Localization of US Wind Turbines Using Satellite Imagery

Jabs (Mohammad) Aljubran
Stanford University
AI for Climate Change Bootcamp
aljubrmj@stanford.edu

Abstract

State-of-the-art wind turbine are based on voluntary surveys and self-reports. This approach is expensive, time-consuming, and prone to human error. Hence, automated high-fidelity wind turbine localization using computer vision and satellite imagery is proposed as a sustainable solution. A pipeline model is developed and evaluated to accomplish this task using two main modules: detection and localization. Detection is casted as a binary classification task, and it is performed by training a DenseNet121 network, achieving accuracy, AUROC, precision, and recall of 0.995, 0.998, 0.981, and 1.000, respectively, on the test set. Grad-CAM is used to generate heatmaps to further understand the model behavior and facilitate counting and localizing wind turbines. Semi- and weakly- supervised models are developed to use the Grad-CAM heatmaps and output count and location of wind turbines. The weakly-supervised model is found to be superior in performance with 0.907 in counting recall. It is observed that counting false negative instances are due to the fact that wind turbines sometimes appear incomplete as they occur at the edge of the tile, and that Grad-CAM heatmap signal is not sufficiently indicative in some cases.

1. Introduction

There is an increasingly high penetration of renewable energy resources (solar panels, wind turbines, etc.) in the world. More than 50% of the global power generation is projected to be renewable by 2035 [6]. In September 10th, 2018, the California governor Jerry Brown signed Senate Bill No. 100 on advancing clean energy and climate protection. The legislation states that 100% of the state's retail electricity supply is to fully come from eligible renewable energy resources with zero-carbon emissions by December 31st, 2045.

Yet, the increase in renewable energy investments puts immense pressure on the electricity grid since renewables are sparsely distributed, intermittent, and unstable. Unfortunately, there is no integrated documentation that specifies renewable energy installation locations, real-time power output, and proximity to the electricity grid. Efficiently constructing a high-fidelity, complete, and accurate database for these resources is crucial for downstream tasks, including monitoring, coordination, and power system controls.

Most of the state-of-the-art approaches are based on voluntary surveys and self-reports, which are quite inefficient and hard to update. Many efforts are focused on documenting solar and wind power resources as they account for the largest share of the renewable energy growth worldwide. For example, The Open PV Project used crowdsourcing to construct an incomplete solar panel installation database in the US without accurate GPS information [16]. Furthermore, US Geological Survey (USGS) used satellite images to verify the wind turbine installations using manual annotation [5]. These manual approaches are expensive and time-consuming as many tasks require updated maps.

However, with the advent of high spatial resolution satellite imagery and recent breakthroughs in deep learning and computer vision, we can automate these tasks and provide real-time data on energy at scale (in both time and space!). Hence, this project will focus on developing convolutional neural network (CNN) models that aim to localize wind turbines across the US using satellite imagery. Input, turbine/non-turbine satellite images, is fed into a pipeline model (CNN binary classification detection module, followed by weakly supervised object counting and localization module) to predict the count and geographical location of wind turbines.

2. Related Work

Automatic object detection using aerial imagery has been used for various localization tasks: buildings [9][18], vehicles [1][13], poverty [10][11], and others. Recent renewable energy research efforts applied deep learning on satellite imagery to localize photovoltaic solar panels at the regional level. Approaches ranged from machine learning to deep learning algorithms, and from pixel-wise [14][15], which suffer from computational inefficiency and low precision/recall, to image-wise techniques [20], which cannot provide installation sizes accurately.

Researchers at Stanford University recently introduced DeepSolar, a deep learning framework that localizes and characterizes the size of solar panels across the US [19]. They utilized an image-level binary classification of solar panels using pretrained Inception-v3 model, then applied semi-supervised segmentation to determine the spatial dimensions and potential output of solar panels. This novel

semi-supervised segmentation approach resulted in significant improvements in computational efficiency and high accuracy while using label-free training for solar panel size estimation. It also eliminates the need for massive pixel-wise, labelled training data which is required by traditional techniques.

Up to the author's knowledge, there is only one published work on wind turbine localization using satellite imagery. This refers to a pixel-wise approach by researchers in China localizing wind turbines in Shanxi Province and Shandong Province [7]. They manually drew polygons around wind turbines, generated binary masks, trained U-Net model (training set involved only 100 wind turbines in total), and classified wind turbines accordingly. Post-processing involved semi-supervised generation of bounding boxes. Interpretation shows that the trained U-Net model identifies wind turbines mostly by capturing their shadow rather than body only. Despite achieving a high F_1score of 0.97, this pixel-based method is hardly scalable to a regional level and requires strong supervision and manual efforts in shaping polygons and generating masks.

This project introduces an image-level classification and weakly-supervised localization pipeline model. It will train on the US wind turbine database and efficiently scale up to scan the entire nation tiles in a computationally affordable framework. Up to the author's knowledge, this project is novel and represents the first large-scale effort on mapping the US wind turbines.

3. Dataset and Preprocessing

Data is acquired from two main sources: The US Wind Turbine Database (USWTDB), and satellite imagery (i.e. Bing Static Map API). Note that initial evaluation was done using Google Static Map API, yet Bing offered more affordable and scalable subscription packages with the same level of resolution.

3.1. Data Description

USWTDB is an open-source database constructed based on voluntary surveys and self-reports. It was recently updated in April 2019, reporting a total of 59,338 wind turbines. It highlights onshore and offshore wind turbines across the US with project information (name, year, etc.), turbine technical attributes (capacity, height, diameter, etc.), and geographical details (longitude, latitude, etc.). **Figs. 1 and 2** show the distribution of wind turbines on the county and state levels, respectively. Note that Texas and California are the most populated states with 13,820 and 8,165 of wind turbines in total, respectively. Kern County claims the lion's share with just over 55% of the total California wind turbine installations.

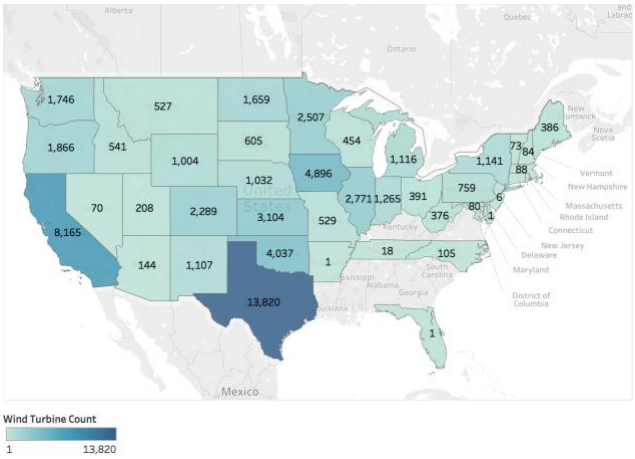

Fig. 1—Wind Turbine Distribution Per State: Population of US wind turbines is illustrated per state. California is ranked second after Texas with a total of 8,165 wind turbines. Note that Alaska, Hawaii, and Guam are not shown to spare space. Tableau Desktop 2019.1 is used to generate this plot.

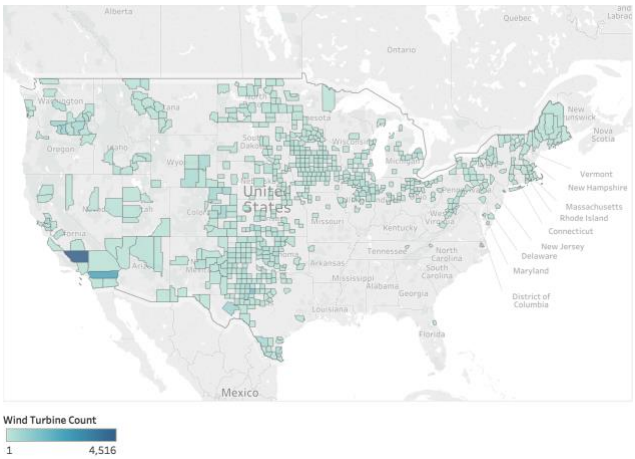

Fig. 2—Wind Turbine Distribution Per County: Population of US wind turbines is illustrated per county. Kern County hosts the largest population with a total of 4,516 wind turbines. Note that Alaska, Hawaii, and Guam are not shown to spare space. Tableau Desktop 2019.1 is used to generate this plot.

3.2. Data Generation

As seen in **Fig. 3**, Bing Static Map API is used to generate images of each wind turbine (positive data points) based on the coordinates provided by USWTDB. Satellite images with no wind turbines (negative data points) are also generated randomly from within the US using a geographical box with latitude and longitude ranges of (49.4, 24.5) and (-66.93, -124.784), respectively.

Fig. 3—Wind Turbine Satellite Image Examples: Two images that show positive wind turbine examples. Note the particularly distinguishable wind turbine structure and shadow.

Note that reducing zoom by one level is equivalent to reducing the required tile dataset size by a factor of four, which is dramatically critical for scalability to the national level. Building a model based on satellite images of low zoom level is desired as it allows for easily scaling up to national level with minimal number of tiles required. Meanwhile, higher zoom levels increase the chances of developing a high-performance model where wind turbines appear closer and clearer. This tradeoff is analyzed by iteratively training and evaluating model performance at zoom level ranges of 16-20. It was found that increasing zoom level beyond 17 shows no significant improvement. Hence, all images are generated with zoom and size dimensions of 17 and 1500x1500, respectively.

A minimum distance in kilometers is maintained between each negative example and all the positive counterparts, seen in **Eq. 1**. The haversine approximation is used to compute the distance between each two examples, and a negative image is extracted only if it meets the minimum distance criteria; otherwise, another random sample is selected.

$$\min\_distance = \frac{156.5430\ x\ \cos(latitude)}{2^{zoom}}\ x\ size \quad (\boldsymbol{Eq.1})$$

Randomly generating non-wind turbine images is efficient, yet it might oversimply the classification process as these random samples may be simple plains and terrains. The model must be challenged and trained over more difficult images with objects that actually may look like wind turbines to the model, i.e. aircrafts, boats, bridges, local roads, removed wind turbines, etc. These difficult negative-label categories, seen in **Fig. 4**, are chosen based on error analysis of the preliminary model results. Descartes Labs GeoVisual Search [4] is used to efficiently extract a total of 1,575 coordinates of the aforementioned objects, which are then fed to Bing Static Map API to generate difficult negative images.

Fig. 4—Difficult Negative Images: Examples of difficult cases (bridges, eliminated wind turbines, local lanes, and aircrafts) to ensure that the model clearly differentiates the wind turbine features from other objects despite the similarities. Note that these categories are not chosen randomly, but based on error analysis and inspection of preliminary models.

3.3. Data Preprocessing

The USWTDB assigns location and attribute confidence levels (1 is least confidence and 3 is most confidence) to each wind turbine, which reflect the reliability of the reported data point. Hence, only wind turbines with USWTDB location and attribute confidence levels of 3 are kept to decrease the likelihood of mislabeled data. Consequently, the overall positive wind turbine data points dropped to 48,271. The same number of negative images is used, including difficult negative images, for training to ensure a balanced dataset distribution.

Data is split into train:validation:test ratios of 98:1:1 for model development and evaluation. Note that if the data is randomly split, then wind turbines of single county, which normally share similar characteristics, will end up in different sets. This could falsely lead to optimistic results during evaluation. Hence, the splitting process is performed such that no two wind turbines of the same county are located in different sets.

Image preprocessing further involved normalization of pixels using ImageNet dataset mean and standard deviation of (0.485, 0.456, 0.406) and (0.229, 0.224, 0.225), respectively. Images are further center-cropped down to size 800x800, followed by a random crop down to size 500x500 to generated off-centered and spatially variable wind turbine training examples. This is finally fed to the deep learning model for training.

4. Methodology

Although end-to-end models are preferred for object localization tasks, they require strong supervision, labelling, and creation of bounding boxes around wind turbines. This approach is constraining as it demands significant manual effort, which deems it infeasible for upscaling to the national level. Hence, a pipeline model is proposed with two main modules: detection and localization.

4.1. Detection Module

The detection module performs binary classification using a CNN model to determine whether or not a tile contains wind turbine/s, then generates a heatmap for each tile indicating which pixels were the most active in making the prediction. This work uses Pytorch framework[1].

Transfer learning framework is used to develop the binary classifier. This approach allows for leveraging knowledge (features, weights, etc.) of other image recognition tasks, which potentially improves model performance and training speed. Since the current dataset is sufficiently large, no layers were frozen during training.

ImageNet-pretrained DenseNet121 model [3][8] is selected to perform binary classification. This architecture connects each layer to every other layer in a feed-forward fashion to form dense blocks, seen in **Fig. 5**. Each layer receives the preceding feature maps as input while passing its own feature maps to all subsequent layers.

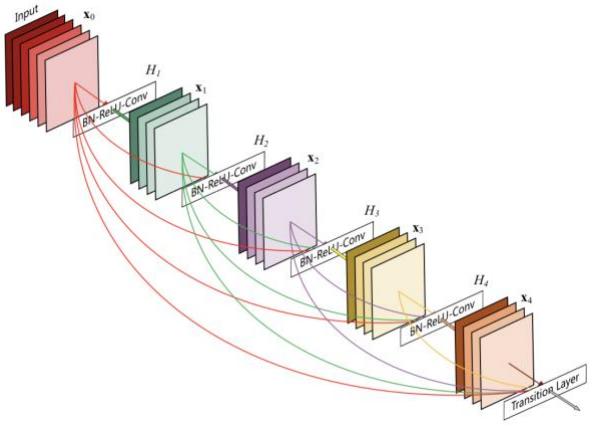

Fig. 5—DenseNet121 Dense Block: Illustration of a 5-layer dense block with growth rate of 4 [8].

Transition convolutional and max pooling layers are used to connect these dense blocks and manipulate feature map sizes. Each convolutional layer is designed to perform batch normalization, rectified linear unit, and convolution (BN-ReLU-Conv) operations in order, denoted by $H(.)$. For a DenseNet architecture with L layers, there are $\frac{L(L+1)}{2}$ direct connections compared to L direct connections traditionally.

Let the output of the l^{th} layer be x_l, **Eq. 2** shows how layers are connected in dense blocks:

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}]) \qquad (\boldsymbol{Eq.2})$$

This architecture is advantageous as it alleviates the vanishing-gradient problem, accelerate optimization, and significantly reduce the number of parameters (Total of 7,978,856 parameters which is relatively low when compared to other state-of-the-art CNN models). Binary cross-entropy loss function is used for training. For an example image i, loss L_i is defined using binary label $y_i \in \{1,0\}$ and predicted probability $\hat{y}_i \in [0,1]$, seen in **Eq. 3**. Since both dataset classes (negative and positive) are quantitatively balanced, and realizing that both false positive and false negative scenarios are equally undesired, so we consider F_1 score and the area (*AUROC*) under the receiver operating characteristic curve (*ROC*). As seen in **Eq. 4-6**, F_1 score is defined with precision (P) and recall (R).

$$L_i = y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \qquad (\boldsymbol{Eq.3})$$

$$P = \frac{true\ positive}{true\ positive + false\ positive} \qquad (\boldsymbol{Eq.4})$$

$$R = \frac{true\ positive}{true\ positive + false\ negative} \qquad (\boldsymbol{Eq.5})$$

$$F_1 = \frac{2PR}{P + R} \qquad (\boldsymbol{Eq.6})$$

Gradient-weighted Class Activation Mapping (Grad-CAM) is then applied to generate a coarse localization heatmap highlighting the important regions in the image [17]. While the original Class Activation Mapping (CAM) setup [21] requires the substitution of fully-connected layers with a global average pooling layer after the convolutional blocks, Grad-CAM is advantageous as it is applicable to a significantly broader range of architectures, as seen in **Fig. 6**.

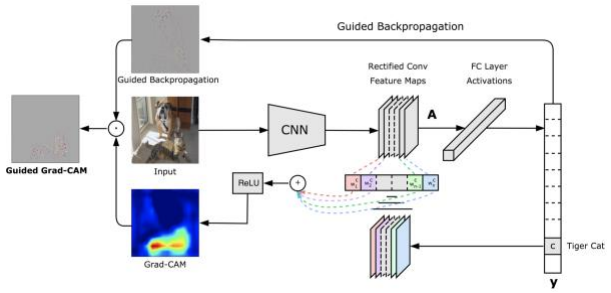

Fig. 6—Grad-CAM Workflow: Image is forward-propagated through the CNN to obtain raw scores. The gradient of the desired class is set to 1 while all others are strictly set to zero. The signal is then back-propagated to rectified convolutional feature maps which generate the coarse localization [17].

[1] Note that read access to the private Github repo is already granted to cs231n-staff at https://github.com/stanfordmlgroup/aicc-spr19-wind.

4.2. Localization Module

The second module of the pipeline receives the generated Grad-CAM heatmaps and performs localization, which aims to count the wind turbine in each tile and determine their respective geographical coordinates (latitude and longitude). Two techniques are examined for this task: semi-supervised and weakly-supervised.

The Semi-supervised approach uses a thresholding technique to segment the heatmap, and thus localize wind turbines[2]. Each heatmap is first smoothed using a gaussian filter with standard deviation of 0.1. A mask is created for each image with $threshold = \tau\ x$ max$_pixesl_intensity$, where $\tau \in [0,1]$ is a hyperparameter that is tuned to an optimal value of 0.5 on the validation set after manually drawing bounding boxes. Binary morphological opening, a mathematical technique [2], is then applied to denoise the heatmap and eliminate sparsely scattered high-intensity pixels. After performing these operations, groups of connected pixels will remain representing probable wind turbine locations. Hence, a rectangular bounding box is created around each group, which concludes the semi-supervised wind turbine localization.

Note that thresholding is optimized by tuning the hyperparameter τ using manually created bounding boxes. Labelbox.com is used to draw bounding boxes around wind turbines in 800 tiles (400 from validation and 400 form test sets), seen in **Fig. 7**. The validation bounding boxes are used as the ground truth for tuning.

Fig. 7—Bounding Boxes: Examples of the manually generated bounding boxes using the 1500x1500 satellite images. Labelbox.com is used to complete this task.

Since this is a single-class localization task, a slightly modified formulation of mean average precision (mAP) is used as a hyperparameter tuning objective. Intersection over union (IoU), seen in **Eq. 7**, is first calculated to determine the overlap between the predicted and corresponding ground truth bounding boxes. It is then binarized iteratively using threshold values from 0.5 to 0.95 with a step size of 0.05. Precision is calculated for each IoU threshold. Average precision (AP) is then computed as the mean of the individual precision values. Finally, mAP is computed as the average of AP over all examples.

$$IoU = \frac{area\ of\ intersection}{area\ of\ union} \qquad (\boldsymbol{Eq.7})$$

Alternatively, a weakly supervised approach is proposed for binary classification problems. Without any preprocessing, each Grad-CAM heatmap is fed into a local maximum detector to detect pixel-wise location and count of peak points, which translates into wind turbines given a single-class localization task. This technique is straightforward, computationally efficient, and requires minor hyperparameter tuning (minimum distance in pixels between each two peaks, which is approximated as 75 given the zoom level to yield high localization performance). Note that this technique can only be used for single-class localization tasks, which makes it suitable for this application. This approach can be evaluated based on the distance of each predicted peak point from the ground truth bounding boxes.

However, it will be seen that both the semi- and weakly-supervised algorithms can fairly well predict the location of the wind turbine object only if they can spot it to start with. In addition, the application of documenting wind turbines can tolerate minor deviation for object centers, but do not tolerate overestimating or underestimating the count of wind turbines. Hence, counting recall per wind turbine object, Eq. 5, is used as the primary metric to compare these two approaches.

5. Results, Experiments, and Discussion

This section will highlight and discuss the model training and evaluation results for both pipeline model modules: detection and localization.

5.1. Detection Task:

Preliminary detection model training is performed using pretrained DenseNet121 with learning and optimization hyperparameters seen in **Table 1**. Literature shows that the selection of large batch size will likely result in significant degradation in the model quality and ability to generalize as the optimization process ends up converging to sharp minimizers on the cost function hyperplane [12]. Hence, batch size of 16 is chosen to train the detection model. Generally, the choice of hyperparameters is interdependent, so iterating over ranges of values is required to optimize the model performance. The model is trained with learning rate of 0.0001 and 5 epochs. Note that using a pretrained model is advantageous where the transfer-learned features represent a desirable initialization on the cost function hyperplane. Otherwise, the model would have to fully rely on the given dataset to learn these features, which might

[2] Note that several SciPy, scikit-learn and scikit-image implementations are used for this purpose, and can be found in the project Github repo.

slow the learning process, and potentially lower the model performance.

Table 1—Training Hyperparameters			
Batch size	16	Epochs	5
Learning Rate	0.0001	Loss	Log loss
Optimizer	Adam	Beta_Momentum	0.9
Beta_Dampening	0.9	Weight_Decay	0

Adam is the optimizer of choice while AUROC is used as evaluation metric on the validation set to save a checkpoint of the best performing model parameters during the training process which acts as means of regularization (similar to using early stopping, but without interrupting the learning process). Note that wind turbines are generally similar, indicating that the training, validation, and testing sets come from a low-variance distribution. Hence, it is expected that no strong forms of regularization are needed to achieve this binary classification task.

Using a 0.5 threshold to binarize probabilities and compute accuracy, precision, recall**, Table 2** shows the preliminary model results on the validation set using the preprocessed raw data. Despite the competent performance, further improvement is highly desired to ensure that the resultant CAM heatmaps are sufficiently accurate for the localization task. When compared to recall, precision is relatively low which indicates a relatively high false positive rate. Therefore, manual inspection of the true positive and false negative dataset is performed to ensure that the USWTBD data is actually correct. To facilitate this process, only true positive images with less than 0.98 probability are inspected (the false negative set was relatively small, so it was fully inspected). It was found that 6,102 data points (12.62% of the level-3 confidence USWTDB!) were mislabeled and did not have wind turbines, seen in **Fig. 7**. These were blacklisted and the model performance significantly improved as a result.

Table 2—Evaluation Metrics on Validation Set		
Metric/Model	Raw Data	Clean Data
Accuracy	0.978	0.997
AUROC	0.999	1.000
Precision	0.961	0.991
Recall	0.995	1.000

Fig. 8—Mislabeled Images: Examples of mislabeled images with removed wind turbine traces (left) and plain terrains (right).

Threshold was optimized using F_1 score to yield an optimal threshold of 0.988. This DenseNet121 model is then evaluated on the test set resulting in accuracy, AUROC, precision, and recall, of 0.995, 0.998, 0.981, and 1.000, respectively. **Table 3** shows the confusion matrix as evaluated on the test set. Even though intentionally challenged with difficult negatives in the test set, this model still performs well. Hence, it is deemed satisfactory for the binary classification task.

Table 3—Test Set Confusion Matrix			
		Actual	
		1	0
Predicted	1	366	7
	0	0	1078

Applying Grad-CAM to the wind turbine detection model results in heatmaps, seen in **Fig. 9**, highlighting discriminative regions of the image which contributed most to the prediction. Observing these heatmaps, we notice how the model considers both the wind turbine main structure as well as its shadow as indicators of positive examples. In addition, some tiles can include parts of a wind turbine (shadow, trunk, blade, etc.) as they fall on the edge which is important to consider when scanning an entire state or nation with this algorithm to avoid double counting wind turbines. The former issue can be resolved using a merging algorithm based on a minimum distance heuristic as long as the wind turbines in each tiles are accurately be localized.

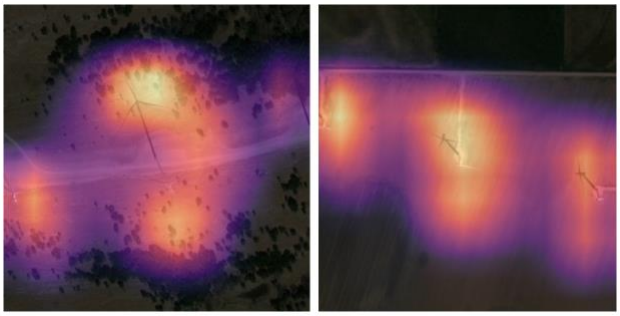

Fig 9—Grad-CAM Heatmaps: Examples of heatmaps generated using the Grad-CAM algorithm. These heatmaps localize the pixels which the model found to be most contributing to the prediction.

5.2. Localization Task:

Semi-supervised thresholding technique and weakly-supervised local maxima technique receive the Grad-CAM-generated heatmaps to draw boxes around wind turbines. **Fig. 10** shows the performance of these two algorithms in counting and localizing wind turbines after tuning the corresponding hyperparameters.

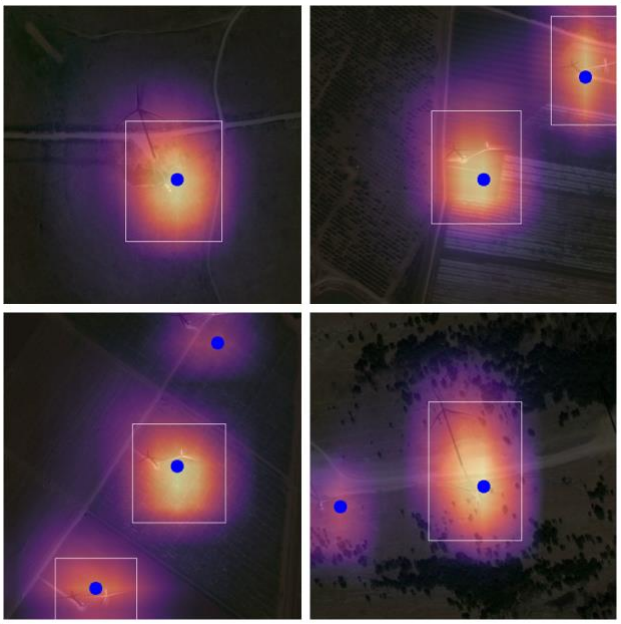

Fig. 10—Semi- and Weakly- Supervised Algorithms: Wind turbine counting and localization using semi-supervised (white boxes) and weakly-supervised (blue dots) algorithms. We observe that the semi-supervised approach captures all wind turbines when the Grad-CAM signal is strong and distinct (top heatmaps), but fails to do so when the signal is strong across some wind turbines but globally weak across others (bottom heatmaps).

Note that the semi-supervised algorithm (white boxes) struggles to capture all wind turbines when the heatmap intensity at a wind turbine is relatively weak. This is because it assigns a threshold value before masking out the heatmap and deciding where the signal is strong. This threshold is dependent on the maximum pixel intensity in each image. Meanwhile, the weakly supervised algorithm uses the local intensity to decide whether or not a given region in the heatmap contains a wind turbine. Hence, it is not affected by the maximum pixel intensity which makes it superior. The provided examples give a sense of these cases where the weakly-supervised algorithm succeeds while the semi-supervised algorithm fails to count wind turbines.

Overall evaluation shows that when they find a wind tur bine, both the semi- and weakly- supervised algorithms actually perform well on the aforementioned localization metrics, such as mAP and peak distance from ground truth bounding boxes. With good localization capabilities, the most critical and differentiating metric here is rather counting recall as this is a single-class task. Evaluation of average wind turbine counting recall performance for the semi- and weakly- supervised algorithms yields 0.814 and 0.907, respectively. Hence, the weakly supervised approach is superior.

Inspecting counting false negatives where the model missed wind turbines, it is found that these are due to two factors: 1) insufficient Grad-CAM performance, and 2) wind turbines falling on the edge of the tiles. **Fig. 11** shows an example where these two factors result in missing the shadow of a wind turbine (right-hand side of the image). Note that the impact of these misses is less significant when considering that fact that these partially showing wind turbines are more completely showing in another tile by default where the model will likely capture them there. Finally, this will have to be accounted for when merging the tiles for the final wind turbine count to avoid double counting.

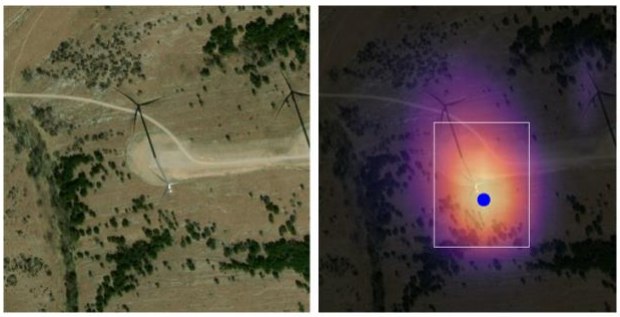

Fig. 11—Weakly-Supervised Algorithm Miss Example: This illustrates where the weakly-supervised algorithm struggles to capture the wind turbines. Note that the quality of the heatmap (low intensity at the right-hand wind turbine shadow) and the fact that the wind turbine shadow is at the edge both caused the weakly-supervised algorithm to miss counting both wind turbines in this tile.

6. Conclusion and Future Work

This novel project introduced a deep learning workflow for high-fidelity wind turbine localization using satellite imagery. A pipeline model is developed and evaluated to accomplish this task using two modules: detection and localization. Detection is casted as a binary classification task, and it is performed by training a DenseNet121 network, achieving accuracy, AUROC, precision, and recall of 0.995, 0.998, 0.981, and 1.000, respectively, on the test set. Grad-CAM is used to generate heatmaps to further understand the model behavior and facilitate counting and localizing the wind turbines. Semi- and weakly- supervised models are developed to use the Grad-CAM heatmaps for counting and localizing wind turbines. The weakly-supervised model is found to be superior in performance with 0.907 in counting recall. It is observed that counting false negative instances are due to the fact that wind turbines sometimes appear incomplete as they occur at the edge of the tile, and that Grad-CAM heatmaps are not sufficiently indicative in some cases. Hence, this pipeline model will benefit from an improved heatmap generation model.

The current work is focused on applying this algorithm on the entire US where tiles are being acquired and stored as of now. After running the model and generating the count and location of the wind turbines, a merging algorithm

(based on minimum distance heuristic) will be developed to avoid double counting wind turbines which happen to be on the edge of adjacent tiles. Next, a regression model will be developed to detect the height, diameter, and capacity characteristics of each detected wind turbine. After overlaying the global wind weather forecast, this will enable real-time energy output of wind resources across the US. This project will be expanded globally afterwards. Ultimately, an atlas of all energy installations and transmission lines is to be developed to allow for optimal energy allocation and investment.

7. Contributions and Acknowledgement

I would like to acknowledge my teammates at the AI for Climate Change (AICC) bootcamp which is run by Prof. Andrew Ng. Namely, I would like to thank the wind team members: Sharon Zhou, Jeremy Irvin, Will Deaderick, and Eva Zhang. The bootcamp provided computational power which was critical to completing this data-intensive project. Also, the wind project teammates were very supportive as they contributed in acquiring satellite images and developing the algorithms. I also would like to acknowledge USGS and Bing for publicly providing wind turbine database and satellite imagery.

References

[1] Chen, X., Xiang, S., Liu, C. L., & Pan, C. H. (2013, November). Vehicle detection in satellite images by parallel deep convolutional neural networks. In 2013 2nd IAPR Asian Conference on Pattern Recognition (pp. 181-185). IEEE.

[2] Coster, M., & Chermant, J. L. (2001). Image analysis and mathematical morphology for civil engineering materials. *Cement and Concrete Composites*, *23*(2-3), 133-151.

[3] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[4] Descartes Labs. (n.d.). Retrieved May 06, 2019, from https://search.descarteslabs.com/?layer=naip_v2_rgb_2014-2015#lat=39.2322531&lng=-100.8544921&skipTut=true&zoom=5

[5] Eerscmap.usgs.gov. (2019). U.S. Wind Turbine Database. [online] Available at: https://eerscmap.usgs.gov/uswtdb/ [Accessed 22 Apr. 2019].

[6] Global Energy Perspective 2019. (n.d.). Retrieved May 15, 2019, from https://www.mckinsey.com/industries/oil-and-gas/our-insights/global-energy-perspective-2019

[7] Han, M & Wang, H & Wang, G & Liu, Y. (2018). Targets mask u-net for wind turbines detection in remote sensing images. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-3. 475-480. 10.5194/isprs-archives-XLII-3-475-2018.

[8] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[9] Jabari, S., Zhang, Y., & Suliman, A. (2014, July). Stereo-based building detection in very high resolution satellite imagery using IHS color system. In 2014 IEEE Geoscience and Remote Sensing Symposium (pp. 2301-2304). IEEE.

[10] Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., & Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty. Science, 353(6301), 790-794.

[11] Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., & Ermon, S. (2018). Tile2Vec: Unsupervised representation learning for spatially distributed data. arXiv preprint arXiv:1805.02855.

[12] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

[13] Madhogaria, S., Baggenstoss, P. M., Schikora, M., Koch, W., & Cremers, D. (2015). Car detection by fusion of HOG and causal MRF. IEEE Transactions on Aerospace and Electronic Systems, 51(1), 575-590.

[14] Malof, J. M., Bradbury, K., Collins, L. M., & Newell, R. G. (2016). Automatic detection of solar photovoltaic arrays in high resolution aerial imagery. Applied energy, 183, 229-240.

[15] Malof, J. M., Bradbury, K., Collins, L. M., Newell, R. G., Serrano, A., Wu, H., & Keene, S. (2016, November). Image features for pixel-wise detection of solar photovoltaic arrays in aerial imagery using a random forest classifier. In 2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA) (pp. 799-803). IEEE.

[16] NREL Data Analysis and Visualization Group. (n.d.). The Open PV Project. Retrieved May 15, 2019, from https://openpv.nrel.gov/

[17] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 618-626).

[18] Youssef, M. M. S., Mallet, C., Chehata, N., Le Bris, A., & Gressin, A. (2014, July). Combining top-down and bottom-up approaches for building detection in a single very high resolution satellite image. In 2014 IEEE Geoscience and Remote Sensing Symposium (pp. 4820-4823). IEEE.

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

[20] Yuan, J., Yang, H. H. L., Omitaomu, O. A., & Bhaduri, B. L. (2016, December). Large-scale solar panel mapping from aerial images using deep convolutional networks. In 2016 IEEE International Conference on Big Data (Big Data) (pp. 2703-2708). IEEE.

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