Implicit Neural Representations For 3D Scene Reconstruction: How Building in Better Inductive Biases Creates Better Models

By Joseph Morales

In this blog, I hope to address implicit neural representations as a method for modeling 3D structures, how they can be leveraged to create models that learn 3D representations of objects in order to create unseen 2D and semantic views, and how this is representative of how including better inductive biases in ML creates better models.

3D Representations

Representing 3D objects is challenging because of a need to encode various modalities of information. There's a goal to qualify where exactly an object is and how much volume it takes up, a goal to maintain an accurate representation of the properties of its surface (e.g. smoothness, texture, appearance, etc), and a goal to semantically label either the entire object or different individual parts of it. Classical representations of objects in 3D space such as voxel grids, meshes, and point clouds face various trade offs; To scale the resolution of voxel grids and meshes in order to store more fine-grained information, storage requirements balloon. Although point clouds are capable of storing points sampled at any given resolution, there is a necessary trade off between poorly representing volumes with sparse clouds and using increased storage to construct a more complete representation. Additionally, point clouds are incapable of representing topological relations on their own, creating additional challenges when trying to redetermine surface properties.

The underlying problem with these methods, and where implicit neural representations differ, is that they are inherently discrete. They rely on constructing grids and shapes, or on storing information in quantized amounts, but important properties of 3D objects have to do with their continuity and smoothness. Recent innovations in implicit neural representations circumvent the unavoidable problems of classical 3D modeling schemes by instead using neural networks as continuous functions, mapping points in 3D space to the properties of whatever is occupying that space. In this way, a neural network of constant size can be trained to return the features at any given resolution of a point on a learned 3D model.

How can we construct 3D models from 2D images?

In order to understand the context in which current use cases of implicit neural representations are innovating machine learning research, let us look at the problem of constructing a 3D representation of an object from exclusively 2D views of the object, such that we can query a model for a new view of that object and it returns an accurate view. This kind of paradigm can be thought of as part of an Inverse Graphics problem - can we reconstruct the object or scene that was used to generate graphics of it? Specifically, given a set of images of a scene, can we generate a new image of the scene that we have not perceived yet?

Work done by Tartarchenko et al. in 2016 approached this as a typical encoder-decoder problem, in which pairs of images of an object and a new camera perspective were fed into a network, and it was expected to output an image of the object from the new perspective and depth map of the resulting image. It was trained by taking the loss between the predicted images and depth maps and ground truth values for them. A picture of the model architecture can be found below, but in summary, the image was encoded using convolutional layers into a representation of the object, and the new perspective was used along with a decoder to expand this representation into a new image. The research resulted in novel results, however it was far from human-like in terms of resolution and accuracy, leaving much to be desired.

A crucial reason for this is the inductive bias of the model; the authors assumed the model would encode the images with some degree of information about 3D structure of the objects contained within, but they didn't explicitly enforce 3D structure as a necessity in that representation. As a result, while the model learned to create representations with 3D information, they were not influenced strongly enough to represent it more accurately. It begs the question, "how can we build in better inductive biases, and can a more explicit 3D bias impact performance on this task?"

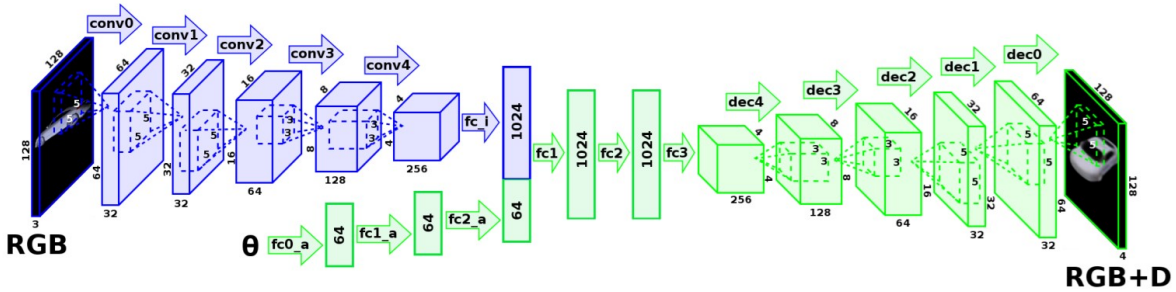

Figure 1. Encoder-Decoder architecture used by Tartarchenko et al. to construct RGB+D images of an object from a new perspective

Scene Representation Networks as a 3D Modeling Method

Work in 2020 done by Sitzmann et al. did exactly that in "Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations" by leveraging the 3D inductive biases inherent to implicit neural representations. At a very high level, their approach was virtually the same as Tartarchenko's – given images of an object, "encode" that object to represent aspects of it in 3D space, and "decode" it from a new camera perspective to get a new view of the same object. The key innovation is the piece of the model architecture Sitzmann used to represent the object, or in this case the entire scene: a fully connected neural network, coined a "Scene Representation Network" or SRN, that takes in an x, y, z coordinate as input and returns the features (color and distance to the nearest object surface) at that point in the scene. Through the inclusion of this 3D interface into the network, they are able to explicitly require a 3D representation of the scene be learned by their model.

More specifically, the representational pipeline of the model can be broken down into 3 parts. It begins with a decoder module that takes in a set of images (e.g. a series of pictures of the same car), and outputs a latent variable **z** representing the scene depicted in the set. This representation is not yet explicitly in 3D, which is the job of the SRN at the end of the pipeline. To connect the decoder and its latent variable to the representation within the SRN, the model uses a fully connected hypernetwork that takes in the latent variable of the decoder, and uses it to output all of the weights of the SRN. When learning occurs, only the hypernetwork and decoder are updated because the two fully determine the parameters of the SRN. To reiterate, this pipeline takes in a set of images of a scene and decodes them all into one latent variable and passes that through a hypernetwork that outputs the SRN weights, such that the SRN can be given an x,y,z coordinate and it will return the features at that point within the scene.

The final part of the model devised by Sitzmann et al. is a neural renderer, which takes in a camera pose and intrinsics, and constructs the image that would be seen from that perspective. It uses an iterative ray tracer that, for each pixel, marches down a ray and queries the SRN for the features at a particular point. If the point is a part of a surface, it returns the feature vector at that point, and if it is not it uses the value given by the distance to the nearest surface to continue marching down the ray. A fully connected network (called the pixel generator) processes the feature vector and returns the RGB value for a particular pixel. By combining the scene representation pipeline, which learns to create an implicit, 3D model of the scene based off of a set of images of it, and the neural renderer, which uses a given camera perspective and the SRN to create new images, we have a complete system that can be trained end-to-end. A comparison of this model with Tartarchenko's can be seen below, and animated examples of SRN performance can be found at https://www.vincentsitzmann.com/srns/ [Not sure how this would render on a webpage, so I can change to "next page" if required].

Figure 2. Qualitative performance of Tartarchenko's encoder-decoder approach can be seen compared to SRN's, along with ground truth.

As we can see from the figure, although derived from a much more complicated system, Sitzmann's approach with an SRN model results in better reconstructed images than Tartarchenko's approach. On top of that, it can be used to directly build a 3D model, whereas in Tartarchenko's paper they describe a process by which multiple images must be generated, overlayed, and further processed to extract a 3D model, which is ultimately less precise. I would like to reiterate and argue that it is not just the increased complexity of Sitzmann's model that is driving it to do better than Tartarchenko's, but that this explicit 3D inductive bias that is being imposed on the model is the determining factor in its performance.

How can Scene Representation Networks be extended?

The ability of SRNs to represent objects in 3D is not limited to just color or distance from the nearest surface. In a direct follow up of the SRN research presented by Sitzmann above, Kohli et al. sought to extend them to perform semantic segmentation. They believed that the 3D representations of SRNs, along with the encoded appearance information (e.g. RGB) would make reusing them for semantic labeling feasible. Plotting the t-SNE embeddings of the SRN (which can be seen in the figure below), they found that existing representations very easily and directly transferred to the semantic segmentation task.

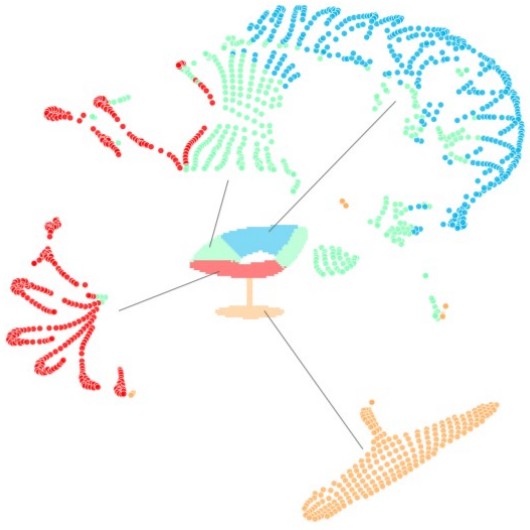

Figure 3. t-SNE plot of the embeddings of pre-trained SRNs on a new office chair example, colored with their semantic labels. The lack of overlap between embeddings of different semantically labeled parts indicates it is easy to distinguish between them.

Figure 4. Semantic labelling of 2 chairs by the modified SRN system from different camera views.

What Kohli specifically implemented was an additional module on top of the entire original SRN model architecture. This additional module is a fully connected neural network that, like the pixel generator, takes in the feature vector returned by the SRN, but instead of calculating RGBD, it determines the semantic label of the given point. Because it is an addition to the original SRN network and not an entirely new architecture, they were able to train the model to perform semantic segmentation with a very small amount of semantically labeled images. The motivation for this task is a result of that fundamental issue – although it is feasible to train the network from scratch on semantic information, in the real world it is very hard to collect a sizable semantically labeled database. Instead, by leveraging the representations already learned by SRNs, they were able to use just 30 semantically labeled examples and achieve impressive performance (as seen in the figure above or at www.computationalimaging.org/publications/semantic-srn/).

This extension of the original SRN research just goes to further reinforce the importance of strong inductive biases in machine learning models. The underlying assumption of the original SRN problem is that explicit 3D modeling of the scene would better allow networks to capture the geometry and appearance of objects, and that assumption was successfully stretched to encompass the structure and bias required for semantics, even though it could be considered an entirely different modality. Importantly, building in this bias did not involve any fancy architectures; All of the representational power of the SRN system came from basic, fully connected neural networks that were intelligently utilized to enforce the structure required to solve the problem. To design more and more innovative systems, both in this field and across machine learning in general, researchers should aim to create systems that similarly don't just rely on increased computational power and resources, but instead maximally leverage what we know about the tasks and how to best represent them with models that we have on hand.