Inferring Shading Parameters from Graphically Generated Images

Ruta Joshi [ruta@stanford.edu]
Anthony Carrington [acarring@stanford.edu]
Madeleine Yip [mjyipstanford@stanford.edu]

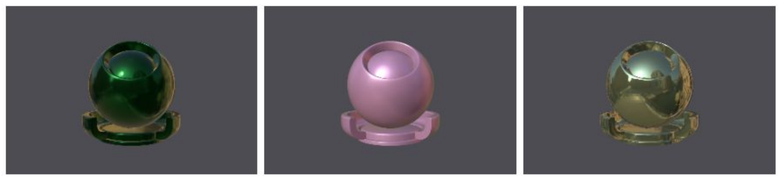
Stanford University

Overview

Our team used deep learning to reverse engineer a 3D image generated with MaterialX, a look development platform for computer graphics used for film. Given a computer-generated image of a mesh, shaded using 10 floating point properties, we regress the float values that define each property. We use transfer learning on existing VGG16 and ResNet50 architectures to evaluate the performance of different architectures on this domain. We were able to achieve the best results from our fine tuned Resnet50 model, which achieved a MSE loss of 0.09 and MAE of 0.26. The network architecture we propose can be used for film and photography applications in which the properties of an image need to be known in order to regenerate similar images using a different platform.

Data

Our dataset consists of 10,000 images that we generated using MaterialX. All images were of a single mesh and shaded using 10 floating point properties: saturation (base), color (3 RGB values), specular, specular roughness, specular color (3 RGB values), and metalness. We first generated the ground truth values for these properties, then used MaterialX to render a mesh shaded with these properties. The input images were cropped and resized to size (224 x 224 x 3) and were labeled with their 10 ground truth property values, all scaled to fall between 0 and 1. RGB values were normalized by dividing by 255. We split the dataset as 65% train, 15% validation, and 20% test for evaluating our model.



Example images from our dataset

Features

The data has (224 x 224 x 3) or 150,528 input features, which consist of the RGB color values that make up the MaterialX images in our dataset. All features are raw input features, since the goal of our project was to create a model that could output the shading properties of an image from its raw depiction. Moreover, we use pre-trained deep neural networks with proven success on computer vision tasks, VGG-16 and ResNet-50, to handle and abstract feature extraction.

Models

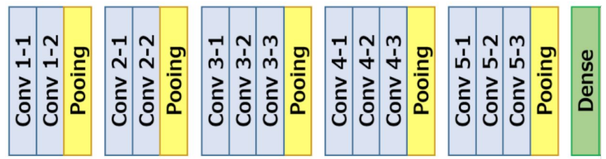
We trained the following models:

1. VGG16 (V1) with a fully connected sigmoid output layer of 10 neurons
2. VGG16 (V2) with an added convolutional layer, spatial dropout, and a fully connected output layer of 10 neurons
3. VGG16 with one-hot vectors of 1001 bins for each of the 10 floats for classification (VGG16 with bins V1)
4. VGG16 with one-hot vectors and added convolutional layers (VGG16 with bins V2)
5. ResNet50 (V2) with an added convolutional layer, spatial dropout, and a fully connected output layer of 10 neurons
6. ResNet50 (V3) with an added average pooling layer, two dense layers, spatial dropout, and batch normalization

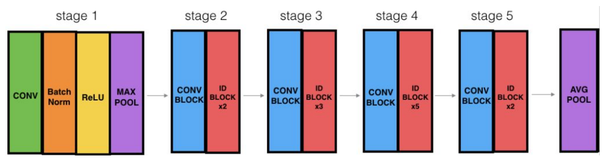
We sought to compare models so that we could identify the best architecture for the task.

Shown are the basic VGG16 and ResNet50 architectures from which we removed the last pooling and dense layers and added our new architecture for models 1, 2, 3, and 4:

VGG16 Base Architecture



ResNet50 Base Architecture



We also created a new loss function, modifying cross entropy loss, for our bins-based architecture. We used this for comparing the bins architecture models 5 and 6.

$$L_i = -\left(1 + \frac{|c_{y_i} - c_{\hat{y}_i}|}{M-1}\right) \sum_{c=1}^{M} y_{o,c} \log(p_{o,c}),$$

Results

Comparison of ground truth labels and predictions from ResNet50 V3, our best performing model.

Images from Ground Truth Labels

Images from Model Predicted Labels

Model	Train MSE	Train MAE	Test MSE	Test MAE
ResNet50-V3 [Added layers]	0.0443	0.163	0.09	0.26
VGG16-Classification	0.1626	0.4032	0.1742	0.4174
VGG16-Regression	0.0772	0.2017	0.12	0.351

Discussion and Future Work

We found from our experiments that classification was an easier task for VGG to learn than regression, and that ResNet models outperformed VGG models by a small amount. All models performed similarly on the test set and validation set, but ResNet models, which are inherently simpler than VGG, trained faster and were easier to interpret.

Our best performing model was a pre-trained ResNet50 (ending before the pool and dense layer) with additionally trainable layers, consisting of an average pool, 2x [512 Dense, Dropout 50%] Block, and a BatchNorm layer. In the future, we would recommend training earlier layers as well.

Given more computational resources and time, we would explore using bins to apply the classify-then-regress trick to the ResNet50 models. We would also tune the number of bins used for classification, conduct a grid search for hyperparameters, and apply learning rate decay to all models.

References

[1] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau, “Single-image svbrdf capture with a rendering-aware deep network,” ACM Transactions on Graphics (SIGGRAPH Conference Proceedings), vol. 37, p. 15, aug 2018.

[2] D. Gao, X. Li, Y. Dong, P. Peers, K. Xu, and X. Tong, “Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images,” ACM Transactions on Graphics (TOG), vol. 38, no. 4, p. 134, 2019.

[3] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, “Robust optimization for deep regression,” in Proceedings of the IEEE international conference on computer vision, pp. 2830–2838, 2015.

[4] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud, “A comprehensive analysis of deep regression,” IEEE transactions on pattern analysis and machine intelligence, 2019.

[5] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang, “Rendernet: A deep convolutional network for differentiable rendering from 3d shapes,” in Advances in Neural Information Processing Systems, pp. 7891–7901, 2018.