Realistic Image Synthesis with Cascaded Refinement Networks

Author: Xiang Zhang

Class: CS560

2018 Spring

Abstract

Image synthesis is the process of creating images. Realistic image synthesis is the process of creating images that are, in some way, accurate representations of a real scene. Often, but not always, the images are meant to be viewed by a human observer. Therefore, the accuracy is with respect to the human visual system. In some cases the image need only be plausible, the viewer is convinced that the scene could actually be real. The Cascaded Refinement Network (CRN) is a single feedforward network with appropriate structure, trained end-to-end with a direct regression objective which could synthesize photographic images from semantic layouts.

Keyword: synthesis; realistic image; CRN

Introduction

A skilled painter could draw images that depict urban scenes that conform to these layouts. Highly trained craftsmen can even create paintings that approach photorealism. Can we train computational models that have this ability? Given a semantic layout of a novel scene, can an artificial system synthesize an image that depicts this scene and looks like a photograph?

CRN is a model for photographic image synthesis from pixel wise semantic layouts. It is a convolutional network, trained in a supervised fashion on pairs of photographs and corresponding semantic layouts. Such pairs are provided with semantic segmentation datasets (Cityscapes Dataset). We use them not to infer semantic layouts from photographs, but to synthesize photographs from semantic layouts. In this case, it can be viewed as the inverse of semantic segmentation.

We show that photographic images can be synthesized directly by a single feedforward convolutional network trained to minimize a regression loss. We show that direct supervised training of a single convolutional network can yield photographic images. Furthermore, the presented approach scales seamlessly to high image resolutions. The synthesize images with resolution up to 2 megapixels (1024×2048), the full resolution of our training data.

Method

3.1. Preliminaries

Consider a semantic layout L ∈ {0,1}m×n×c, where m×n is the pixel resolution and c is the number of semantic classes. Each pixel in L is represented by a one-hot vector that indicates its semantic label: L(i, j) ∈ {0, 1}c

$$\text{s.t.} \quad \sum_p L(i, j, p) = 1$$

One of the c possible labels is ‘void’, which indicates that the semantic class of the pixel is not specified. Our goal is to train a parametric mapping g that given a semantic layout L produces a color image I ∈ Rm×n×3 that conforms to L.

In the course of this project we have experimented with a large number of network architectures. As a result of these experiments, we have identified three characteristics that are important for synthesizing photorealistic images. We review these characteristics before describing our solution.

Global coordination. Globally consistent structure is essential for photorealism. Many objects exhibit nonlocal structural relationships, such as symmetry. For example, if the network synthesizes a red light on the left side of a car, then the corresponding light on the right should also be red. This distinguishes photorealistic image synthesis from texture synthesis, which can leverage statistical stationarity. Our model is based on multi-resolution refinement. The synthesis begins at extremely low resolution (4×8 in our implementation). Feature maps are then progressively refined. Thus global structure can be coordinated at lower octaves, where even distant object parts are represented in nearby feature columns. These decisions are then refined at higher octaves.

High resolution. To produce truly photorealistic results, a model must be able to synthesize high-resolution images. Low resolution is akin to myopic vision in that fine visual features are not discernable. The drive to high image and video resolutions in multiple industries is a testament to resolution’s importance. Our model synthesizes images by progressive refinement, and going up an octave in resolution (e.g., from 512p to 1024p) amounts to adding a single refinement module. The entire cascade of refinement modules is trained end-to-end.

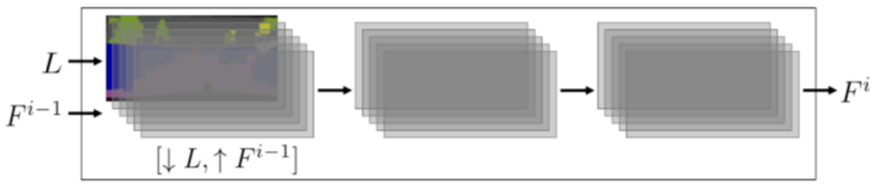


Figure 3. A single refinement module.

3.2. Architecture

The Cascaded Refinement Network (CRN) is a cascade of refinement modules. Each module M i operates at a given resolution. In our implementation, the resolution of the first module (M 0) is 4 × 8. Resolution is doubled between consecutive modules (from M i−1 to M i). Let wi × hi be the resolution of module i.

The first module, M0, receives the semantic layout L as input (downsampled to w0 ×h0) and produces a feature layer F 0 at resolution w0 × h0 as output. All other modules M i (for i≠ 0) are structurally identical: M i receives a concatenation of the layout L (downsampled to wi×hi) and the feature layer F i−1 (upsampled to wi × hi) as input, and produces feature layer F i as output. We denote the number of feature maps in F i by di.

Each module M i consists of three feature layers: the in- put layer, an intermediate layer, and the output layer. This is illustrated in Figure 3. The input layer has dimensionality wi ×hi ×(di−1 + c) and is a concatenation of the downsampled semantic layout L (c channels) and a bilinearly upsampled feature layer F i−1 (di−1 channels). Note that we do not use upconvolutions because upconvolutions tend to introduce characteristic artifacts . The intermediate layer and the output layer both have dimensionality wi × hi × di .

Each layer is followed by 3 × 3 convolutions, layer normalization, and LReLU nonlinearity.

The output layer F ̄ı of the final module M ̄ı is not followed by normalization or nonlinearity. Instead, a linear projection (1×1 convolution) is applied to map F ̄ı (dimensionality w ̄ı ×h ̄ı ×d ̄ı) to the output color image (dimensionality w ̄ı ×h ̄ı ×3). The total number of refinement modules in a cascade depends on the output resolution. For our main experiments on the high-resolution Cityscapes dataset, the number of modules is 9, accounting for a resolution increase from 4 × 8 to 1024 × 2048. For the number of feature maps di, we use1024for i = 0..4, 512 for i = 5,6, 128 for i = 7, and 32 for i = 8.

3.3. Training

The CRN is trained in a supervised fashion on a semantic segmentation dataset D = {(I,L)}. A semantic layout L is used as input and the corresponding color image I as output. This can be thought of as "inverse semantic segmentation". It is an underconstrained one-to-many inverse problem. We will generally refer to I as a "reference image" rather than "ground truth", since many valid photographic images could have yielded the same semantic layout.

Given the underconstrained nature of the problem, using an appropriate loss function is critical, as observed in prior work on image synthesis. Simply comparing the pixel colors of the synthesized image and the reference image could severely penalize perfectly realistic outputs. For example, synthesizing a white car instead of a black car would induce a very high loss. The basic idea is to match activations in a visual perception network that is applied to the synthesized image and separately to the reference image.

Let Φ be a trained visual perception network (we use VGG-19). Layers in the network represent an image at increasing levels of abstraction: from edges and colors to objects and categories. Matching both lower-layer and higher-layer activations in the perception network guides the synthesis network to learn both fine-grained details and more global part arrangement.

Let {Φl} be a collection of layers in the network Φ, such that Φ0 denotes the input image. Each layer is a three- dimensional tensor. For a training pair (I, L) ∈ D, our loss is

$$\mathcal{L}_{I,L}(\theta) = \sum_l \lambda_l \|\Phi_l(I) - \Phi_l(g(L;\theta))\|_1.$$

Here g is the image synthesis network being trained and θ is the set of parameters of this network. The hyper parameters {λl} balance the contribution of each layer l to the loss.
For layers Φl (l ≥ 1) we use 'conv1 2', 'conv2 2', 'conv3 2', 'conv4 2', and 'conv5 2' in VGG-19 [43]. The
Hyper parameters {λl} are set automatically. They are initialized to the inverse of the number of elements in each layer. After 100 epochs, {λl} are rescaled to normalize the expected contribution of each term ‖Φl(I) − Φl(g(L; θ))‖1 to the loss.

3.4. Synthesizing a diverse collection

The architecture and training procedure described so far synthesize a single image for a given input L. In our experiments this already yields good results. However, since a

given semantic layout can correspond to many images, it also makes sense to generate a diverse set of images as output. Conditional synthesis of diverse images can be approached as a stochastic process. We take a different tack and modify the network to emit a collection of images in one shot, with a modified loss that encourages diversity within the collection.
We further build on this idea and formulate a loss that considers a virtual collection of up to kc images. (Re- call that c is the number of semantic classes.) Specifically, for each semantic class p, let L_p denote the corresponding channel L(·, ·, p) in the input label map. We now define a more powerful diversity loss as

$$\sum_{p=1}^{c} \min_{u} \sum_{l} \lambda_l \sum_{j} \left\| L_p^l \odot \left(\Phi_l^j(I) - \Phi_l^j(g_u(L;\theta)) \right) \right\|_1$$

where Φjl is the jth feature map in Φl, L^l_p is the mask Lp downsampled to match the resolution of Φl, and ⊙ is the Hadamard product. This loss in effect constructs a virtual image by adaptively taking the best synthesized content for each semantic class from the whole collection, and scoring the collection based on this assembled image.

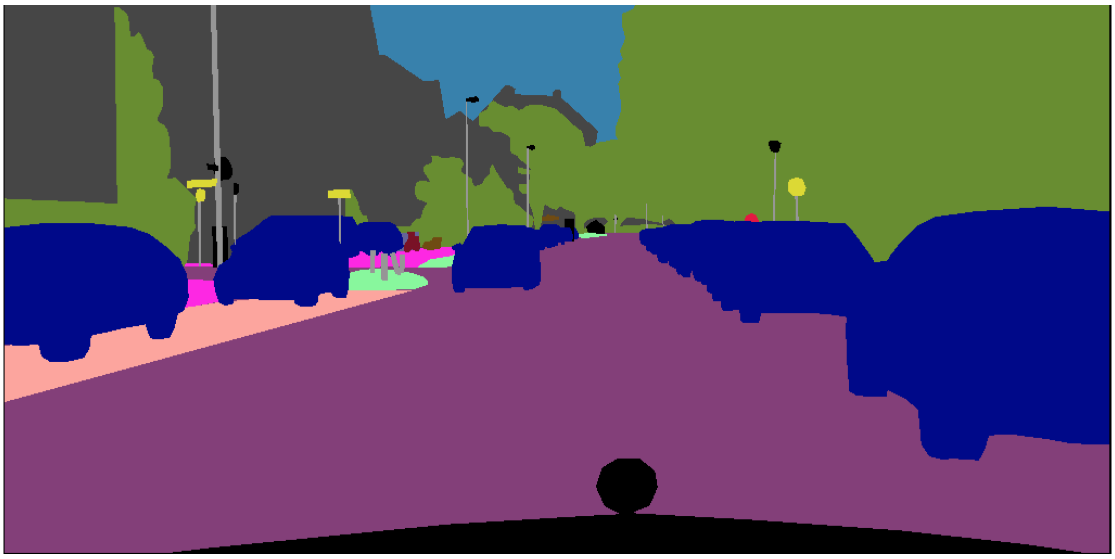
Experiments

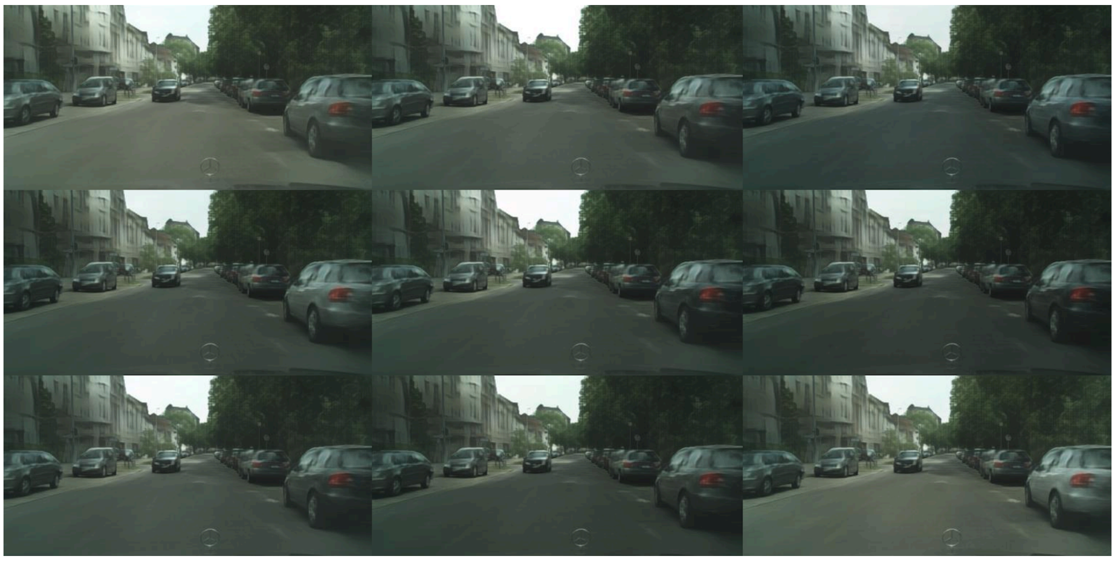
Requirement

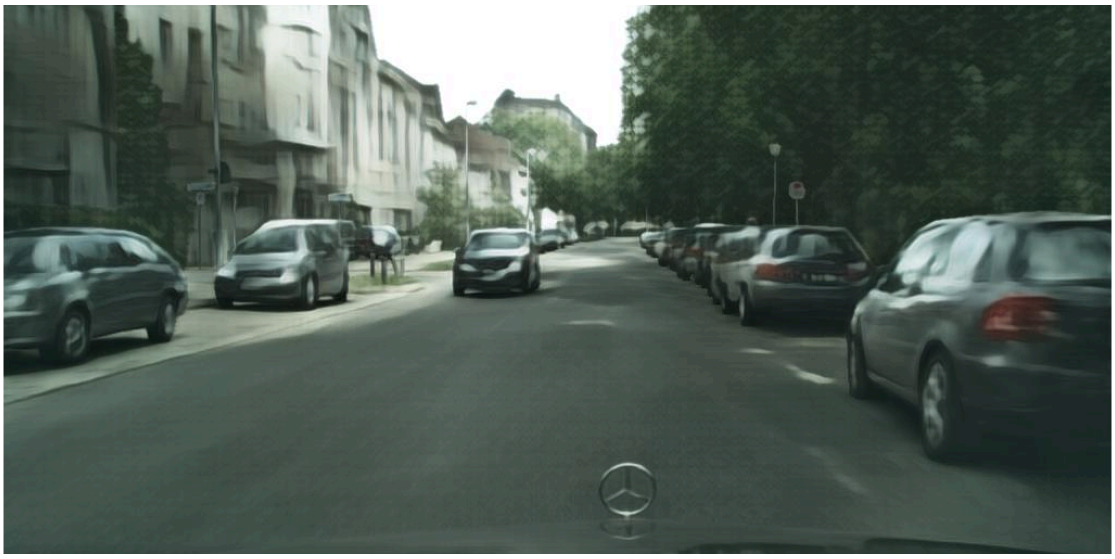
Required python libraries: Tensorflow (>=1.0) + Scipy + Numpy + Pillow.

Datasets. We use the dataset of Cityscapes, which has become the dominant semantic segmentation dataset due to the quality of the data . We train on the training set (3K images) and evaluate on the validation set (500 images). (Evaluating “inverse semantic segmentation” on the test set is impossible because the label maps are not provided.)

Result







Conclusion

We have presented a direct approach to photographic image synthesis conditioned on pixel wise semantic layouts. Images are synthesized by a convolutional network trained end-to-end with a regression loss. This direct approach is considerably simpler than contemporaneous work, and produces much more realistic results.

Reference:

Photographic Image Synthesis with Cascaded Refinement Networks. In CVPR, 2017
Towards principled methods for training generative adversarial networks. In ICLR, 2017
Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
The Cityscapes dataset for semantic urban scene understanding. In CVPR, 2016.