Solar PV in Aerial Imagery

IDS 705 – Spring 2019

Team A – Emma Sun, Chang Shu, Iuliia Oblasova, Joe Littell

Abstract:

The advancement of Deep Machine Learning (ML) and Computer Vision (CV) have spurred numerous leaps forward in various other fields of science and study. From benefiting oncologist in identifying malignant tumors earlier, keeping them from becoming extremely dangerous to the patient's survival, to allowing human rights observers to identify and catalog criminal atrocities from satellite imagery from half way across the globe. Convolutional Neural Networks (cNN) have allowed such advancements to flourish through their feed forward mechanisms, however areas of complexity and challenge still exists. As such, this paper chronicles the use of Machine Learning and multiple Computer Vision techniques to identify various solar photovoltaic (PV) panels in configuration from images without these panels. These images are compiled from various residential and suburban settings with 101 by 101-pixel dimensions with RGB color channel. As such, this differs from classification problems which typically utilize extremely high resolution, multispectral imagery for their model building.

Introduction:

Climate Change is global concern that requires a global commitment and solution. Countries from Italy to Vanuatu to the United States are already facing the economic and political weight of rising tides and less fertile farmlands. [1] As these occurrences continue to increase, investment into clean and scalable energy solutions becomes extremely necessary. As such, being able to rapidly identify between locations with solar PV panels and those without allow for numerous beneficial applications, from being able to distinguish areas which have minimal carbon footprints, to ideal locations for expansion or green technologies, to identifying avenues of potential political and economic issues subverting the use of these technologies. [2]

Figure 1: The devastating effects of Climate change in Italy [3] and the United States [4]

This problem set becomes increasingly more significant as two factors change rapidly. One is the astoundingly negative effects climate change continues to add to our planet. Polar ice caps are rapidly melting, storms are increasing in intensity and frequency, and huge swaths of land are becoming infertile. The other is the numerous green technologies are seeing rapid price reduction as the manufacturing of these devices becomes more efficient. Of these technologies, Solar Photovoltaic (PV) panels are seeing the greatest reduction in cost, expected to surpass even wind by 2019. This fact is incredibly important as solar energy has the largest individual consumer outreach due to the size and efficiency of the devices allowing households to greatly reduce their carbon footprint without major installation cost. Where as other technologies, like wind or hydroelectric require vast amounts of land and infrastructure to harness, solar can be employed on pre-existing structures with minimal adaptation requirements.

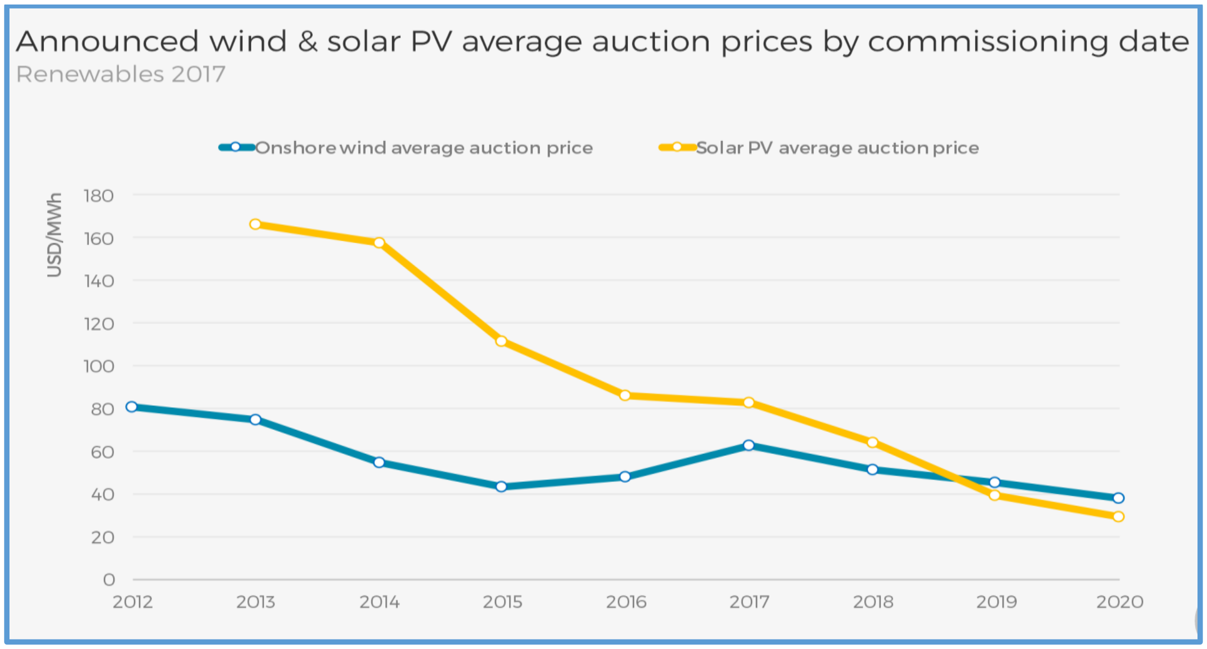
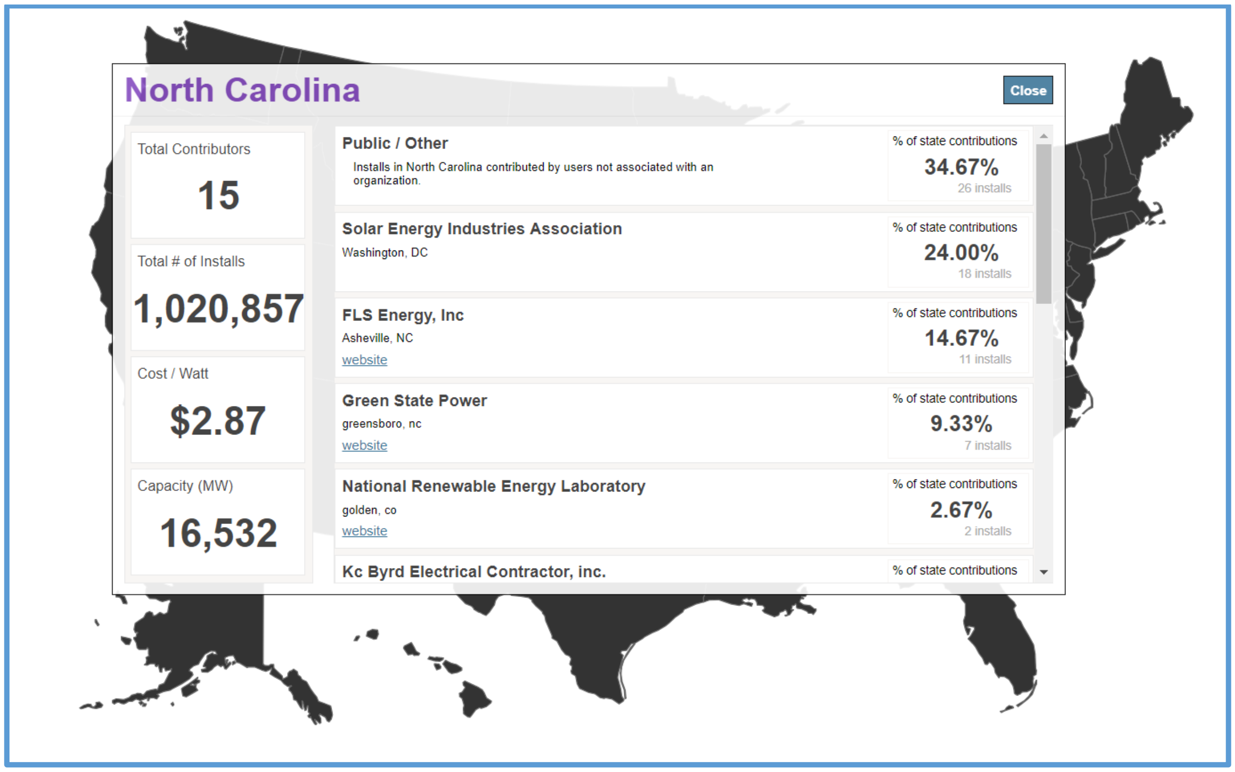


Figure 2: Projected cost per Mega-Watt Hour of Solar PV (orange) and Wind through 2020. [5]

Background:

Thus far, there have been numerous entities have attempted develop a method for identifying solar PV panels. On one end of the spectrum is the National Renewable Energy Laboratory (NREL) at the Lawrence Berkeley National Laboratory (LBL) which manages a wide-reaching array of traditional survey methods through the help of the United States Federal and State Governments, Panel producers, installers and customers as a part of the Open PV project to identify Solar PV market penetration. [6] In this ongoing study, outreach is required to identify and contact all producers, installers, and customers across the 50 states in order to determine final user penetration of the PV panels. While this information can be highly accurate, it requests large buy in from multiple entities as well as large overhead in order to bring to fruition.


North Carolina

Close

Total Contributors	15
Total # of Installs	1,020,857
Cost / Watt	$2.87
Capacity (MW)	16,532

Contributor	% of state contributions
Public / Other Installs in North Carolina contributed by users not associated with an organization.	34.67% 26 installs
Solar Energy Industries Association Washington, DC	24.00% 18 installs
FLS Energy, Inc Asheville, NC website	14.67% 11 installs
Green State Power greensboro, nc website	9.33% 7 installs
National Renewable Energy Laboratory golden, co website	2.67% 2 installs
Kc Byrd Electrical Contractor, inc.	

Figure 3: An example of contributors to the Open PV project utilizing a sample of North Carolina’s data.

On the other side of the spectrum is Sanford’s Deep Solar project, which utilized transfer learning from Google’s Inception V3 model. [7] Deep Solar’s model utilizes a ground truth [8] image segmentation model through the use of Class Activation Maps after classification has occurred to reduce the number of incorrectly classified images (in this case false positives) in order to develop a more precise classification of occurrences. This model, completed in November 2018, utilizes Google’s Static Map API with a 15-centimeter resolution to acquire training and test images, as well as Amazon Mechanical Turk to hand label the 366,467 images. This ultimately led to minimal overhead before attempting identify solar PV installation base across the 48 contagious US. [9]

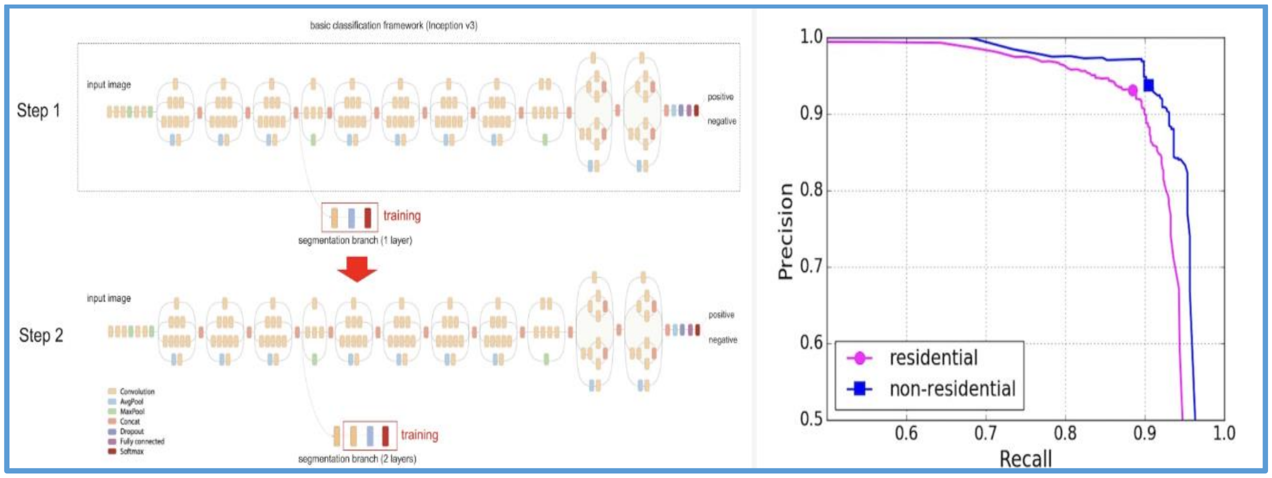


Figure 4: Deep Solar's Model for Solar PV Identification and Classification which utilizes transfer learning from Google's Inception V3 Architecture.

Due to scaling and cost restrictions upon our identification process, a cutdown version of the Stanford study fit our needs the best. As such, we utilized a simple cNN with color channel adjustments to better identify features that will allow for increased classification of panels.

Data:

Contrary to the Deep Solar project, resources are not available to conduct largescale use of Mechanical Turk for labeling additional images. Therefore, we solely utilized the 1500 training images as well as the 1500 test images that were given for our model. These images are all overhead satellite views of suburban and residential locations. Each image is 101 pixels in width and 101 pixels in height. They are all color images utilizing the RBG (Red-Blue-Green) color channels.

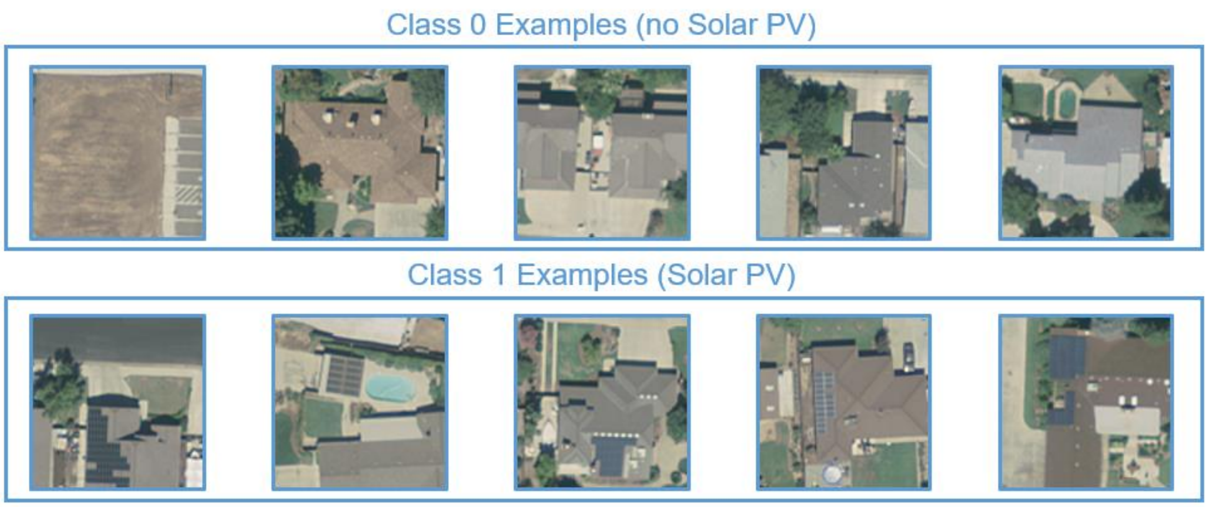


Figure 5: 10 examples of Classification 0 and Classification 1 images from the training set; Images which containing a Solar PV panel fall within Classification 1

As observed in Figure 5, The variation between images can be minute. Outside of homes not containing a residence (similar to the image in to top left of Figure 5), only the solar PV panel stands out. The variance in size and color of the panels (bottom row of Figure 5.) can also vary highly. In order for our model to work well, we have to determine a way in order to highlight to the model where the difference is. Given that we have 30,603 features per image (101 pixels x 101pixels x 3 color channels), we need to simplify what the model observes to determine a classification.

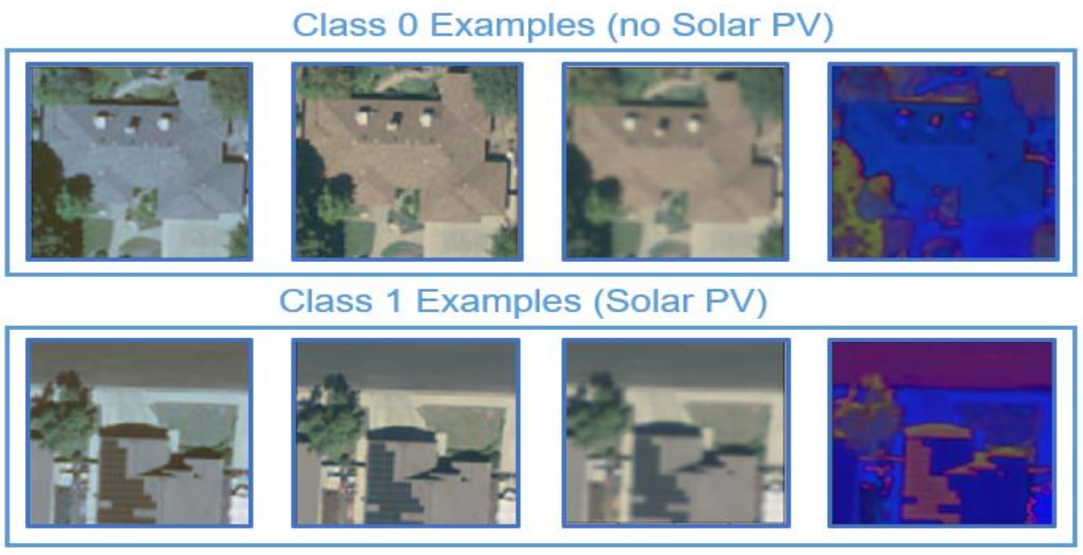


Figure 6: Utilizing color channel conversion and Gaussian Blur to extra features.

As seen in Figure 6, as the image is read from its directory (far left), it will be converted from BRG to RGB color channel. A gaussian blur will be applied to deduce the sharpness of the image. From the blurred image the color channels will be converted from RGB to HSV. This limits the number of colors available, thus highlighting the solar PV. Finally, the image will be divided by 255 to make all pixel values ranged between 0 and 1, as opposed to 0 and 255. All of this is accomplished with the OpenCV [10] package within python.

A potential pitfall in this methodology may occur with large, roof mounted air conditioning units. Their shape and coloring are similar to Solar PV panels, which may incur a false positive classification. This can be slightly seen in Figure 6, where the small air conditioning units in the Class 0 example ends with a similar color to the solar panel in the Class 1 example, albeit substantially smaller in size.

Methods:

Initially we utilized different models to observe increases in accuracy given the problem set. We utilized K Nearest Neighbor (kNN) [11] and Logistic Regression [12] with and without to the previously mentioned conversions to identify how accurate methodologies could produce given the problem set. Which both had a greater accuracy than random chance, however they capped out at 71% accuracy for the regression.

After additional research into the subject, it was determined that a Convolutional Neural Network (cNN) could produce the best results. [13] We chose cNN because it works with a forward propagation to determine unique features as the model examines each individual image to determine the characteristics that differ between our labeled testing images. As the name suggest, the model works similar to the human brain by utilizing neurons to identify the features. Our model utilizes the Tensor Flow environment through the Keras frontend and works as follows;

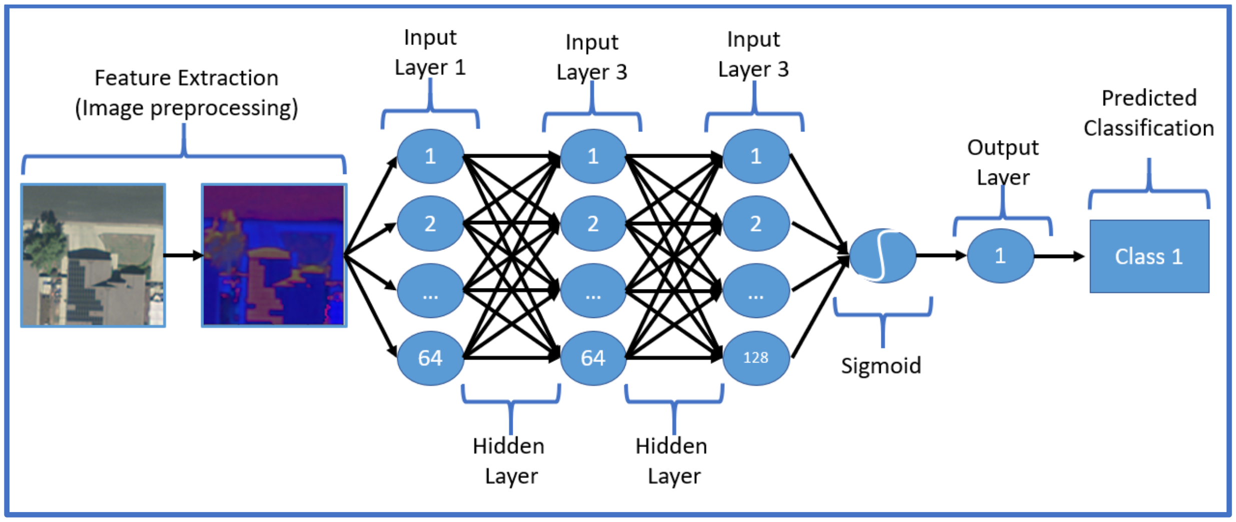


Figure 7: Visualization of 3-Layer cNN for Image Classification.

We created a sequential network that utilized three 2D convolution layers [14], the first of which utilized 64 nodes or neurons, a rectifier linear unit *('ReLU')* [15] or ramp function for activation, and a 2 by 2 max pooling [16]. Each layer identifies a main feature of the images, for example one layer may determine if a house is present, the next if there is anything on the roof, and then the last if that object is a solar panel. The max pooling layer allows the data to be 'down sampled' or simplified in order to generalize the feature so that its dimensionality is reduced, and assumptions are allowed to be made.The second layer is identical to the first and the third layer contains 128 nodes in order to determine smaller features and details once it has made its way through the larger layers. These operate as explained above in the first layer.

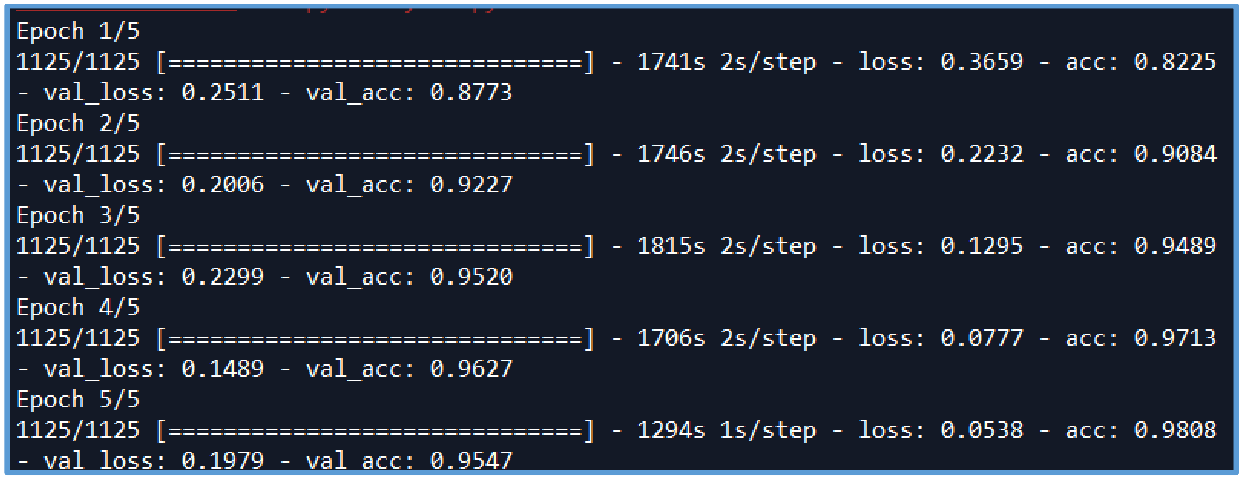
Finally, a connective layer, or dense layer [17] is added to connect the above layers. This layer contains dropout to prevent overfitting, or the model being too exact to the training images. The model is compiled with a *sigmoid* [18] activation function, similar to logistic regression, since there are only two possible classifications. The optimizer utilized is '*adam*' [19] as it allows for the learning rate, how much we determine a feature to matter when creating weights, to be adjusted per epoch, and does so automatically. The loss function is *binary cross* *entropy* [20] since we are utilizing a sigmoid function, and there are only two classifications. Finally, we utilize *'accuracy'* [21] for the metric to evaluate our predictions, resulting in a number between 0 and 1, corresponding to a percentage if multiplied by 100.

From the classifier model, we utilized a 75/25 split for our training set so that we can more accurately gage our model without biasing it from the test set. The sklearn *train_test_split* [22] package added this functionality.

The final set to the model is utilizing Keras' *DataImageGenerator* [23], which is built in to the *keras.preprocessing.image* package. This package allows for an artificial increase in data by rotating, flipping, shearing, shifting and zooming the training images to produce new images with accurate labeling without having to require additional resources.

Results:

Overall, despite the lack of transfer learning and a relatively lightweight architecture, our model performed closely to the other models within the competition. Utilizing the above method with only 5 epochs, we were able to reach a high of 96.27% accuracy on our validation steps over the 4th epoch.



```
Epoch 1/5
1125/1125 [==============================] - 1741s 2s/step - loss: 0.3659 - acc: 0.8225
- val_loss: 0.2511 - val_acc: 0.8773
Epoch 2/5
1125/1125 [==============================] - 1746s 2s/step - loss: 0.2232 - acc: 0.9084
- val_loss: 0.2006 - val_acc: 0.9227
Epoch 3/5
1125/1125 [==============================] - 1815s 2s/step - loss: 0.1295 - acc: 0.9489
- val_loss: 0.2299 - val_acc: 0.9520
Epoch 4/5
1125/1125 [==============================] - 1706s 2s/step - loss: 0.0777 - acc: 0.9713
- val_loss: 0.1489 - val_acc: 0.9627
Epoch 5/5
1125/1125 [==============================] - 1294s 1s/step - loss: 0.0538 - acc: 0.9808
- val_loss: 0.1979 - val_acc: 0.9547
```

Figure 8: Training the cNN for classification of Solar PV Panels. The model utilizes 3 convolution layers, 5 epochs, and a 27/75 split between training and validation sets.

When in comparison to a simple Logistic Regression model, our cNN preformed substantially better as seen in Figures 9 and 10. Our area under the curve is a 95.5 % accurate, a 35.9-point increase compared to the Logistic Model at 59.6%. This is slightly miss leading however, as the Keras model uses fully feature extracted images when the Logistic Regression model does now. With out the color conversions, the cNN preforms in the 77.5% accuracy range, a 17.9-point increase.

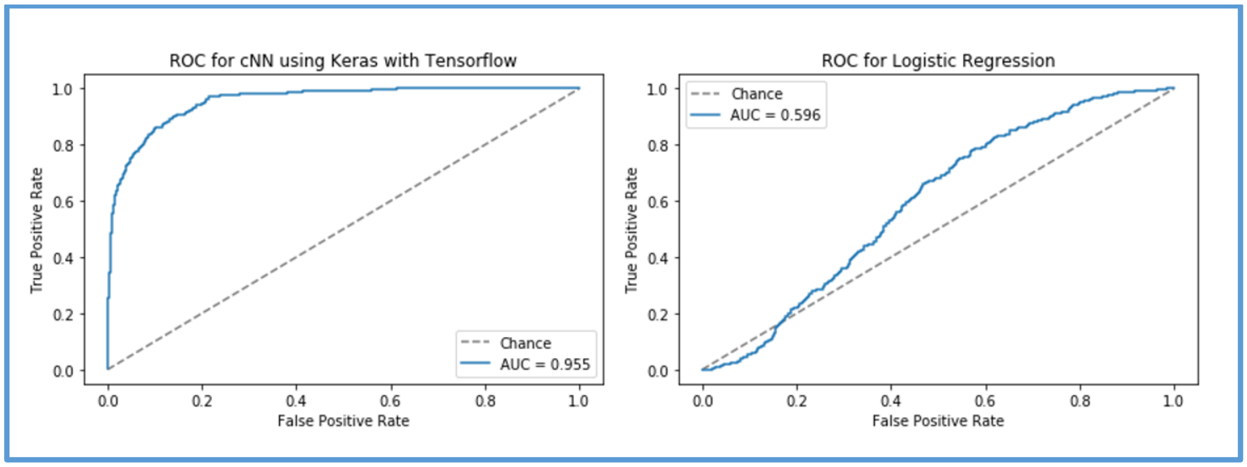


Figure 9: ROC comparison for cNN (left) Logistic Regression Model with base images

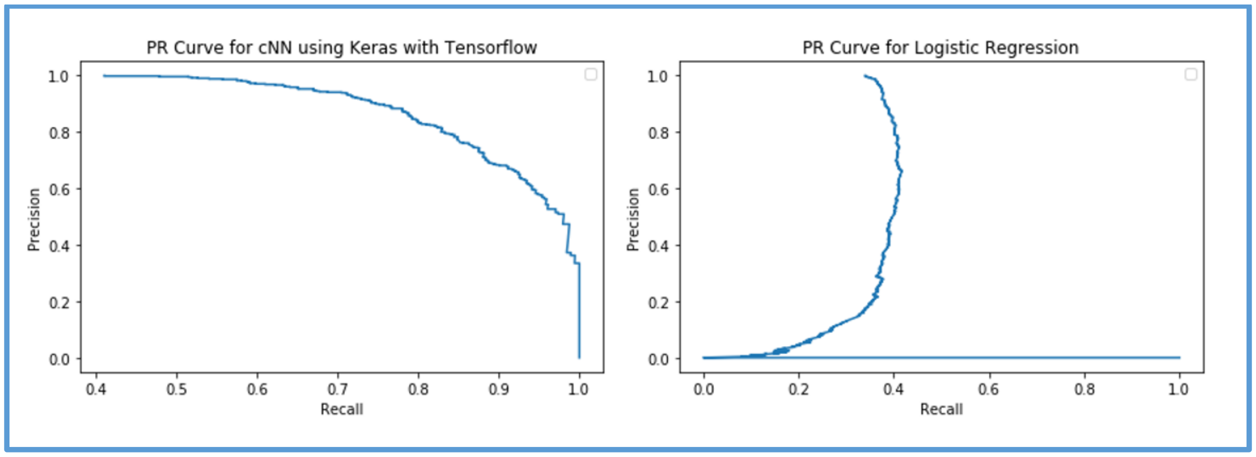


Figure 10: PR Curve comparison for cNN Model (left) and Logistic Regression Model (right)

While the above visualizations is utilizing the 75/25 test and validation split, the cNN model performed no less than 98.294% accuracy on unseen data. This is in comparison to the above logistic mode which performed at no less than 58.197% accuracy, leading us to believe that the cNN is a strong generalized model which will perform well in similar data.

Our model failed in precisely the same manner where we had previously assumed. This is where there are large man-made structures that are rectangular and similar color to the panels. Even if we used an additional step of image manipulation for segmentation, similar to the Deep Solar model, we could see similar results. This leads us to believe that our model may suffer in accuracy in more urbanized location where such structures are more prevalent. Additional images for training that clearly shows a number of images where this might occur could potentially train the model to better distinguish these features and elements of the landscape, however without more testing we can not be certain.

Conclusions:

Although Convolutional Neural Networks require more computational power than more traditional Machine Learning techniques and methodologies, they produce substantially more accurate results in image classification. This, coupled with the fact that we utilized a limited training set of only 1500 low resolution images, produced an accuracy range greater than 98% on all test data. These accuracy metrics occured without the utilization of transfer learning. If utilized, transfer learning more than likely would have increased the model's accuracy even more, given than cNN work better the more data they have to process during the learning phase. Given more images, of higher resolution, we would assume that our model would see increased accuracy in evaluating if solar PV panels are present or not.

As stated in our introduction, the ability to give policy makers accurate data to which they will utilize to make intelligent and informed decisions on climate change on behalf of their constituents is of utmost important. Our model is extremely lightweight compared to other transfer learning models, requiring less computational power. Despite this we see a less than 1% drop off in accuracy on testing data. This gives policy makers a fast and efficient method to identify key locations in order to influence their decisions. Only through actively understanding the problems, through facts and not assumptions, will we be truly able to curb the degradation of our planet.

Roles:

Joe Littell: Overall acted as the project lead. Was in charge of scheduling and timelines. Developed the on aspects of the cNN model itself as well as the final report.

Emma Sun: Developed our initial theory and model for Logistic and kNN models for testing prior to movement to the cNN. Helped with aspects of testing of the code to understand how to translate the initial code functionality to that of the cNN.

Iuliia Oblasova: Developed many of the feature extraction for the training images as well as researched the DataImageGenerator for use to increase model accuracy.

Chang Shu: Worked closely with Emma on research and initial model development. Worked on research for cNN and feature extraction experimentation with Iuliia.

References:

[1] Glennon, Robert. "The Unfolding Tragedy of Climate Change in Bangladesh." Scientific American Blog Network. April 21, 2017. Accessed March 04, 2019. https://blogs.scientificamerican.com/guest-blog/the-unfolding-tragedy-of-climate-change-in-bangladesh/

[2] "Exploring Recurrent and Feedback CNNs for Multi-Spectral Satellite Image Classification." NeuroImage. October 23, 2018. Accessed March 04, 2019. https://www.sciencedirect.com/science/article/pii/S1877050918319987

[3] "World Heritage Sites at Risk." Union of Concerned Scientists. Accessed March 04, 2019. https://www.ucsusa.org/global-warming/science-and-impacts/impacts/world-heritage-tourism-sites-climate-change-risks

[4] McLaughlin, Kelly. "It Was a Thriving Town of 27,000, and Now It's Smoldering Ash; Here's How Paradise, California, Became a Fire Trap." INSIDER. November 13, 2018. Accessed March 04, 2019. https://www.thisisinsider.com/paradise-california-was-a-fire-trap-2018-11

[5] "Announced Wind and Solar Average Auction Prices." October. October 04, 2017. Accessed March 04, 2019. https://www.iea.org/newsroom/energysnapshots/announced-wind-and-solar-average-auction-prices.html

[6] NREL Data Analysis and Visualization Group. The Open PV Project. Accessed March 04, 2019. https://openpv.nrel.gov/about

[7] Jiafan. "The DeepSolar Project." HOPES Huntington's Disease Information. Accessed March 04, 2019. http://web.stanford.edu/group/deepsolar/home

[8] "Image Segmentation." Reconstructing an Image from Projection Data - MATLAB & Simulink Example. Accessed March 04, 2019. https://www.mathworks.com/discovery/image-segmentation.html

[9] Deep Solar Supplemental Information https://www.cell.com/cms/10.1016/j.joule.2018.11.021/attachment/b91312aa-483a-44ad-9d75-60f0762cf709/mmc1.pdf

[10] "Getting Started with Images." Image Denoising - OpenCV-Python Tutorials 1 Documentation. Accessed March 04, 2019. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html

[11] "Sklearn.neighbors.KNeighborsClassifier." 1.4. Support Vector Machines - Scikit-learn 0.19.2 Documentation. Accessed March 04, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[12] "Sklearn.linear_model.LogisticRegression." 1.4. Support Vector Machines - Scikit-learn 0.19.2 Documentation. Accessed March 04, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[13] Agarwal, Yash. "Create Your First Image Recognition Classifier Using CNN, Keras and Tensorflow Backend." Medium.com. July 08, 2018. Accessed March 04, 2019.

https://medium.com/nybles/create-your-first-image-recognition-classifier-using-cnn-keras-and-tensorflow-backend-6eaab98d14dd

[14] “Keras 2D Convolution Layer Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/layers/convolutional/

[15] “Keras Activation Documentation (ReLU).” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/activations/

[16] “Keras Max Pooling Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/layers/pooling/

[17] “Keras Core Layers Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/layers/core/

[18] “Keras Activation Documentation (Sigmoid).” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/activations/

[19] “Keras Optimizer Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/optimizers/

[20] “Keras Loss Function Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/losses/

[21] “Keras Metric Documentation.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019. https://keras.io/metrics/

[22] "Sklearn.model_selection.train_test_split¶." 1.4. Support Vector Machines - Scikit-learn 0.19.2 Documentation. Accessed March 04, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[23] “Keras Image Preprocessing Documentaton.” Keras - 2.2.4 / 3 October 2018. Accessed March 04, 2019.https://keras.io/preprocessing/image/