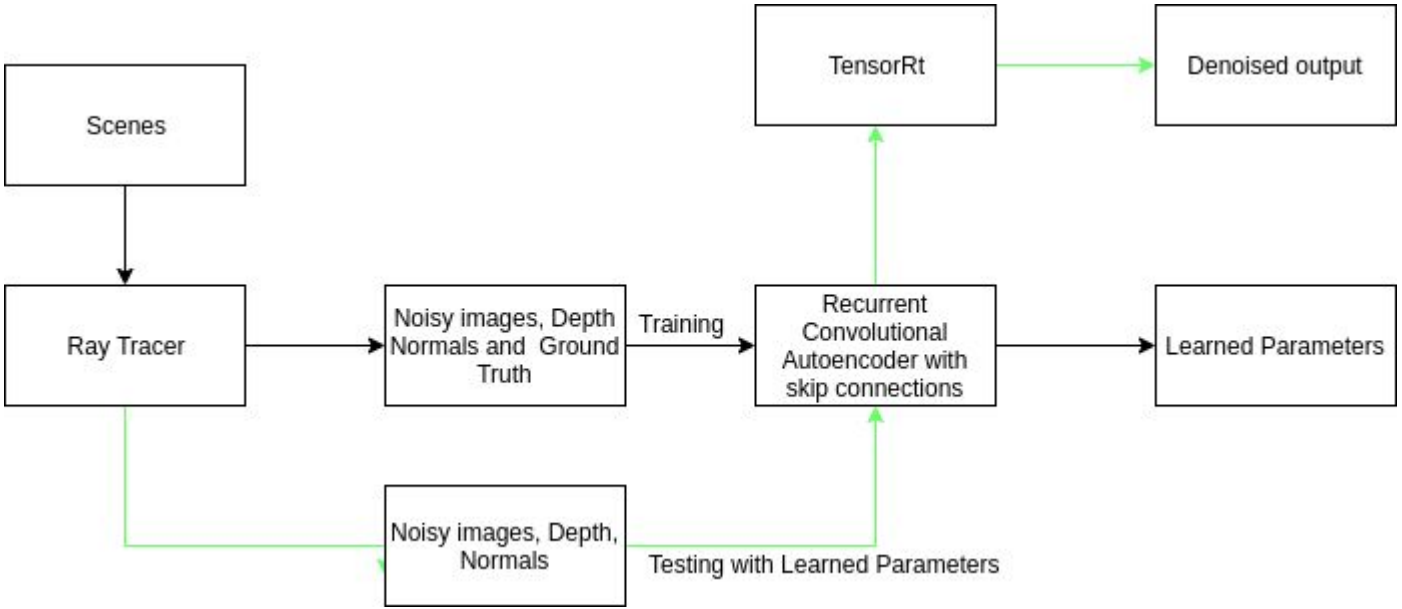
Another Image (AI) Denoiser

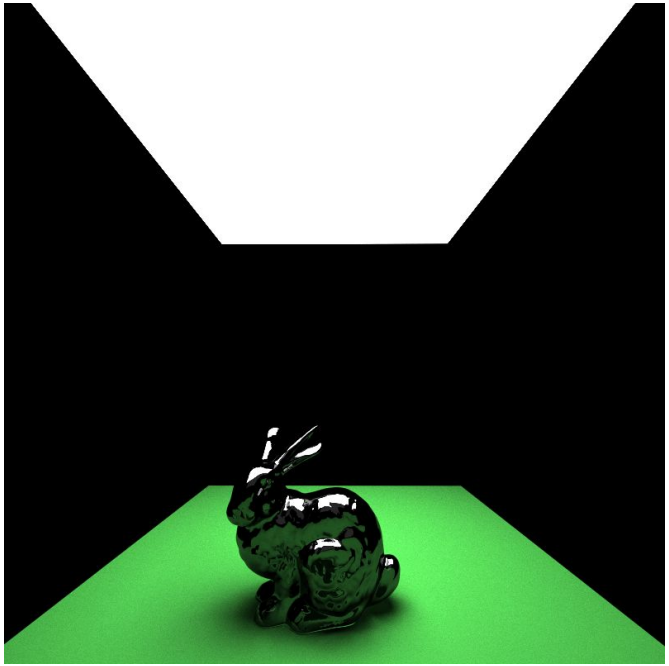
Dewang Sultania & Vaibhav Arcot

Workflow


Scenes
Ray Tracer
Noisy images, Depth Normals and Ground Truth
Training
Recurrent Convolutional Autoencoder with skip connections
TensorRt
Denoised output
Learned Parameters
Noisy images, Depth, Normals
Testing with Learned Parameters

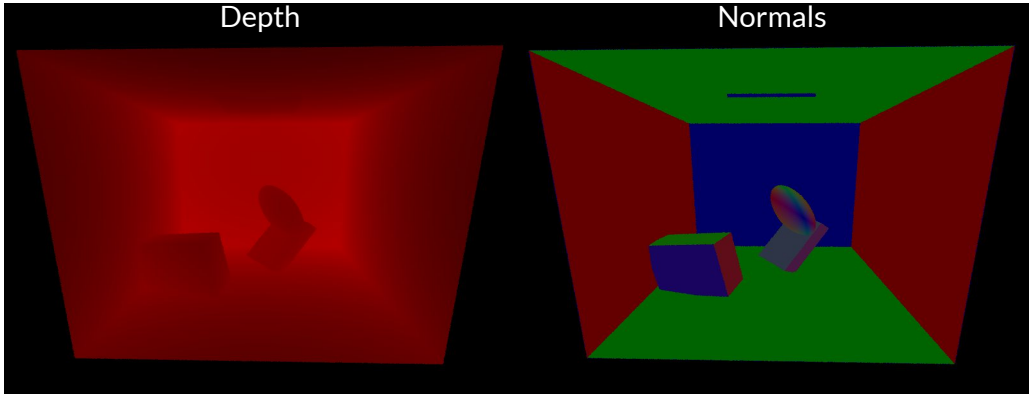
Ray Tracer

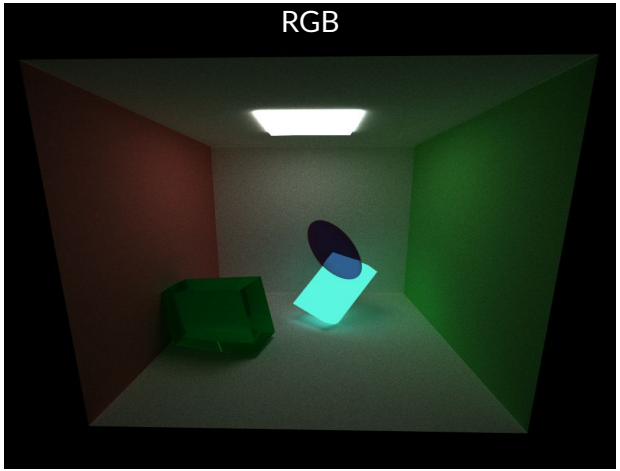
- Features completed
  - Meshes
  - Dielectrics
  - Specular reflections
  - Parsing mtl files
  - Extracting normals, albedo and depth maps
- Features in progress
  - Texture loading
  - KD Tree



Data Generation

- Procedural Data Generation
  - Reflection, Refraction
  - Scale, Translation, Rotation
  - Color, Emittance
  - Camera eye
- Generated 80Gb of data
  - 30 scenes, 2503 images per scene





Architecture - Recurrent Convolutional Autoencoder with Skip Connections

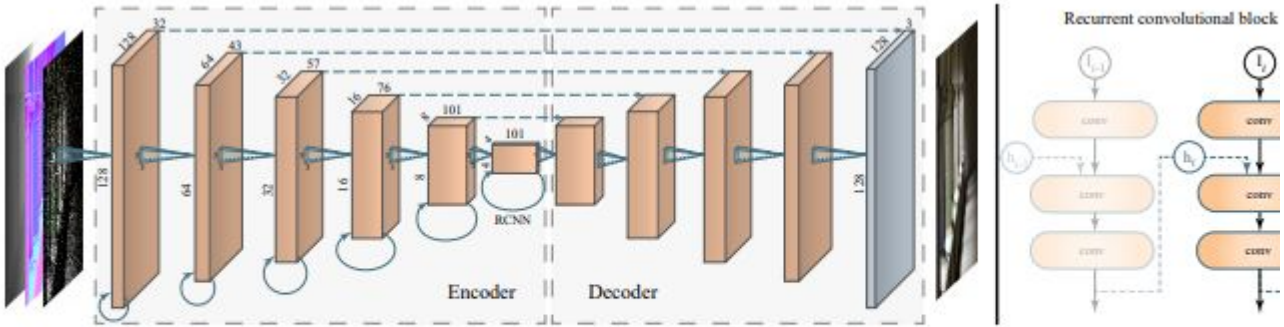


Fig. 2. Architecture of our recurrent autoencoder. The input is 7 scalar values per pixel (noisy RGB, normal vector, depth, roughness). Each encoder st
a convolution and 2×2 max pooling. A decoder stage applies a 2×2 nearest neighbor upsampling, concatenates the per-pixel feature maps fron
connection (the spatial resolutions agree), and applies two sets of convolution and pooling. All convolutions have a 3×3-pixel spatial support. On t
we visualize the internal structure of the recurrent RCNN connections. I is the new input and h refers to the hidden, recurrent state that persists b
animation frames.

- Convolutions
- Autoencoder
- Recurrent Layer
- Skip connections

Challenges

- Compute
  - Complicated architecture which requires a lot of GPU compute for training
- Data Storage:
  - 30 scenes with 2503 images each are 80 GB worth of data
  - Resizing and storing might be a solution, but that happens at the expense of performance.

Next Steps

- Benchmark PyTorch model
- Implement the neural network in cuDNN.
- Generate more scenes with textures.
- Speed up with TensorRt