SolarX: Solar Panel Segmentation and Classification

Spencer Paul
Stanford University
spaul2@stanford.edu

Rodrigo Nieto
Stanford University
rjnieto@stanford.edu

Ethan Hellman
Stanford University
hellman1@stanford.edu

Abstract

Increased emissions from fossil fuels has expedited climate change creating a pressing need to shift to renewable sources of energy. Solar photovoltaics (PV) is a promising form of renewable energy, but government and corporate stakeholders lack a comprehensive mapping of the current distribution of PV's. Knowledge of where PV cells are and how many there are is critical information for the purpose of energy generation capacity estimation. We sought to create a model that could segment and detect PV cells from aerial satellite imagery. For detection, we trained a ResNet-34 to achieve an AUC-ROC score of .99. For segmentation we trained a U-Net to achieve an IoU of score of .789. Both models were trained on a data set of 20,900 224x224 satellite images from the cities of Fresno, Stockton, Oxnard, and Modesto. Both models outpreformed prior benchmarks set by DeepSolar. Through various hyper-parameter tuning and experimentation, we seek to optimize a model for the task of PV segmentation and classification.

1. Introduction

Unprecedented levels of carbon dioxide in the Earth's atmosphere have resulted in detrimental climate and environmental impacts that threaten planetary extinction. As a result, governments around the world have been trying to shift from fossil fuels to renewable energy sources. Solar energy has emerged as one of the premier candidates for renewable energy because of its endless availability and limited environmental.

Solar photovoltaic (PV) is an exponentially growing form of renewable energy and many countries have been making efforts to install solar cells on rooftops of homes, business, and other suitable locations due to the promising environmental benefits of the energy source compared to fossil fuels. Generally, distributed solar PV's are installed on rooftops, agricultural lands, and water surfaces. However, the scaling of PV plants is limited by land availability and integrity. To be able to assess energy needs and determine correct allocation of grid resources, governments need a reliable mapping of distributed PV cells. Because PV's are often privately owned and historical data is unreliable, traditional data collection methods have failed to provide an accurate map of the PV landscape.

Furthermore, not all installed PV panels are accurately registered and not all records are up to date. This can result in issues for the renewable energy market as operators need to be able to predict total rooftop solar PV generation over numerous areas, among other concerns. However, the usage of satellite imagery with deep learning can be used as fruitful tool to be able to identify solar PV's to overcome this problem.

We have experimented with multiple machine learning architectures to optimize detection accuracy. Our training approach is split up into two phases. First, we train a classifier to identify whether or not a solar panel is present in the given satellite image. Then, we use the classifier output as a downsampling base for U-net convolutional upsampling which segments the images to locate the positions of the solar PV's.

One can think about this training process as a two step process. The first process, through the help of ResNet34 model that pre-trains on Imagenet data, is used to train a classifier, which will detect whether there is a solar panel present in the photo. The classifier employs a binary cross entropy loss function to evaluate its performance. Furthermore, for the classifier's evaluation metrics, we take two metrics: the AUC - ROC curve and the F1 score. With this classifier established, we are then able to transition into the second half of the training.

For the second phase of the training, the segmentation process is necessary for isolating the polygon shapes where the solar panel is in the photo. To do this, we use a U-net architecture, which is a common model architecture for semantic segmentation. Semantic segmentation is the process of associating each pixel in an image with a class label. In this case, the class of interest would be the area that contains the solar PV's and the output of the model would be the area in which the solar PV's are present in the image. The segmentor is evaluated based off the intersection over union (IoU).

To train the U-net and to test our model, we used a dataset of distributed aerial photography and satellite imagery of solar panels across high-resolution images in California. The dataset is publicly available and contains the geo-spatial coordinates and border vertices of solar cells from the cities of Fresno, Stockton, Oxnard, and Modesto in California. Finally, the data had to undergo several preprocessing operations, such as a mask generator and cropping, before using them for training and testing.

2. Related Work

2.1. Image Segmentation

The objective for semantic segmentation is to assign each pixel of a given image to one of multiple classes. Specifically, it identifies which objects are shown in an image and where exactly are they located in the image. There have been early approaches to semantic segmentation such as the work of He et al. (2004) [1] and Shotton et al. (2009) [2]. Deep neural networks and convolutional neural networks have played a dominant role in the field of image classification and have resulted in deeper and more complex architectures (He et al, 2016) [3].

The cornerstone of semantic segmentation begins with the Fully Convolution Network (FCN) (Long *et. al* 2015) [4] that can be applied to images of any dimension. The architecture uses a convolutional network architecture for the first layers (convolution and max pooling layers are utilized until the image is downsized sufficiently enough). Then the outputs are scaled up to its original size using upsampling and deconvolutional layers. The down stream extracts contextual information and the up stream reconstructs more detailed spatial information. A problem that FCN addresses with using deep neural networks for semantic segmentation is that detailed, deep structures do not get lost. This is because FCN is not a relatively deep architecture and introduces skip connections. Skip connections skip some of the layers of a neural network and feed the output of one of the layers as the input to the next layers.

Skip connections were improved with the introduction of U-net architectures which built on the FCN architecture (Ronneberger *et al.* 2015) [5]. The U-net consists of symmetric contractive and expansive paths with the corresponding layers of both paths connected by skip connections. The U-net is a relatively simple architecture that has become popular in the field of semantic segmentation and has inspired later architectures with its encoder-decoder structure.

2.2. Solar Panel Segmentation

The area of solar panel segmentation is a novel research field; that being said, there have already been several promising approaches. The approaches that have gone down the path of image segmentation typically assign a probability to each pixel (with a classifier) or through CNNs, but with additional layers that output a probability map.

One of the earliest, effective approaches to solar panel segmentation came from Malof *et al.* (2016) [6] at Duke University and utilized a random forest classifier to assign probabilities to each pixel enabling segmentation. This was improved by the usage of CNNs that consisted of convolutional layers and max-pooling layers Malof *et al.* (2016) [7].

A big development in the field was made with the introduction of DeepSolar by Yu *et al.* (2018) [8] at Stanford University. Their project invovles two phases: a CNN based classification followed by semi-supervised segmentation. The group was able to identify solar PV's across the contiguous United States at a precision of 93.1% and recall of 88.5%. The results of this project were groundbreaking for the field and is still considered state-of-the-art today as it influenced updates to other models, such as the CNN utilized by Yuan *et al.* (2016) [9] at Oak Ridge National Lab.

In 2020, the field was heavily influenced by the introduction of the U-net architecture. The paper by Li Zhuang *et al.* (2020) [10] presents an algorithm for automatic segmentation of residential solar panels with a cross learning driven U-net. Since the U-net architecture overcomes disadvantages of FCNs and has demonstrated its performance on analyzing satellite images, it is promising that this project adapts the U-Net to small-scale residential solar panels segmentation. This advancement of the U-net was a great development; however, a potential disadvantage is that due to its specific initialization method, considered in the CrossNet, its application would be constrained to networks that consist of a encoder and decoder structure.

3. Methods

3.1. Overview

We aim to solve two problems: (a) PV classification - a binary classification task predicting if an image contains any solar panels and (b) PV segmentation - generating pixel masks for the areas in an image that contain solar panels. For both our architectures, we used fastai's GitHub repo as a base, tweaking their model to fit our desired output and experimenting with various hyperparameters . [11]

3.2. PV Classification

3.2.1 Model

For PV classification we used a 34-layer residual network (ResNet-34) that can be seen in figure 1. [12] ResNet-34 has achieved state of the art performance - 3.57% error on the ImageNet test set. Given that solar panel detection is a significantly more trivial problem, we figured it was a suitable architecture. [13]

ResNet-34 capitalizes on the idea of "the deeper the better" in the case of CNNs. ResNet-34 is able to go deeper while avoiding vanishing gradients seen by other models by adding skip connections from later layers to initial filters. This connection sums the input (x) and output of each 3x3 convolution block (f(x)):

$$f(x) + x = h(x) \tag{1}$$

When we take the gradient we will never get zero - even if:

$$\frac{\partial f(x)}{\partial x} = 0 \tag{2}$$

from the convolution:

$$\frac{\partial x}{\partial x} = 1 \tag{3}$$

ensuring gradients are non-zero.

The network begins with a 7x7 convolutional layer followed by a pooling layer. Then we preform a series of 3x3 convolutions where input dimensions are persevered by adding padding of 1. Each dotted line in figure 1 shows where the dimensions of the input volume are reduced. This is done by increasing the stride from 1 to 2, instead of a max pooling operation. The structure of this is replicated throughout the network in each of the colored layers. Finally our input is flattened into a FC layer. We then added a sigmoid activation function to the final layer:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

so that the output would be probabilistic. Our threshold was set to .5 to determine the presence of solar panels.

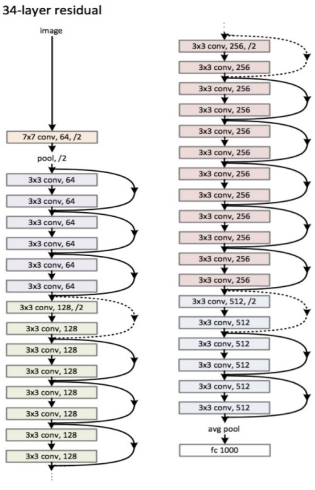


Figure 1. A residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increases dimensions.

3.2.2 Loss Function: Binary Cross-Entropy

$$\text{Loss} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{5}$$

We used binary cross entropy as our loss function as it is standard for binary classification tasks. BCE is easily differentiable and allows us to penalize the probabilities based on the distance from the expected value.

3.2.3 Optimizer: Adam

Adam combines ideas from RMSprop and SGD with momentum to create an ideal optimizer. [14] It uses squared gradients to scale the learning rate as training progresses by v_t. It also takes a moving average of the gradient m_t. See equation where g is the gradient of the current mini batch and the betas are a hyper parameters with default values of .9 and .999 respectively:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{7}$$

Bias correction is preformed to generate $\hat{m}_t$ and $\hat{v}_t$:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1} \tag{8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2} \tag{9}$$

Finally the weights are updated using the following equation:

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \tag{10}$$

3.2.4 Hyperparameters:

For our classification model, the main hyperparameters that we focused on were batch size and learning rate. For our final model, we trained on batch sizes of 16. Our final learning rate was set at 0.0005.

3.3. PV Segmentation

3.3.1 Model

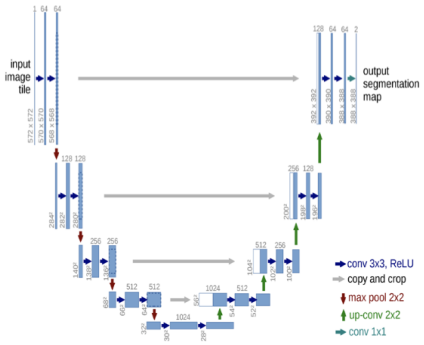


Figure 2. U-net architecture

For segmentation of PV's, we decided to use a UNet architecture designed in 2015 to process biomedical images as seen in figure 2. [15] The network consists of a contracting path (left side) and an expansive path (right side). The contracting path behaves like a typical CNN, applying two 3x3 unpadded convolutions, followed by a ReLU and 2x2 max pooling operation with a stride of 2. For each downsampling block, the number of feature channels are doubled. In the expansive path, the model consists of 2x2 up-convolutions that halve the number of feature channels. Then a concatenation is preformed with the corresponding feature map from the contracting path (visually the gray arrows in figure 1) and two 3x3 convolutions-ReLU are applied. The final layer of the network is a 1x1 convolution used to map the component feature vetcor to the desired number of classes which is 2 in our case. We modified the architecture to take 224x224 inputs instead of the original 512x512 images used by Ronneberger et al. Additionally, we added a sigmoid activation function a the end of the network to generate per-pixel probabilities for containing PV's.

3.3.2 Loss Function: Dice-BCE Loss

$$Loss_{m-bce} =$$

$$-\frac{1}{N}\sum_{i}\beta(y - log(\hat{y})) + (1+\beta)(1-y)(log(1-\hat{y}))$$

$$\text{BCE-Dice Loss}(y,\hat{y}) = \alpha L_{m-bce} - (1-\alpha)DL(y,\hat{y})$$

$$DL(y,\hat{y}) = 1 - \frac{2y\hat{y}+1}{y+\hat{y}+1}$$

For segmentation, we implemented a combination between a binary cross entropy loss function and a dice loss function. In a proper review [16] of an array of loss functions applied to the task of image segmentation, Jadon et al., found a combination of binary cross entropy loss and Dice loss to be competitive with other benchmark loss functions. Compared to other loss functions, this performed better on our task. The Dice-BCE Loss functions as a weighted sum of the Dice and BCE loss functions. Binary Cross-Entropy measures the difference between two probability distributions. For segmentation, this works well at a pixel-level. The Dice coefficient is widely used as a metric in computer vision to calculate the similarity between two images. It was later adapted as a loss function.

3.3.3 Optimizer

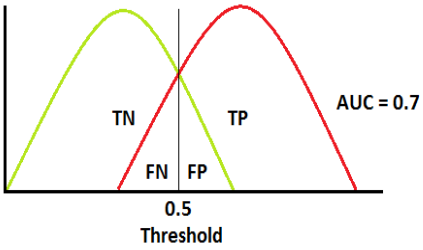
For segmentation, we used the same optimizer (Adam) as our classifier. Dice-BCE Loss fit the segmentation task since we are outputting 224x224 binary masks giving the probability of a pixel containing a PV. In essence, this is the same as classification except on a per pixel basis. Adam also works well here capitalizing on taking smaller steps as we get closer to a global minimum while also keeping momentum as a weighted average of our steps at prior epochs.

3.4. Evaluation Metrics

3.4.1 AOC-RUC Curve

The AUC (area under the curve) ROC (receiver operating characteristics) curve is a frequently employed evaluation metric for checking or visualizing the performance of multi-class classification. More specifically, the ROC is a probability curve, while the AUC indicates the measure of separability — in other words how capable the model is at distinguishing between classes.

Now we will use the ROC curve, which is a curve of probability, to plot the distributions of the overlap between true positives and true negatives, to extract AUC, our evaluation metric.



As we can see, because there is an overlap between the true negatives and the true positives, our AUC evaluates to a score of 0.7. This indicates to us that the model is able to distinguish the difference between the positive and negative classes 70% of the time. This evaluation metric is critical for multi-class segmentation as it infers how well the model does at segmenting classes accurately.

3.4.2 Intersection over Union (IoU)

Intersection over union is a common metric for segmentation models. It measures the overlap between two bounding boxes or masks. This metric gives us information on if the image was segmented correctly and how perfectly the image was segmented. The formula for IoU is as follows:

$$\textbf{IoU} = \frac{\text{Area of overlap}}{\text{Area of union}}$$

The IoU will analyze the output of the model with the output label and calculate the IoU as a metric for how well the model performed.

3.4.3 F1 Score

The F1 score is a popular metric in machine learning that combines the precision and recall of a classifier. Let us define precision, which is the proportion of true positives over all instances that were classified (true positive and false positive) as true in the model. Let us define recall, which is the

proportion of true positives over all instances that are true (true positive or false negative) in the model. The F1 score is thus:

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

We opted to use AUC-ROC as our primary evaluation metric for classification over F1. AUC-ROC is essentially F1 scores at various threshold probabilities so it is more explanatory than an arbitrarily chosen threshold for F1.

4. Dataset

4.1. Description

Due to licensing restrictions and lack of solar cells in typical satellite images, there are very few publicly available data sets for PV classification and segmentation. In 2016, Bradbery et al. sought to decrease the information gap in distributed solar PV arrays by producing a high quality data set for PV detection which we use in our project [17]. The data set consists of 601 5000-by-5000 pixel TIF ariel images and geospacial coordinates for over 19,000 solar panels. Images span multiple cities in California (Fresno, Stockton, Oxnard and Modesto) and contain a diversity of landscapes including urban, suburban and rural regions. Cities were selected to meet two criteria: (a) the spatial resolution had to be higher than 30cm to ensure every panel would be represented by multiple pixels and (b) the regions had to have a high density of solar cells to prevent class imbalance. Each solar panel's geospatial coordinates and vertices were hand labeled by 2 annotators and the union of the two labels were used as the final ground truth label.

(a) Satillite Image (b) Corresponding Mask

Figure 3. Example of PV image and generated mask

4.2. Preprocessing

The data set required a significant amount of preprocessing to get it in an acceptable form to pass into our U-Net (see Appendix Figure 4). Initially, we passed in the images as well as their corresponding PV coordinates into a mask generator that created a 5000x5000 boolean numpy array mask with 0's where solar panels were not present and 1's where they were present for each image (figure 3b). The images were also converted to 3x5000x5000 arrays. To generate more data and to allow our model to train more efficiently we took 224x224 crops of the 601 satellite images and masks using centroid coordinates of the PV's to ensure our crops contained enough solar cells. This brings our total dataset to 20,900 224x224 images to be fed into the model. We randomly split the data using an 80-10-10 distribution for our training, validation, and test sets. The test set contained a split of 48 percent negative samples and 52 positive samples, representing a balanced class distribution. We used this existing code-base for prepossessing. [18]

5. Results

5.1. PV Classification

5.1.1 Architecture

Classification Architecture	Classifier AUC-ROC	F1
Resnet18	0.9954	0.978
Resnet34	0.9993	0.95
Resnet50	0.9991	0.989
Resnet101	0.9984	0.985
Resnet152	0.9972	0.986

Table 1. AUC-ROC vs F1 score for different classification architectures

For PV classification we used three channel satellite images with size 224x224 pixels. The data set of 2,090 images was devised as 80% for training, 10% for validation and 10% for testing. All architectures were pretrained on imagenet as this would give us a better weight initialization for the earlier residual layers in our network. Additionally, we employed early stopping during training after we saw validation loss not improve for 5 epochs. We selected a ResNet-34 architecture for the model (see table 1). The decision was based on prioritizing AUC-ROC over F1 (explained in Methods). We acknowledge that ResNet-50 may have also been a suitable choice as it had the best F1 score of .989 on the test set. Regardless, both of these out-preformed the baseline .914 F1-score established by Stanford research team Deep Solar who used an inception v3 architecture. [8] We achieved a final ROC-AUC of .99 and F1 score of .95 which can be seen in figures 6 and 7.

5.1.2 Optimizer

Optimizer	Classifier AUC-ROC	F1
Adam	0.9993	0.95
RMSProp	0.5	0.6919
Nadam	0.9966	0.9801

Table 2. AUC-ROC vs F1 score for different optimizers. All tests were executed using a resnet34 classification, a batch size of 64, and a binary cross entropy loss function.

Next we experimented with optimizers (see table 2). RMSProp preformed significantly worse than Nadam and Adam. We hypothesize that this is because RMSProp

doesn't account for momentum of the weights and instead just decays the update. Nadam and Adam both incorporate momentum. We opted to choose Adam since the ROC-AUC score outperformed Nadam. Nadam is a slightly more optimized version of adam that works by looking ahead at the next x to preform our current update. The update rule for Nadam can be seen in figure 5.

Algorithm 2 Nesterov-accelerated Adaptive Moment Estimation (Nadam)

Require: $\alpha_0, \ldots, \alpha_T$; $\mu_0, \ldots, \mu_T$; ν; ϵ: Hyperparameters
$\mathbf{m}_0$; $\mathbf{n}_0 \leftarrow 0$ (first/second moment vectors)
while θ_t not converged **do**
$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$
$\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + (1-\mu_t)\mathbf{g}_t$
$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$
$\hat{\mathbf{m}} \leftarrow (\mu_{t+1}\mathbf{m}_t / (1 - \prod_{i=1}^{t+1} \mu_i)) + ((1-\mu_t)\mathbf{g}_t / (1 - \prod_{i=1}^{t} \mu_i))$
$\hat{\mathbf{n}} \leftarrow \nu \mathbf{n}_t / (1 - \nu^t)$
$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{\mathbf{n}}_t + \epsilon}} \hat{\mathbf{m}}_t$
end while
return θ_t

Figure 5. Nadam Algorithm [19]

5.1.3 Hyperparameters

The two hyper-parameters we experiment with are learning rate and batch size.

Class. Batches	Class AUC-ROC	F1
2	0.9963	0.9765
16	0.9967	0.979
32	0.9965	0.9766
64	0.9957	0.975
128	0.9953	0.9784

Table 7. Analyzing the AUC-ROC score and F1 score for different batches for the classifier portion of the training. All tests utilize a BCE loss function, an Adam optimizer, a Resnet34, and learning rate of 0.001

From our results, we find that marginal benefits come from a slightly smaller batch size when training our classifier. As compared to a batch size of 32, a batch of 16 yields an increase in AUC-ROC of .0002. Nonetheless, this is the batch size that proved most optimal and is what our final model was trained on.

Class. Architecture	Class. LR	Class. AUC-ROC	F1
Resnet34	0.005	0.9964	0.976
Resnet34	0.001	0.9967	0.979
Resnet34	0.0005	0.9976	0.982

Table 3. Analyzing the AUC-ROC and F1 score for the classifier with different learning rates and a set classification batch size of 16 for all tests.

Looking at the table above, we find that incremental changes in the learning rate by a factor of 2 has a learning rate of .0005 outperforming the other learning rates by a small amount. As such, our final classification model trains with a learning rate of .0005.

5.2. PV Segmentation

5.2.1 Loss Function

Segmentation Loss Function	Optimizer	IoU
BCE	Adam	0.7552
Dice Loss	Adam	0.7626
BCE-Dice Loss	Adam	0.8049
IoU Loss	Adam	0.7518
Focal Loss	Adam	0.7552

Table 5. Analyzing the IoU scores for different segmentation loss functions with Adam set as the optimizer for all tests.

In line with the formal assessment of different image segmentation loss functions [16], we chose to train on 5 different loss functions: Binary Cross-Entropy, Dice, Binary Cross-Entropy Dice Combination, IoU, and Focal loss. Different loss functions promise to address certain aspects. BCE loss is standard for comparing probability distributions between two sets. Dice score is a standardized metric for comparing the similarity between images and has been adapted to a loss function for training. BCE-Dice loss combines the two aforementioned formulas and benefits from a standard image segmentation metric and assessment of probability distribution. IoU is another standardized evaluation metric to compare the similarity between two images which is adapted as a loss function. Lastly, Focal loss can also be seen as a variation of BCE. It down-weights "easy" examples and enables the model to focus more on "hard" examples. It typically works well for highly imbalanced datasets.

Overall, we find that the combination of BCE and Dice Loss outperformed all other loss functions. Of note, the second best is the uncombined Dice loss. The third best was the BCE loss. As such, balancing the contributions of both enables peak performance, as promised in previous analysis [16].

5.2.2 Optimizer

We decide to train using an Adam optimizer for simplicity. Our experimentation timeline did not afford us sufficient time to test different optimizers for our segmentor.

5.2.3 Hyperparameters

Segmentation Loss Function	Batch Size	IoU
BCE-Dice Loss	2	0.771
BCE-Dice Loss	16	0.7182
BCE-Dice Loss	32	0.8105
BCE-Dice Loss	64	0.7786
BCE-Dice Loss	128	0.7689

Table 6. Analyzing the IoU scores for different batch sizes with Adam set as the optimizer and with a segmentation loss function set as BCE-Dice Loss for all tests.

In our experiments, we tune the batch size to be able to find the ideal trade off between a smaller and larger batch size.

A larger batch size implies less noisy gradients, but it necessitates more memory and is slower. A smaller batch size would be more noisy; however, it would converge faster. We evaluated differences in batch size performance using IoU. Overall, it was determined that a batch size of 32 produced the greatest IoU score, and is thus the optimal batch size for this model.

Segmen. Optimizer	Segmen LR	IoU
Adam	0.005	0.7317
Adam	0.001	0.789
Adam	0.0005	0.7768

Table 4. Analyzing the IoU score for the segmentor with different learning rates and a set segmentor batch size of 32 and the segmentation loss function set as BCE-Dice loss for all tests.

Analyzing the results our learning rate optimization, it is clear that a learning rate of .001 outperforms both a higher and lower architecture. Though more precise changes in the learning rate could potentially make a difference, we found this middle point to be the most optimal learning rate of those we tested.

5.3. Visualization

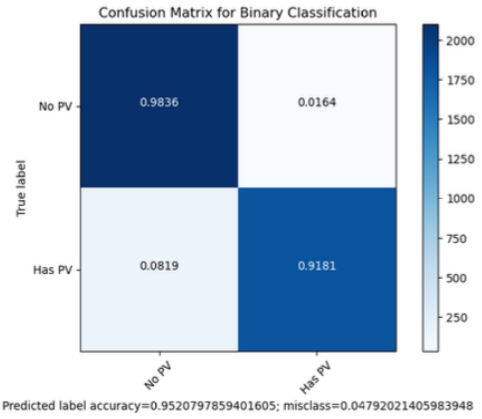


Figure 6. Confusion matrix at threshold = .5 for ResNet-34

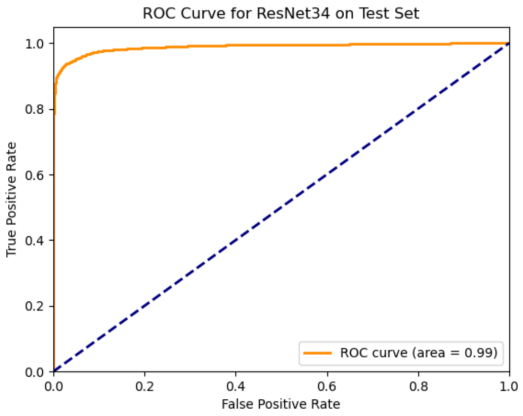


Figure 7. ROC-AUC Graph for Resnet-34 Classifier

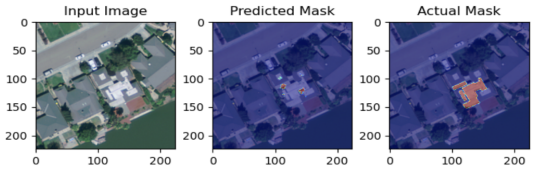


Figure 8. Example of poor segmentation, IoU = .055

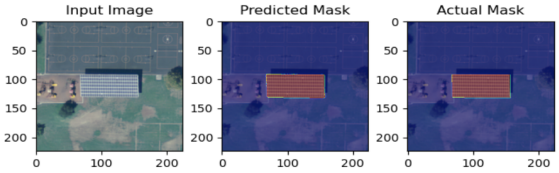


Figure 9. Example of good segmentation, IoU = .98

6. Conclusion

In this paper, we present a solar panel segmentation model that works to classify and segment solar PV's in a given image. The model divides the training portion into two phases: a pre-trained Resnet34 model for classification and a U-net model for segmentation. We experiment with multiple classification architectures, such as different forms of Resnet. In addition, we experiment with other variables such as the segmentation batch size, the optimizer, and segmentation loss function. Our model produced promising results and if we compare it with the likes of DeepSolar as a benchmark,

our model is able to classify solar PV's more effectively.

6.1. Future Work

While our methodology seeks to converge on an optimal model for solar panel segmentation from satellite imagery, there are a few areas which could benefit from further exploration. Given that we choose to use AUC-ROC as our standardized evaluation metric for different classification models, we deemed an Adam optimizer to be better than a NAdam optimizer. That being said, if instead, the F1 score was used as a standardized evaluation metric, a NAdam optimizer could be considered more optimal. As such, in the future, further exploration should be done to evaluate the performance of a classifier using a NAdam optimizer with the various other hyperparameters that were tuned. By the same logic, a different architecture may have proven more optimal as well. Table 1. shows how a Resnet50 may have outperformed other Resnet architectures if the evaluation were the F1 score. Therefore, further tests should be done to evaluate this architecture compared to a Resnet34 with optimally tuned parameters.

Lastly, for the segmentation task, it is most common to see U-Net architecture. However, that does not mean that a different architecture such as a Joint Pyramid Upsampling module, as in FastFCN [19], a two-stream Gated Shape CNN, as in Gated-SCNN [20], an Atrous CNN, as in DeepLab [21], or a pixel-wise bounding box and semantic segmentation CNN, as in Mask R-CNN [22], would not have performed better. Future work should include a broader exploration of different architectures potentially suited to this task.

7. Contributions

This research would not be possible without code provided from fastai for initial models as well as gabrieltseng for data preprocessing (see refrences).
We also could not have done this work without the following python libraries (numpy, pytorch, sklearn, matplotlib, PIL). [23] [24] [25] [26] [27]

Spencer Paul was responsible for initial research into PV segmentation and selection of the model architectures alongside Rodrigo Nieto. He also was responsible for writing the abstract, methods and results section of the paper. Additionally, he wrote the evaluation and visualization code for graphics in results section. He helped with training and experimentation although the majority was done by Ethan Hellman.

Ethan Hellman was responsible for model experimentation and designing the research process so as to evaluate hyperparameters, loss functions, and architectures. He did the majority of training for the models. Additionally, Ethan Hellman was chiefly responsible for the sourcing of data. This includes the final dataset used in this study. He also supplied coffee for everyone.

Rodrigo Nieto was responsible for the election of the PV segmentation and selection model with Spencer Paul. In addition, Rodrigo Nieto researched and procured the literature and state-of-the-art models of current solar PV segmentation projects to be able identify promising techniques, architectures, and to build a better understanding of our benchmark. He was also involved in the data pre-processing portion of the project.

8. Appendix

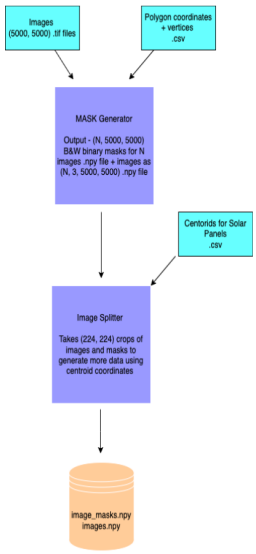


Figure 4. Preprocessing Pipeline

References

[1] Zemel R. S. He, X. and Carreira-Perpiñán. Multiscale conditional random fields for image labeling. *IEEE*, 2004. 2

[2] Winn J. Rother C. Shotton, J. and A. Criminisi. Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput*, 2009. 2

[3] Zhang X. Ren S. He, K. and J. Sun. Deep residual learning for image recognition. *IEEE*, 2016. 2

[4] Shelhamer E. Long, J. and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE*, 2015. 2

[5] Fischer P. Ronneberger, O. and T. Brox. U-net: convolutional networks for biomedical image segmentation. *Medical*

image computing and computer-assisted intervention (MICCAI), 2015. 2

[6] Leslie M. Collins Jordan M. Malof, Kyle Bradbury and Richard G. Newell. Automatic detection of solar photovoltaic arrays in high resolution aerial imagery. *Applied Energy*, 2016. 2

[7] K. Bradbury J. M. Malof, L. M. Collins and R. G. Newell. A deep convolu-tional neural network and a random forest classifier for solar photovoltaic arraydetection in aerial imagery. *IEEE*, 2016. 2

[8] Arun Majumdar Jiafan Yu, Zhecheng Wang and Ram Rajagopal. Deepsolar:a machine learning framework to efficiently construct a solar deploymentdatabase in the united states. *Joule*, 2018. 2, 5

[9] O. A. Omitaomu J. Yuan, H. L. Yang and B. L. Bhaduri. Large-scalesolar panel mapping from aerial images using deep convolutional networks. *IEEE*, 2016. 2

[10] Li Zhuang, Zijun Zhuang, and Long Wang. The automatic segmentation of residential solar panels based on satellite images: A cross learning driven u-net method. *ELSEVIER*, 2020. 2

[11] Fastai python library. "https://github.com/fastai/fastai", 2022. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[13] Vincent Feng. An overview of resnet and its variants. *Towards Data Science*, 2017. 2

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 3

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 4

[16] Shruti Jadon. A survey of loss functions for semantic segmentation. *IEEE TPAMI*, pages 1–6. 4, 6

[17] Kyle Bradbury. Distributed solar photovoltaic array location and extent data set for remote sensing object identification. *Sci Data*, 2016. 5

[18] "https://github.com/gabrieltseng/solar-panel-segmentation", 2019. 5

[19] Kaiqi Huang Kongming Liang Yizhou Yu Huikai Wu, Junge Zhang. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. 2019. 8

[20] Varun Jampani Sanja Fidler Towaki Takikawa, David Acuna. Gated-scnn: Gated shape cnns for semantic segmentation. 2019. 8

[21] Iasonas Kokkinos Kevin Murphy Alan L. Yuille Liang-Chieh Chen, George Papandreou. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. 2017. 8

[22] Piotr Dollár Ross Girshick Kaiming He, Georgia Gkioxari. Mask r-cnn. 2017. 8

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. 8

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 8

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 8

[26] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007. 8

[27] P Umesh. Image processing in python. *CSI Communications*, 23, 2012. 8