Solar Panel Segmentation

Weicheng Dai

November 2021

1 Introduction

Solar panels are important for people these days, because it can generate heat as well as power. In some cities, however, there are few solar panels installed among residential areas. The number and the area of solar panels are good measurements to find the financial situation and human residential rate, as well as the power consumed in certain area. In this article, some methods are applied to segment solar panels on satellite images. The result is that UNet[3] has the highest intersection of union, over 45%. The github link of this report can be found here [1].

2 Related Work

2.1 Segment-based Classification

Alex Halcomb [2] proposed a segment-based classification. First, an image is segmented to several pieces based on color, circleness, et al.. Then, for each of the segmentation, features are extracted in terms of color space, perimeter, area. After that, logistic regression and random forest algorithm are applied to classify if an area belongs to a solar panel or not. The result of accuracy was very good, over 90%.

However, the accuracy of segment-based classification requires that the segment was very precise. In some cases this is very useful, e.g. the object contains components of similar color. In other cases it is not. Think of a white dog playing in snow.

2.2 Multi-task Segmentation

He K.[1] proposed mask R-CNN to segment multiple objects. First, models related to regions of interest are trained that contain exactly one object, then, another model is trained related to classify what that object is. Two pipelines are trained in this article, and it requires time and computing resources. The accuracy was fair. It reaches 40% in both medium items and large items.

Mask R-CNN is good for segmenting multiple items, but it requires resources. In our work, only one item, which is solar panel, needs to be segmented, so it is also not a perfect match.

3 Dataset

Satellite images of solar panels taken by Bradbury K. [3] are used as the dataset in this article. It contains about 500 satellite images with about 20000 solar panels, labeled in coordinates or vertices. The images are taken in 4 cities: Fresno, Stockton, Oxnard, and Modesto.

The first thing to do was to convert coordinates into masks. matplotlib[4] has a function called "Polygon", which helps draw polygons on an image. Therefore, for each solar panel sample, $128 * 128$ size of the image was cut out to form the X data. Correspondingly, $128 * 128$ size of the mask was also cut out for Y. This work was very time consuming and takes about 12 hours. The codes are in *dataset.ipynb*.

[1] https://github.com/WeichengDai1/Solar-panel-segmentation.git

[2] https://alexhalcomb.github.io/

[3] https://doi.org/10.6084/m9.figshare.c.3255643.v1

[4] https://matplotlib.org/

The dataset was split into two parts, training and testing. Specifically, images from Fresno and Modesto are used as training data, with a sum of 15292 items. Images from the other two cities are used as testing data, with a sum of 4141 items. The codes are also in *dataset.ipynb*. From here because the work was time consuming, so I start using .py files instead of .ipynb files.

To increase the number of training set, and the robustness of model, data augmentation was applied on the training dataset. Specifically, each item is rotated 90°, 180°, and 270°, for both the image and the mask. The codes are in *dataAug.py* and *dataAug_Y.py*. The dataset generated in this part is called $'X\_train\_aug.npy'$ and $'Y\_train\_aug.npy'$.

4 Methods

Three methods are applied on the training session. They are image-based k-nearest neighbour, fully convolutional network (FCN)[2], and UNet[3]. Different from Alex [2], FCN and UNet are both based on pixel, instead of segmentation.

The metric to evaluate a model is called intersection over union (IOU). Let P denote the prediction of a model, and let T denote the true label, the definition is below. Note that a minor number of $1 * e^{-6}$ is added to avoid being divided by zero.

$$IOU = \frac{P \cap T}{P \cup T + 1 * e^{-6}}$$

In FCN and UNet, the optimizer that is used is the *Adam* with weight decay of $1 * e^{-6}$. Learning rate was initially set $1 * e^{-4}$, with a step learning rate which changes the learning rate into 1/10 every 30 epochs. The loss function was the $BCELoss$, aka binary cross entropy loss. I have tried 100 and 500 as epoch number, but it seems that there is no big difference.

4.1 K-nearest neighbour

Since the size of training data is quite large, it will be very burdensome to apply pixel-based k-nearest neighbour. Thus, image-based k-nearest neighbour is applied. Different from pixel-based, which compares a pixel with all of the pixels existed in all images, image-based k-nearest neighbour compares the Euclidean distance between test images and all training images. In this part, k can be switched from 3, 5 and 7. After the k-nearest images are found, the union of their corresponding masks are shown as the prediction of a given image. The result is in Table 1. As is seen in the table, the result is not so good, with the best IOU of 25.02% when $k = 7$. The code is in *knn.py*.

Table 1: Result of image-based k-nearest neighbour

k	IOU
3	17.63%
5	21.90%
7	25.02%

4.2 Fully Convolutional Network

Fully convolutional network[2], also known as FCN, is a network that contains both an encoder and a decoder. In the encoder part, convolutional layers and pooling layers are used to extract features on the input image and sample the important ones. In the decoder part, convolutional transpose layers are used to map the features to predictions. The structure of FCN is in Figure 1. Different from the original structure, I used several convolutional transpose layer to avoid miss of tiny solar panels on the image. Training 100 epochs of FCN takes about 10 hours with 3 GPU. The result of training and testing is in Figure 2. As is seen from the data, the IOU is about 20%, which is not a very good result. The code of FCN is in *FCN.py*.

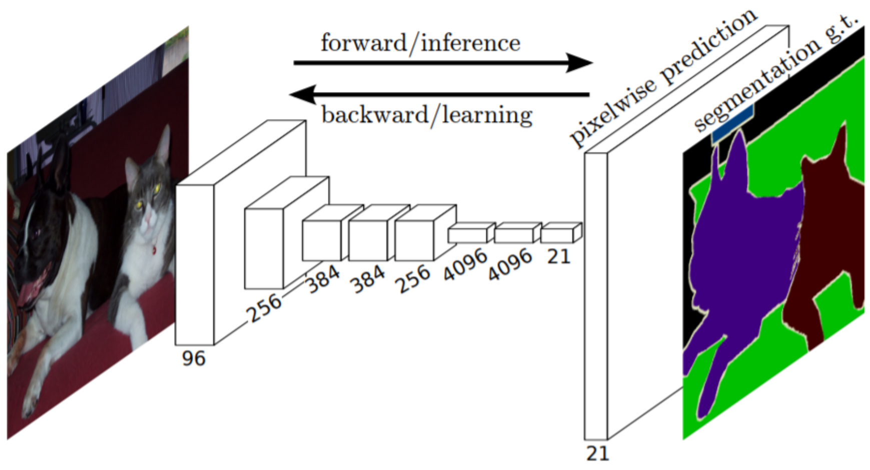


Figure 1: FCN structure

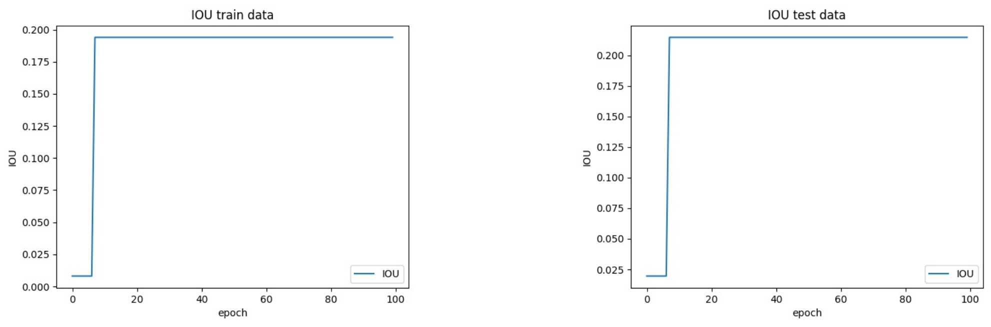


Figure 2: training and testing IOU of FCN

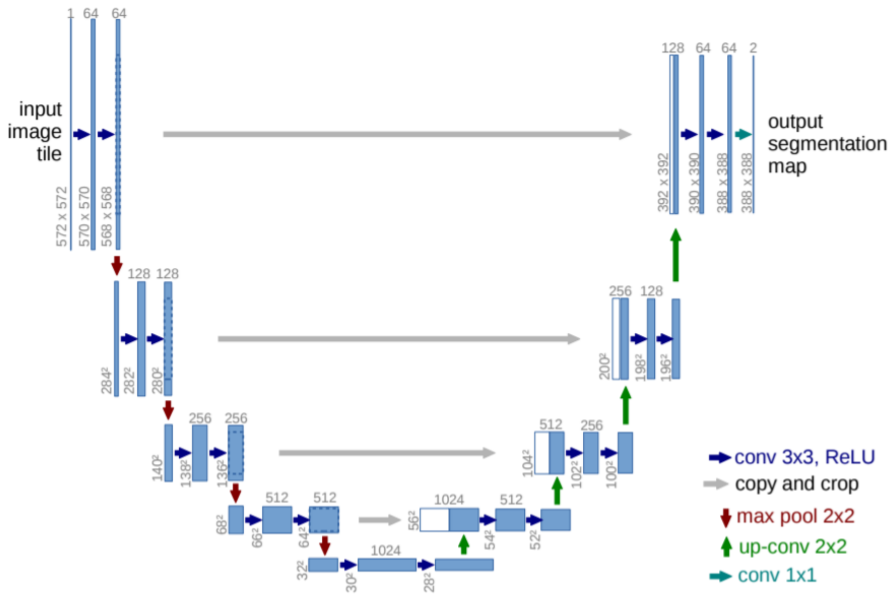


Figure 3: UNet structure

4.3 UNet

UNet[3] is named after the structure of its model3, which is similar to the letter 'U'. Similar to FCN, UNet also has an encoder and a decoder, but the difference is that UNet takes some outputs of the encoder and concatenates them with the outputs of the decoder. In this experiment, since the input image and the output mask have exactly the same width and height, padding should be added to ensure that the size never changes. Training 100 epochs of UNet takes about 15 hours with 3 GPU. The result of training and testing is in Figure 4. As can be seen, the IOU of UNet reaches 45%. The code of this part is in *unetmodel.py*.

5 Results

Comparing the three models that are applied in this experiment, UNet has the highest IOU. However, it seems the IOU of 45% is not a very high value, but as I tried to modify some of the hyperparameters, and tried to add some dropout layers, the result is still the same. Therefore, I tried to show some of the predictions of the model, as is shown in Figure 5. After visualizing some results in testing dataset, conclusion can be drawn that UNet model is quite effective in this task, and IOU is merely a criterion. The code of this part is in *imageShow.py*.

6 Discussion

In this paper, Three methods are applied on the segmentation of solar panels. As a result, UNet reaches the highest IOU of 45%. Compared with Mask R-CNN[1], which has a mAP of 43.%2 for medium objects and 22.1% for small objects, UNet is fair in this task.

As for the other models, k-nearest neighbour may gain better performance if it is pixel based, because that is more plausible, but it is also very time consuming. and actually the FCN seems stop learning after certain epochs, because the curve is a straight line.

Also, as the rise of transformer[4] in many tasks, it seems in segmentation, transformers may also have better performance. Sadly I have not understood the theory in transformer, so I have not tried this model.

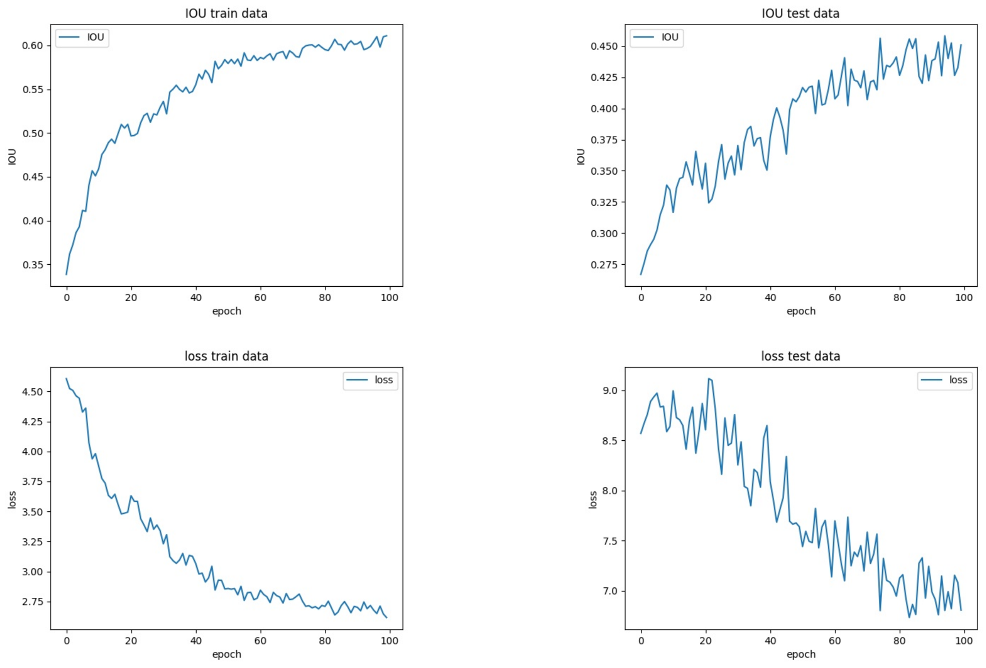


Figure 4: IOU and loss in training and testing of UNet

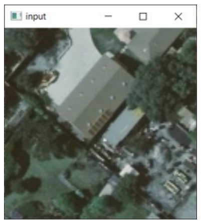
References

[1] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn, 2018.

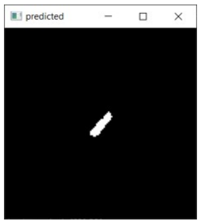
[2] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation, 2015.

[3] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015.

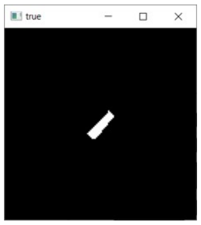
[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.



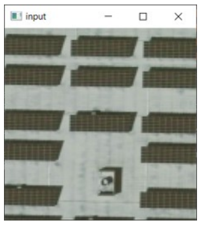
(a) input 1



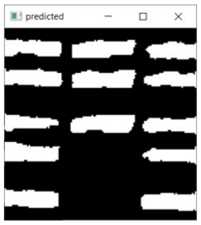
(b) predict 1



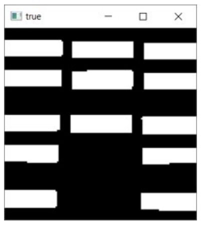
(c) true 1



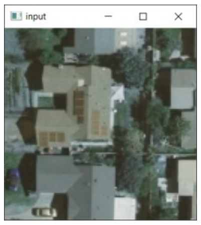
(d) input 2



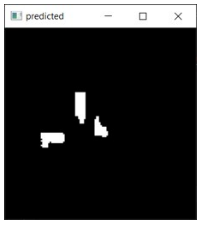
(e) predict 2



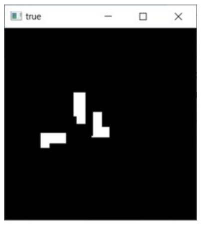
(f) true 2



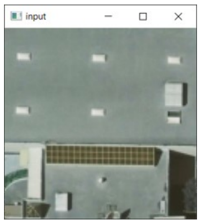
(g) input 3



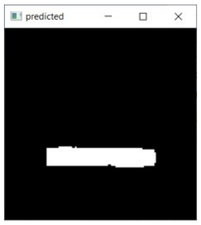
(h) predict 3



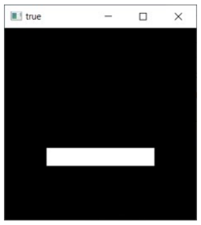
(i) true 3



(j) input 4



(k) predict 3



(l) true 3

Figure 5: Result of UNet