– Independent Work Report Fall 2021 –

A Step Towards a Better Solar Panel Development Tracking System

Lucas Gen
Adviser: Alan Kaplan

Abstract

Given the need to transition to renewable energy sources such as solar panels, a universal, reliable, thorough, and accurate system of tracking the development of these solar panels needs to be available. Many companies or organizations post their individual solar panel generation information, but getting a cumulative dashboard of all the solar panels in an area, regardless of who they belong to is not currently an easy task. Previous attempts tend to rely on user-volunteered information which is hard to come by and therefore incomplete. Using a series of convolutional neural nets, one of which uses the popular UNet architecture, we perform both binary classification and image segmentation on images from a given region in order to detect if there are solar panels present, and if so, where they are. Our models performed well on the whole, with binary classification accuracy of up to 85% and image segmentation accuracy rates of up to around 89%. These promising results indicate that a reliable, accurate, and thorough automatic solar panel development tracker is a very realizable goal.

1. Introduction

1.1. Motivation

As global temperatures rise annually, and earth's natural resources continue to be depleted, renewable energy sources are becoming more and more necessary. Continuing to rely on fossil fuels and other polluting energy sources has already caused irreparable damage to the earth, the damage of which will only continue to get worse if drastic changes aren't made soon. In light of this, many home owners and communities are increasingly making the transition to renewable energy sources such

as solar panels. Not only are individuals switching to these energy sources, governments are also beginning to implement required transition deadlines. For example, in its Puerto Rico Energy Public Policy Act (Act 17), passed in April of 2019, Puerto Rico has required their energy come from 40% renewable energy resources by 2025, 60% by 2040, and 100% by 2050[6]. While this is a worthy plan, the government so far has not been on track to meet these goals. At the time of this paper's publication, almost three years have passed since the passing of Act 17, and yet very little progress has been made. As of 2021, three years after Act 17 was passed, only 3% of the energy generated by Puerto Rico comes from renewable energy sources[6].

In addition to the environmental benefits that solar panels have, for the people of Puerto Rico, they have tremendous other benefits. Puerto Rico has a history of power grid issues and even after switching energy providers from a public company (PREPA) to a private one (LUMA) in 2021, blackouts have still been plaguing the commonwealth[4]. On top of the unprovoked power outages due to extremely unstable providers, Puerto Rico has also been hit in the past by devastating natural disasters. One of the latest, Hurricane Maria, left many residents stranded in Puerto Rico for months without power[8]. Therefore, on top of the environmental impacts, solar panels for people in Puerto Rico provide independence from the fragile power grid and a more reliable supply of sometimes life-saving power.

Despite the transition to renewable energy sources, specifically solar panels, being of such importance to the people of Puerto Rico, there currently isn't a good way for Puerto Rican residents to see the progress (or lack thereof) towards their renewable energy goals. Tracking the current progress of the transition needs to be made as public as possible so that the people of Puerto Rico can stay up to date and hold the government accountable to the acts that have been passed. The closest resource that exists is known as the Puerto Rico Solar Map, as can be seen in Figure 1. While still being a helpful resource, this map has many flaws. Namely, it's not up-to-date and it's incomplete. The creators of this map have recorded a total of 412 solar panel systems on this map with battery size information down to the thousandth of a kilowatt-hour, but they rely solely on people volunteering this information. Because of this, the map has nowhere near all of the

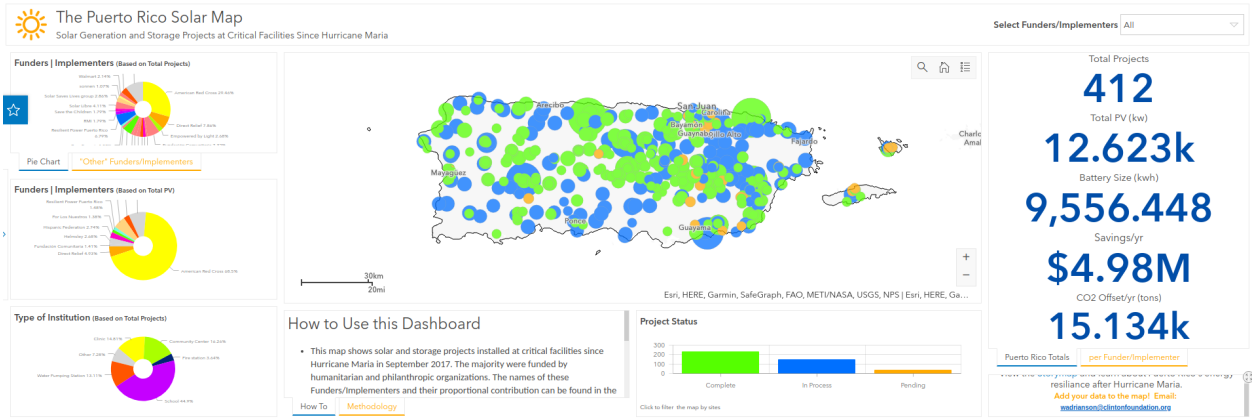

Figure 1: Puerto Rico Solar Map[5]

solar panels in Puerto Rico documented. In addition, this map is updated by hand which takes an incredible amount of time and manpower, resulting in the map being updated at most biannually, and often much less frequently.

1.2. Goal

Given the need for the transition to renewable energy sources, including laws such as Puerto Rico's Act 17, there is more of a need than ever to track the amount of renewable energy being produced in specific areas. In an attempt to assist this cause, this project focuses on the development of solar panels. While previous projects such as the Puerto Rico Solar Map have attempted to track the solar panels in a given area (Puerto Rico), there needs to be a more reliable and thorough system than this. While the approach to accomplish this task is very different, the goal of this project is very similar to that of the Puerto Rico Solar Map. This project aims to thoroughly track the development of solar panels in a specific region, using only reliable and accurate information.

2. Related Work

Many projects in machine learning have attempted to track the development of solar panels in specific regions before. These projects have a wide range in scope, from smaller, personal projects, to teams attempting to track every solar panel on the planet.

On the smaller end are projects similar to Jose Padarian's approach[11]. This project attempts to detect solar panels in a given set of images using a convolutional neural net. The intriguing

aspects of this project are that it's relatively simple, detects individual solar panels on a small scale, and reports very high accuracy rates. On the training set, this project reports a ninety-five percent accuracy rate, as well as a ninety-two percent accuracy rate on the validation set[11]. The drawback to this project however, is that the datasets used to train and test the machine learning model are quite small. This model is trained on only 124 labelled solar panels which were used to create the training and validation sets[11]. Because of this, not only is the model likely not generalizable at all, but the accuracy rates reported are on a very small group of actual solar panels and therefore are not as significant as the rates would seem to indicate. More projects, with larger and more robust datasets should be developed in order to create a more powerful method of detecting solar panels.

A larger-scale example of a project in this field was created by Kruitwagen et al[14]. This project also uses machine learning techniques to automatically detect solar panels from aerial images. Specifically, this project uses satellite images to detect industrial, large-scale solar panel installations. This project is powerful in many ways, namely in its ability to scan the entire earth in search of solar panels. It has been used to locate many solar panels which weren't recorded previously, locating over sixty-eight thousand solar panels in total[14]. However, the machine learning model used for this project is only trained to detect large, industrial solar panel installations. Because of this, it completely ignores all of the residential and smaller-scale solar panels that are becoming increasingly common and accounting for more and more energy output. While a very powerful project, more models should be trained to help detect not only large-scale installations, but smaller ones as well.

Perhaps the most significant previous work in this field is known as DeepSolar[16]. This project has scanned over a billion image tiles, and has detected solar panels all across the contiguous United States. This project is compelling since it performs both classification on the images it is fed to determine whether or not there are any solar panels within the image, as well as image segmentation to determine which pixels of the image are part of the solar panel. In addition, the neural net this project uses to perform image segmentation is semi-supervised, meaning that it doesn't need ground-truth labels in order to make predictions about where a solar panel is located[16]. Finally,

this project not only detects solar panels, but also approximates their size and location to record into a database of the contiguous United States. This is a very powerful project with many strengths. However, more related projects should still be developed. One reason is that this model has only been trained and tested in the contiguous United States. It has not produced any valuable information for the people of Puerto Rico who want to make sure their government is on track to reach their clean energy goals. In addition, while this project is fantastic for larger-scale implementation, it is still very worthwhile to create more localized machine learning models to make predictions on specific areas since these will likely have higher accuracy rates for that area than one which needs to be more general in scope.

3. Approach

While the goal for this project is very similar to many other projects, the approach to achieving this goal is different than most. Unlike the Puerto Rico Solar Map which is limited by the unreliable source of information coming from volunteers, this project aims to accomplish the same goal, but based on a much more reliable source of information. While it's difficult to communicate with everyone who owns a solar panel in a given area, it's relatively easy to obtain satellite images from these areas. Therefore, aerial images are a much more reliable source of information for this goal than volunteered information. In addition, these aerial images have to contain almost all of the solar panels in the area they cover. This is evident since, in order for solar panels to function they need to be exposed to the sunlight and therefore the sky, where aerial photos are being taken from.

Given that this project will rely on aerial imagery as the source of information for helping to track the development of solar panels, the first step to doing this is finding out whether or not an image even has a solar panel within it at all. In other words, given an image, can we classify this image as either an image with a solar panel, or an image without a solar panel? Due to the nature of this question, it lends itself quite well to a binary classification problem in machine learning.

3.1. Binary Classification

Classification problems in machine learning are problems in which the machine learning models are designed to assign a class to the inputs. A binary classification problem is, quite simply, a classification problem in which there are only two labels, so each input has to be labeled with one class or the other. For this project, the the two labels will correspond to whether or not the image contains a solar panel or not.

There are a number of machine learning models that are commonly and effectively used to perform binary classification. Examples include logistic regression, k-nearest neighbor, and even decision tree. However, this project attempts to perform binary classification using a convolutional neural net (CNN). CNN's are very popular machine learning models due to their adaptability and power. They also lend themselves quite well to problems that need to take in images as input, and as such, will be used as the machine learning model for the binary classification aspect of this project[12].

3.1.1. Neural Nets

Neural nets are machine learning models whose design is loosely modeled around the human brain (hence "neural"). They consist of a series of layers, each with a number of nodes which the data is passed through. Figure 2 depicts a simple neural net with three layers, depicted by the three columns of circles connected through arrows to the other columns. Each of these three layers have a number of nodes belonging to them, labelled as $x_1 ... x_n$, $h_1 ... h_n$, and $y_1 ... y_n$. The first layer, with nodes labelled $x_1 ... x_n$ is denoted as the "input layer" and the final layer with nodes labelled $y_1 ... y_n$ as the "output layer" as can be seen in Figure 2. Neural Nets are fed data into the input layer, and they pass this data along the arrows in Figure 2 to the next layer and the next, etc. Each arrow in the diagram represents a connection from one node to the next, and along this connection a weight is multiplied to the value passed from the previous node which allows some features of the input to be valued more heavily than others in the final prediction. Then, at each node, a non-linear activation function can be applied to the values received from the previous layer, which gives neural nets the power to predict more than only linear functions which would be the case without some non-linear operation.

Also at each node, a bias can be added, which is simply a value added at the end of each node's calculation before the total value is passed to the next node. This bias value can be used to skew the model's predictions in one direction or the other. For example, a machine learning model that predicts whether or not a patient has cancer may want to err on the side of predicting the patient has cancer when they don't as opposed to the opposite since the opposite has much larger consequences than the first. The final values that are calculated at the output layer are the model's predictions. After this forward pass through the neural net is complete, the error of this prediction is calculated by comparing the predictions made by the model with the ground-truth labels of the dataset through one of many loss functions. Finally, the neural net is traversed in the reverse order, the weights and biases being adjusted at each node in an effort to make the error shrink. This process is repeated as many times as needed in order to get the best predictions possible.

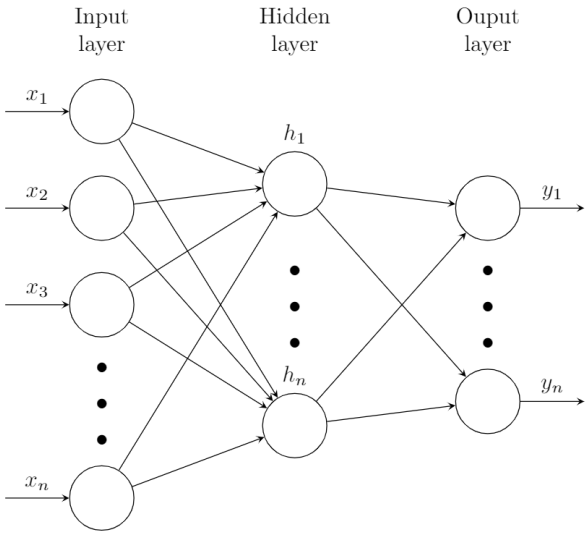

Figure 2: Diagram of a simple neural net[3]

3.2. Image Segmentation

After this project makes predictions about which images contain solar panels or not, we want to then identify exactly where the solar panels are in the image. This type of machine learning task is known as image segmentation. This project will predict where the solar panels are on a pixel-by-pixel basis,

meaning that, for each pixel in the input image, it will compute the probability that pixel is part of a solar panel. Once it does this for every pixel, we will output the new image, with it's prediction of where solar panels are in the image. An example of one such prediction, made by this project, can be seen in Figure 3. By first detecting whether an image has a solar panel, and then predicting

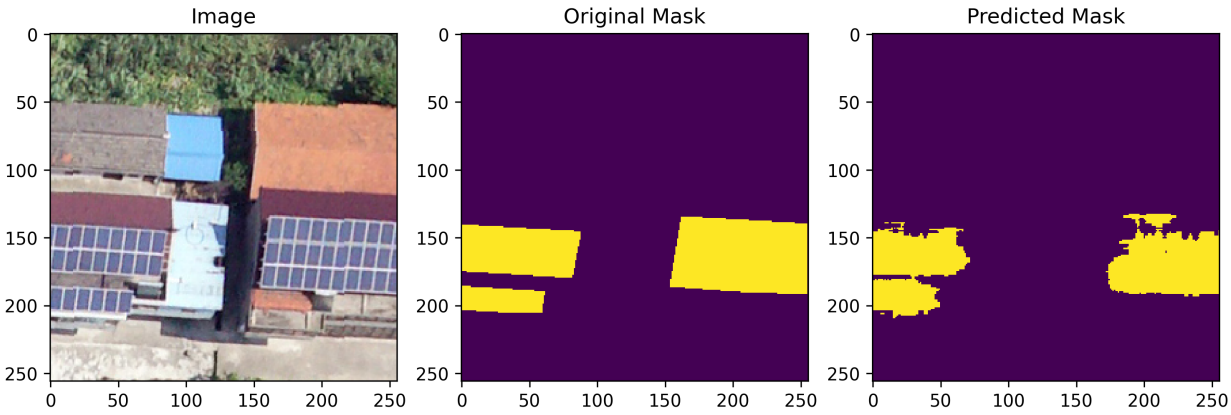

Figure 3: Model Segmentation Prediction

where that solar panel is in the image, we can gain a lot of information about the solar panels in a given area.

3.2.1. UNet Architecture

For this image segmentation task, we will use a convolutional neural net as well. However, we will use a series of convolutional layers in a very specific architecture that is know as UNet. The UNet architecture is very popular for this type of image segmentation problem and it was originally developed for biomedical image segmentation tasks[10]. The uniqueness about this architecture is that it is based around a series of encoding and decoding layers that can be seen in Figure 4. The left side of Figure 4 shows the encoding side of the neural net, in which data is abstracted, and then transformed into lower dimensions in order to process higher-level features. The right side of the diagram on the other hand shows the decoding side of the neural net in which the data is translated back into it's original dimensions, decoded from the higher abstractions.

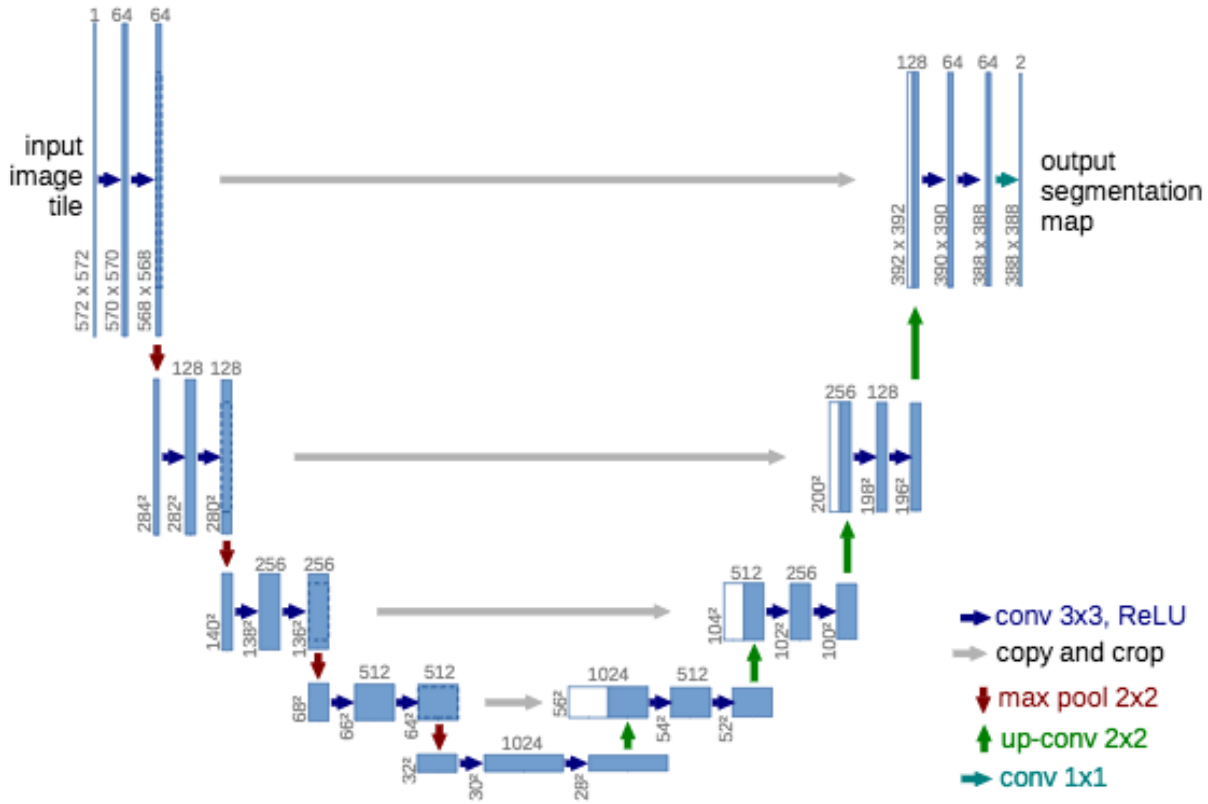

Figure 4: Example Diagram of UNet architecture[15]

4. Implementation

The following subsections describe the steps this project took to implement the machine learning approaches described above in section 3.

4.1. Dataset

As with most machine learning projects, this project's performance is highly dependent on the amount of high-quality data it is trained on. High quality data is data that it as relevant to the given topic as possible. In addition, both binary classification and image segmentation are usually supervised models, meaning that these models need to be trained on datasets that have labels for each of its images. These labels allow the machine learning model to calculate how wrong its predictions are, and adjust to minimize the loss accordingly. For the binary classification problem of predicting whether or not an image has a solar panel, labels consist of some sort of indication of whether a given image in the dataset has a solar panel or not. For the image segmentation problem

of attempting to determine where a solar panel is in a given image, the labels for these images need to have some indication of where the solar panels are in the image.

4.1.1. Obtaining Segmentation Images

Creating a dataset for the image segmentation problem of predicting where solar panels are in an image would be very time-consuming. Therefore, this project uses a public dataset which can be found on Zenodo.org, titled "Multi-resolution dataset for photovoltaic panel segmentation from satellite and aerial imagery"[13]. This dataset contains three different groups of solar panel images, with spatial resolutions of 0.8m, 0.3m, and 0.1m respectively. These groups are further split up according to the type of land in the background of the images. Examples of the type of land in the background include shrub land, flat concrete, and water surface. This dataset is relatively large, with over seven total gigabytes worth of images, consisting of both pictures and labels. However, this project is only able to use the group of images with 0.1m spatial resolution as the others are taken from too far away for the goal of this project.

This dataset is used as it contains labels for each image which will help our image segmentation model predict where solar panels are in an image. Each image in this dataset has a label that is also an image file, but has differently colored pixels, denoting where the solar panel(s) exist in the original image. An example of one of these labels alongside its original image can be seen in Figure 5.

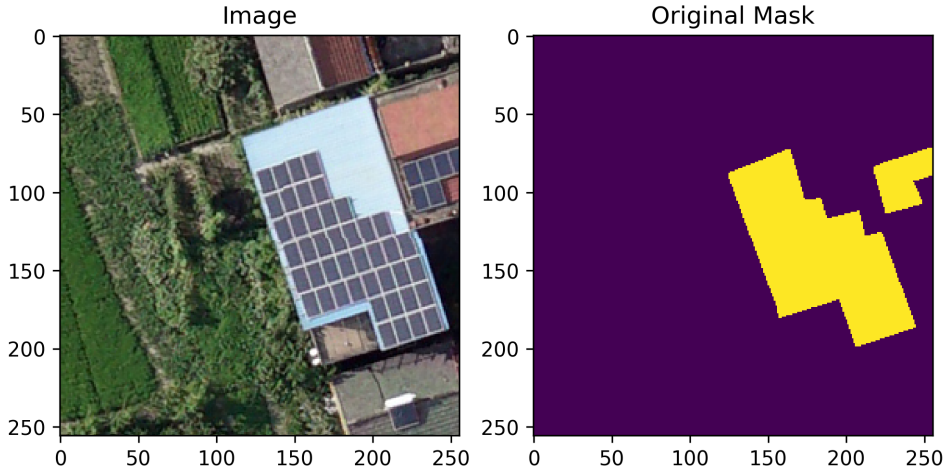

Figure 5: Image Segmentation Dataset Image and Label

4.1.2. Obtaining Classification Images

For the binary classification of predicting which images contain solar panels and which do not, the neural net needs a dataset consisting of images that both contain solar panels and don't. In order to get this dataset, this project combines two datasets, one with images only with solar panels, and one with images without solar panels. For the dataset containing images with solar panels, this project reuses the dataset from 4.1.1 since it is a high-quality dataset that only contains images with solar panels.

For the second dataset consisting of only images without solar panels, this project uses a dataset from "Fractalytics" who created this dataset for their machine learning project of automatically classifying rooftops to a certain type [1]. An example of one of the images can be seen in Figure 6 Even though this dataset was designed for a different classification problem, it consists of images

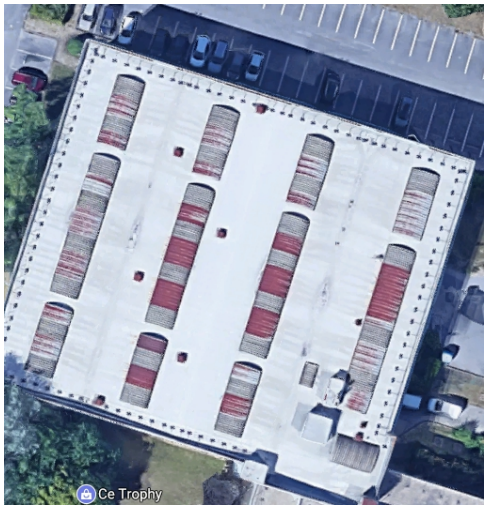

Figure 6: Classification Dataset Example With No Solar Panel

of rooftops from very similar angles and distances as the other dataset from 4.1.1. While this dataset was not created to ensure that none of the images have solar panels as the other dataset was created to ensure that every image has a solar panel, upon inspection, there are very few images in the dataset that contain solar panels. Even if a few images with incorrect labels exist in the cumulative dataset, it shouldn't drastically affect the predictive power of the model as long as there are sufficiently enough images in the dataset. Another problem with this dataset is that the images

are all different dimensions. Neural nets have set numbers of nodes in the input layer, and therefore need a constant size image to be the input. As the first dataset from section 4.1.1 has constant image size of 256x256 pixels, the images from this second dataset were all resized to fit these dimensions. Using the Python Imaging Library (PIL), all of the images from this second dataset are iterated through, and resized to this 256x256 resolution.

After standardizing the size of every image, the cumulative dataset for this binary classification problem of predicting which images have solar panels and which don't was created by combining the datasets with solar panels and the one without. This cumulative dataset for binary classification consisted of an equal number of images with solar panels as without. This is important, as having much more of one class of image as the other could result in the model being too eager to predict the class with a higher representation in the dataset. By including an equal number of both classes in the dataset, we ensured that the model was attempting to detect actual features of images with solar panels in them as opposed to defaulting on guessing the higher represented class while still having a relatively low error rate. The cumulative dataset for this binary classification problem ended up with 1290 images in total, half being images with solar panels, and half without.

The last task remaining to prepare this dataset was to label every image. This was done by running the cumulative dataset through a function which checks each file for it's filetype. Since the filetypes from the two different datasets are different, this function appends a "class" label to each image's file with an integer describing whether it has a solar panel or not.

4.1.3. Making Pytorch Datasets

In order to use these datasets of images that have been prepared, they need to be loaded into a form that PyTorch can understand since this is the Python library that this project uses for a lot of the neural net heavy lifting. Fortunately, PyTorch has a Datasets and Dataloaders API which simplifies this process a bit. To define the dataset for PyTorch, we first import PyTorch's `toch.utils.data.Dataset` class and then define a custom dataset class that overrides a couple of the built-in functions to fit our specific dataset.

Then, we need to create a Trainloader for this dataset. Trainloaders are simply tools that PyTorch

created to allow people to iterate through their datasets while also easily utilizing more advanced machine learning techniques, such as batching the data, shuffling the data, and also loading data in parallel to take advantage of multiple processors. Batching is when a dataset is split up into "batches" of smaller subsets which are then passed through a neural network as opposed to putting the entire dataset into the network at once. This can save a tremendous amount of memory since loading the entire dataset into the network at once could be extremely costly, and can also speed up training since our model can update it's weights and biases after each iteration through a batch instead of after each iteration through the entire dataset. Since it's best to utilize these helpful machine learning techniques, we will use PyTorch's built in Trainloader class, making sure to set up batching for these datasets.

4.2. Binary Classification

This project uses a convolutional neural net to make predictions about whether an image has a solar panel in it or not. Using PyTorch's infrastructure to create a convolutional neural net involves first, importing the `torch.nn.Module` class which sets up a neural net, and then overriding a few key functions to adapt it properly to the problem at hand.

The first function to override is the `__init__` function which declares the layers of our Convolutional Neural Net. This project uses two convolutional layers, connected through one MaxPooling layer and three Linear layers as depicted in Figure 7.

```
def __init__(self):
    super().__init__()
    self.conv1 = nn.Conv2d(3, 6, 5)
    self.pool = nn.MaxPool2d(2, 2)
    self.conv2 = nn.Conv2d(6, 16, 5)
    self.fc1 = nn.Linear(59536, 120)
    self.fc2 = nn.Linear(120, 84)
    self.fc3 = nn.Linear(84, 2)
```

Figure 7: Binary Classification Layers

The second function that needs to be declared is the `forward` function which describes how this neural net should perform a forward pass. The code is shown in Figure 8. As shown in Figure 8,

```
def forward(self, x):
    x = self.conv1(x)
    x = F.relu(x)
    x = self.pool(x)
    x = self.pool(F.relu(self.conv2(x)))
    x = torch.flatten(x, 1) # flatten all dimensions except batch
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.fc3(x)
    return x
```

Figure 8: Binary Classification Forward Function

this neural net passes the data(x) through each layer defined in Figure 7, using the ReLU activation function often in between layers in order to allow for extra non-linear complex predictions. After the data is passed through each layer, the final value is returned at the end as the neural net's prediction.

Now that the model is declared and knows how to process the data fed into it, it needs to be trained. For this, the model needs a loss function and optimizer. The loss function tells the model the unit of measurement to use to calculate how far away its predictions are from the truth. The optimizer on the other hand tells the model how to improve, and minimize this loss. This neural net uses `nn.CrossEntropyLoss()` which is a standard method for calculating loss. This loss function is designed to measure the error in a model who's outputs are probabilities between zero and one, as this project's binary classification model is. To better understand this function, the code is shown in Figure 9.

```
def CrossEntropy(yHat, y):
    if y == 1:
      return -log(yHat)
    else:
      return -log(1 - yHat)
```

Figure 9: Cross Entropy Loss Code[2]

The optimizer this project uses is known as stochastic gradient descent, which is a very popular machine learning method for optimization. In order to minimize the loss of the model, stochastic gradient descent attempts to travel down the curves of the model's loss function until it gets to the minimum. To do this, the algorithm starts at a random point on the function, calculates the gradient

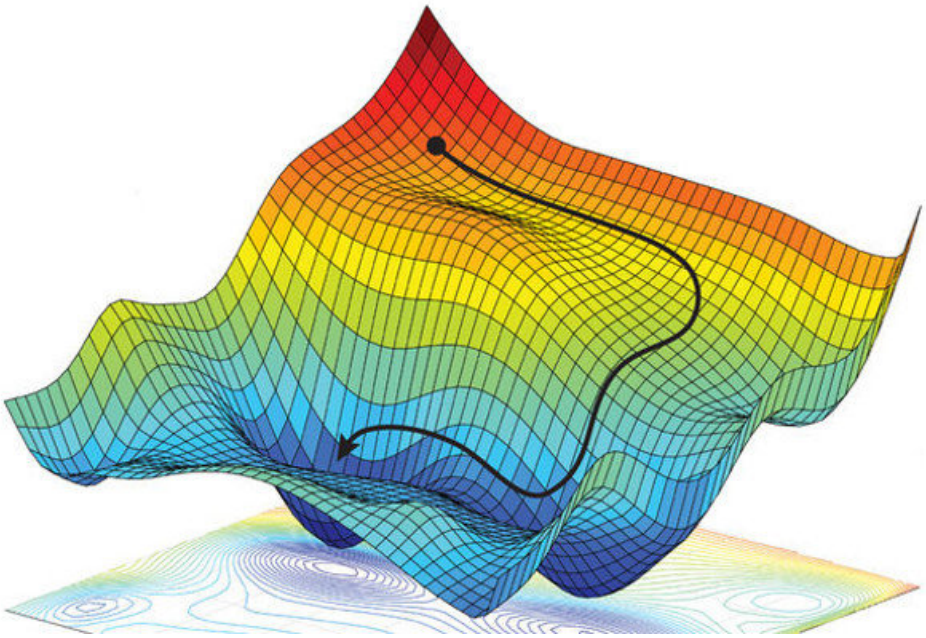

Figure 10: Diagram of Gradient Descent[7]

at that point, and then moves in the direction of the gradient, with the idea being that this is the direction of the minimum. In Figure 10, the black line depicts the path the gradient took in its efforts to get to the minimum value. Note that this is an iterative algorithm, so it does not know where the true minimum is, it can only calculate at each point, the direction that goes down the fastest, and follow this path. The difference between stochastic gradient descent and gradient descent is that instead of calculating the gradient with respect to the entire dataset every time which can be very expensive to do, stochastic gradient descent generates a collection of random data points each time and calculates the gradient of those. This algorithm has proven to have very good results and is much less expensive than gradient descent.

4.2.1. Training

Now that the neural net for this project's binary classification problem has been set up, it needs to be trained. In order to train this model, the dataset we created was iterated through using the Trainloader we built. The images were fed into and passed through our neural net, the loss was calculated at the end, and then the weights and biases of the neural net were updated according to which direction the stochastic gradient descent algorithm calculated would decrease loss the most. Then this process was repeated. Many of Pytorch's built-in functions made these steps easier. After

experimenting with the parameters of this model, we produced our best results when training this model for 50 epochs with a batch size of 64.

4.3. Image Segmentation

For the image segmentation task of predicting which pixels in an image are part of a solar panel, this project also uses a convolutional neural net, but this time with a much different architecture that's known as UNet. In order to implement some of the architecture as described in section 3.2, the first thing we need to do is setup our dataset for this segmentation model. Fortunately, this dataset setup is very similar to that of the binary classification problem.

4.3.1. Segmentation Model

There are several different ways to go about creating the UNet architecture. The method this project uses is based off of Shivam Chandhok's method, as demonstrated in their article "U-Net: Training Image Segmentation Models in PyTorch"[9], due to it's modular design and utilization of the PyTorch library. For this method, there are a few classes that need to be created.

The first class being a "Block" class, which will be used to create both the encoding layers and the decoding layers. Each block will define two convolutional layers connected through a ReLU function and defines a forward function which takes in an input, runs it through these layers, and returns the number of values that was specified in the Block definition. Keeping this Block class dynamic as it is, allows it to be used to create both the Encoder and Decoder classes from it.

Next, the Encoder class needs to be created, which starts with the input to the neural net and reduces the dimensionality of this input while creating higher levels of abstraction. This class creates a list of Blocks upon instantiation, and passes the data through these blocks when trained, increasing the dimensionality of each block as it goes down as the number of feature channels increases.

The Decoder class is similar in structure to create, but has the number of channel features decrease from block to block as the data is passed upward, and the number of feature channels decreases.

Finally, we put all of these classes together as we create a final UNet class. Upon instantiation,

this class creates the Encoders and Decoders, and inside it's `forward` function, this class passes the input image, down through the Encoders and back up through the Decoders until the final prediction is shown.

4.3.2. Segmentation Training

After creating this different convolutional neural net for image segmentation, the training itself is very similar to that of the binary classification model. We simply load up our dataset and dataLoader, and run through the dataset multiple times, calculating the loss, and backpropagating to update the neural net's weights and biases. For the training of our image segmentation model, we experimented with many different batch sizes and number of epochs run, but got our best results when we trained our dataset with 50 epochs with a batch size of 4.

5. Evaluation

5.1. Binary Classification

A fairly simple algorithm is used in order to evaluate how powerful the binary classifier for this project is. After the model is trained on the training dataset, each image from the testing dataset is then run through the model, with the model making predictions on whether the image contains a solar panel or not being made on each one. Since the testing dataset has ground-truth labels for each of these images, the percent of correct predictions can be calculated by counting the number of predictions this model made correctly, and then dividing by the total number of test images. After doing exactly this process, this model receives up to around an eighty-five percent accuracy rate. Figure 11 shows this project's binary classification predictions for one batch of images.

5.2. Image Segmentation

The algorithm for determining how well the image segmentation model performed is a bit different. Since this model is attempting to predict where solar panels are and not just if there is one or not, whether or not the model is "accurate" is more nuanced. It would be unproductive to calculate the percent of images this model predicted "accurately", since a model that predicts the class of the vast

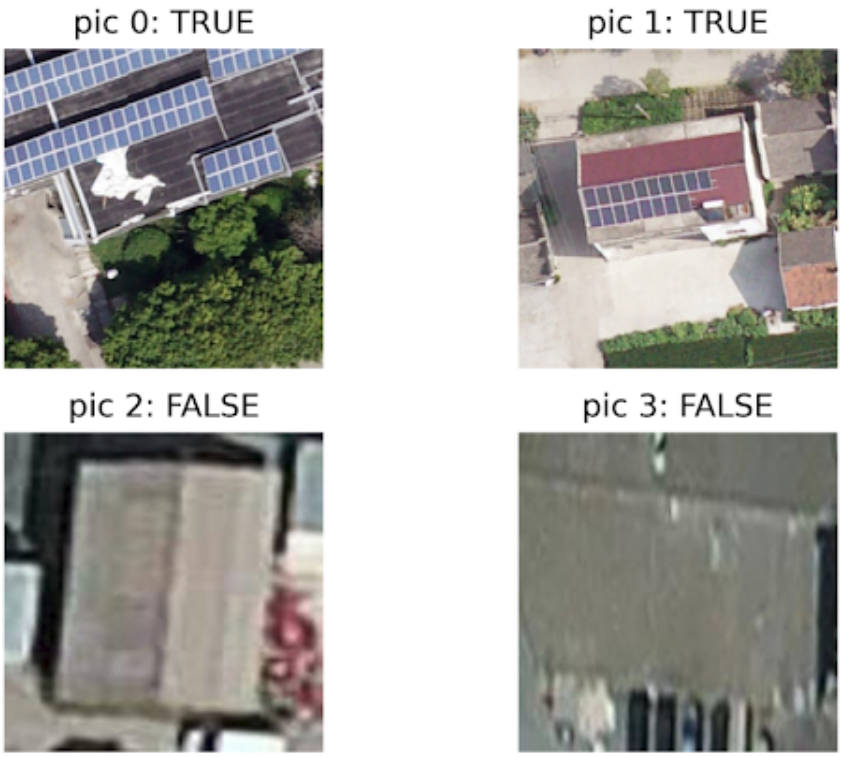

Figure 11: Binary Classification Predictions on One Batch

majority of pixels correctly should likely be considered a powerful model despite not predicting any images perfectly. Essentially, for this image segmentation problem a model can still be thought of as performing well even if it didn't predict the solar panels pixel-perfect. For example, Figure 3 can be thought of as showing a relatively accurate prediction despite not classifying every pixel correctly. Therefore, instead of finding the percentage of images that the model predicted "correctly", this project calculates the average loss of each image's prediction in the test dataset, and uses this as the measure of how well this model can determine where a solar panel is in an image. After training this image segmentation model to predict the pixels that are part of a solar panel and evaluating it on the test set, this model achieves an accuracy rate of around eighty-nine percent. Figure 12 shows the image segmentation model's loss over 50 epochs.

Figure 12: Image Segmentation Loss vs Epochs

6. Conclusion

Given how vital the need to switch to more renewable energy sources has become and the prevalence of solar panels as a source of renewable energy, there needs to be better methods of tracking the development of these solar panels. In order for this system of tracking to be as powerful as possible, it needs to have three qualities. First of all, it needs to be almost fully automated so that it can be run repeatedly, without any human-power, constantly getting the latest information and staying up to date. Secondly, this system cannot rely on incomplete or unreliable information. Finally, this system needs to be as accurate as possible. This project intends to take a step towards this goal of a powerful solar panel tracking system.

The best system for tracking the development of solar panel needs to be as automated as possible. In this project, the only manual aspect of running the models, is collecting the datasets from the region that the models need to be run on. This is a necessary human-powered task unfortunately, but one that cannot currently be bypassed. All of the classification and segmentation however, is fully automated, and requires no human-power. Therefore this project has taken a step towards this

goal of a better solar-panel tracking system by creating a process that is heavily automated, with very little human-interaction.

The best system for tracking the development of solar panels also cannot rely on incomplete or unreliable sources of information. Current options that attempt to track solar panel projects, such as the Puerto Rico Solar Map will never be a complete documentation of the solar panels in Puerto Rico despite still being a very useful, educational resource. The Puerto Rico Solar Map's reliance on people to volunteer their solar panel information is the cause of this. However, this project relies only on images to gather information, as opposed to people. As long as relatively recent aerial images can be obtained, most likely via satellite, this project will be able to run and make estimates of the solar panels in an area. Given that satellite images are taken frequently all over the world, a source of satellite images will be considered a reliable source of information. In addition, due to the nature of solar panels needing to be exposed to sunlight and therefore the sky, the vast majority of solar panels will be able to be seen in these satellite images. Therefore, satellite images will be considered a complete source of information about the solar panels present in a certain region.

Finally, a powerful system for tracking the development of solar panels needs to be as accurate as possible. While no necessary machine learning model will be one hundred percent correct all the time, this project attempts to attain the most accurate results as possible by creating powerful machine learning models, and experimenting with tweaking the parameters of the model in order to minimize testing loss. In addition, the datasets used as the input to run this project on may technically not be one hundred percent accurate, since even if all solar panels are visible in them, a few of them may be out of order or have some other deficiency, which our models would likely have no way of distinguishing. Despite these small caveats, the results achieved by training our models on the limited datasets we currently have, are enough to determine that our models have a significant degree of accuracy.

Therefore, this project has taken great steps towards creating a better solar panel development tracking system by creating a system that tracks solar panels with a high degree of autonomy, based on very reliable, complete, and accurate information, and high a credible degree of accuracy. Our

binary classification model was able to correctly classify a given image as either containing a solar panel or not with rates of up to around eighty-five percent. Our image segmentation model was able to predict where these solar panels existed in the image, with rates of up to around eighty-nine percent. However, much more work still needs to be done.

7. Future Work

In the future, this project can be expanded upon and further developed in order to make it more accessible and powerful. In order to make this project more powerful, perhaps the most important next step is to create better datasets. As mentioned previously, many of the limitations of this project likely come from limited quality datasets. While the dataset used for image segmentation was a very high-quality dataset, the other was much lower quality in terms of resolution. In addition, in order to get this second dataset fed into the neural nets, the images were forced to be resized into the dimensions of the other dataset, meaning that images were sometimes slightly warped or cropped in order to become the same resolution as the other dataset. Finally, datasets with a much larger number of images, would be tremendously helpful.

Another necessary step in the further development of this project is to have the program count the actual solar panels itself instead of only pointing out where they are. In addition, the program could also attempt to make predictions about the square area footage of the solar panels in the image. However, these goals require labeled datasets that weren't available for this project. Therefore, this is left as a next reference for future work to still be done. These goals would take this project to the next level and take another large step towards creating a better solar panel development tracker.

8. Ackowledgements

I would like to thank Dr. Kaplan for all of his help, guidance, and advice this semester and last summer.

9. Honor Code

This paper represents my own work, in accordance with university regulations.

- Lucas Gen

References

[1] "Deep-learning: Rooftop type detection with keras and tensorflow." [Online]. Available: http://fractalytics.io/rooftop-detection-with-keras-tensorflow

[2] "Loss functions." [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[3] "Neural network diagram." [Online]. Available: https://tex.stackexchange.com/questions/497859/neural-network-diagram

[4] "Power outages plague puerto rico despite luma takeover." [Online]. Available: https://www.nytimes.com/2021/10/19/us/puerto-rico-electricity-protest.html

[5] "Puerto rico solar map." [Online]. Available: https://www.puertoricosolarmap.org

[6] "Puerto rico territory profile and energy estimates." [Online]. Available: https://www.eia.gov/state/?sid=RQ

[7] "Spatial uncertainty sampling for end-to-end control." [Online]. Available: https://www.researchgate.net/figure/Non-convex-optimization-We-utilize-stochastic-gradient-descent-to-find-a-local-optimum_fig1_325142728

[8] "Two major power outages in a week fuel fear in puerto rico — and memories of hurricane maria." [Online]. Available: https://www.washingtonpost.com/nation/2021/06/18/puerto-rico-power-outages/

[9] "U-net: Training image segmentation models in pytorch." [Online]. Available: https://www.pyimagesearch.com/2021/11/08/u-net-training-image-segmentation-models-in-pytorch/

[10] "Unet." [Online]. Available: https://en.wikipedia.org/wiki/U-Net

[11] "Weekend project: Detecting solar panels from satellite imagery." [Online]. Available: https://towardsdatascience.com/weekend-project-detecting-solar-panels-from-satellite-imagery-f6f5d5e0da40

[12] "When to use mlp, cnn, and rnn neural networks." [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/#:~:text=The%20benefit%20of%20using%20CNNs,important%20when%20working%20with%20images.

[13] J. Hou, Y. Ling, and L. Yujun, "Multi-resolution dataset for photovoltaic panel segmentation from satellite and aerial imagery," Aug. 2021, Data document can refer to the preprint https://essd.copernicus.org/preprints/essd-2021-270/. Available: https://doi.org/10.5281/zenodo.5171712

[14] L. Kruitwagen *et al.*, "A global inventory of photovoltaic solar energy generating units," *Nature*, vol. 598, no. 7882, pp. 604–610, 2021.

[15] A. Lou, S. Guan, and M. H. Loew, "Dc-unet: rethinking the u-net architecture with dual channel efficient cnn for medical image segmentation," in *Medical Imaging 2021: Image Processing*, vol. 11596. International Society for Optics and Photonics, 2021, p. 115962T.

[16] J. Yu *et al.*, "Deepsolar: A machine learning framework to efficiently construct a solar deployment database in the united states," *Joule*, vol. 2, no. 12, pp. 2605–2617, 2018. Available: https://www.sciencedirect.com/science/article/pii/S2542435118305701