches before ultimately finalizing a CNN. These initial models included logistic regression, KNN, naive bayes, and random forest. We chose these models based off of previous research and the ability to use these models in order to predict a binary outcome (Khurshid et al., 2015; Ali et al., 2018, Malof et al., 2016).

As we iterated over our model selection phase of analysis were able to monitor our performance in two different ways: AUC/validation accuracy on our partitioned validation set, and test accuracy through submission to a seperate test set housed on Kaggle. Our accuracy on the test dataset housed on Kaggle was the metric that ultimately our final model. However, given the rules of the Kaggle competition, only three submissions to the Kaggle dataset were allocated per day. Due to this restriction, only our best performing models were applied to the test set housed on Kaggle.com .

For our logistic regression, KNN, naive bayes and random forest models we evaluated the performance using receiver operating characteristic (ROC) curves and Area Under the Curve (AUC). We used a baseline level of AUC of 0.5 to represent a model that performs by chance and interpreted higher AUC as improved model accuracy. When attempting a CNN approach, we primarily evaluated model performance using validation accuracy. This was primarily due to the fact that the Kaggle competition was evaluated based on validation accuracy. Additionally, we also noted the training accuracy and loss of our models, but these metrics were ultimately not used to determine our model performance. In general, an overreliance on metrics using only the training set can lead to overfitting.

In each modeling approach we attempted we changed several hyperparameters in order to improve our performance. Specifically for our final CNN, these hyperparameters included the

number of epochs, the number/type of layers (model complexity), and the number of training/validation steps.

Convolutional Neural Networks

For our final model we used convolutional neural networks. These are a specialized type of neural network for processing data that has a grid-like topology. CNNs are widely known in computer vision for being successful for tasks such as classifying images, clustering images, and object recognition (Karpathy, 2018). At they're core, CNNs use the convolution operation, instead of matrix multiplication in at least one of their layers. As other neural networks, they are conformed by a sequence of layers. The layers of CNN have neurons arranged in three dimensions: width, height and depth. There are different type of CNN architectures, but there are main type of layers (Goodfellow et al., 2016; Skymind, Accessed 2019) (see Figure 8 for a visual representation):

1. Convolutional Layer: This layer applies a convolution operation, the output is passed to the next layer.
2. Pool: This layer performs a down sampling operation, by combining the outputs of neurons at one layer into a single neuron in the next layer.
3. Dropout: This layer performs an analogous feature selection operation as random forest, by randomly turning off some proportion of all nodes.
4. Flatten: This layer flattens the pooled feature map into a column
5. Full-Connection: The layer will compute the class scores, each neuron in this layer will be connected to all the neurons in the previous one.

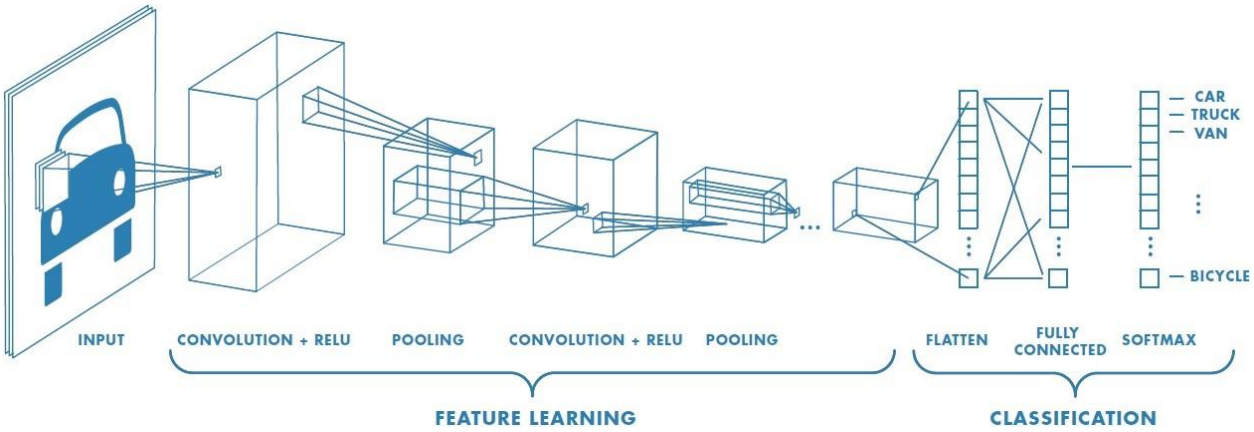

Fig 8. Neural network with many convolutional layers (Prabhu, 2018)**.** The convolutional layer applies a convolution operation, the output is passed to the next layer. The pool layer performs a down sampling operation by combining the outputs of neurons at one layer into a single neuron in the next layer. The flatten reshapes the feature map into a column. The full-connection layer will compute the class scores, each neuron in this layer will be connected to all the neurons in the previous one.

Results

As discussed in the previous section, we chose to separate our original dataset into a single training and validation set. The training set contained 80% of the original images and the validation set by 20%. We chose to have the majority of the images in the training set because each of our approaches required a significant number of training images to result in accurate performance. Additionally, we had access to a test set of images house on Kaggle.com that we used to evaluate our best performing models.

In terms of our logistic regression (both traditional and with PCA), KNN, naive bayes models, and random forest, we measured our validation performance using Area Under the Curve (AUC). The results for these models are shown in Figures 9-13. For our CNN approach, we measured our validation performance primary on validation accuracy. Because our CNN model showed such strong performance, we chose to submit CNN models to our test set housed on Kaggle.com and therefore can also report the Kaggle test set performance of our CNN model.[1] The results of our CNN model are shown in Figure 14.

Comparing each approach, we found that our traditional logistic regression only performed slightly above chance with an AUC of 0.597 (a model performing at chance would result in an AUC of 0.5). Resulting in a slightly better performance, our KNN classifier and our gaussian naive bayes models performed rather similarly to each other with AUCs of 0.639 and 0.643 respectively. With the addition of our generated relative luminance and gradient channels, logistic regression was able to achieve a 0.785 AUC. Random forest modeling performed even stronger with and AUC of 0.824. Finally, our CNN resulted in validation accuracy of 0.957. Although AUC and validation accuracy are not measured on the same scale, the results suggest our CNN model produces meaningful improvement in accuracy. See Figure 15 for a comparison of our key metrics across each approach.

Notably, our results suggest that the addition of relative luminance does not seem to add much predictive power to our models. However, the addition of the gradient channel does seem to increase our models' performance. These findings suggest that solar PV absorption of light

[1] According to the rules of the Kaggle competition, only three submissions were allocated each day for the entire team. Therefore, it was in our best interest to select only the most accurate models to apply to the test dataset on Kaggle.com. If this three-submission limit had not been in place, we would have applied each of our model approaches to our test dataset to compare performance.

does not have much effect on relative luminance, but that the sharp edges of the solar PV do, in fact, help to distinguish solar PV from their environment in aerial images.

Although logistic regression, KNN, gaussian naive bayes, and random forest can be successful in many context, convolutional neural networks appears to achieve the best model fit for this context. Instead of handling each pixel as features individually, CNN essentially looks at pixels in context. This type of approach is ideal when considering image recognition analysis because pixels only have meaning in relation to surrounding pixels. Given algorithm behind CNNs in without results, CNN or similar neural network approaches appear to be a feasible and strong choice for similar image detection analyses.

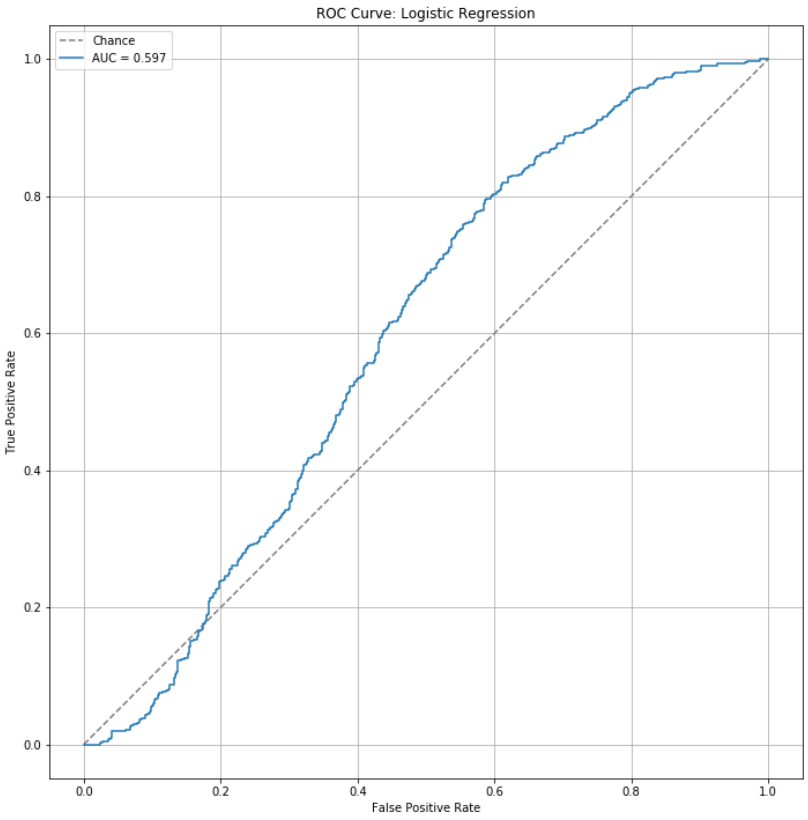

Fig 9. ROC curve for logistic regression. A model performing at chance would have an AUC of 0.5. Our logistic regression does not perform much better than chance with and AUC of 0.597.

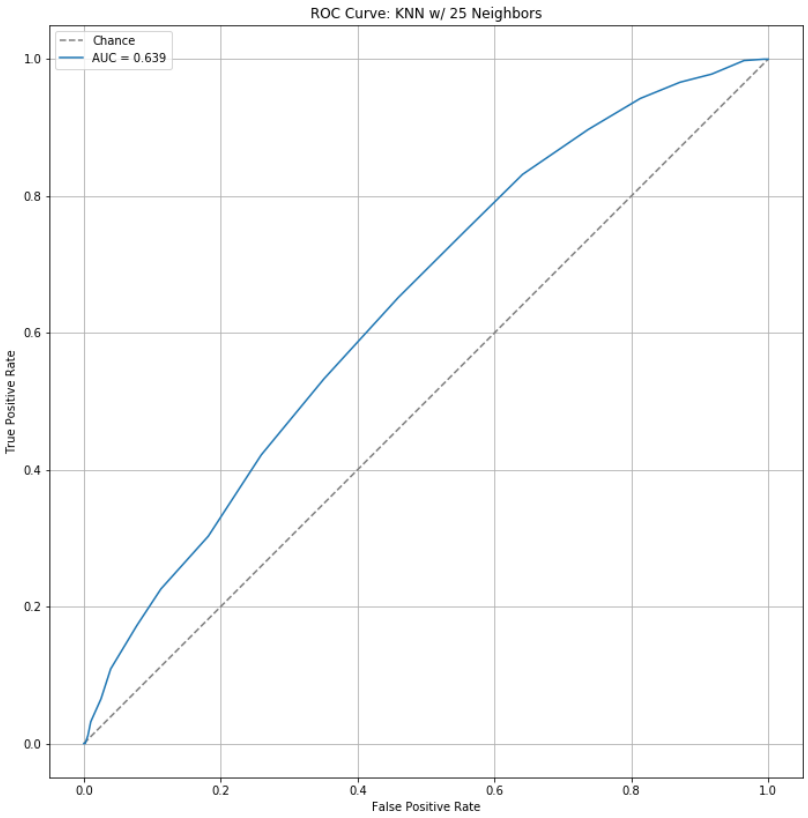

Fig 10. ROC curve for k-nearest neighbors model (K = 25). A model performing at chance would have an AUC of 0.5. Our KNN model performs better than our logistic model, but still only achieves an AUC of 0.639.

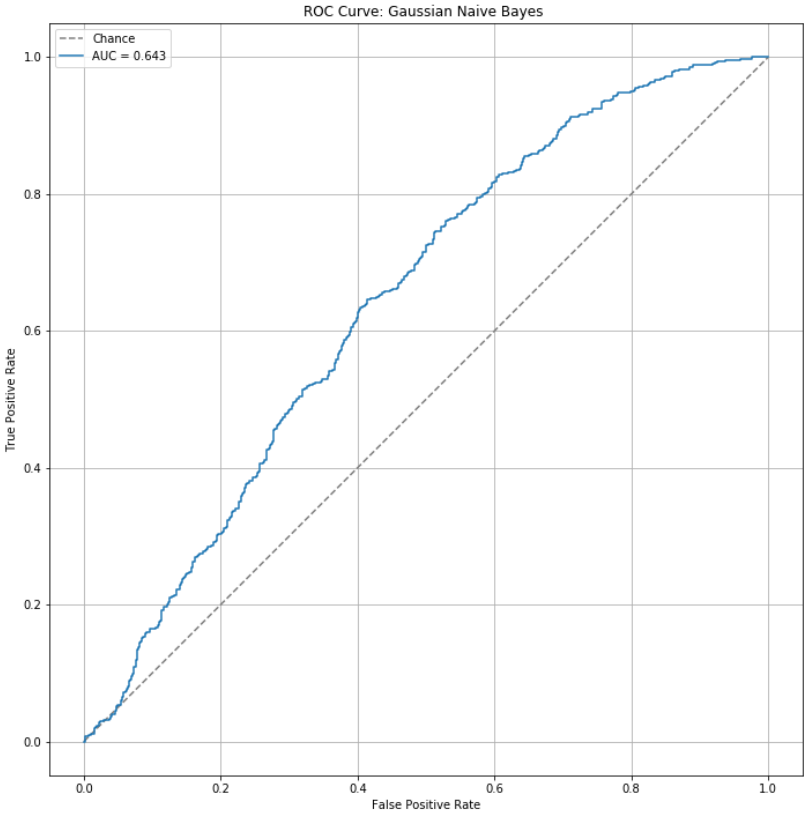

Fig 11. ROC curve for gaussian naive bayes model. A model performing at chance would have an AUC of 0.5. Our naive bayes model performs better than our logistic model, but performs close to our KNN model which only achieves an AUC of 0.643.

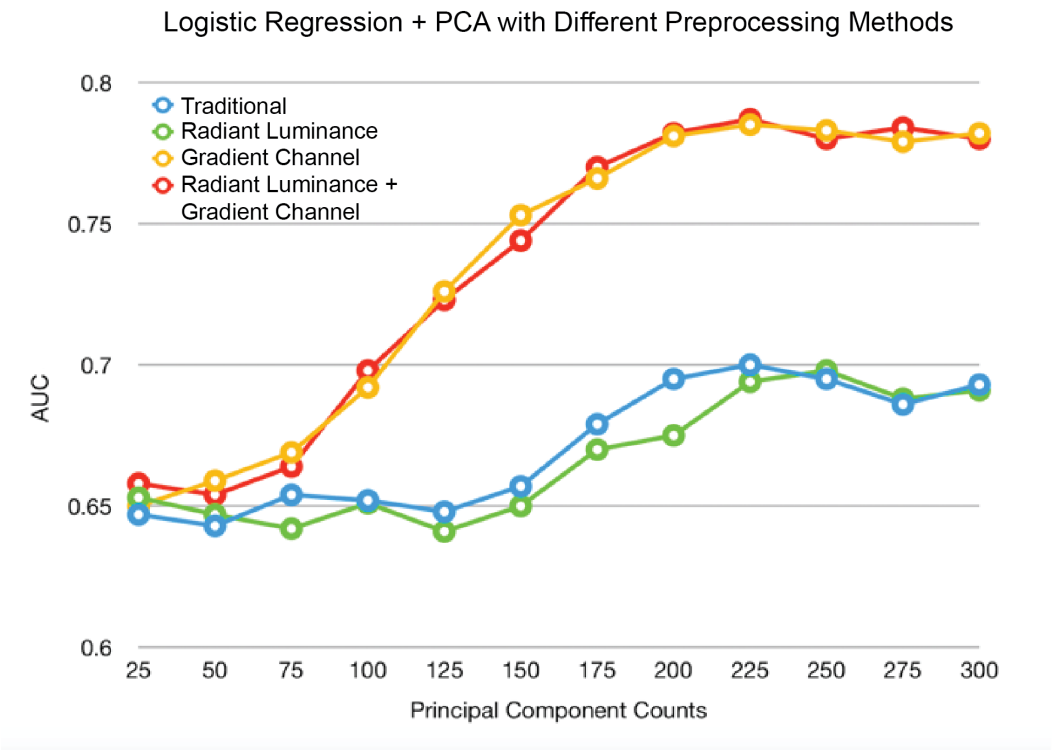

Fig 12. AUC for logistic regression + PCA with different preprocessing methods. Adding gradient channels increases model performances for every choice of PCA, while adding relative luminance has little improvement on the model. The optimal choice of PCA components is 225.

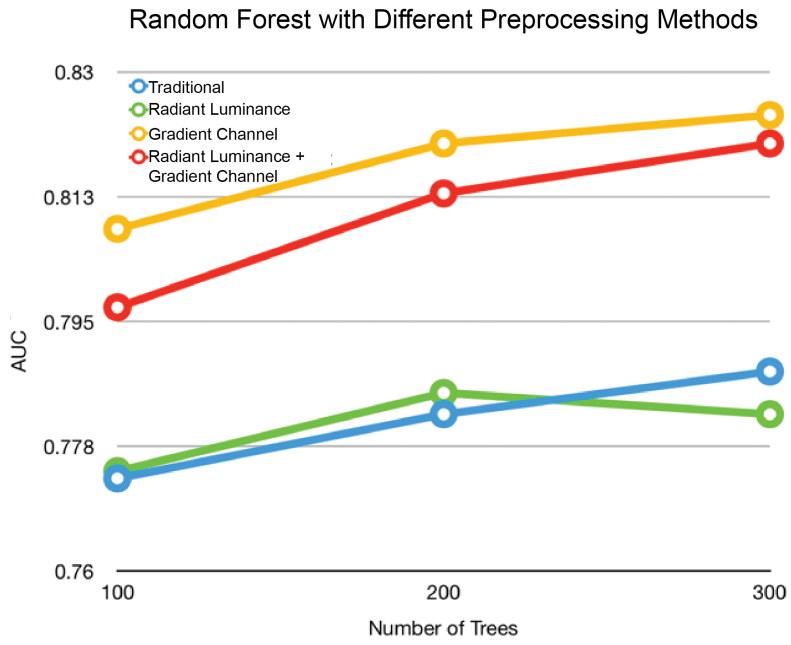

Fig 13. AUC for random forest models using different preprocessing methods. Adding gradient channels increased model performances for every choice of tree number, while adding relative luminance might sometimes hurt the performance. The optimal choice of tree number is 300.

Metric	Performance
Training Loss	0.0700
Training Accuracy	0.9746
Validation Loss	0.2258
Validation Accuracy	0.9567
Test (Kaggle) Accuracy	0.9854

Fig 14. Performance metrics for convolutional neural network. Our CNN model is not evaluated using AUC, but instead is primarily evaluated on validation accuracy. Our CNN model is highly accurate at .9567 validation accuracy and .9854 test accuracy. Due to the large discrepancy between the accuracies our model approaches, only CNN was applied to the test dataset housed on Kaggle.com.

Model	Metric	Validation Set Performance
Chance	AUC	0.500
Logistic Regression (Traditional)	AUC	0.597
KNN	AUC	0.639
Gaussian Naive Bayes	AUC	0.643
Logistic Regression[2] + PCA	AUC	0.785
Random Forest[3]	AUC	0.824
CNN	Validation Accuracy	0.957

Fig 15. Performance comparison across logistic regression (both tradition and with PCA), k-nearest neighbors, gaussian naive bayes, random forest, and convolutional neural networks. Our traditional logistic regression only performed slightly above chance with an AUC of 0.597. Our KNN classifier and our gaussian naive bayes models performed slightly better than our traditional logistic regression and rather similarly with AUCs of 0.639 and 0.643 respectively. By adding image gradient into feature space and PCA our logistic regression was able to achieve AUCs of 0.785. Our random forest model had an even better performance of 0.824 AUC respectively. Finally, our CNN resulted in validation accuracy of .957. Although AUC and validation accuracy are not measured on the same scale, our CNN showed a meaningful improvement in accuracy of detection.

[2] This logistic regression was run with 225 principal components and utilized the gradient channel.
[3] Our best performing random forest model used 300 trees and utilized the gradient channel.

Conclusion

As the solar industry continues to expand, it is becoming increasingly important to monitor solar PV consumer adoption. Traditional data sources such as consumer surveys, are costly, time-consuming, and can often only give a partial or biased view of the market. Satellite imagery allows us to acquire views of households all over the country. The utilization of aerial images to detect consumer solar PV may result in a more consistent and cost-effective view of solar adoption in the US.

Previous studies have explored the use of machine learning for satellite image classification. Specifically, machine learning techniques have shown to be successful at detecting solar PV using random forest and convolutional neural networks. In this analysis, we examined a dataset consisting of 1,500 satellite images each image contains a 101 pixels x 101 pixels size and has three color channels (Red, Green, Blue). Each image in our dataset was associated with a human-assigned class label - with solar PV or without solar PV. These labels allowed us to explore several supervised learning approaches.

Our findings align with previous research and suggest that approaches such as traditional logistic regression, k-nearest neighbors, and gaussian naive bayes may not be sufficient for identification with high rates of accuracy. That said, by adding additional features, such as image gradients, performance can be improved even when using simplest models. This may be a relevant if shorter training time is prefered. Although random forests in particular have been shown to be successful in this context (Malof et al., 2016), our findings suggest a CNN approach produces the highest accuracy of detection. By essentially evaluating pixels in context of surrounding pixels, CNNs are well-suited for image recognition and, in this analysis, resulted in a validation accuracy of 0.957.

In comparison to previous research on supervised learning and solar PV detection, this analysis was rather small. Our dataset contained only 1,500 images compared to others with almost double the sample size (Malof et al., 2016). By increasing our training data size, we would be able to improve our model and improve its generalizability to other satellite images. Also, additional approaches could yield increased accuracy, such as transferring learning from pre-built models.

For our analysis, we only focused on detecting whether a solar PV was present in an aerial image. However, future research should examine the use of supervised learning techniques to not only identify the existence or absence of solar PV, but also estimate the energy consumption associated with each installation. This could be achieved by integrating other disparate data sources, such as geographic data and energy usage patterns.

Roles

In general, all team members attended weekly meetings to discuss progress and future steps. In terms of specific roles:

- **Azucena** was heavily involved in modeling. Specifically, she was in charge of the first CNN and made several submissions to Kaggle. Azucena also contributed to writing the Kaggle report.
- **Viggy** was also heavily involved in modeling. Specifically he created and submitted the baseline models (KNN, Logistic, and Naive Bayes). He also contributed to the CNN modeling and submissions to Kaggle.com. Finally, Viggy also contributed to writing the Kaggle report.
- **Anna** was primarily responsible for writing and editing the Kaggle report. Anna also attempted a CNN modeling approach.
- **Allen (Yifei)** was also heavily involved in the CNN modeling effort and also contributed to the CNN submission to Kaggle.com as well as logistic and random forest models. He provided and tested different preprocessing methods on baseline classifiers. He also contributed to writing the preprocessing methods of the Kaggle report.
- **Sicong** was also heavily involved in the CNN modeling effort and also contributed to the CNN submission to Kaggle.com.

References

A Beginner's Guide to Convolutional Neural Networks (CNNs). Skymind Retrieved from https://skymind.ai/wiki/convolutional-network.

Ali, Hazrat, et al. "Supervised classification for object identification in urban areas using satellite imagery." *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018.

Bradbury, Kyle, et al. "Distributed solar photovoltaic array location and extent dataset for remote sensing object identification." *Scientific data* 3 (2016): 160106.

Chollet, F. Building powerful image classification models using very little data". The Keras Blog. (June 5, 2016). Retrieved from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.

Chollet et al. Keras. Retrieved from https://keras.io.

Energy Transition Outlook 2018. DNV GL. Retrieved from https://eto.dnvgl.com/2018/.

Golovko, Vladimir, et al. "Convolutional neural network based solar photovoltaic panel detection in satellite photos." *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1. IEEE, 2017.

Goodfellow, Ian, Bengio Yoshua, and Courville, Aaron, 2016. "Convolutional Networks". Deep Learning. MIT Press. Retrieved from http://www.deeplearningbook.org.

Janes, M., 2014. "Predictive models help determine which consumers buy solar equipment and why". *Phys.org*. Retieved from https://phys.org/news/2014-05-consumers-solar-equipment.html.

Jonsson, E. (2008). Channel-coded feature maps for computer vision and machine learning (Doctoral dissertation, Institutionen för systemteknik).

Karpathy, A. CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/convolutional-networks/.

Khurshid, Hasnat, and Muhammad Faisal Khan. "Segmentation and classification using logistic regression in remote sensing imagery." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.1 (2015): 224-232.

Kubat, Miroslav, Robert C. Holte, and Stan Matwin. "Machine learning for the detection of oil spills in satellite radar images." *Machine learning* 30.2-3 (1998): 195-215.

Lewinson, E.. Mario vs. Wario: Image Classification in Python. (24 Jul 2018). Retrieved from https://towardsdatascience.com/mario-vs-wario-image-classification-in-python-ae8d10ac6d63.

Malof, Jordan M., Leslie M. Collins, and Kyle Bradbury. "A deep convolutional neural network, with pre-training, for solar photovoltaic array detection in aerial imagery." 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, 2017.

Malof, Jordan M., et al. "Automatic detection of solar photovoltaic arrays in high resolution aerial imagery." *Applied energy* 183 (2016): 229-240.

Matasci, S. "Solar tax credit". EnergySage. (6 Jan. 2019). Retrieved from https://news.energysage.com/congress-extends-the-solar-tax-credit/.

Mnih, Volodymyr, and Geoffrey E. Hinton. "Learning to detect roads in high-resolution aerial images." *European Conference on Computer Vision*. Springer, Berlin, Heidelberg, 2010.

Prabhu, R.. "Understanding of Convolutional Neural Network (CNN)" (3 Mar. 2018).Retrieved from https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

Rai, Varun, D. Cale Reeves, and Robert Margolis. "Overcoming barriers and uncertainties in the adoption of residential solar PV." *Renewable Energy* 89 (2016): 498-505.

"Renewables on the Rise 2018 - Environment America." Retrieved from https://environmentamerica.org/sites/environment/files/reports/AME_Renewables-on-the-Rise_Jul18-Web.pdf.

Solangi, K. H., et al. "A review on global solar energy policy." *Renewable and sustainable energy reviews* 15.4 (2011): 2149-2163.

"Solar Industry Research Data". SEIA Retrieved from https://www.seia.org/solar-industry-research-data.

Stokes, M., Anderson, M., Chandrasekar, S., & Motta, R. (1996, November 5). A Standard Default Color Space for the Internet. Retrieved from https://www.w3.org/Graphics/Color/sRGB

Venkatesh, T. "Simple Image Classification using Convolutional Neural Network—Deep Learning in python". Retrieve from https://becominghuman.ai/building-an-image-

classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8.

Appendix

Appendix 1: Gradient Channel Calculation

Mathematically, we use normalized two-dimensional Gaussian function for interpolation to first reconstruct a continuous signal function $C(x, y)$from original image thus making the derivation possible. Next, we down sample the continuous derivative functions to make the gradient the same size of $101 \times 101$and calculated the magnitude. These steps are illustrated by equation (1) - (3).

$$C(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) G_x(x - j, y - i) \tag{1}$$

$$D_x(x, y) = \frac{\partial C}{\partial x}(x, y) = \frac{\partial}{\partial x} \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) G(x - j, y - i) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) G_x(x - j, y - i) \tag{2.1}$$

$$D_y(x, y) = \frac{\partial C}{\partial y}(x, y) = \frac{\partial}{\partial y} \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) G(x - j, y - i) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) G_y(x - j, y - i) \tag{2.2}$$

$$I_c(r, c) = D_x(x, y)|_{(c,r)} \qquad I_r(r, c) = D_y(x, y)|_{(c,r)} \tag{3}$$

In practice, we do not need an explicit expression of the continuous signal function $C(x, y)$ but only the down sampled derivatives $I_c(x, y)$ and $I_r(x, y)$. Therefore, we use a normalized two-dimensional truncated Gaussian function $G(x, y)$ to calculate the derivative along the x- and y-axis and then use their magnitude as the gradient value. The truncation point α were set to be 3.5σ. These steps are shown by equation (4) - (7)

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} = g(x)g(y) \tag{4}$$

$$G_x(x, y) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} = d(x)g(y) \qquad G_y(x, y) = -\frac{y}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} = g(x)d(y) \tag{5}$$

$$I_x(r, c) = \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I(i, j) G_x(c - j, r - i) \qquad I_y(r, c) = \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I(i, j) G_y(c - j, r - i) \quad \text{where } \alpha = 3.5\sigma \tag{6}$$

$$Gradient = \sqrt{I_x^2 + I_y^2} \tag{7}$$

Where coordinate system (r, c) denotes a discrete function and (x, y) denotes a continuous function (Jonsson, 2008).