**Abstract**

**Information about rooftop solar photovoltaic (PV) installations can be quite useful for decision-makers in local energy companies and government. To generate such information accurately and efficiently, researchers have proposed ways to collect solar photovoltaic images by high resolution color satellite imagery and further process these images by computer vision algorithms. However, current computer vision algorithms still showed limited accuracy in classifying suspicious input images.**

**Deep convolutional neural network (CNN) has demonstrated its impressive classification and object detection ability on ImageNet benchmark since 2012. Among different CNN models, Resnet is a representative model proposed in 2015. We fine-tuned a pre-trained Resnet with only 1000 images. The fine-tuning process takes less than five minutes and model achieved around 99% classification accuracy, ranked 1st in Kaggle competition. Compared with human-labeled prediction with 97% accuracy, our algorithm is proved to exceed human performance on this benchmark.**

**Introduction**

Image recognition has been of great interest data scientists and machine learning experts for decades now, with uses ranging from handwriting recognition to reverse image searching. The ability for a computer to mimic a human’s ability to visually classify images using learning patterns has been challenging, although much progress has been made in recent years. Face recognition can be found on even entry-level digital cameras, and we can speed through tolls at 60 mph with confidence that a camera recorded and parsed our license plate number. There is much value in these classification problems, and learning and understanding the basics of these learning algorithms is essential in furthering one’s knowledge of machine learning.

In this relatively basic project, a binary classifier was designed to detect solar panels in satellite imagery. A variety of training images were given, covering both classes: contains a solar panel and does not contain a solar panel. The goal of this task was to accurately identify whether a test image included a solar panel or not. Although only binary classification, this question has large implications, if implemented accurately. Automating the process of collecting solar panel data could be hugely beneficial to the energy industry, especially if used in conjunction with other data. Satellite imagery is enormously abundant, and can be matched to location data or addresses. For example, weather location can be correlated with positive solar panel matches to determine the best locations for solar panel placement. Utility companies can use satellite data to determine which customers are using solar panels, without sending an employee to investigate. Finally, as the use of solar panels rise, new data relating to the use of panels by geographical location can be collected, and further research into refining the algorithm to determine size and capacity of the panels can be used to better understand and predict renewable energy trends.

**Background**

<https://www.researchgate.net/publication/300416092_Automatic_solar_photovoltaic_panel_detection_in_satellite_imagery>

<https://www.sciencedirect.com/science/article/pii/S0031320304000548>

<https://www.sciencedirect.com/science/article/pii/S0924271609000884>

<https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/>

**Data:**

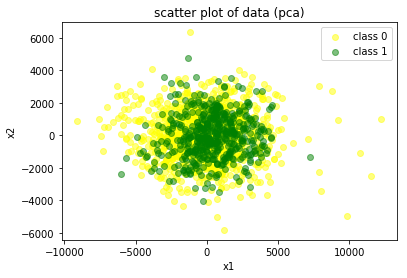
**                   **

**                    **

There are no obvious patterns in the location, number and size of solar panels. Solar panels may be at any location in the images, and their size varies from one to another. Some pictures contain only one solar panel, while the others may contain more than one solar panel. The background of solar panels is highly variant. Furthermore, whether the complete solar panel is contained in the image is random. Hence, it is difficult to detect the existence of solar panels based on these features.

Every sample in the training set is a RGB image with the size of 101×101×3. To visualize the whole training set, we applied principal component analysis to these images and set the number of principal components to be 2. It is obvious from the scatterplot the two classes do not appear separable. The solar panels are not the principal components of the images. Moreover, the two classes are not of the equal size, the number of Class 0 is much higher than that of Class 1. For this task, simple models such logistic regression, KNN and plain convolutional neural networks are unlikely to extract enough features from the images. A more complex model is necessary. We need a large number of feature detectors to extract features from the images.

The size of training set is small, which may cause overfitting. Hence, data augmentation is needed to increase dataset size. In addition, we can see from the images, most of the solar panels are not near the edges of the pictures, so center cropping during data augmentation may not significantly influence the data. The images from the training set are not highly different from the ones of ImageNet. Hence, we can consider applying fine tuning when designing our algorithm. We can train a pretrained neural network on our dataset.

****

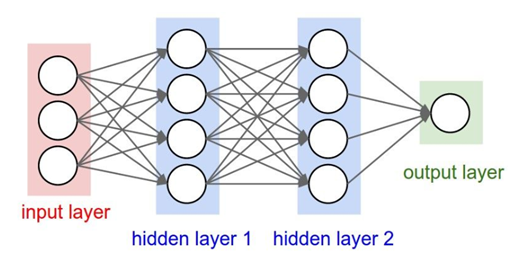
**Methods:**

**Theoretical Framework:**

**Data Augmentation**

Neural networks need a large amount of training data to achieve good predictive performance. Overfitting may be caused due to small training datasets, which leads to high testing error ratio. To avoid this, data augmentation is used before training to increase the size of training set. Common data augmentation methods contain adding noise, flipping horizontally or vertically, rotation and cropping.

**Neural Networks**



(“2-hidden-layer Neural Net”)

Neural Networks take a vector from the input layer, and transform it through several hidden layers, which are made of neurons. For classification tasks, the last layer (output layer) represents the class scores. Neurons are fully connected with all neurons in the previous layer. Outputs from the previous layer are inputs for the current layer with associated weights and biased, and they are processed by the activation function to introduce nonlinearity for the network. Neurons in the same layer are independent with each other.

**Convolutional Neural Networks**

A convolutional neural network is a class of deep, feed-forward artificial neural networks that are composed of several layers. There are three main kinds of layers that are stacked to form CNN: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

Convolutional Layer:

It is the core building block of CNN. Convolutional layers apply convolution filters on the input, then pass the result to the next layer. These convolutional filters can detect features of the image.

-Each convolutional neuron processes data only for its receptive field.

This is inspired by the fact that cortical neurons of animals respond to stimuli only in a limited region. Furthermore, when processing images, it is impractical to connect the neuron to all neurons in the previous layer, since the number of parameters will become extremely large. So, every neuron in CNN is only connected to a subset of the input, which is different from original neural networks. Local connectivity reduces the number of parameters and makes it possible to train a CNN.

-All neurons in the same feature map share the same weights.

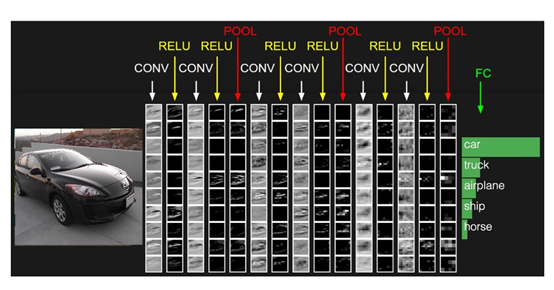
It is assumed that feature detectors are useful for the whole image regardless of the locations. For one specific feature, only one convolutional filter is used to extract this feature. Parameter sharing helps to decrease the number of parameters dramatically.

Pooling Layer:

Pooling layer is another kind of building block that is periodically inserted among convolutional layers. Max pooling, the most popular kind of pooling, takes the maximum value in every sub region. Pooling layer progressively decreases the dimensionality of data but does not change the depth dimension. It makes the network more invariant to small changes in the initial input and controls overfitting.

Fully-connected Layer:

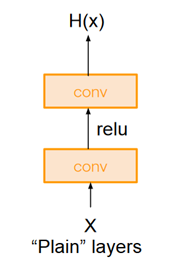
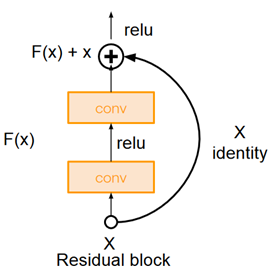
At the end of CNN is fully-connected layer as in ordinary neural networks. Neurons in fully-connected layer are fully connected with outputs from the previous layer.



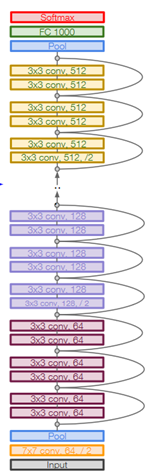
(Typical architecture of CNN)

**Residual Network:**

Deeper neural networks are more difficult to train. Meanwhile, accuracy of the neural network encounters degradation problem when continuously increasing the network depth. Residual network is proposed to solve this problem. Residual networks are easier to train and reveals better accuracy. It uses network layers to fit a residual mapping rather than directly trying to fit a desired underlying mapping. As shown in the graph below, in CNN, H(X) is learned directly, while in residual networks, H(X)-X is learned.

**                           **

ResNet architecture:



Residual blocks are stacked to form residual networks, and every residual block has two 3 x 3 convolutional layers. Batch normalization is applied after each convolutional layer to accelerate training process and make the network less sensitive to initialization. Periodically, the number of convolutional filters is doubled, and at the same time, the data is down sampled to half in each dimension. Additional convolutional layer is added at the beginning of the ResNet. Stochastic gradient descent with momentum is applied in back propagation. In each iteration, parameters are updated:

https://lh6.googleusercontent.com/4LVBWY4RfVIGEdh9Ue3a4eKol5mvYlGsZogiwsI4IdGLerdj0af4CZNKVo-NWwomUL26lL4gmSpPUCci5xd6yVl2lwxesOnkdzLsilYGCjNohtMrOZtnrCzGW_YNlwNCqy8yOZv7

Momentum helps to converge in the right direction faster and reduce oscillation.

Fine tuning:

In practice, training a deep neural network is time-consuming because of the huge number of parameters and limitation of computation speeds. The ResNet can be pretrained on a large dataset, such as ImageNet, instead of being trained from scratch.

ImageNet is a large image database that contains more than fourteen million URLs of images. The images of solar panel detection task are not significantly different from those of ImageNet. The ResNet pretrained on ImageNet has already learned features that are relevant with our task. We can use a pretrained RetNet to fit the training data of solar detection to acquire better performance.

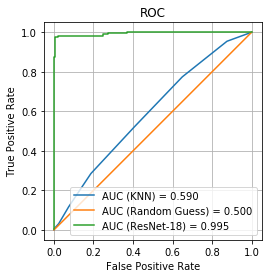
We use PyTorch, an optimized tensor library for deep learning using GPUs and CPUs, to implement our algorithm.  
  
Preprocessing:  
The whole training set is divided into training set (1000 pictures) and validation set (500 pictures) to evaluate predictive performance of our model. In Preprocessing, images from training set are randomly resized and cropped to 224×224, horizontally flipped randomly, transformed to a tensor, then normalized with mean of [0.485, 0.456, 0.406], and standard deviation of [0.229, 0.224, 0.225]. The image size of 224×224 matches with the parameters of ResNet-18 in PyTorch. The mean and standard deviation of normalization is calculated based on data from ImageNet.

https://lh5.googleusercontent.com/UbvwI9q4x6pXUFHLL_AO_nIe4qYcCzxkJmoEJDQdPPnrMRHBTHQBOuERX7qPx0EjgOwMqnXGMy89EMjBr3UfNDU0atVoWlCETCXqmQKPeUt-AZchKXYZPHJCtbS1Yl0a0nkzy9K5

**Model:**

We use ResNet-18 in PyTorch to train the data, which is already pretrained on ImageNet. The data is similar to that of ImageNet, so we can assume features learned from ImageNet are relevant with those in our images of solar panels. We remove the last fully-connected layer of ResNet-18, since it outputs 1000 class scores for ImageNet task, but we need a binary classification. The other layers of ResNet-18 remain the same. We take cross-entropy loss as the loss function for this binary classification task. In backpropagation, stochastic gradient descent with momentum is applied. Batch size is set to 4, momentum is set to 0.9, and learning rate is set to 0.001. Five ResNet-18 models are trained, and the final labels of validation set are determined by these five models together. One image is classified as Class 1 if more than two models classify it as Class 1, otherwise it is labeled as Class 0.

**Results:**



The whole training data is divided into training set and validation set to measure the predictive performance of models. ROC and AUC of ResNet, KNN and random guessing is shown above. It is obvious that ResNet significantly outperforms other models, among which KNN is better than random guessing. The AUC of ResNet-18 is almost twice as much as that of KNN and random guessing. Besides, the ROC of ResNet-18 is almost ideal for our detection task. For the k-nearest-neighbors classifier, k is set to 7. The probability of the class of one observation is determined by its seven nearest neighbors. This method does not extract enough information to classify, and its result is only a little better than random guessing. Simple models, like KNN, are easy to implement, however, they cannot perform well in this detection task due to the complex features hidden in images. For the training set, the two classes are imbalanced since the number of class 0 (995) is twice as much as that of class 1 (505). Assigning all observations to class 0 will yield a higher accuracy ratio than that of KNN, however, ROC performance of this method is poor.

For our validation set, only five pictures are wrongly classified. These five pictures are shown below, with five examples that our model works well on.

In the five wrong classifications, the first four of them are TypeⅡerrors. Perhaps the solar panels in these images are too small to be detected by our algorithm, or maybe they are cropped out of the image during the preprocessing. To improve our model, we may consider applying more methods to augment our training data, which can introduce more variation of our data. We also can use a deeper neural network to train the data and regularization can be considered to avoid overfitting.

**Conclusion:**

Our solution for this task is transfer-learning + CNN. It proved to work well on Kaggle competition0.

The most challenging part is that our dataset is not large enough.

Our further research would be appling CNN on some pictures that are not natural images and have more than 3 channels.

Here are some key takeaways and experiences we hope to share with our readers.

1. Comment on CNN:

Just like the in the Large Scale Visual Recognition Challenge (ILSVRC), our result has also demonstrated the impressive performance of CNN in images classifications. This algorithm has kept its dominant position in computer vision algorithms for several years. Thus, for supervised learning on natural images, CNN should be the first algorithm to consider.

Personally, I believe there are two most charming strength of CNN: strong feature extraction ability and complex enough model. The first strength, automatic feature extraction by convolutional+pooling layers, greatly save the effort on manual feature selection when dealing with natural images. The second strength is that it is a complex enough model to fit on huge amount of data and learn extremely complex hidden functions. (1000 cases on ImageNet). Furthermore, there are many good tricks to avoid overfitting in such a complex model, like L1/L2 regularization, dropout and batch normalization.

2. Why PCA+SVM is not a preferable method:

Personally, I think PCA+SVM may not be a suitable method for this task. Firstly, the performance of PCA+SVM is not comparable with CNN on MNIST, a small dataset. What can be worse is that the target in this competition, a solar PV in image, is not likely to be a ‘major component’ in a image. PCA can view target as some ‘unnecessary’ detail of a image.

Compared with MNIST, it is more like an object detection task, even though it is doing binary classification. CNN has proved to be effective in objection detection task.

3. Coping with limited data:

Only 1500 images are given in this Kaggle competitions. It is a small-sized dataset compared to MNIST with about 50000 images. This can be bad news for a complex model like CNN. Thus, in this case, two options are definitely worth trying: data augmentation and transfer learning.

Data augmentation can be helpful in improving model robustness. For example, simply by rotating, flipping images horizontally or vertically, 2\*2\*2 = 8 different images are generated by one single input image. Moveover, random scaling and cropping can further increase the variety in input images. One important thing to notice is that such augmentation can be done randomly during training, rather than generated during preprocessing before training actually starts. Since it would occupy a lot of storage to store those pre-augmented data and loading those data would make training process even slower. Instead, cropping / flipping operations during training can be super fast on numpy array.

Transfer learning, also named fine-tuning, is also a very good choice, especially when training data is limited. Since the model pre-trained on ImageNet can already efficiently capture many features in natural images, performing gradient descent on that model would converge much faster and to a much better local point.

In addition, there are two choices in fine-tuning process: either fine-tune the weights on all layers, or fix those previous layers and only fine-tune last a few layers. (especially fully connected layers) There has been a lot of discussion in deep learning community on which scenario each method should be used. My personal experience on this task is that fine-tune the weights would give a slightly better performance.

4. About speed:

GPU is usually an essential for a faster training and testing during deep learning. On one hand, this made a higher request for users of CNN, but on the other side, it can be much faster than some other machine learning algorithms that do not scale well with GPU.

For example, a SVM with RBF kernel typically has O(n^2) computational complexity, which can be super slow for large training data. GPU is not common to accelerate SVM. But CNN can cope with large amount of data pretty well. The training time should be linear in this case O(n).