Photographic Image Synthesis with Cascaded Refinement Networks Implementation

Deep Learning for Computer Graphics Final Project

Saurabh Kumar
Department of Computer Science
& Engineering
Texas A&M University
College Station, Texas, USA
thesaurabh@tamu.edu

ABSTRACT

This is the final project for my Deep Learning for Computer Graphics course. In this project, I have implemented a previously published paper in the field of Computer Graphics. I have implemented it using pytorch and generated my results with slight modifications. We will discuss the entire project in more detail in the following sections.

So, the overall objective of this project is that given a semantic layout of a scene, we want to generate a photorealistic image of a scene that conforms to the input layout. The generated image should be realistic in appearance compared to the real-world scenes. The input to this project is a semantic label map. I have compared the output images generated by my implementation with the images generated by the original paper implemented by the authors in this paper and provided my feedback about areas of improvements.

1. Introduction

Refer to the figure below for input image for this algorithm. The input image is a semantic label map. The output will be an image describing a real-world scene. From the image below, we realize that there are multiple valid and correct output images that can be generated for a single input image.

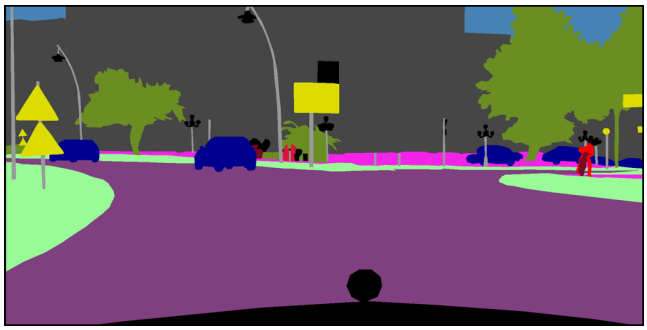

Figure 1: Input Semantic Layout

A sample corresponding output image to be generated by this system can be as follows.

Figure 2: Output Photorealistic Image

This process can be described as the reverse of Image Segmentation process. Image segmentation in computer vision is the process of partitioning a digital image into multiple segments. So this process is the exact opposite of image segmentation where we generate a digital image from the segmented image.

As already mentioned, the input which is a semantic label map is a representation of a scene where the different components of the scene are represented by different semantic classes and are color coded.

2. Related Work

A lot of work has been done in this area in the previous researches. This paper is the implementation of one such publication that describes a method of generating photorealistic images from semantic layouts. The paper is titled "**Photographic Image Synthesis with Cascaded Refinement Networks**" authored by **Qifeng Chen** and **Vladlen Koltun.**

This paper proposes an approach for synthesizing photographic images by direct supervised learning of single feedforward convolutional network while trying to minimize regression loss. The algorithm presented in this paper is proven to work seamlessly for generating images of up to 2MP resolution.

The approach implemented functions as a rendering engine that takes a two-dimensional semantic specification of a scene and produces a corresponding realistic photographic image. Unlike contemporaneous work, this approach does not rely on adversarial training.

I have implemented the exact algorithm as presented in the paper above with slight modifications. The paper proposes a 10 module Network for synthesizing images of resolution 1024*2048 where as I use a 8 module network to synthesize images of resolution 256*512. There are some other differences as well such as the weights to calculate the contribution of various modules to the overall loss in the loss function which I have chosen randomly based on experiments for good results. There are also differences in the mechanism of up sampling the feature maps and down sampling the semantic label maps for each module.

KEYWORDS

Semantic Layout, Deep Learning, Image Segmentation, Computer Graphics

Reference:

Qifeng Chen and Vladlen Koltun. 2017. Photographic Image Synthesis with Cascaded Refinement Networks: In *International Conference on Computer Vision (ICCV'17). Venice, Italy,*

3. Algorithm

The algorithm in this paper uses the Cascaded Refinement Network (CRN) which is a cascade of refinement modules. Each module Mi operates at a given resolution. In my implementation, the resolution of the first module is 8×4. Resolution is doubled between consecutive modules in the CRN. Let wi ×hi be the resolution of module i. Then the first module, M0 receives the semantic layout L as input (down sampled to 8×4) and produces a feature layer F0 at resolution 8 ×4 as output. All other modules Mi (for i6= 0) are structurally identical in behavior. Mi receives a concatenation of the semantic layout L down sampled to wi×hi and the feature layer F i−1 up sampled to wi×hi as input and produces feature layer Fi as output.

Each module Mi consists of three feature layers, an input layer, an intermediate layer, and an output layer. The 3 layers in a module can be visualized as below in figure 3. The input layer has dimensionality wi×hi×(di−1 + c) and is a concatenation of the down sampled semantic layout L (c channels) and a bilinearly up sampled feature layer Fi−1 (di−1 channels).

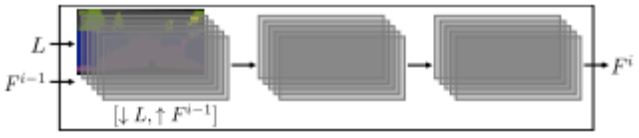

Figure 3: Cascaded Refinement Network Module

Each layer is followed by 3×3 convolutions, layer normalization, and LReLU nonlinearity. The output layer of the final module is not followed by normalization or nonlinearity. Since this is an exact implementation of an original paper, please refer to the original paper for all the details about the architecture.

4. Implementation

The original paper was implemented by the authors using TensorFlow. I have implemented the same paper using Pytorch. This implementation was trained and tested successfully. The challenges faced while implementation and the details for implementing the project.

Talking about the challenges faced during the implementation of this project, the most important challenge that I faced was getting familiar with the pytorch framework for overall implementation of the project. I don't have a lot of experience with Python development let alone pytorch and using it for the overall development was definitely a gruesome task. Once I was able to get a model implemented that was in execution ready state, the execution seemed impossible. The execution part for training the model just did not work even on google colab. Google colab provides free GPU's for our tasks but the training kept on running out of memory. I had to look for alternatives and TAMU HPRC seemed like a good idea. I used my TAMU HPRC account since it provides GPU's for computing, but I was not able to find the python packages installed on the HPRC machines. I tried to install those packages myself, but the system did not let me. Looking for other alternatives to test my program, it turned towards free trail access to Amazon Web Services. Although the AWS website said free trial, it basically relied on the pay for resources as you go model and just the signup seemed to be free. So, I then tried out Google Cloud Platform which gives 300$ as starting credit and a free trail for 12 months. On google cloud, we have to select the zone where the VM instance is being created and we can also select the hardware resources to be used in our VM such as the number of CPU's, the amount and type of memory and the number of GPU's. I finally realized that the usage of GPU's was not allowed with the free account and upgraded to the paid account in order to run my training.

Another challenge was to obtain the dataset for training. I had to write scripts to filter out and organize the dataset from the overall data provided on the website for my training purpose.

Talking about the other details pertaining the project, I implemented the project to only generate images of resolution 256*512. I used the python notebook on google colab for initial development of the project. It allowed me to perform syntax and semantic checks on my python code as well as test out the CUDA code for using GPU's. Since colab does provide access to GPU's, I could test out the program syntactically and only ran out of memory while training.

For training, I experimented with using both the cityscapes dataset and the NYU indoor scene dataset as I mentioned in my project proposal. It seemed like the model also learnt the way those semantic label maps are generated and produced very blurry results when both of them were used. So finally, I decided to just use the cityscapes dataset to train and test my model. I used the gtFine_trainvaltest dataset for the semantic images and the leftImg8bit_trainvaltest for the corresponding real images as ground truth for training the model.

I also tried out random sample semantic label maps from the internet for testing my model, but the results obtained were not good.

5. Results

The results of this implementation are shown below. Figure 4 shows a sample input semantic image and Figure 5 shows the corresponding photographic image generated by my system. The resolution of the input and output images are 256*512.

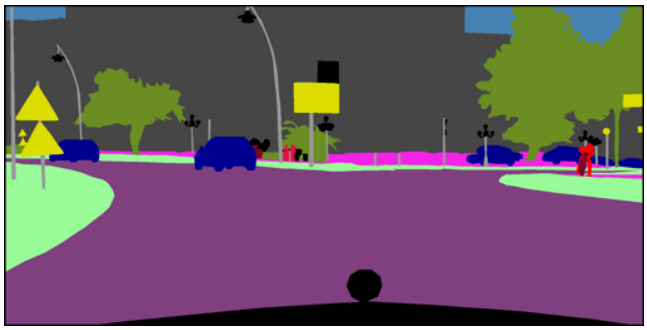

Figure 4: Input Semantic Layout to my system

Figure 5: Output Image generated by my system

As we can see in the images above, although blurry, the system generated realistic looking image of the scene depicted by the semantic label map. We can correlate the objects in the semantic label map with the objects present in the generated image such as the cars, the trees, the building and the Mercedes logo.

Figure 6: Input Semantic Layout to Original Authors Implementation

Figure 7: Output Image generated by Original Authors Implementation

Comparing with the results obtained by the original authors of the paper, the figures above shows the results achieved by authors of the original paper. The Figure 6 above shows a sample input semantic label map and Figure 7 shows the corresponding photographic image generated by the authors implementation. The generated image is the same resolution as ours in order to make a fair comparison in our approaches. As we can see, the output image generated by the authors implementation looks much more vivid and realistic as compared to my image.

One of the reasons due to which my results are not as good could be the fact that I did not train my model on entire cityscapes dataset whereas the author of the paper did. I trained my model only on one part of the cityscape's dataset consisting of less than 3000 images. Secondly, I trained my model in one go whereas training the model in module wise parts could have resulted in way better results.

Overall, I am satisfied with the results generated by my experiment given the scope of the project and the constraints in terms of time and resources. I certainly believe that the results can be improved provided sufficient training and tweaking of the parameters is done as well as incremental training is performed on the model.

ACKNOWLEDGMENTS

I would really like to thank the authors of the original paper for providing a detailed approach which really helped me in this project.

REFERENCES

[1] Qifeng Chen and Vladlen Koltun. 2017. Photographic Image Synthesis with Cascaded Refinement Networks: In *International Conference on Computer Vision (ICCV'17). Venice, Italy,*

[2] PyTorch Tutorials, https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html

[3] GPU accelerated computing with python, https://developer.nvidia.com/how-to-cuda-python