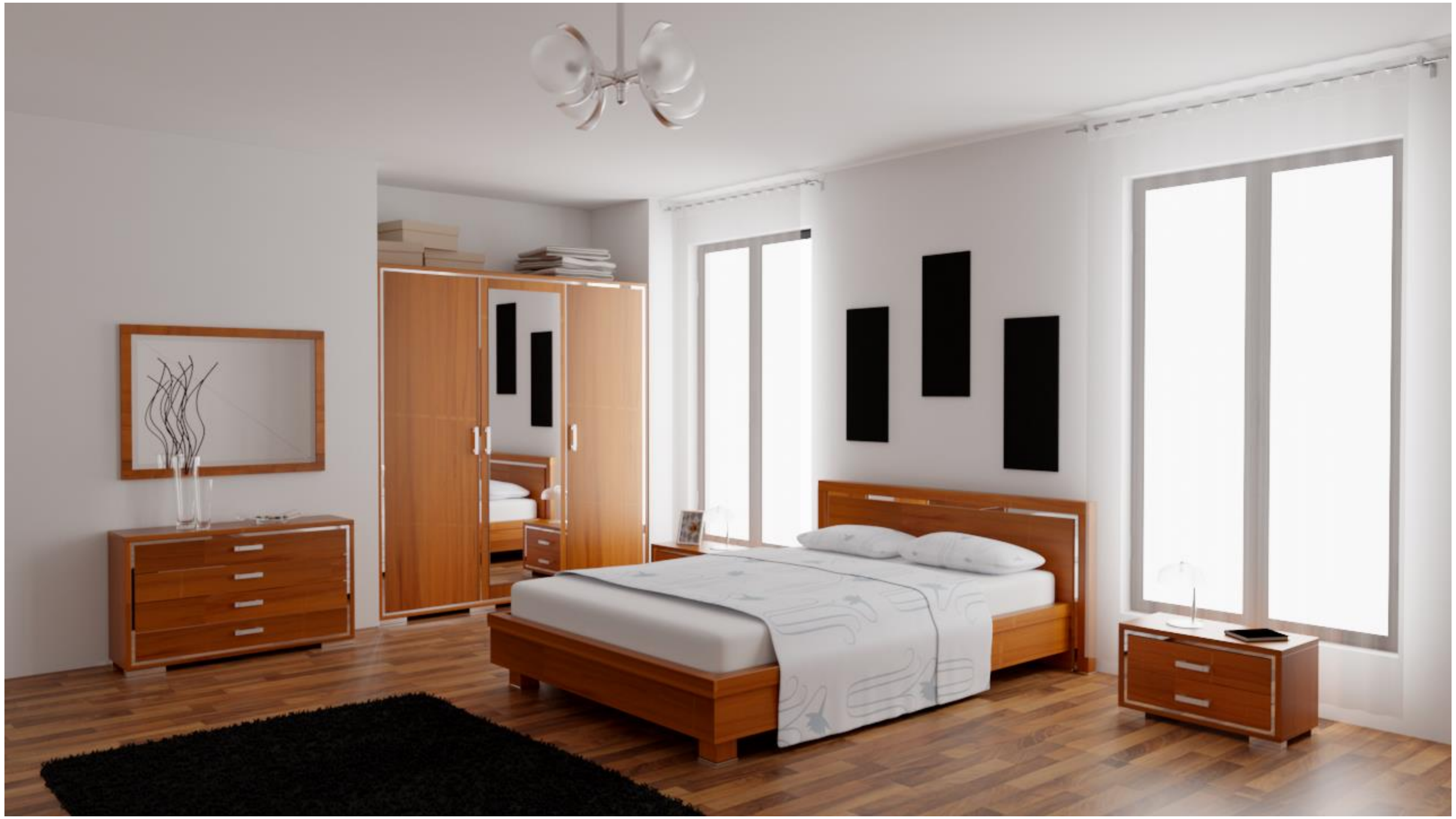
Neural Sequence Transformation

Sabyasachi Mukherjee[1] Sayan Mukherjee[2] Binh-Son Hua[3,4] Nobuyuki Umetani[1] Daniel Meister[1]

1 東京大学 THE UNIVERSITY OF TOKYO 2 bluegat 3 VinAI RESEARCH 4 VINUNIVERSITY

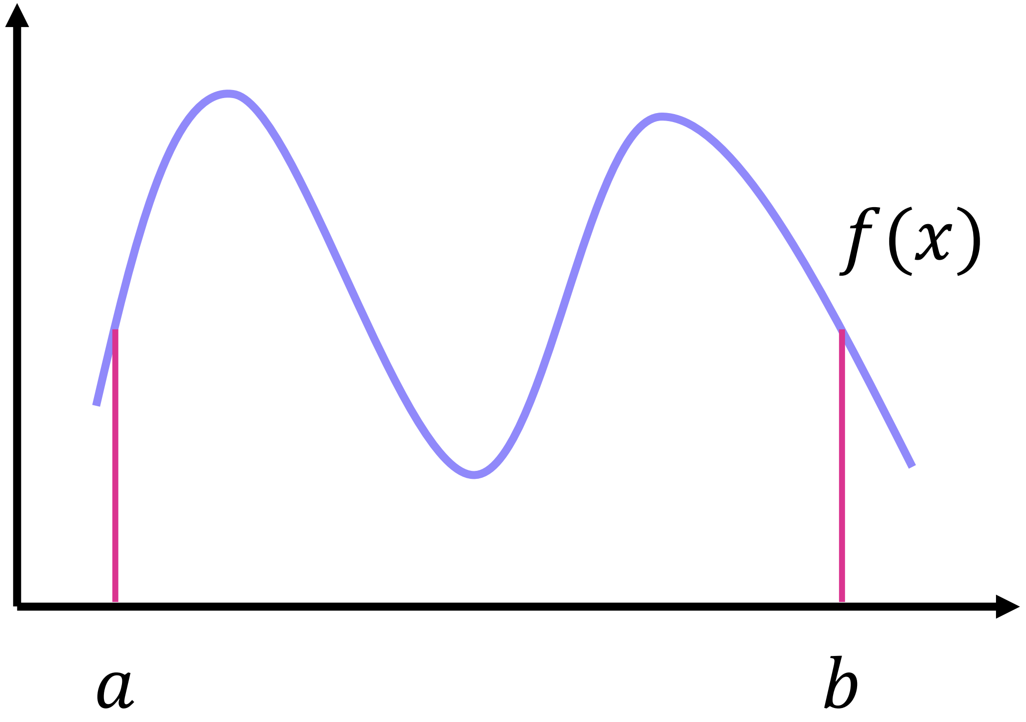
Background

Physically Based Rendering



https:// benedikt-bitterli.me/resources/images/bedroom.png

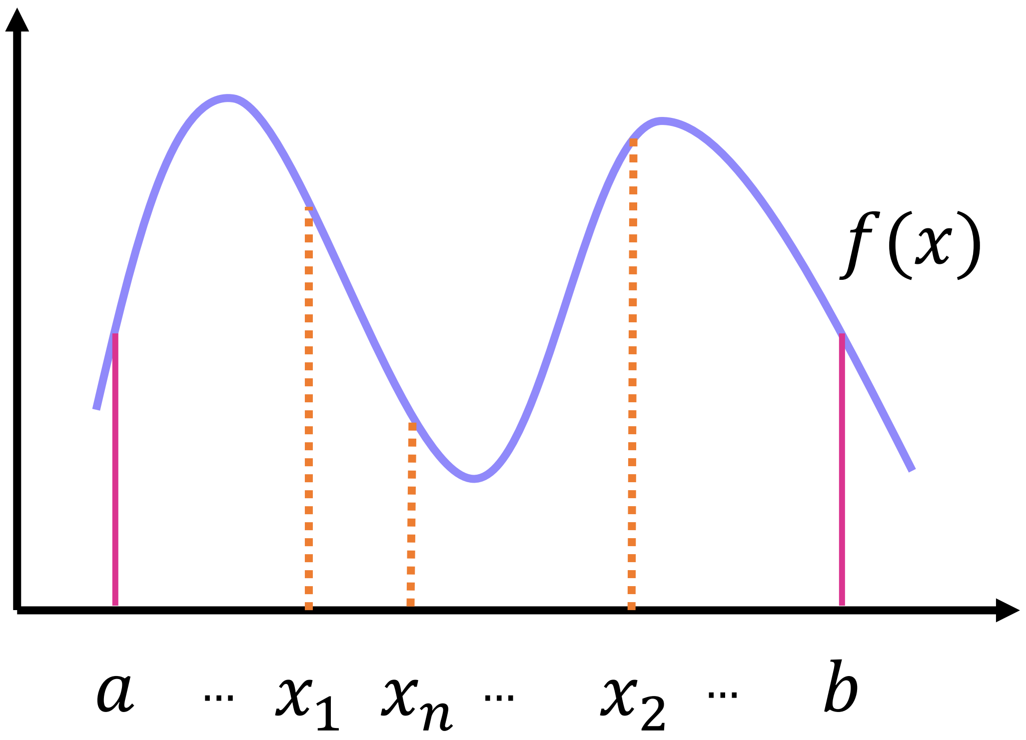
Monte Carlo Integration



We want to find

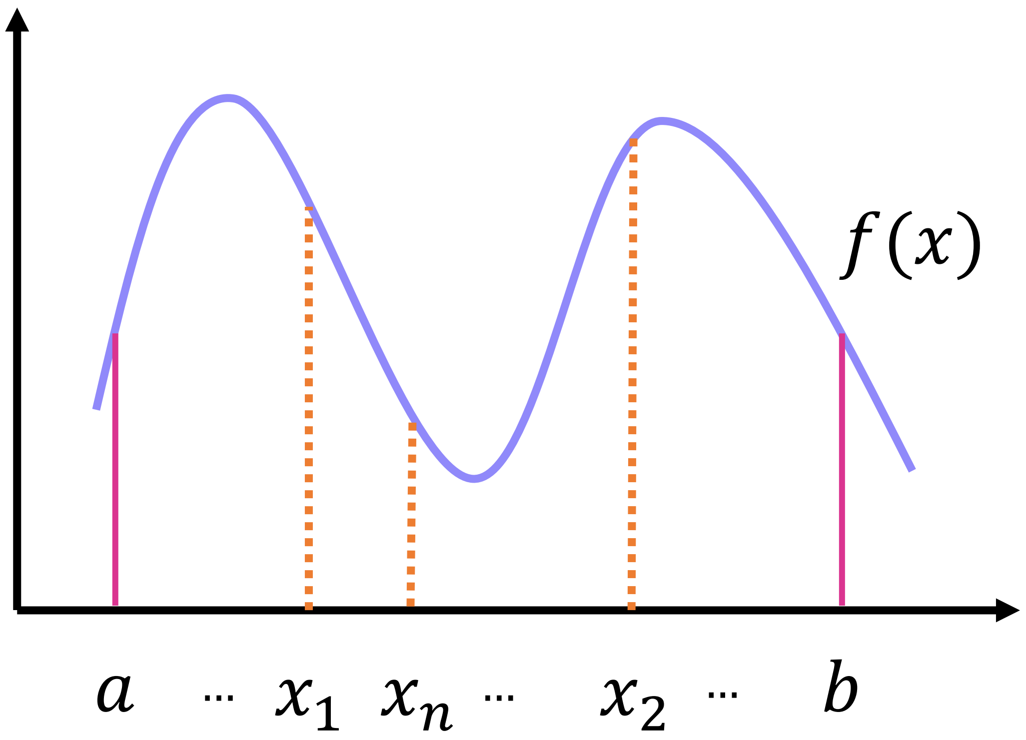
$$\int_a^b f(x)dx$$

Monte Carlo Integration



1. Draw n samples randomly in (a, b)

Monte Carlo Integration



2. Calculate

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

$$\to \int_a^b f(x)dx \;\; as\, n \to \infty$$

Disadvantages of Monte Carlo Integration

- Converges at a slow rate of $O(\sqrt{n})$
- We propose to improve convergence using Sequence Transformation

Sequence Transformation

- A mapping $\mathcal{T}$ from a sequence (s_n) to another sequence (t_n)

$$\left(1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \dots\right)$$

$$\Big\downarrow \quad \mathcal{T} \coloneqq t_n = s_n^2$$

$$\left(1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}, \dots\right)$$

Sequence Transformation

Example: Aitken's Δ^2 Process

$$t_n = s_n - \frac{(s_{n+1} - s_n)^2}{s_{n+2} - 2s_{n+1} + s_n}$$

Background

$$I = \int_0^1 f(x)dx = \int_0^1 e^{-x^2}dx \approx 0.746824$$

$\hat{I}_1 =$ 0.991071

$\hat{I}_2 =$ 0.990075

0.874699

… 0.757849

… 0.767839

… …

$\hat{I}_{16} =$ 0.746318

Motivation

$$I = \int_0^1 f(x)dx = \int_0^1 e^{-x^2} dx \approx 0.746824$$

0.991071 , 0.990075 , 0.874699 , 0.791553 , 0.767839 , … , 0.746318

- Monte Carlo integration → a sequence of terms converging to I
- Apply sequence transformation methods to Monte Carlo integration

Related Work: $a_n g_n$ transformation [BZ91]

$$S_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

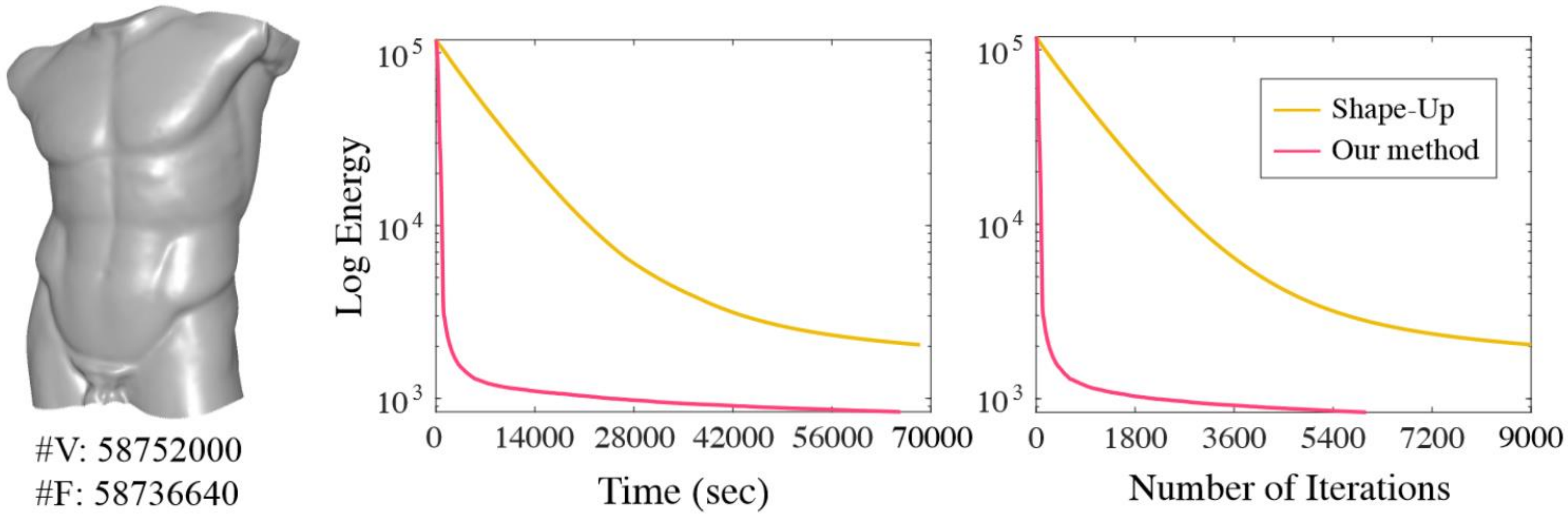
- Transformation:

$$T_{n+1} = S_n - \frac{S_{n+1} - S_n}{g_{n+1} - g_n} g_n$$

- g_n is arbitrary function of n
- However, we show $(\mathrm{T_n})$ has slightly *higher* variance than (S_n) regardless of choice of g_n

Related Work: Using Anderson Acceleration[PDZ*18]

- Showed improvements in geometry processing and physics simulation using Anderson Acceleration [And65]
- However, applicable to fixed point methods
- Monte Carlo integration is difficult to formulate as one



Peng et al. Anderson Acceleration for Geometry Optimization and Physics Simulation, 2018.

Our Method

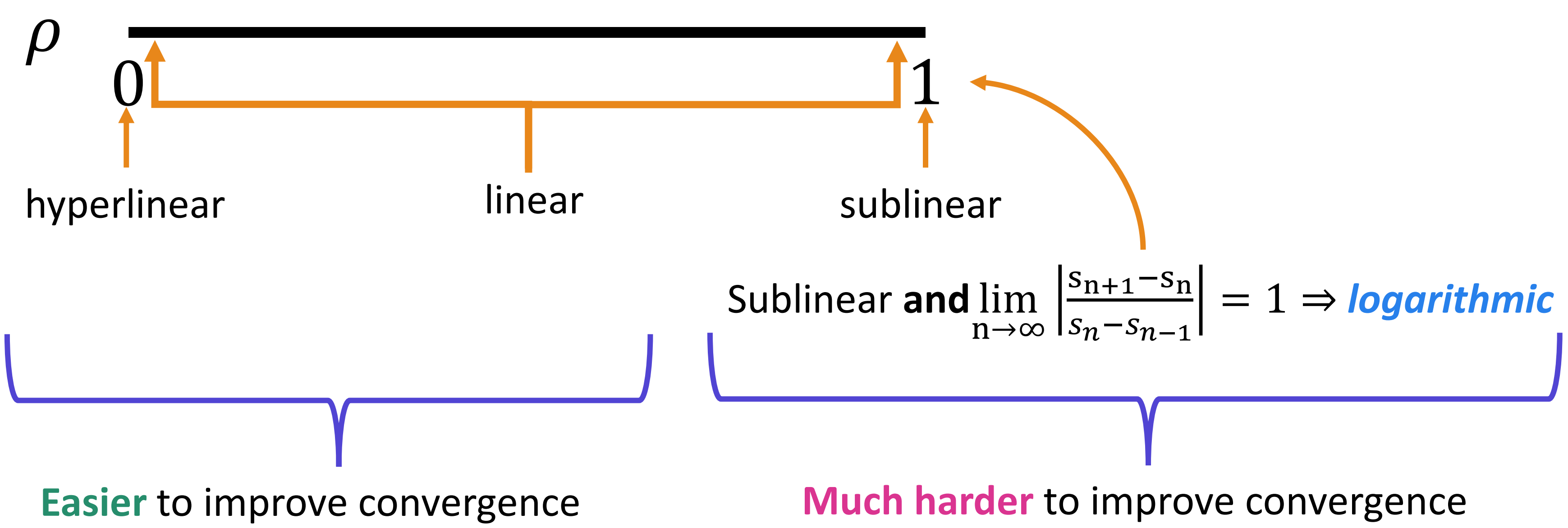
Background:
Analysis of Monte Carlo convergence

- There is no universal sequence transformation method for all sequences [DGB82]
- Determine *type of convergence* before applying sequence transformation methods

Our Method

Type of Convergence [Wen89]

$$\rho = \lim_{n\to\infty} \left| \frac{s_{n+1} - s}{s_n - s} \right|$$



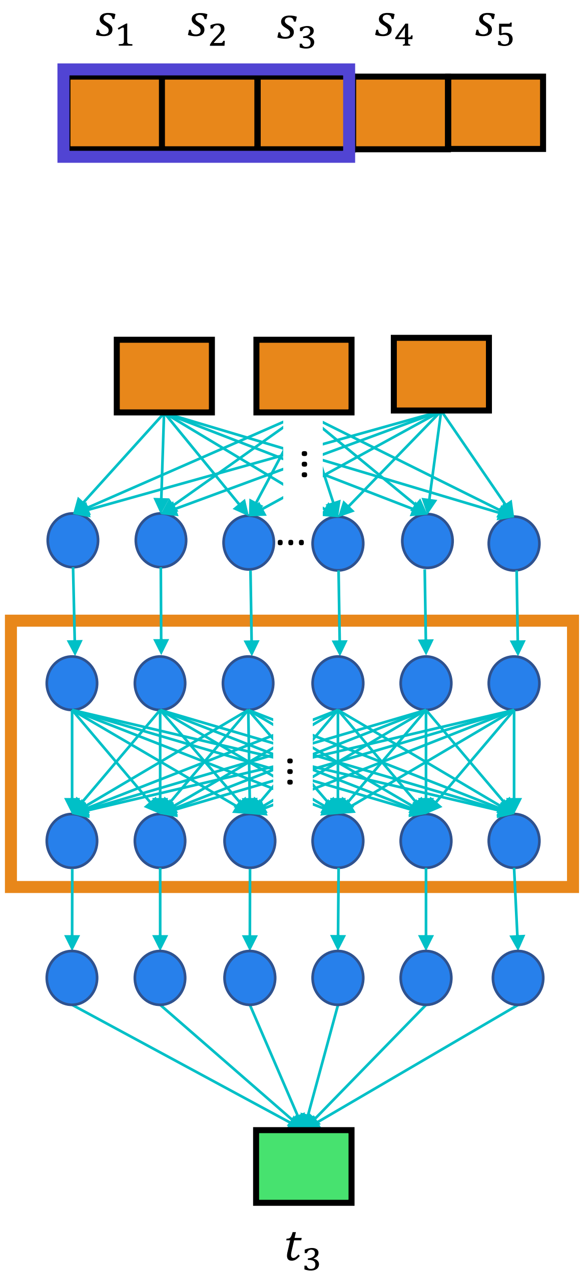
Contribution 1: Analysis of Monte Carlo convergence

- *Type of convergence* is defined for deterministic sequences only
- We show that Monte Carlo estimates converge like a logarithmic sequence

Contribution 2: A data-driven approach to learn sequence transformation

- A simple MLP architecture is proposed
  - Pass in sequence values in a sliding window fashion
  - Output of the network is a single value per sliding window

Our Method

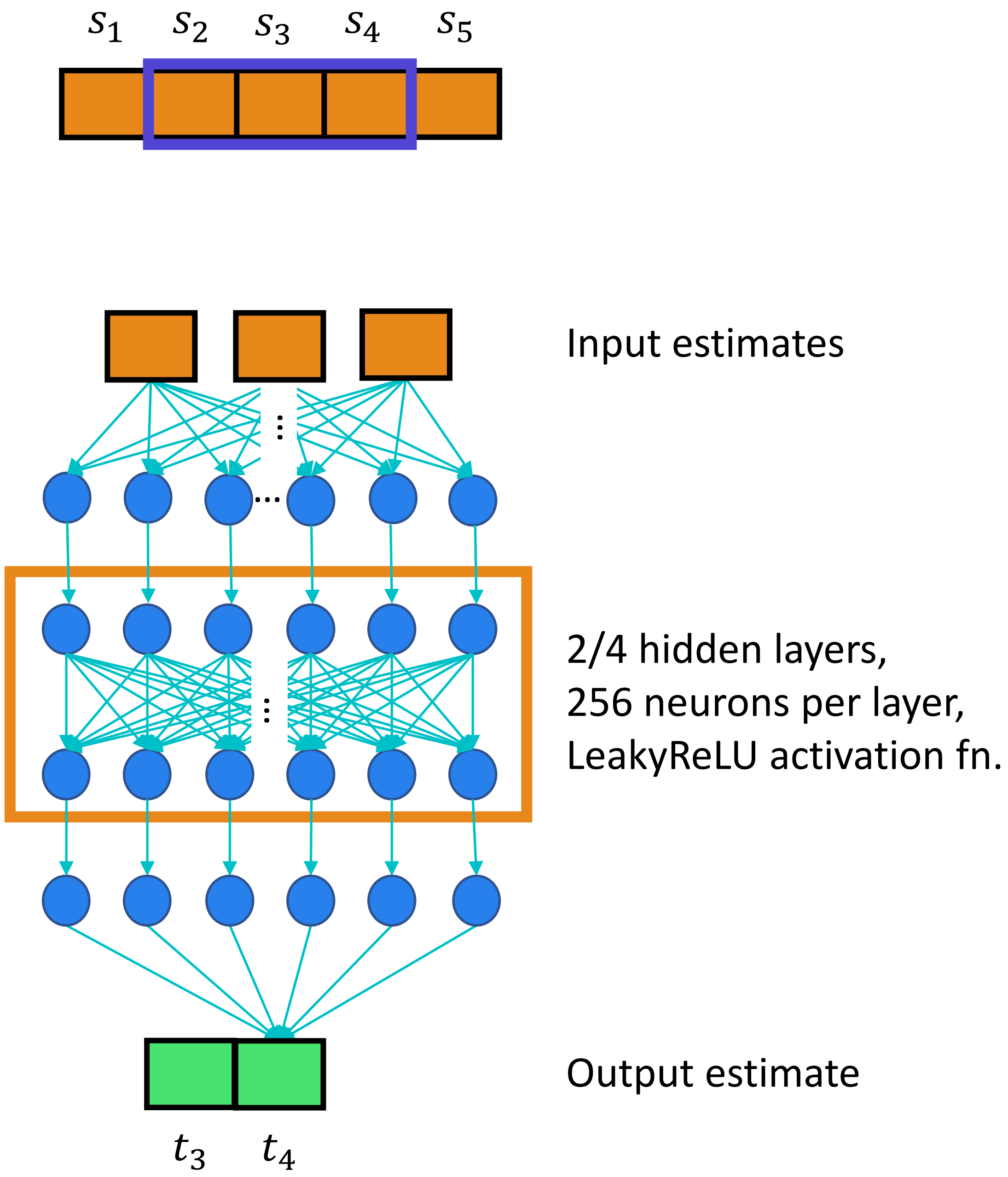


Input estimates

2/4 hidden layers,
256 neurons per layer,
LeakyReLU activation fn.

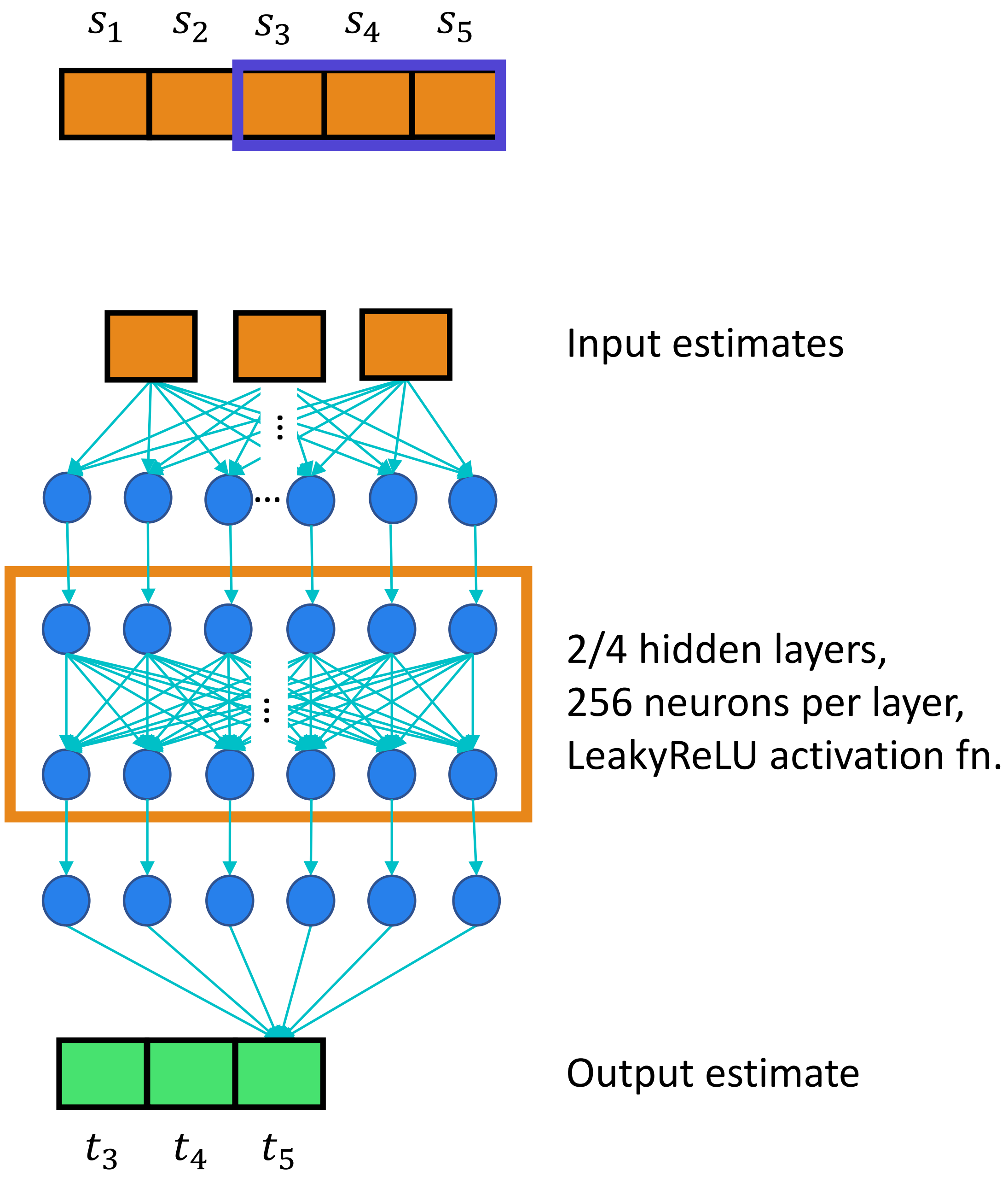
Output estimate

Our Method

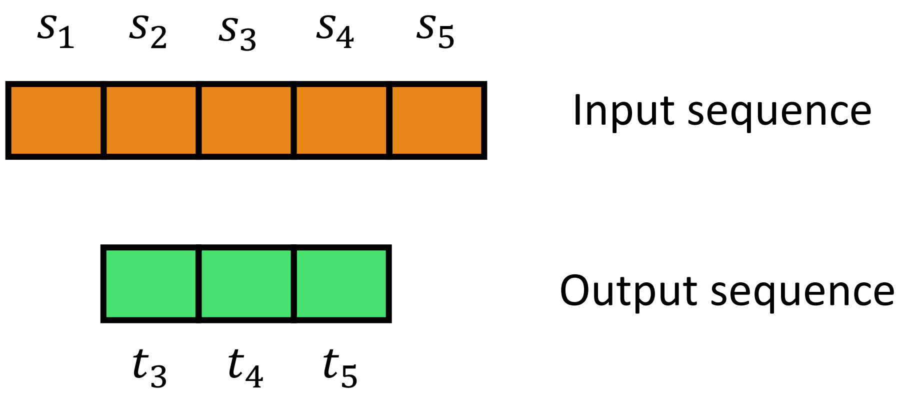


s_1 s_2 s_3 s_4 s_5
Input estimates
2/4 hidden layers,
256 neurons per layer,
LeakyReLU activation fn.
Output estimate
t_3 t_4

Our Method


s_1 s_2 s_3 s_4 s_5
Input estimates
2/4 hidden layers,
256 neurons per layer,
LeakyReLU activation fn.
Output estimate
t_3 t_4 t_5

Our Method

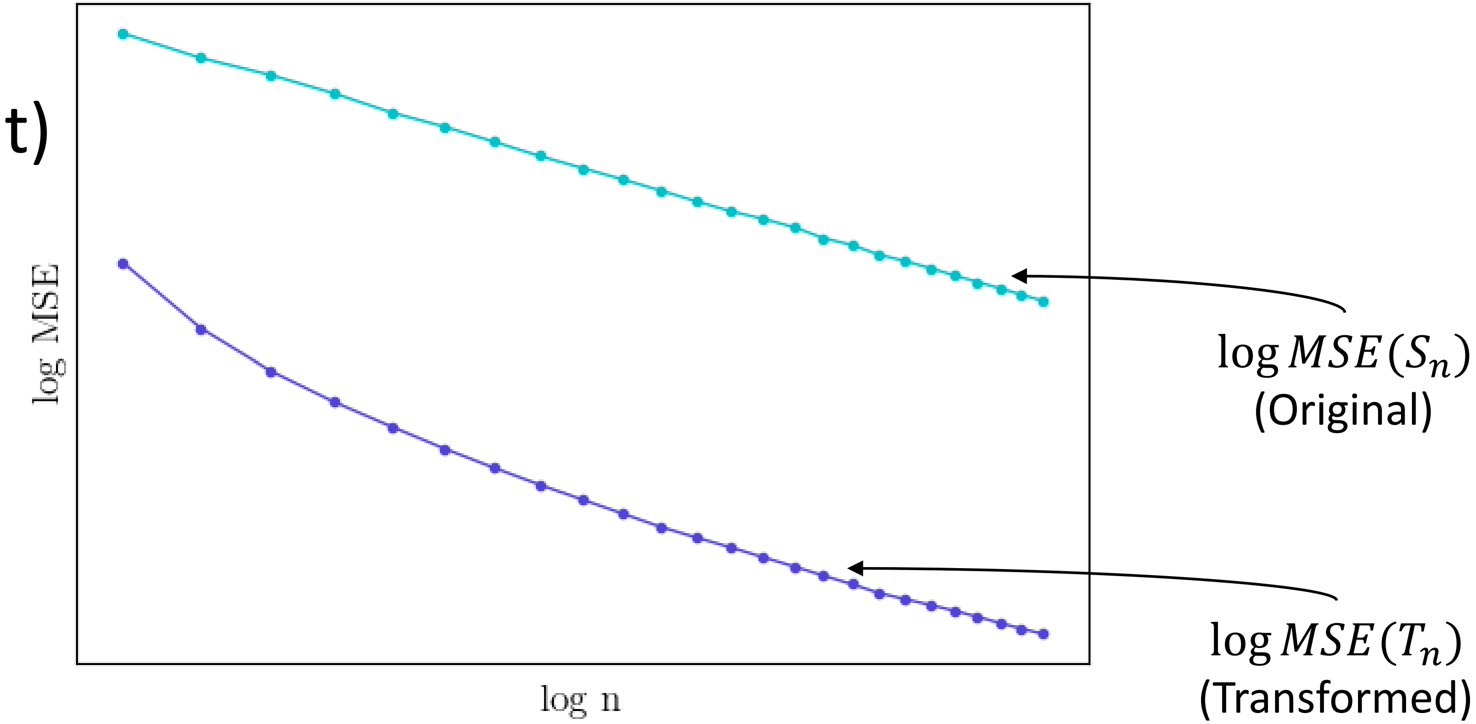


Contribution 3: Loss function

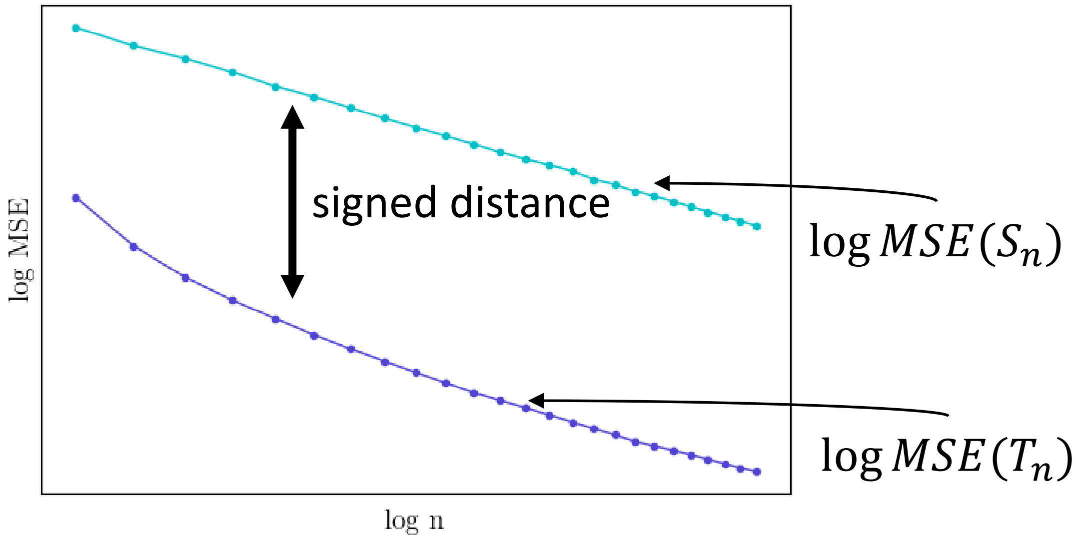
- We propose a novel loss function tailored to Monte Carlo integration
- Output sequence requirements:
  - Must have lower error than input sequence
  - Should converge at a faster rate than input sequence

Convergence Graphs

- Plot log(MSE) vs. log(Sample Count)



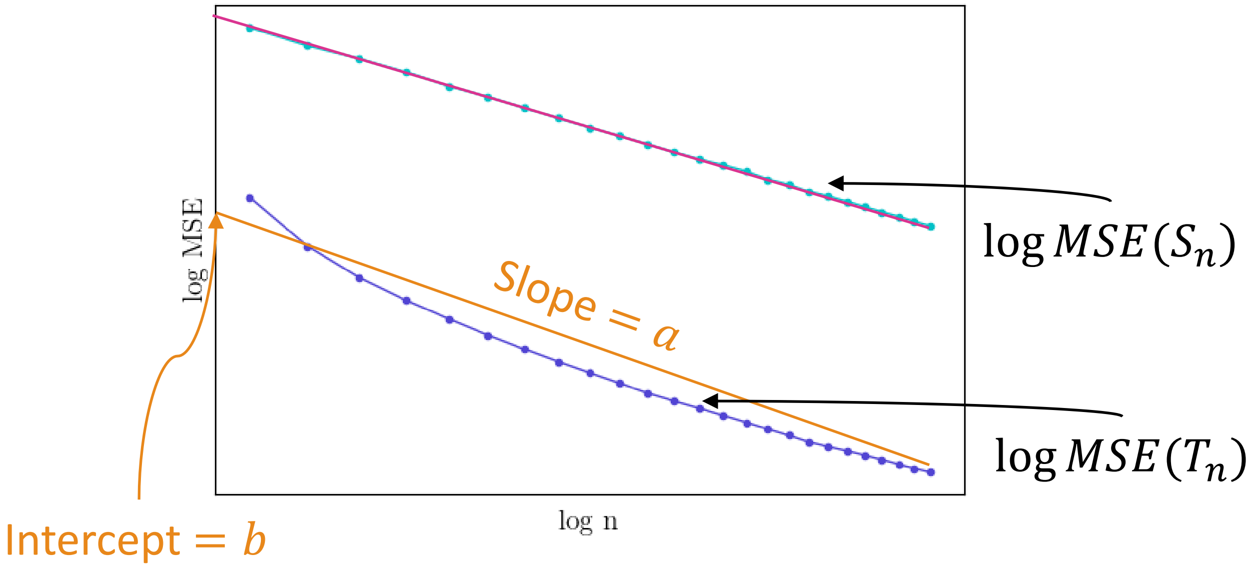
Loss Function Requirement 1: Lower Error



- Minimize the total signed distance between the output and input sequences:

$$\log(MSE(T_n)) - \log(MSE(S_n))$$

Loss Function Requirement 2: Faster Convergence



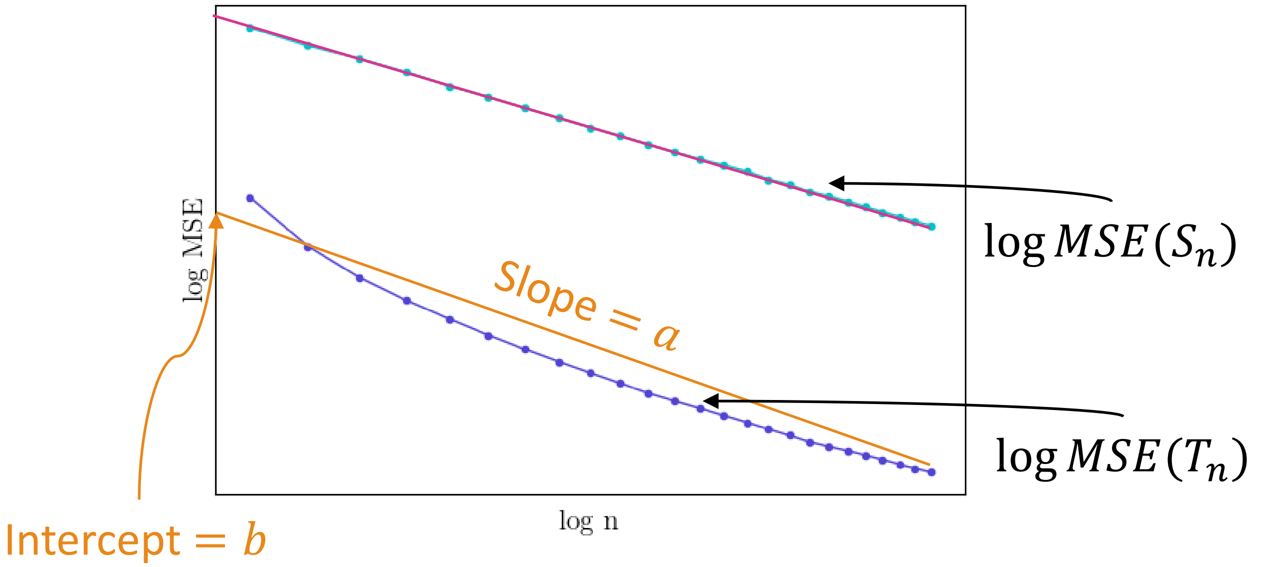
Faster convergence :=

Better "slope" of $\log MSE(T_n)$

$$\log MSE(T_n) \approx a \log n + b$$

slope (↑ a) intercept (↑ b)

Loss Function Requirement 2: Better Slope



Minimize $a \log n + b, n$ small $\Rightarrow$ b dominates

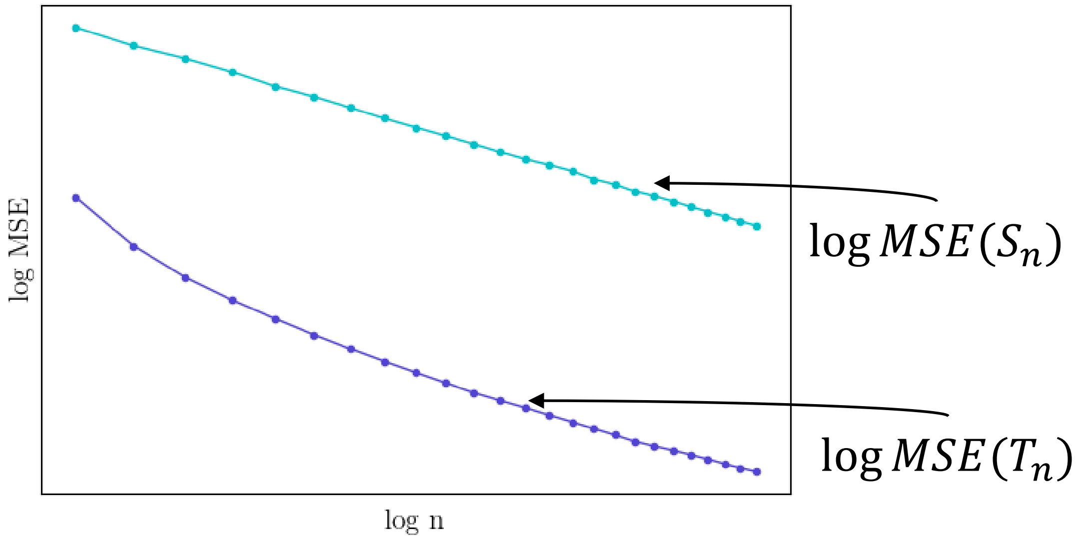
Want optimizer to make a more negative

Proposal: minimize $a + \frac{b}{\log n}$ instead

$\Rightarrow$ Minimize $\frac{\log MSE(T_n)}{\log n}$

(Since $\log MSE(T_n) \approx a \log n + b$)

Loss Function Requirement 2: Better Slope



Proposed loss function:

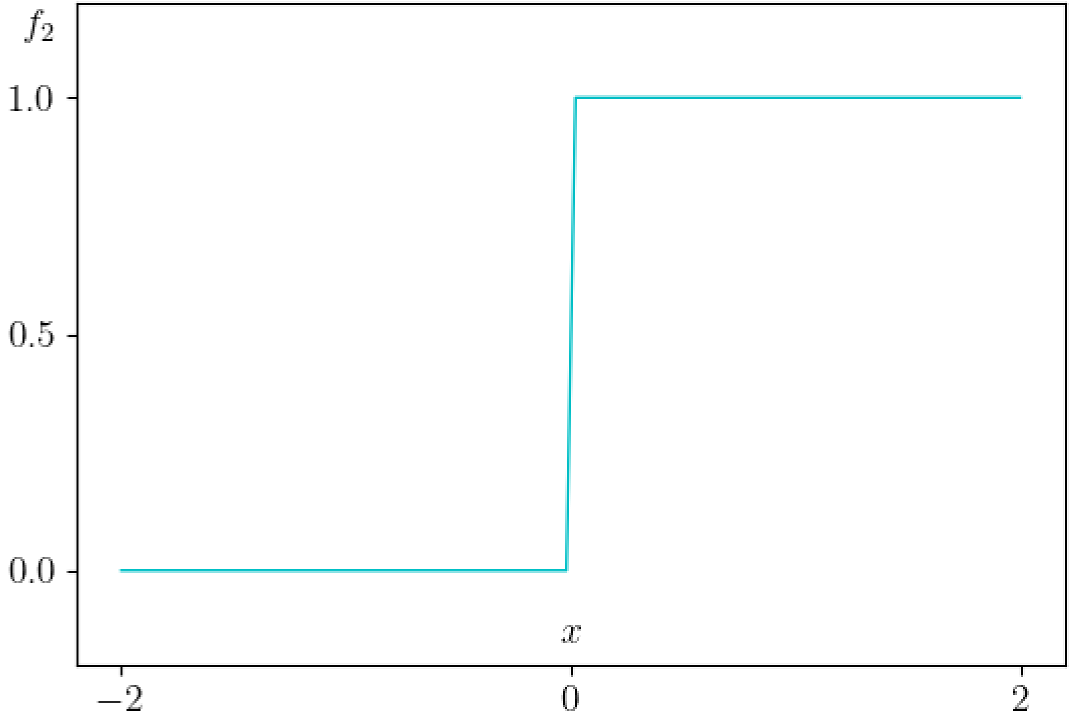
$$\mathcal{L}_n = \frac{\log MSE(T_n)}{\log n} - \log MSE(S_n)$$

A Quick Recap

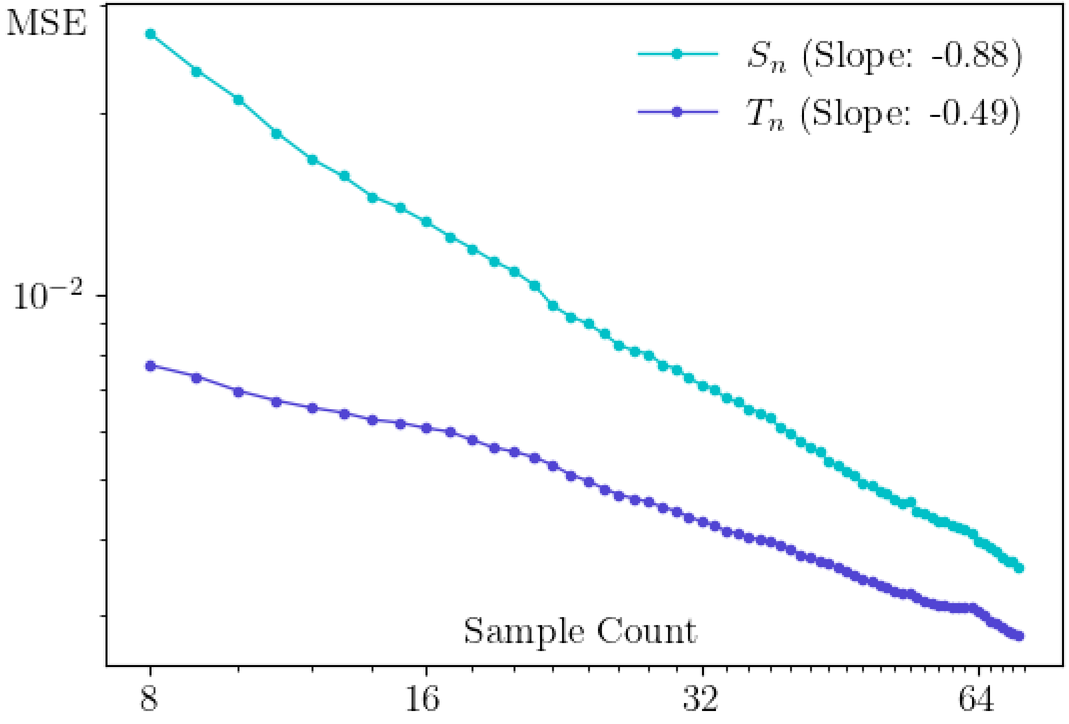
- Monte Carlo integration has slow convergence and can be viewed as a sequence
- We consider a data-driven approach to sequence transformation to improve it
- We design a neural network to learn sequence transformation
- Now we apply our method to 1D integrals and images

Results

1D Result: Step Function

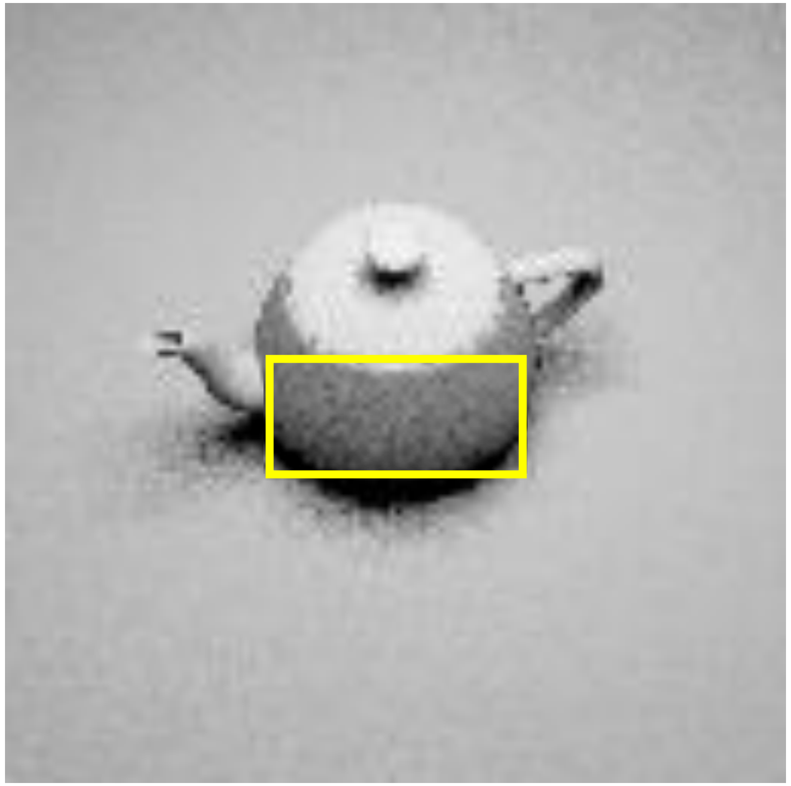


Shape of the Step function

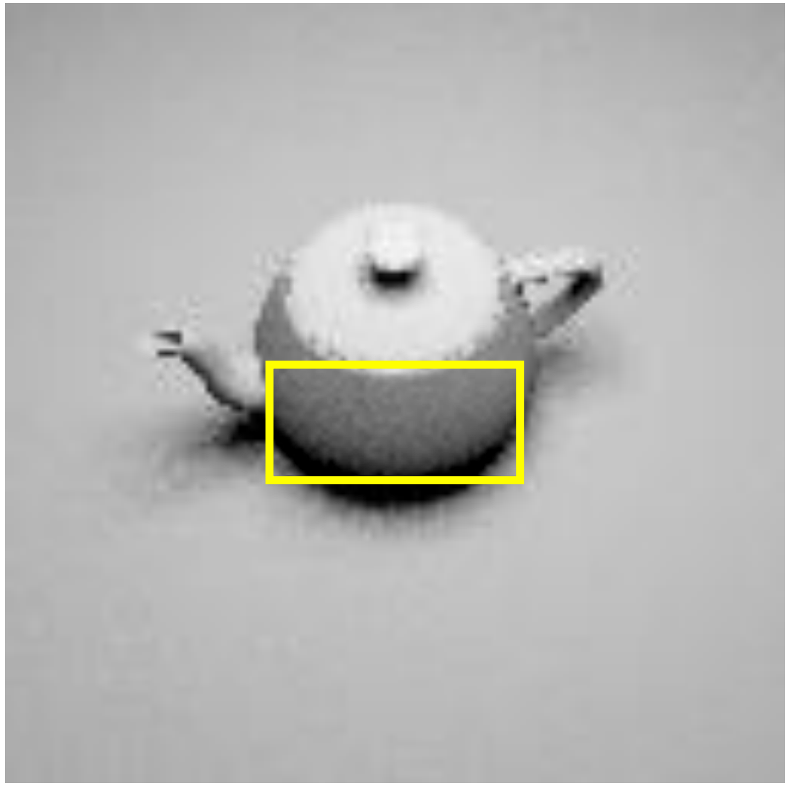


Convergence graph of Step function

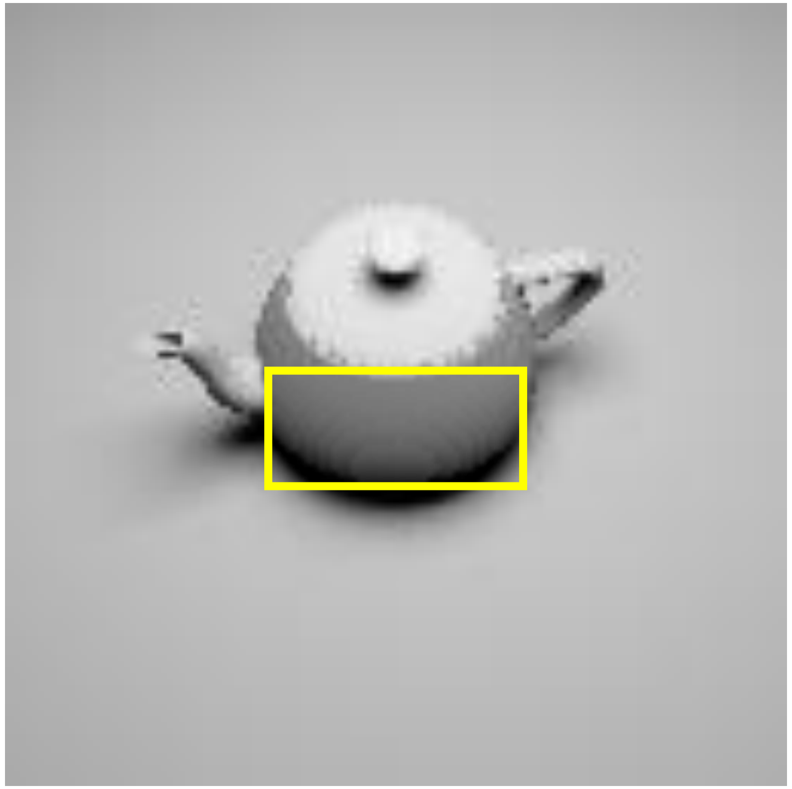
Results: Diffuse Teapot Scene



Input, 8 samples per pixel



Output, 8 samples per pixel

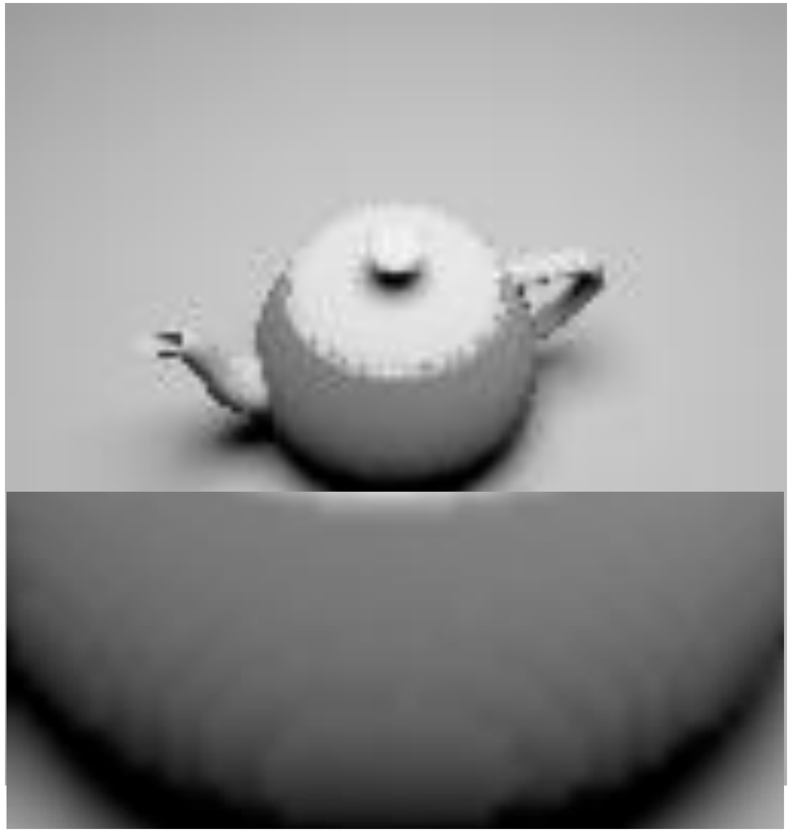


Reference, 8192 samples per pixel

Results: Diffuse Teapot Scene

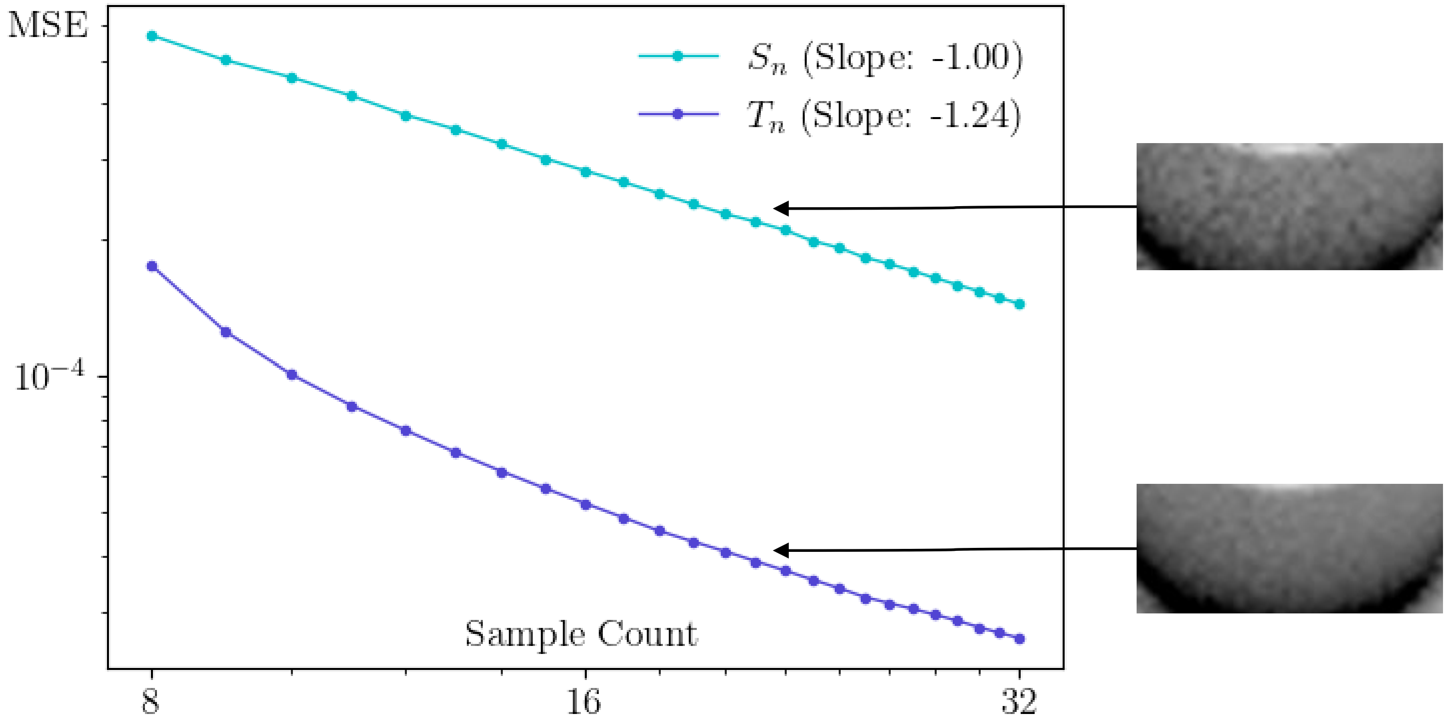
Input, 8 samples per pixel

Output, 8 samples per pixel



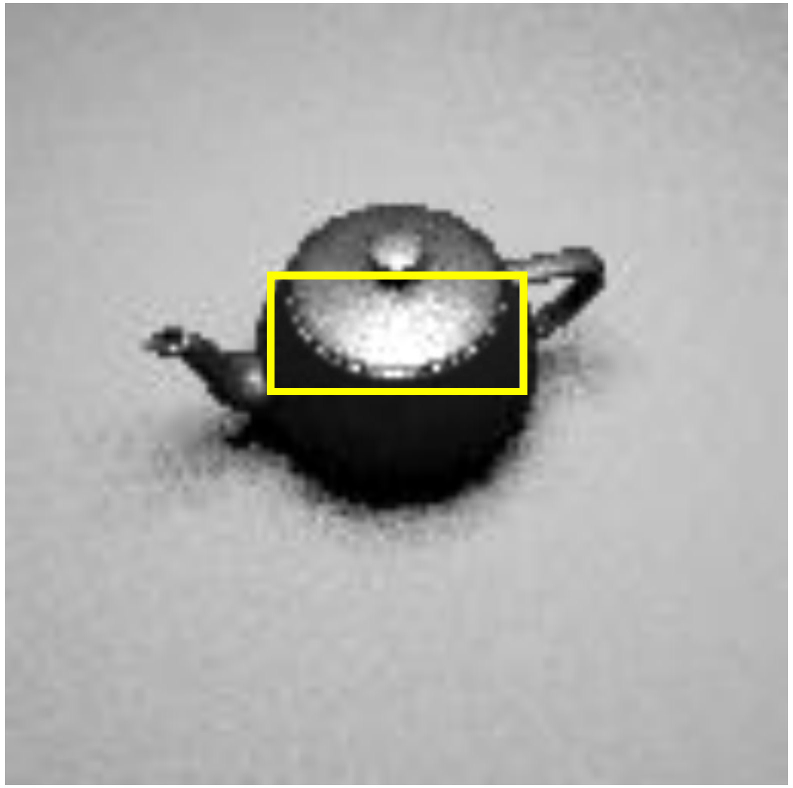
Reference, 8192 samples per pixel

Results: Diffuse Teapot Scene

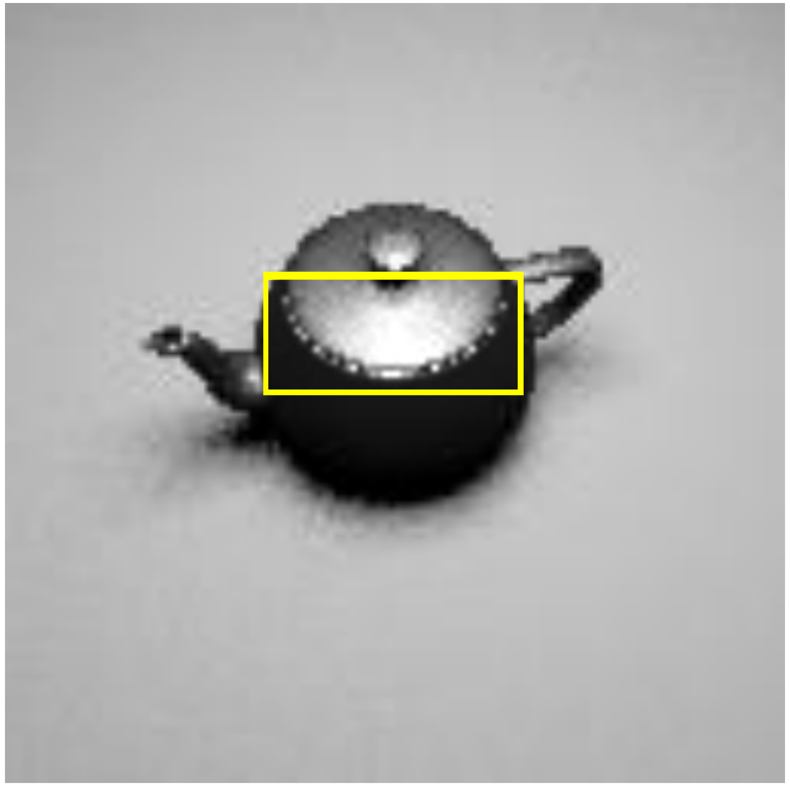


Convergence graph of Diffuse Teapot scene

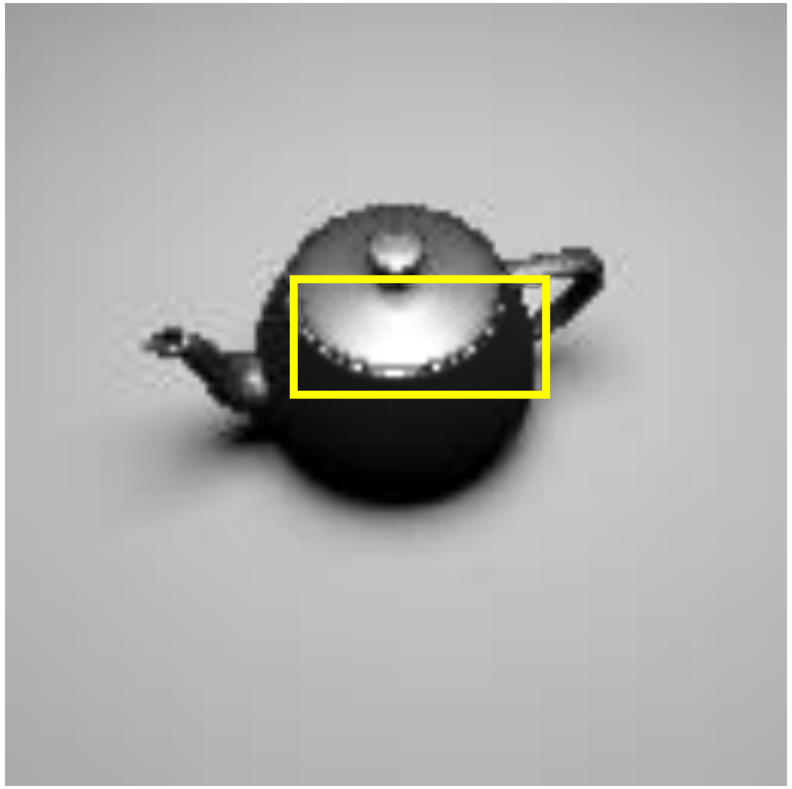
Results: Glossy Teapot Scene



Input, 8 samples per pixel

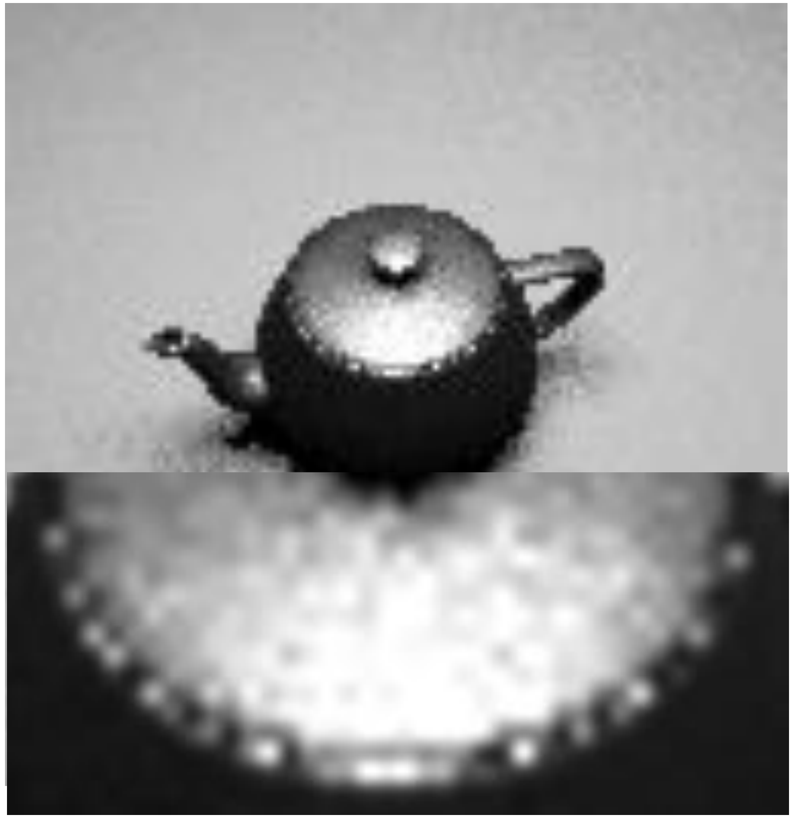


Output, 8 samples per pixel

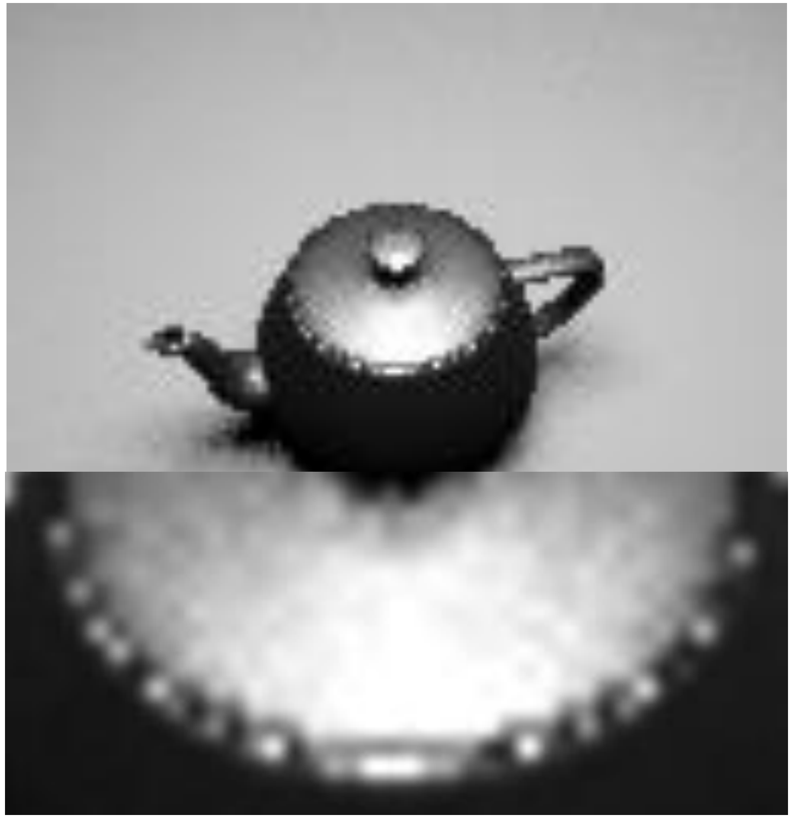


Reference, 8192 samples per pixel

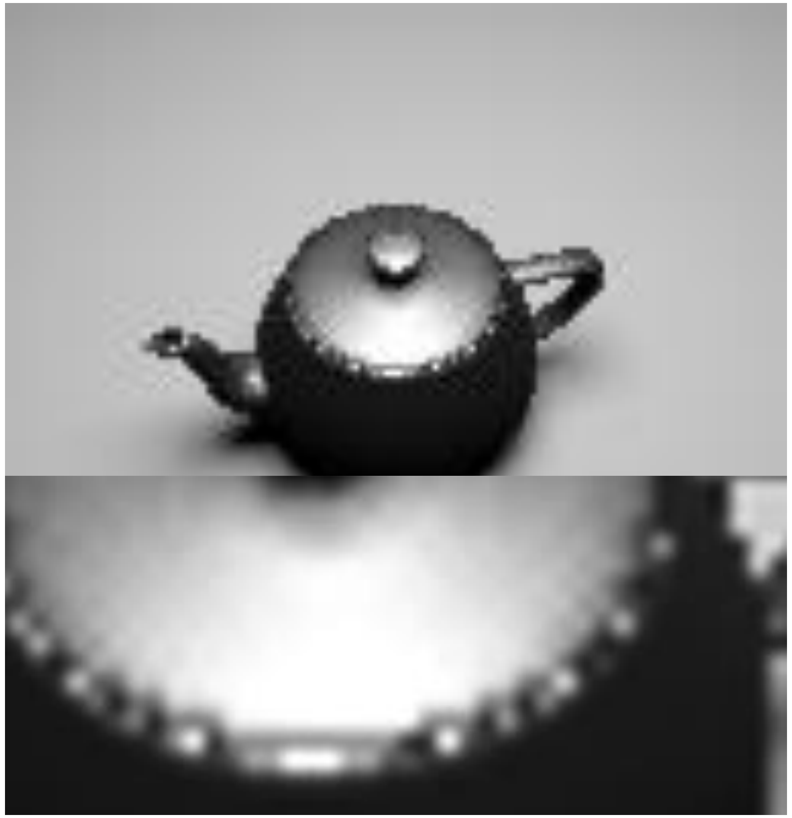
Results: Glossy Teapot Scene



Input, 8 samples per pixel

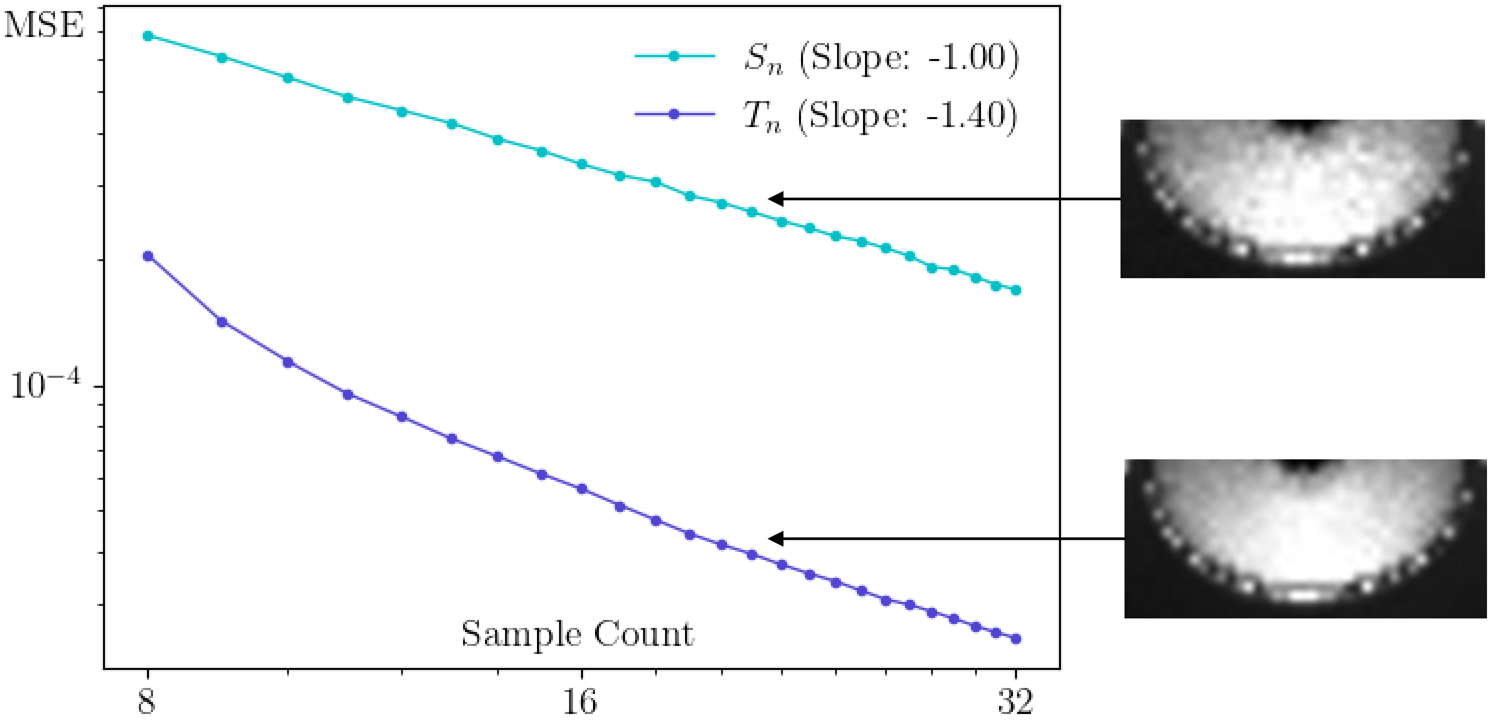


Output, 8 samples per pixel



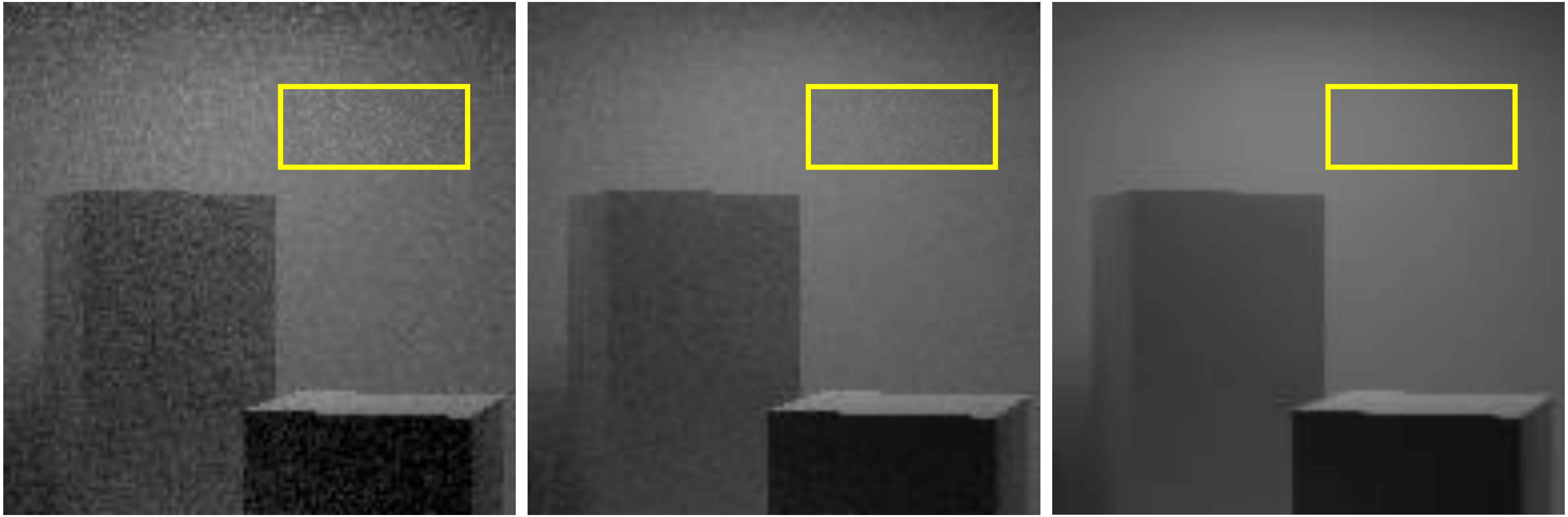
Reference, 8192 samples per pixel

Results: Glossy Teapot Scene



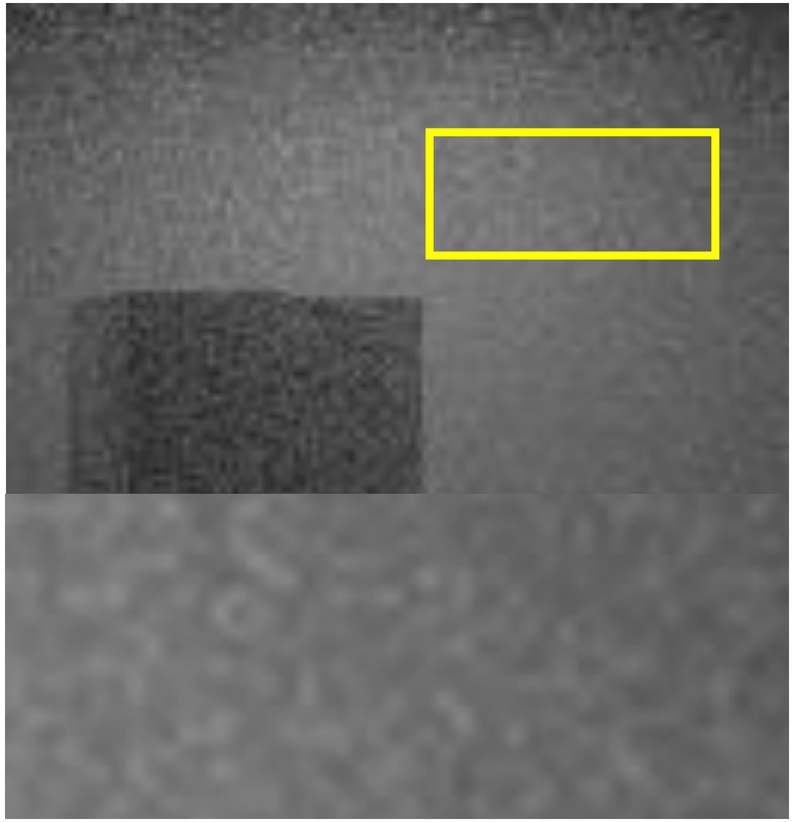
Convergence graph of Glossy Teapot scene

Results with Denoiser: Cornell Box Scene

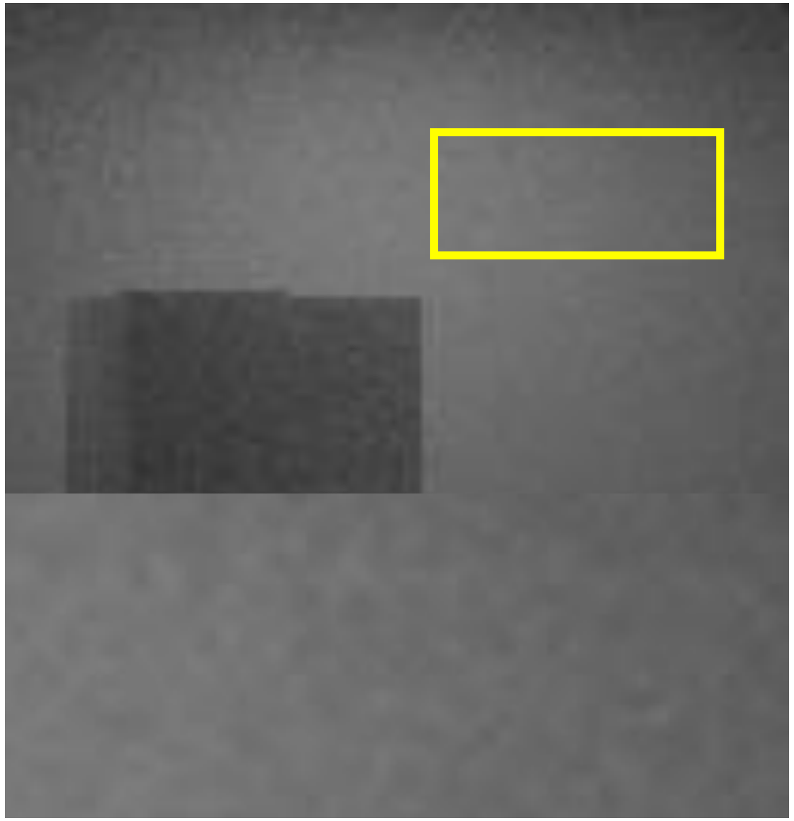


Input, 8 samples per pixel   Output, 8 samples per pixel   Denoised output, 8 samples per pixel

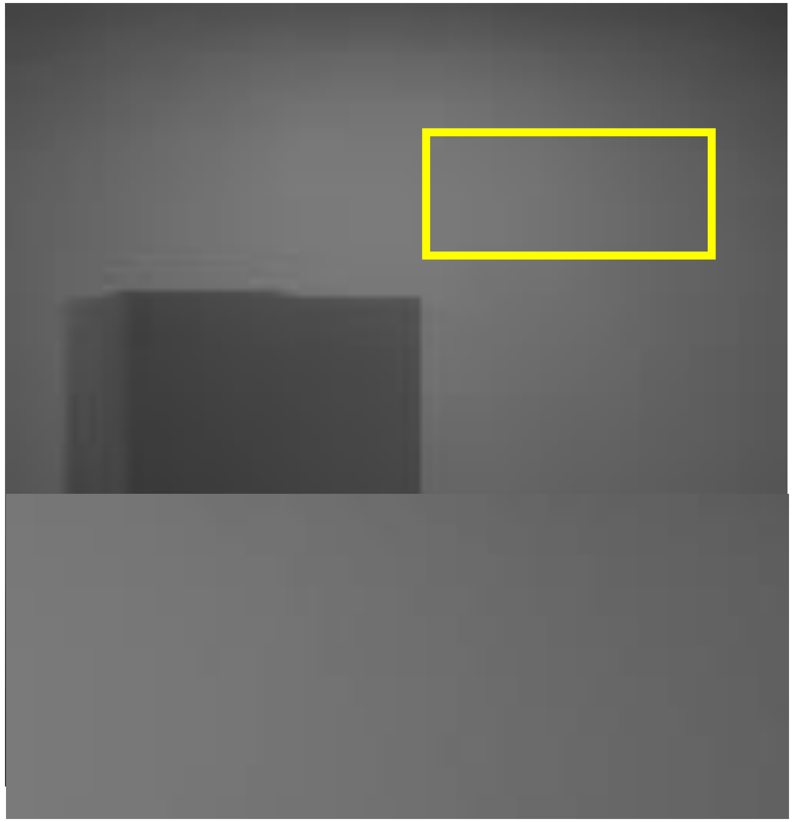
Results with Denoiser: Cornell Box Scene



Input, 8 samples per pixel

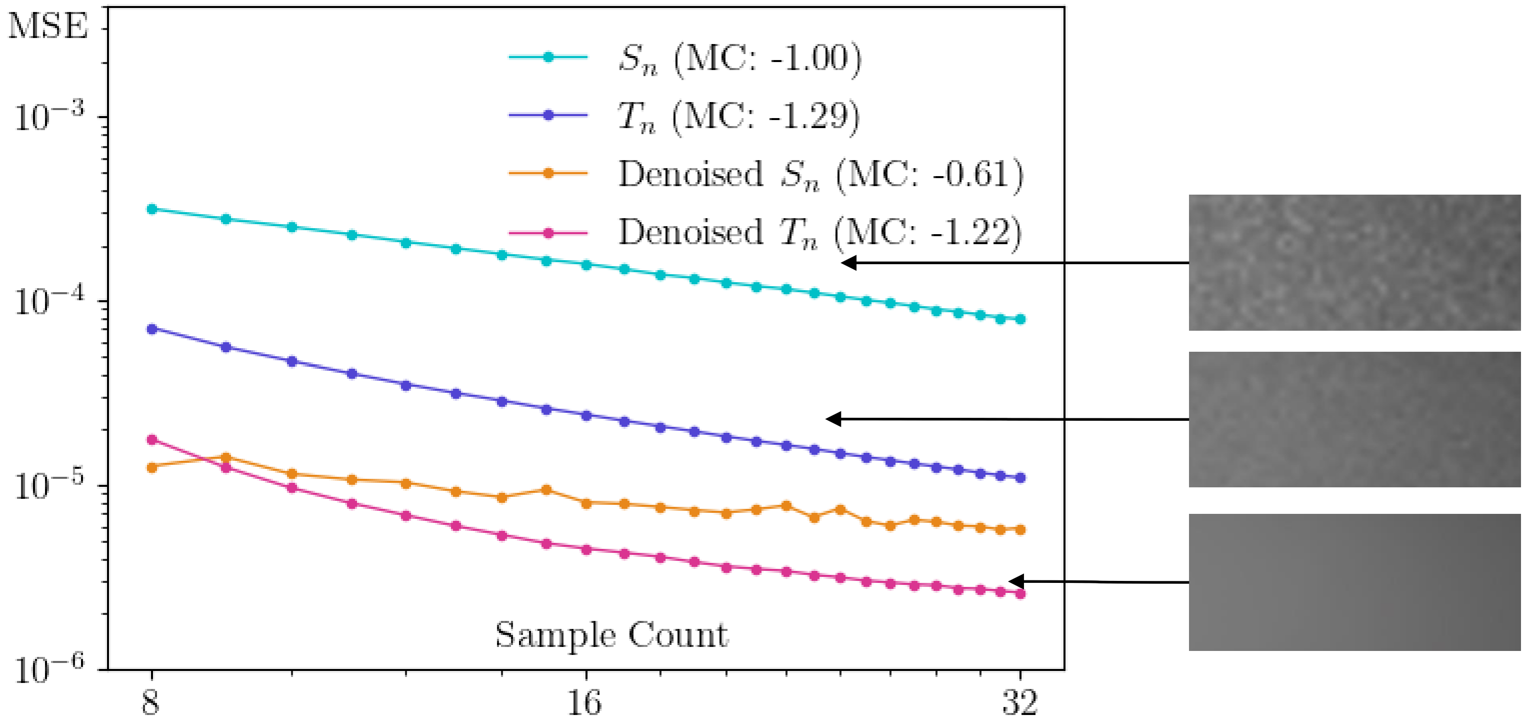


Output, 8 samples per pixel



Denoised output, 8 samples per pixel

Results with Denoiser: Cornell Box Scene



Convergence graph of Cornell Box scene

Summary and Future Work

- Monte Carlo estimates converge close to logarithmic rate.
- Proposed a data-driven neural network approach to learn a sequence transformation that can improve Monte Carlo integration.
- Proposed a custom loss function tailored to Monte Carlo integration.
- Obtained improvements for both 1D integrals and rendered images.

- Make our method real-time
- Explore possibilities of application along with analytical sequence transformations

Neural Sequence Transformation

Sabyasachi Mukherjee[1] Sayan Mukherjee[2] Binh-Son Hua[3,4] Nobuyuki Umetani[1] Daniel Meister[1]

1 東京大学 THE UNIVERSITY OF TOKYO

2 blueqat

3 VinAI RESEARCH

4 VINUNIVERSITY