Applying Deep Learning in Street View Imagery for Environmental Health Research

AN ABSTRACT OF THE PROJECT OF

Yifang Zhang for the degree of Master of Science in Computer Science presented on November 29, 2017.

Title: Applying Deep Learning in Street View Imagery for Environmental Health Research.

Abstract approved:

Dr. Lizhong Chen

Urban green space is associated with multiple physical and mental health outcomes. Several benefits of green space, such as stress reduction and attention restoration, are dependent on visual perception of green space exposures. However, traditional green space exposure measures do not capture street-level exposures. In this project, we apply deep learning models to measure green space in street view imagery. We train Faster R-CNN model on PasadenaUrbanTree dataset and equip it with the ability to detect trees, which is then used to count the number of trees in the test images. We also employ a PSPNet model that pretrained on ade20k dataset to do semantic segmentation on street view images and compute the portion of green space. Combining the outcomes from object detection and semantic segmentation, the green space in street view imagery can be measured quantitatively.

Applying Deep Learning in Street View Imagery for Environmental Health Research

by
Yifang Zhang

A PROJECT

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented November 29, 2017
Commencement June 2018

Master of Science project of Yifang Zhang presented on November 29, 2017

APPROVED:

___

Dr. Lizhong Chen, representing Electrical Engineering and Computer Science

___

Director of the School of Electrical Engineering and Computer Science

___

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

___

Yifang Zhang, Author

ACKNOWLEDGEMENTS

I would first like to thank my Major Advisor Dr. Lizhong Chen of EECS Department at OSU. The door to Dr. Chen's office was always open whenever I ran into trouble or had a question about my project or research. He consistently allowed me to do my own work, but steered me in the right direction whenever he thought I needed it.

I would like to thank my Committee Members Dr. Mike Bailey and Dr. Amir Nayyeri for constant support and guidance. Dr. Mike Bailey's Courses such as Computer Graphics help me to create foundation for computer vision; Dr. Amir Nayyeri's course Algorithm helped me to think about algorithm in a different way.

Finally, I must express my very profound gratitude to my Parents for providing me unfailing support and continuous encouragement throughout my years of study, and through process of development.

Yifang Zhang

TABLE OF CONTENTS

Page

1. Introduction

1.1 Introduction on deep learning

Deep learning has become prevailing today and has led to a series of breakthroughs on many complicated problem, such as object detection, semantic segmentation, and natural language processing. Deep learning is an approach for Artificial Intelligence which is intelligence displayed by machines. A major challenge for artificial intelligence is to handle problems that are easy for human to solve but hard for human to describe formally, such as understanding speech or recognizing objects from images, which can be solved by human intuitively, and which are difficult for computer systems. Machine learning provides a solution for this kind of problem, which allows algorithms to extract patterns from features and to learn knowledge. For example, the logistic regression, which is a basic machine learning algorithm, can do classification on images after being properly trained by training data. The key is that some patterns of the training data are extracted and assembled into the parameters of the algorithm during the training process. However, it is often the case that the performance of such simple algorithm enormously depends on how we represent the features. Deep learning, as a type of machine learning, solves this problem by representing complex concepts in terms of other, simpler concepts, and often gets higher performance when it is applied directly on raw data. Deep learning understands the world in terms of a hierarchy of concepts, from simpler ones to complex ones, from low level to high level, layer by layer. This hierarchy structure gives deep learning the ability to represent some complex functions that can simulate the real world.

It seems that deep learning is a new technology which is becoming more and more popular because of its more impressive performance on many complicated problems than other traditional algorithms. In fact, deep learning can date back to the 1940s. Deep learning was relatively unpopular for several years preceding its current popularity because of the limitation of computational resources. Nowadays, thanks to the rapid progress on computational resources, especially the emergence of high-performance GPUs, deep learning algorithms start to exhibit their advantages and become prevailing.

1.2 Components of a neuron network

The basic computational unit of a deep learning model is a neuron. The most common deep learning models are actually hierarchical frameworks of neurons with multiple layers. Neurons here in some sense resemble the neurons in neuroscience which are much more complicated. A neuron in deep learning models takes as input a vector of data, multiplies the data by the corresponding weight vector the neuron stores, then add a bias to the output of the multiplication then passes the output to an activation function. Figure 1 illustrates the structure of a neuron. This "neuron" take x1, x2,and x3 as inputs, and +1 as bias, then do the following computation:

$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$$

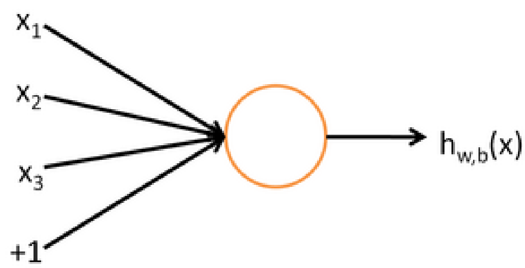

Figure 1. The structure of an neuron

Function f, which provides non-linearity, is called activation function. Some popular activation functions that are commonly used in deep learning are sigmoid, tanh, and ReLU (Rectified Linear Unit). The graph below shows the properties of those functions according to their mathematical forms. Containing the attributes that is described above, a neuron itself can be seen as a linear classifier which has capacity to like or dislike its input.

The framework of deep learning model combines thousands of hundreds of neurons in a hierarchical style which can extract high-level, abstract features from raw data, layer by layer, from simple to complex. Neurons are connected in an acyclic graph since cycles would imply infinite loops during the forward pass of a network. The most common layer type in a neuron network is fully-connected layer in which each neuron connects to all neurons of former layer, and has no connection with other neurons in the same layer. Figure 2 shows structure of a simple deep learning model with multiple fully-connected layers, each circle in this scheme represents a neuron described above.

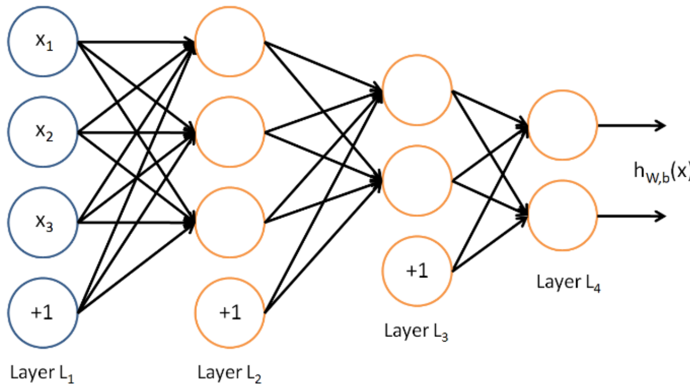

Figure 2. Structure of a simple deep learning model with multiple fully-connected layers

The framework of deep learning model also stores state information, so-called "weights", that helps the model to make sense of the input. The weights in a deep learning model are trainable by giving training samples. Those weights can be updated using gradient descend method. There are some common strategies to apply gradient descend, such as SGD, Adagrad, RMSprop. There are also some important hyperparameters that can be set during the learning process, such as learning rate, momentum, learning rate decay, and batch size.

Before deep learning algorithm becoming popular today, researchers usually used traditional computer vision algorithms for object detection in images, such as Deformable Part Model(DPM)[1] or Selective Search[2]. However, after the dramatic improvement brought on by R-CNN[3], more and more papers are focusing on designing deep learning models for image processing. The most common deep learning model on image processing is convolutional neural network, which has several convolutional layers that can extract the features of the images using fewer weights.

The convolutional layer is the core part of a CNN that performs as feature extractor. The most important property of a convolutional layer is that it has capacity to share parameters between neurons. The way convolutional layers work likes sliding a small window on the input features/images with each sliding window has a set of parameters that would be shared by some neurons.

1.3 Current popular deep learning algorithms

Deep learning has ability to handle tasks that may be too difficult to solve with fixed programs that written by human beings. Among many kinds of tasks that can be solved with deep learning, image/video processing and natural language processing are probably the most common, on which many papers of top conference focus, such as NIPS, ICML, CVPR etc.

Image/Video processing. The most basic aspect of this kind of task is image classification, which can assign a label to an image for the category it belongs to. Large number of papers were working on this aspect and had proposed many famous deep learning models, such as AlexNet[4], ZF Net[5], GoogLeNet[6], and VGGNet[7] etc. Those models are widely used by other models that perform more complex tasks as basic components. Object detection, which is a bit more complex than image classification, can detect instances of semantic objects of a certain class (such as people, cats, or cars) from digital images or videos. Among papers on object detection, some models are designed for general object detection, such as Faster RCNN[8], Single Shot MultiBox Detector(SSD)[9], and Yolo[10], other models are designed for specific object detection, like face detection or pedestrian detection. Semantic segmentation, which attempts to partition the image into semantically meaningful parts, can label each pixel with a class of objects (such as car, cat, or person) and non-objects (such as sky, road, or water). There are also many deep learning models performing this kind of task, such as Fully Convolutional Networks(FCN)[11], fully convolutional instance segmentaion(FCIS)[12], and Mask RCNN[13]. Other tasks, like image captioning which can describe an image with a meaningful sentence, and video tracking which can locate a moving object over time using a camera, are also popular topics of image/Video processing.

Natural language processing. NLP is a way for computers to analyze, understand, and derive useful meaning from human language, and have multiple applications, such as generating keyword tags, machine translation, and speech recognition. NLP algorithms are typically based on deep learning algorithms.

1.4 This project

Urban green space is associated with multiple physical and mental health outcomes. Nowadays, over 80% of the US population lives in urban areas, and the built environment has much effect on human health. Green space, as an indispensable component of built environment, has been identified multiple potential pathways through which it may influence an individual’s health, including reductions in air pollution, decreased noise pollution, and increased physical activity. Several benefits of green space, such as stress reduction and attention restoration, are dependent on visual perception of green space exposures. However, our ability to understand the effect of urban green space on human health is limited by the lack of quantitative data of urban green space. Most studies rely on survey data or birds' eye view measures or satellite derived measure, and thus do not capture street-level information which shows the most direct connection between individuals and urban green space. The emergence of geo-referenced imagery in the United States, such as Google Street View Imagery, provides a convenient window to access the street-level information.

In this project, we apply deep learning algorithms to measure green space quantitatively from street view imagery. We train a Faster R-CNN model based on PasadenaUrbanTree dataset and make it have capacity to do tree detection, then we are able to count the number of trees in the test images. For identifying other specific urban green space features, such as grass, shrubs, and flowers, since they are not objects that have fixed shape but stuff (e.g. sky, road) that is hard to detected using bounding boxes, we apply semantic segmentation models on those features. We use a PSPNet model that pretrained on ade20k dataset to do semantic segmentation on street view images and compute the portion of green space. Combining the outcomes from object detection and semantic segmentation, we are able to measure green space from street view imagery quantitatively.

2. Apply object detection models

Urban green space includes multiple components, such as trees, grass, shrubs, and flowers, and we are now focusing on tree detection.

2.1 Choose Faster RCNN as our model to do tree detection

There are many models that can be used to do tree detection. Table 1 shows the comparison between Faster R-CNN, YOLO, and SSD. Yolo can run very fast, at almost 155fps, but it has lower accuracy. SSD, although has a high accuracy, must have the image reshaped to fit the size it needs, for example, SSD300 can only handle image of size 300*300, and SSD512 only handle image of size 512*512. However, the GSV images which we would like to use as training/testing data normally has very high resolution or diverse sizes, and we also want our model can handle images of any size. Faster R-CNN, while has a relatively high accuracy (higher than YOLO, and almost equal to SSD), is able to take as inputs images of any size, especially images of very high resolution, since it takes advantage of the idea from Spatial Pyramid Pooling Networks (SPP Networks)[14]. This is why we choose Faster R-CNN.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512

Table 1. Compare Faster RCNN with YOLO and SSD

2.2 Introduction of Faster RCNN

For now, we are using the publicly available implementation of Faster R-CNN to detect green space objects from street view imagery and we are now focusing on tree detection. Faster R-CNN is one of the state-of-the-art methods on objects detection, which is composed of convolutional neural networks and region proposal networks.

The convolutional neural network part, as mentioned above, locates at the bottom of the whole model, is used to extract features from input images. It can take as inputs images of any size then output a feature map of that image for following use. ZF net and VGG net are used as feature extractors which make the channel of the feature map be 256 and 512 respectively. Both ZF net and VGG net are pre-trained ImageNet models. At least 3 gigabytes of GPU memory will be needed if we train Faster RCNN with ZF network, while around 11 gigabytes of GPU memory are necessary if we train Faster RCNN with VGG16.

The region proposal network(RPN), takes as input the feature maps output from the convolutional neural network part, then outputs a set of rectangular object proposals, each with its score associated with its probability of object or not object. For generating region proposals, RPN slides a 3*3 window on the input feature map then outputs a new feature map in which each element is mapped to a 3*3 region of the input. This process can be implemented as a convolutional layer whose kernel size is 3 that locates at the top of the input feature map. For each sliding-window location (each 3*3 region of the input feature map), a set of 9 anchors which all have the same center but with 3 different scales and 3 different aspect ratios are generated simultaneously. Furthermore, for each anchor, the intersection over union (IoU) of the anchor with respect to the ground-truth bounding box is computed. The new feature map after the 3*3 convolutional layer is then fed into 2 sibling fully connected layers, one is a classifier that computes the scores and the other is a regressor that encodes the coordinates (with respect to the original image) of the proposal boxes. These two fully connected layers are implemented as two sibling convolutional layers each with kernel size of 1. Then for each anchor, a predicted bounding box and a probability p that indicates whether the predicted box contains an object or not are computed by these two sibling small networks. If a predicted bounding box contains an object, then this bounding box will be passed to the following classification network as a valid proposal. Figure 3 illustrates the architecture of the region proposal network.

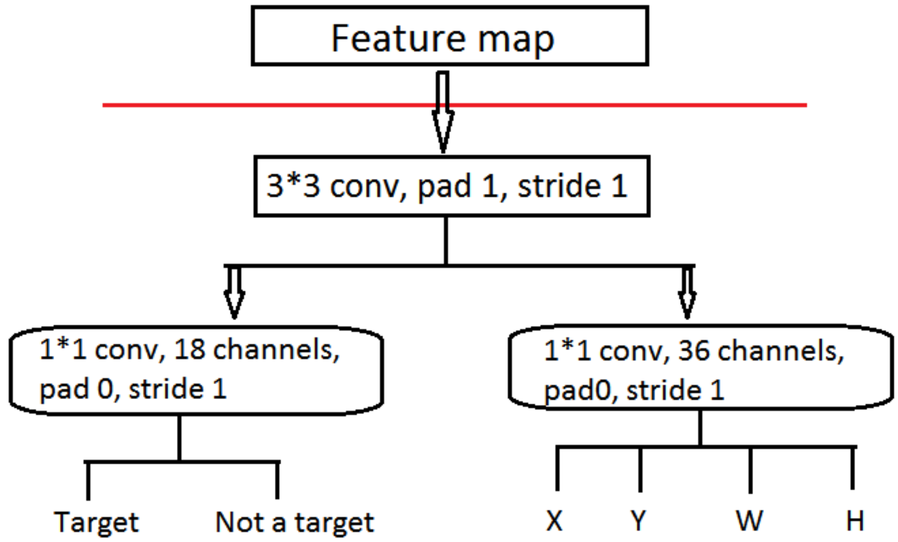

Figure 3. The structure of RPN

RoI max pooling (Region of Interest Max Pooling) can convert the features inside region of interest of any size into a small feature map of fixed size H*W, where H and W are hyper-parameters that we can modify. When a region of interest of size h*w is passed to the pooling layer, it would be divided into an H*W grid of sub-windows of size h/H * w/W in which max pooling is applied. The RoI pooling layer is in fact a special case of SPP (spatial pyramid pooling layer) with only one pyramid level. Figure 4 illustrates what RoI does. RoI pooling is nessesary since the proposals the RPN generates are at different size, but following fully connected layers which would be used as classifiers can only take fixed-size inputs. RoI pooling makes the whole modle have ability to handle images of any size.

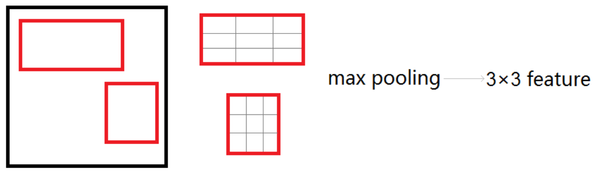

Figure 4. What RoI does

Combining the components described above, Faster R-CNN gets the feature vector of each proposal which is a bounding box that has been predicted to contain an object inside it, then a classifier will take those feature vectors as inputs and determine which categories they belong to. Since we may get multiple proposals for a single object, non-maximum suppression(NMS) is applied after classifier to keep the proposal with highest score. Figure 5 shows what NMS does. Then the bounding box of this proposal with a category label would be the final result of the detection.

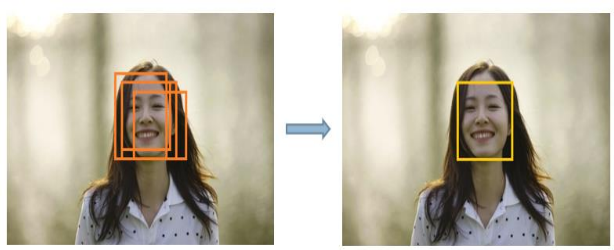

Figure 5. Example of NMS

2.3 Use Faster RCNN and the results

Faster R-CNN is a general algorithm that can detect any type of objects if we train it with corresponding training data. We got a dataset called PasadenaUrbanTrees which contains aerial and street-view imagery for a section of Pasadena. PasadenaUrbanTree dataset is a dataset of 80,000 trees and it also contains meta-data(such as longitude and latitude of a tree) that can be analyzed to locate trees on the street-view images. However, this dataset has no annotations for ground-truth bounding box of trees directly which are necessary and important for object detection. Instead, the ground truth bounding box is generated from the longitude and latitude of the trees. Since the dataset does not contains the height of every trees, the algorithm that does this transformation has assumed a fixed height for all trees, which makes the ground truth bounding boxes could not match the real trees perfectly and hence introduces some defects. Although with this drawback, the outcomes we get from Faster R-CNN trained with this dataset are enough to count the number of trees in an image.

We took 8000 images from the PasadenaUrbanTree dataset, on which had trees labeled, as training data and transformed those images to a format that Faster R-CNN could accept. We also modify some code of Faster RCNN to make it match the transformed image format. We train Faster RCNN model on a GTX980 with 8-gigabyte memory. And since 8-gigabyte is too small to train the model with VGG network which requires at least 11-gigabyte memory during training process, we train faster RCNN with ZF network which only needs 3-gigabyte memory during training process. We modified the prototxt file which defines the architecture of Faster Rcnn model, since we only need 2 categories (background and tree). We tried many version of hyper-parameters, such as different number of iterations at each training step, different learning rate and different learning rate decay and so on. Each version of hyper-parameters took several hours to train. We compared the performance of each hyperparameter version and chose one of them as our final model. Figure 6 shows the outcomes of detecting trees using the final model. Based on those outcomes, we wrote a script that can pass multiple images into the model and then count the number of trees on each image. Figure 7 shows the outcomes of counting trees.

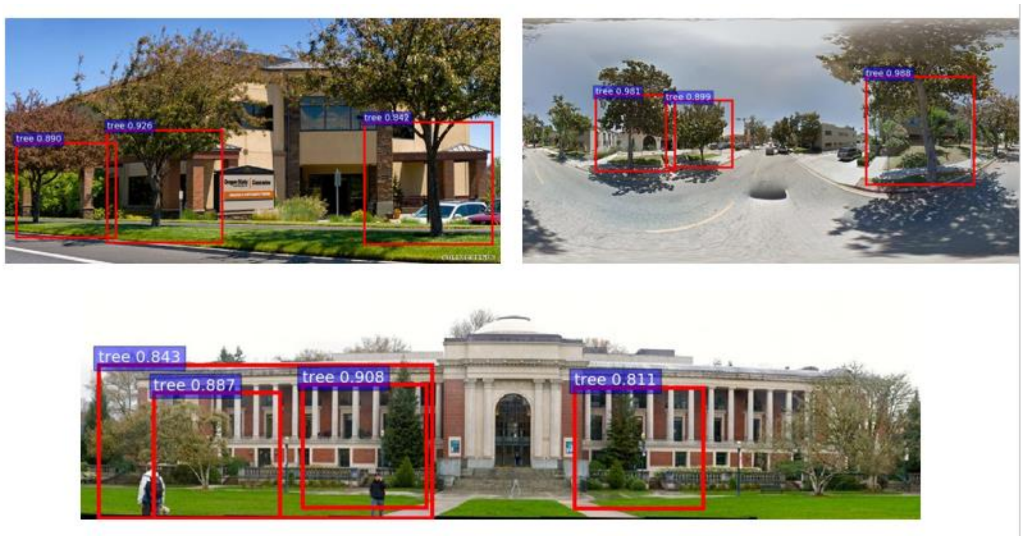

Figure 6. Results of tree detection

Image Name	# of Trees	Score For Each Tree					
5fJkOGJQ72fS8axowng.jpg	4	0.932	0.926	0.863	0.824		
-FxhNMYUZMPDq0bPgGw.jpg	2	0.900	0.866				
4fyaT86MPpuuSVL9cWg.jpg	5	0.933	0.925	0.886	0.880	0.879	
bl4cjHfNCh9cPhPAXIw.jpg	4	0.958	0.948	0.935	0.754		
1PS4wWEa6kH3zvfFUQg.jpg	5	0.966	0.933	0.926	0.777	0.769	
bnluIhC_nmZZMNf3pJw.jpg	6	0.923	0.917	0.888	0.854	0.833	0.790
AQt9mliQado-ewFMxTQ.jpg	5	0.934	0.912	0.872	0.828	0.799	
W0DVdXvn_oB18jpaIPw.jpg	2	0.864	0.761				
0-eh-vM8NYyueu2Hi4w.jpg	4	0.950	0.946	0.909	0.848		
bACBNSvOjTv6tj1xZOA.jpg	6	0.976	0.925	0.880	0.840	0.820	0.787

Figure 7. Results of tree counting

3. Apply semantic segmentation models

3.1 Why we need to do semantic segmentation

Other green space objects, such as grass or shrubs, are very different from trees. Since they are not discrete objects like persons, cars, or trees which has relatively fixed shapes, they are stuffs like sky, road or water which could not be detected as a bounding box, it should be hard for Faster R-CNN to handle this kind of objects. We think semantic segmentation would work well if we have proper training dataset.

3.2 Introduce semantic segmentation as well as PSPNet

Semantic segmentation, as described above, takes as input an image and does classification for every pixel in the image. It assigns each pixel in the input image a category label. After comparing several impressive deep learning models on semantic segmentation, we choose Pyramid Scene Parsing Network (PSPNet)[15] as out model to do segmentation on street-view images. And since the PSPNet framework is based on the fully convolutional network (FCN), understanding FCN is necessary for understanding PSPNet. FCN is a famous model that many state-of-the-art segmentation models are based on it. Unlike prior models which labels each pixel with the class of its enclosing region, FCN does pixelwise prediction directly on every pixel and hence leads to efficiency. Two important components of FCN are convolutional layers that are cast from fully connected layers and upsampling layers, also called deconvolutional layers, which enlarges the size of their inputs.

3.2.1 Casting fully connected layers to convolutional layers

FCN uses normal convolutional layers of other CNN models, such as the AlexNet, VGG net, and the GoogLeNet, as feature extractor, then discards the final fully connected layers and converts them to convolutional layers. In fact, this is the reason FCN names FCN. Note that a fully connected layer can be equivalently expressed as a convolutional layer. The only difference between fully connected layers and convolutional layers is the way their neurons connected to former layer. Each neuron in a fully connected layer fully connects to its input while each neuron in a convolutional layer only connects a local region of its input. However, neurons in both layers compute dot multiplication and have identical functional form. Therefore, it is natural to convert between these two kinds of layers. For example, for a fully connected layer that has 4096 neurons and has its input size of 7*7*512 (512 is the number of channel and 7*7 is the size of each channel), a convolutional layer which has 4096 filters of size 7*7 with zero padding and 1-step stride would get a 1*1*4096 output which is identical of the output of the original fully connected layer. In practice, the converting is done by manipulating the weight matrix of a FC layer into convolutional filters. When FC layers in a model are converted to convolutional layers, it also implies that this model has capacity to handle input of any size.

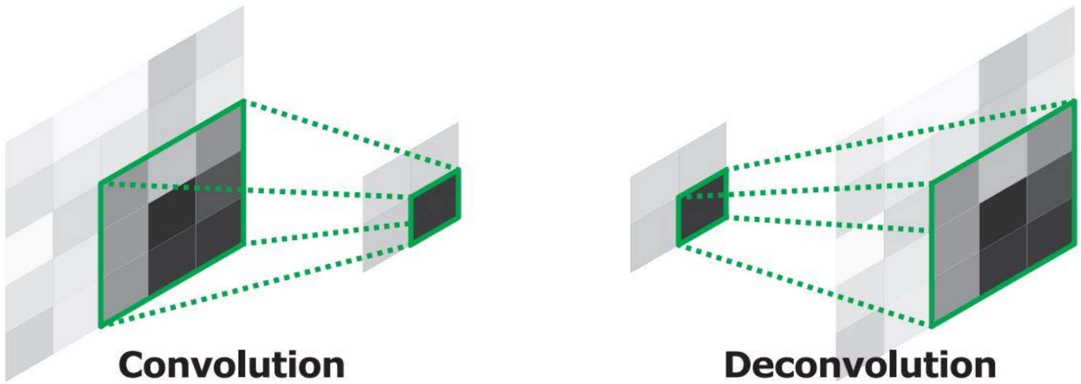

Figure 8. Convolution and deconvolution

3.2.2 Deconvolutional layer

The deconvolution operation reverses the forward and backward passes of convolution. It converts its input from lower resolution to higher resolution (e.g. from a feature map that has smaller size to an output that has the same resolution with the original input image). In practice, a forward pass of a convolution layer can be implemented as matrix multiplication CX, where C is a sparse matrix in which the non-zero elements are the weights of the convolution kernels/filters, and X is the input features. Moreover, the backward pass of a convolution layer is performed by multiplying the loss with the transpose of C. When doing a deconvolution operation, the forward and backward passes of normal convolution are swapped. A forward pass of deconvolution is performed by multiplying X with the transpose of C while the backward pass is performed by multiplying the loss with C. Therefore, deconvolution has also been called transposed convolution. Figure 8 illustrates what deconvolution does as well as convolution. It is obvious that a deconvolution filter is similar with a convolution filter, and the weights in the filter can be learned during training stage. However, in the implementation of FCN, the lr_mult parameter of deconvolution layer has been set as 0 which means they use upsampling (e.g. bilinear upsampling) that uses fixed deconvolution filters.

3.2.3 Pyramid pooling module

PSPNet embeds a pyramid pooling module in an FCN based pixel prediction network. This module is where the power of PSPNet comes from. The major issue of current FCN based models are related to lack of ability to utilize the global context information. Global context information of the image is important for understanding the whole scene as well as improving the performance of segmentation. For example, a predictor may not predict a boat that has car-like appearance as a car if it could use the global context information that this object is over a river. The pyramid pooling module embeds global contextual features into the FCN based framework by fusing information from different sub-regions with different scales. Pyramid pooling module takes as input feature map from the last convolutional layer of a CNN and fuses those features under four different pyramid scales. Then upsampling (bilinear interpolation) is applied on features under different scales to make them have the same size with the original feature map. After this, the upsampled features are concatenated with the original feature map to form the final global feature map which contains both local and global context information.

3.3 Use PSPNet model and the results

In this project, we use a publicly available implementation of PSPnet model which has been pretrained using ade20k dataset. The Ade20k dataset is a dataset for semantic segmentation and scene parsing which contains 150 stuff/object category labels and 1038 image-level scene descriptors. Since category labels in ade20k includes green space objects/stuff, such as tree, grass and plant, the model pretrained on ade20k can be used in our project to do green space segmentation from street-view imagery. We run the PSPnet model on a GTX1080 ti which has 11-gegabyte memory, and only keep the categories we concern with and discard other unrelated categories. The outcomes that we get from PSPnet are shown in Figure 10.

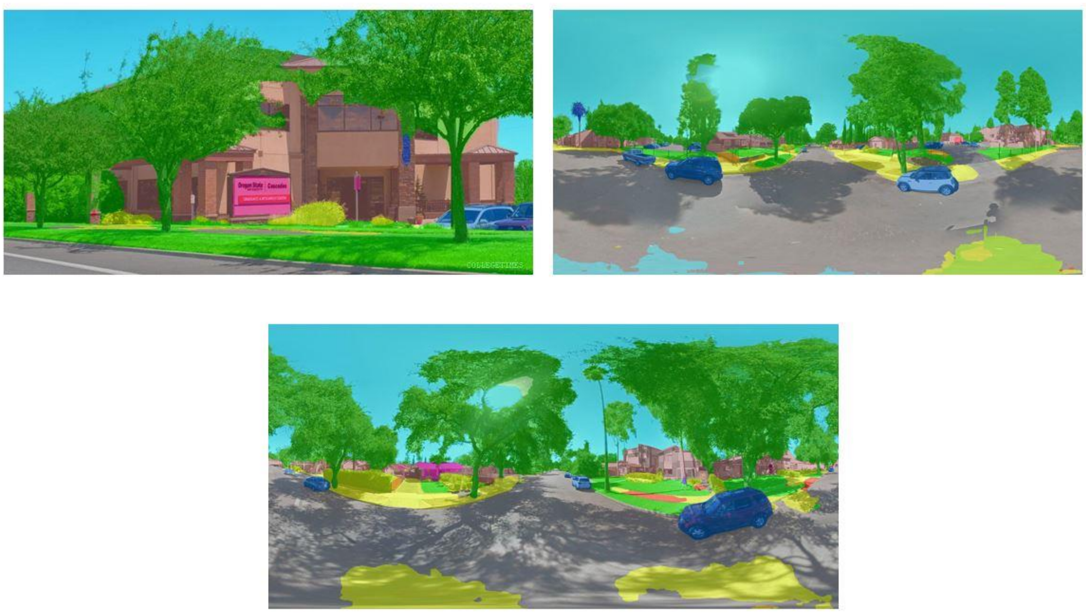

Figure 10. Results of applying PSPNet

4. Combine the output from Faster RCNN and PSPNet

Faster Rcnn model is able to detect trees from the input image, and give each tree a bounding box with credit score. Using the output from Faster Rcnn, we are able to count the number of trees on each street view image. PSPNet model is able to predict whether each pixel in the image is belong to green space or not. Using the outcomes from the PSPNet model, the proportion of green space in an image could be computed. Then for each image, we can use a feature vector to represent what we have measured. For example, a feature vector of an image can have form as:

[number of trees, credit score of each tree, proportion of grass, proportion of grass, …, proportion of shrubs]

A concrete example of a feature vector would be [3, [0.9,0.81,0.7], 0.31, …, 0.25]

Combining feature vectors we generate above with the label or credit about restorative potential of environments of the corresponding image, we can then create a training dataset for restorative potential prediction of environments. When predicting restorative potential of environments, if we want to indicate the environment in an image is good or bad, there are multiple methods, such as logistic regression or support vector machine (SVM, one of the classification method in machine learning). A logistic regression or SVM model can take as input a vector of feature then classify it to one of two classes (good or bad, healthy or unhealth…); we also can use a linear regression model or a deep learning method (specifically, we can use fully connected neural network here) to predict a credit for each image about the restorative potential.

5. Conclusion and future work

The objective of this project is to apply deep learning models to measure green space and then use the measure output to investigate the connection between individuals and urban green space. The first part of objective has been achieved but there are still more problems that should be handled in the future. One of these is that the PSPNet model would take almost 60s to do semantic segmentation on a single image (running on a single GTX1080 Ti) if some hyperparameter has been set to make the model have its best performance. Although the performance of model seems well, 60s is still too long to apply the model in practice, especially when we have thousands of hundreds of images that would be processed. Therefore, we are going to modify some part of the PSPnet model to simplify it and make it faster. We are also going to run the PSPnet model parallelly on several GPUs. Another issue is that we are now running the models on a Linux server using command line. This is not a convenient way, so we plan to write a user interface so that the models can be run in an easy way. For the second part of the objective, training dataset that combines the feature vectors of the image with the label or credit about restorative potential of environments of the corresponding image should be created.

6. Bibliography

[1] Felzenszwalb, Pedro, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

[2] Uijlings, Jasper RR, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition." International journal of computer vision 104.2 (2013): 154-171.

[3] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[5] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

[6] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, and Dragomir Anguelov, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[7] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[8] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[9] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[10] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[11] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[12] Li, Yi, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. "Fully convolutional instance-aware semantic segmentation." arXiv preprint arXiv:1611.07709 (2016).

[13] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." arXiv preprint arXiv:1703.06870 (2017).

[14] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition." European Conference on Computer Vision. Springer, Cham, 2014.

[15] Zhao, Hengshuang, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid scene parsing network." arXiv preprint arXiv:1612.01105 (2016).