Kaggle report: Solar PV in Aerial Imagery

## NN-method

### General approach

The training data provided was firstly divided into training and validation sets by randomly select a subset for validation during model training. After the model results appeared to be satisfactory, the whole training data given was used for training the model then applied to test set for obtaining final scoring metrics.

### Data augmentation

It is very easy to overfit using the limited training dataset given. Data augmentation techniques were therefore used to artifically expand the size of the training set by creating tranformed versions of images. A transformation pipeline provided from Pytorch libarary was used. It firstly random sized crops to 224 x 224 which matches parameters of ResNet18 in PyTorch. Then we performed random horizontal flips, followed by tensor conversion and normalization which is based on data from ImageNet for easier processing.

There were slight difference with respect to the validation and test set as they were not in need to augmentation. As most solar panels concentrated in the center of the picture, in order to increase accuracy, the images were scaled larger up to 256 and center cropped to match the same size 224 x 224, then followed by the same procedures of tensor conversion and normalization.

### Data loading

`ImageFolder` from \*\*torchvision\*\* is used for data loading, which assumes the images are orgainzed in a way where within one main folder, there are subfolders whose names correpsond to class labels. Script `generateDataset.py` was used to automatically separate true and false images into two sepearte folders within the root directory and one subfolder was created under testing folder as well in order to feed into such framework. Combined with `torch.utils.data.DataLoader`, we were able to load multiple samples parallelly in a batch size of 30. Batches between epochs will not be look alike by shuffling the order in which examples are fed to the classifier to make the model more robust.

### Transfer learning and fine tuning

The model used was ResNet-18 from PyTorch, which is already pretrained with ImageNet, one of the most popularly used datasets composing of millions of classified images. The ImageNet contains one thousand classes and satellite imageries are also included. It can be assumed that the features learned from the pretrained network are closely related with the provided solar panel satellite images and turned out to be great result withi high prediction accuracy. Instead of predicting 1000 class scores from the pretrained ResNet model, the last fully connected layer was replaced with a new one that composed of two binary classes only. Here the criterion was set to be cross-entropy loss which the network would try to minimize with respect to the weights. The optimizer during back propagation was chosen to be stochastic gradient descent. A momentum was applied which can gives a larger effective learning rate along the directions of low curvature. It uses one row of data and update the weights each time and is more likely to reach a global minimum with higher flexibility.

## Mechanisms

### Neural Networks

#### Architecutre

Neural networks take input vectors in the input layer, transformed them through N-hidden layers which comprised neurons that are fully pairwise connected between adjacent layers in an acyclic graph. Neurons within a single layer are independent of one another. The outputs of the neurons in one layer become inputs to the neurons in the next layer with associated weights and bisaes. They are processed by the activation function to introduce nonlinearity to the network. In the end, the output layer produces the classification scores for each class.

A close up of text on a white background

Description automatically generatedA picture containing building

Description automatically generated

Figure. Single neuron (left), 2-layer neural network (right)[[1]](#endnote-1)

#### Activation function

The activation function evaluates the signal responses and determines whether or not to activate the neuron given the input. One of the most commonly used activation function is Rectified Linear Unit (ReLU) $f(x) = max(0,x)$, which simply threshold a matrix of activations at zero. It can greatly accelerate the convergence of stochastic gradient descent compared to sigmoid or tanh function.

#### Backpropagation

The weights and biases are updated by assessing the error of the network by backpropogation, which is a recursive application of the chain rule.

The training process of ANN works like the following:[[2]](#endnote-2)

* Randomly initialse weights to small numbers close to 0
* Input first observation of the dataset in the input layer, each feature in one input node
* Forward-propogation from left to right: The neurons are activated in a way that the impact of each neuron’s activation is limited by the weights. Propogate the activations until getting the predicted results
* Compare the predicted result to the actual label. Measure the generated error
* Back-propagation from right to lft. The error is back-propagated. Update the weights according to how much they are responsible for the error. How much the weights are updated is decided by the learning rate
* Repeat all the above and update the weights after each observation/only after a batch of observations.
* It makes an epoch when the whole training set has passed through the ANN. Redo more epochs

### Convolutional Neural Networks (CNN)

There are three main layers in a CovNet, Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

The first layer in a CNN is a Convolutional layer where each convolution filter (or feature identifiers) detects features within its receptive field and output feature maps as the next input. As it is impractical to connect neurons to all neurons from the previous layer, each neuron is connected to only a local region of the input volume (effectively the filter size),[[3]](#endnote-3) each computing a dot product between their weights and a small region they are connected to in the input volume.[[4]](#endnote-4)

The pooling layer performs a downsampling operation along the width and height, which is inserted periodically in between successive Conv layers. By using MAX function on every depth slice of the input (max pooling), it progressively reduces the dimentionality and also helps to control overfitting.

In the end, the fully-connected layer computes class scores where neurons are fully connected to all activations in the previous layer as seen in a regular Neural Network.

|  |  |  |
| --- | --- | --- |
| page11image797040 | | |
|  |  |  |

Figure. CNN Architecture Example

### ResNet

Developed by He et al., residual network is currently by far the state of art CNN models and has shown outstanding performance in image classification tasks.

A close up of a piece of paper

Description automatically generated

Figure. Evaluation of different NN architectures[[5]](#endnote-5)

It features special skip connections and heavy usage of batch normalization.

It was hypothesized that it is easier to optimize the residual mapping than to optimize the original mapping.

Residual blocks are stacked together forming a residual network. Within each residual block, there are two 3x3 convolutional layers. After each convolutional layer, batch normalization is applied, reducing the network’s sensitivity to initialization.

A close up of a logo

Description automatically generated

Figure. Residual learning: a building block[[6]](#endnote-6)

## Non-NN method

### Histogram of Oriented Gradient (HOG)

* Divide image into 8x8 pixel regions
* Within each region quantize edge direction into 9 bins where dominant edge direction of each pixel is computed
* Quantize those edge directions into several buckets

### Support Vector Machine (SVM)

### References

1. Cs231n [↑](#endnote-ref-1)
2. Udemy ML [↑](#endnote-ref-2)
3. <https://www.quora.com/What-is-a-receptive-field-in-a-convolutional-neural-network> [↑](#endnote-ref-3)
4. <http://cs231n.github.io/convolutional-networks/>

   <https://stanford.edu/~shervine/blog/pytorch-how-to-generate-data-parallel>

   <http://iamtrask.github.io/2015/07/27/python-network-part2/>

   <https://towardsdatascience.com/comprehensive-introduction-to-neural-network-architecture-c08c6d8e5d98>

   <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/> [↑](#endnote-ref-4)
5. <https://www.semanticscholar.org/paper/Evaluation-of-neural-network-architectures-for-Canziani-Culurciello/690ebe31326fec38da40e838509bd9a3482f6c11> [↑](#endnote-ref-5)
6. <http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf> [↑](#endnote-ref-6)