**CSC420 Project Report**

**Chen Liang 1002100120**

**Intro:**

When I was travelling in Paris years ago, I was always marvelled by some magnificent yet elegant architectures along the streets. At that time, I was so annoyed because I could bearly tell which building they were due to my lack of background knowledge of Paris. When I learned deep learning from CSC420 this semester, I realized that I could solve the dilemma I encountered when I was a kid by merely applying a convolutional neural network to judge what architecture it is. To achieve this, my teammate and I read through the Facebook research on Detectron (<https://github.com/facebookresearch/Detectron>), and want to recreate our version of object detection based on Detectron. We expect our object detector could categorize each architecture to its corresponding class or label accurately and draw a ground truth box that tightly bounds the outer frame of the architecture. We also expect the detector to reveal how much "confident" such classification is correct by showing a prediction confidence score alongside each ground truth box. One of our main challenges would be there is no existing dataset online, so we have to create a dataset containing 14 different kinds of architectures in which every single image needs to be downloaded from the Internet manually. Our second main challenge is to establish a particular environment to make our object detector work. More specifically, we need to install all dependencies for Detectron, including Caffe2, then set up the Detectron module. Setting up Inference might appear to be simple, but ensuring the environment is indeed the one we want would be super time consuming, and the whole process proves to be very gruelling. Our third main challenge could be drawing bounding boxes for every single image, which would take a lot of time and effort. Since Detectron could implement detection modules like Faster RCNN, MaskRCNN, it is possible to detect multiple objects in a single image with a relatively faster speed while keeping comparatively high accuracy. In this way, the most significant limitation would be the size of the dataset we created, where the maximum number of images for each class never exceeds 300 images. The size of our data set is drastically smaller than the size datasets commonly used, such as the COCO dataset or VOC2007 dataset. Due to the limitation of dataset size, a shoot of architecture from a different angle might not be recognized. Another limitation that needs to be addressed is if we apply filters on the input videos, our model might not recognize these landmarks as precisely as wanted.

Methods:

Data Collecting: My teammate and I built a Paris dataset from scratch. This dataset contains14 classes and for each class, the number of images ranges from 100 to 300. Then we use label Img(<https://github.com/tzutalin/labelImg>) to label ground truth boxes for all 2503 images manually and use annotation/paris\\_building\\_train.json file to save information for all ground-truth boxes, including their image ids, the starting width, height and starting width, starting height for each box.

 Architecture: We chose to use ResNet50 as the backbone rather than VGG 16 due to the reason that replacing VGG16 with ResNet could be spotted improvements as much as 28% (<https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>). Moreover, as ResNet50 could tackle the vanishing gradient problem, ResNet 50 allows us to train relatively deep neural networks without its performance gets saturated or even starts degrading rapidly. (<https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>).

Model: We chose to use Faster RCNN based on FPN to train our dataset. FPN is very competitive with state-of-the-art detectors because it generates multiple feature map layers with better quality information than the regular feature pyramid for object detection within a comparatively inference time. (<https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c>)

The below graph illustrates Faster RCNN on FPN is an expert in extracting masks for images, and it performs better than models like G-RMI, Attraction-Net, etc.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Training Model: We created a YAML file named Paris\\_Detector\\_ResNet50\_FPN\_s1x-e2e.yaml under configs folder. In this YAML file, we configured hyperparameters for training, setting the maximum iterations as 90000, the number of ROIs per image as 512, images per GPU as 2, the number of top-scoring RPN proposals as 2000 (<https://github.com/facebookresearch/Detectron/tree/master/configs/getting_started>) Then we started the training process. The training was done on an NVIDIA GTX 1070 GPU, achieved >92% accuracy, and the weight was saved as model\_final.pkl under model folder.

Below is our resulting average precision visualization, it might appear to be a little bit overfitted due to the number of iterations we set (90000 iterations).

Testing On Input Videos: After getting trained weights, we first test on our test dataset and the testing result for all classes are all over 88%. Then we apply the

Map Feature and Filters:

Conclusion:

We managed to accomplish the main task of detecting and classifying buildings in the video, for problems arises:

Buildings could be spotted from so many different angels. Some of architecures in this video are even taken as aeroshot. In this way, the accuracy that are compromised

The second task is we could detect multiple objects within one frame. Below is an example.

A view of a city

Description automatically generated

It could even intelligently detect building even part of the building is shown:

A picture containing building, sky, outdoor, clock

Description automatically generated

Reference:

[1] <https://github.com/facebookresearch/Detectron/tree/master/tools>

[2] <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>

[3] <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>

[4] <https://github.com/tzutalin/labelImg>

[5]