Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks

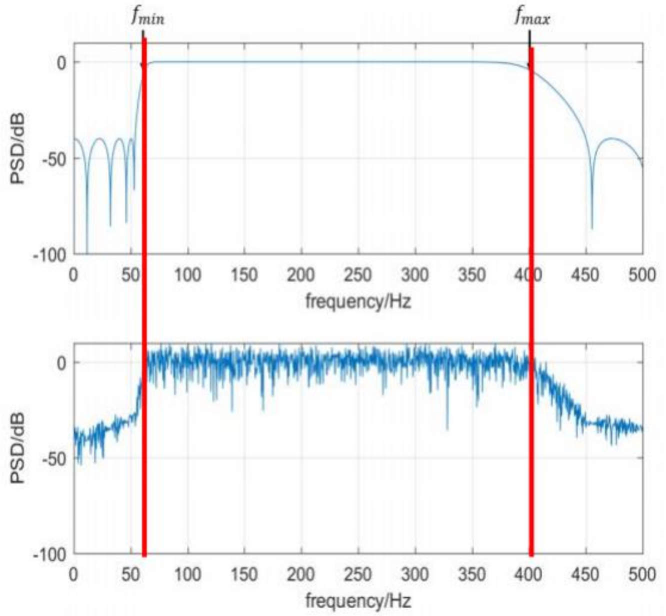
SIMON KALLWEIT, Disney Research and ETH Zürich et al.

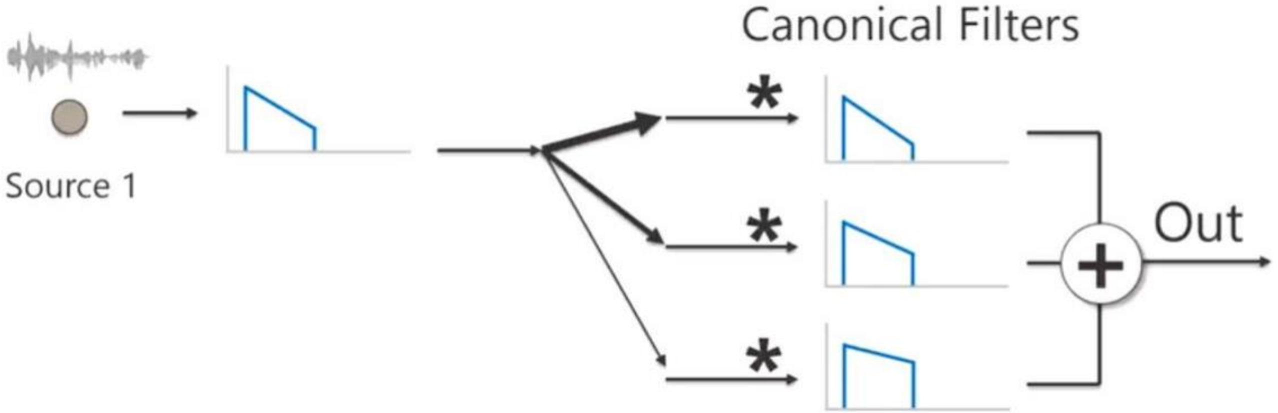
ACM Transactions on Graphics, Publication date: November 2017

Presenter: MinKu Kang

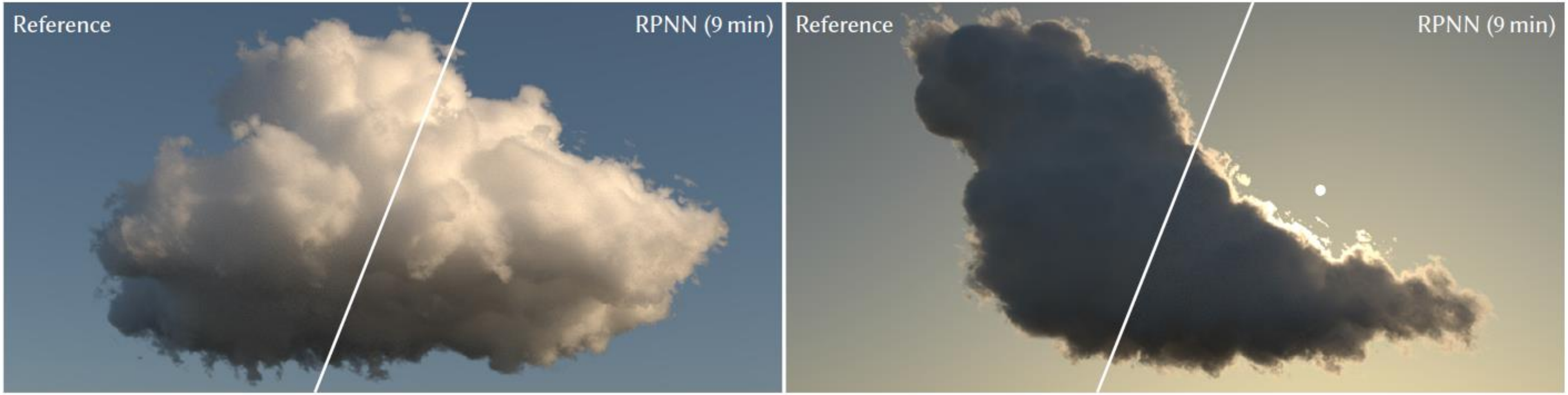
In Previous Talk from Dennis

Ambient sound propagation





Cloud Rendering

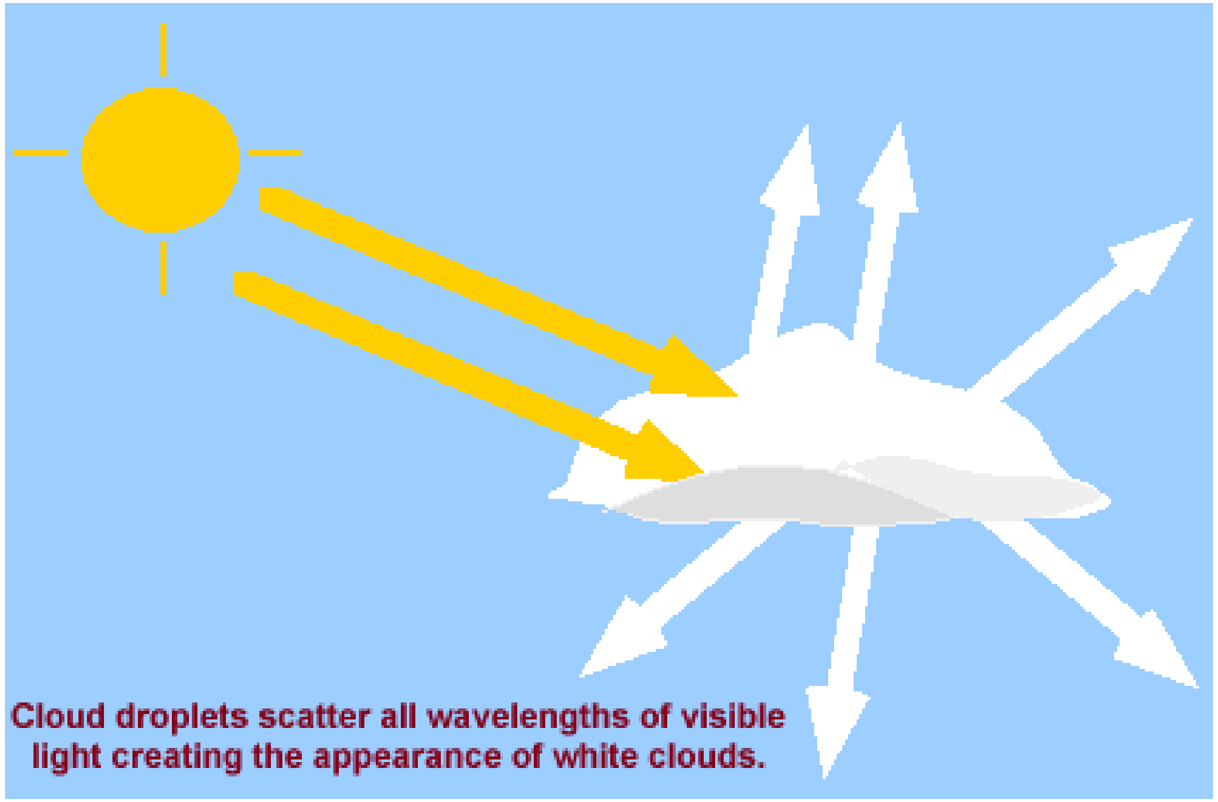


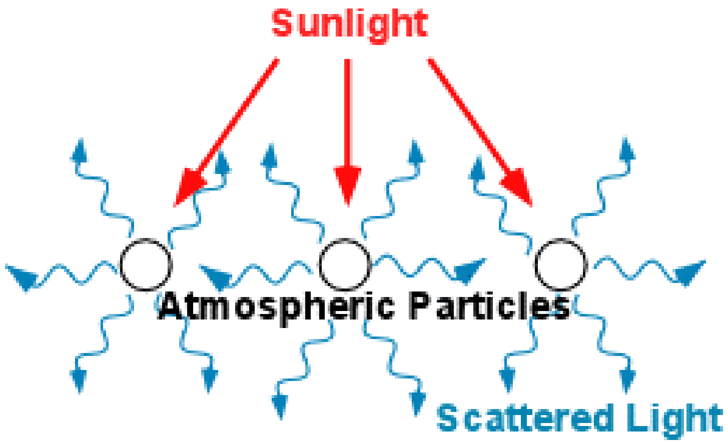
edge-darkening effects

silverlining

https://www.youtube.com/watch?v=0MJl9IF_3fI

Scattering of Light

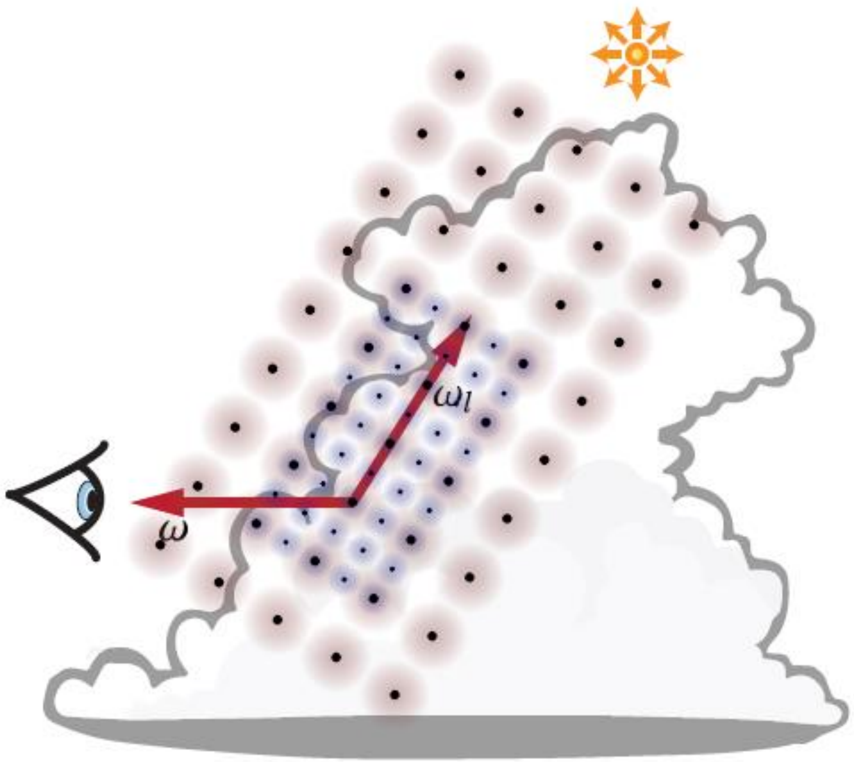




Light scattering in **microscale**, not just in macro scale

http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/opt/mch/sct.rxml

Problem Configuration & Notation



ω: *direction*

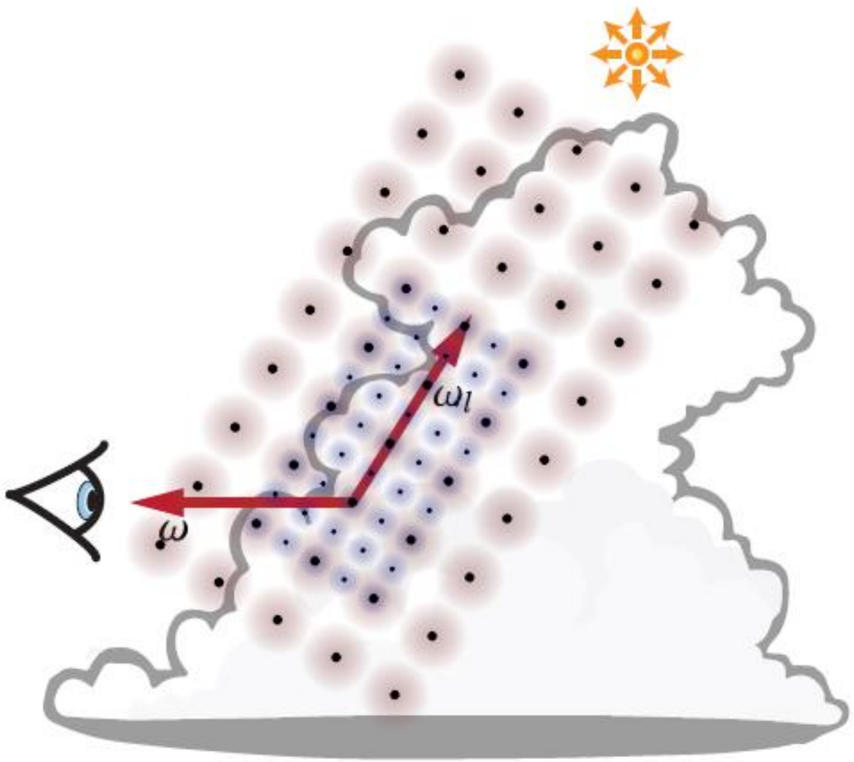
x: *location*

We want to know (compute) the **radiance** at (x, ω)

To render a whole cloud image,
We need to know the radiance at all (visible) **positions** and **directions**

Problem: How to efficiently compute the radiance at a specific position and a direction ?

Problem Configuration & Notation



ω: *direction*

x: *location*

We want to know (compute) the **radiance** at (x, ω)

To render a whole cloud image,
We need to know the radiance at all (visible) **positions** and **directions**

Problem: How to efficiently compute the radiance at a specific position and a direction ?

But, there are too many discrete **particles** to consider (they are not even polygons!).
Is this possible to use **rendering equation** we have learned ?

Radiative Transfer

The **radiative transfer equation**

$$(\omega \cdot \nabla)L(\mathbf{x}, \omega) = -\mu_t(\mathbf{x})L(\mathbf{x}, \omega) + \mu_s(\mathbf{x}) \int_{S^2} p(\omega \cdot \widehat{\omega})L(\mathbf{x}, \widehat{\omega})\, \mathrm{d}\widehat{\omega},$$

Integrating both sides of the differential **RTE** along ω

$$L(\mathbf{x}, \omega) = \int_0^\infty \exp\left(-\int_0^u \mu_t(\mathbf{x}_v)\mathrm{d}v\right) \mu_s(\mathbf{x}_u) \int_{S^2} p(\omega \cdot \widehat{\omega})L(\mathbf{x}_u, \widehat{\omega})\mathrm{d}\widehat{\omega}\, \mathrm{d}u$$

transmittance $T(\mathbf{x}, \mathbf{x}_u)$ $\mathbf{x}_u = \mathbf{x} - u\omega$

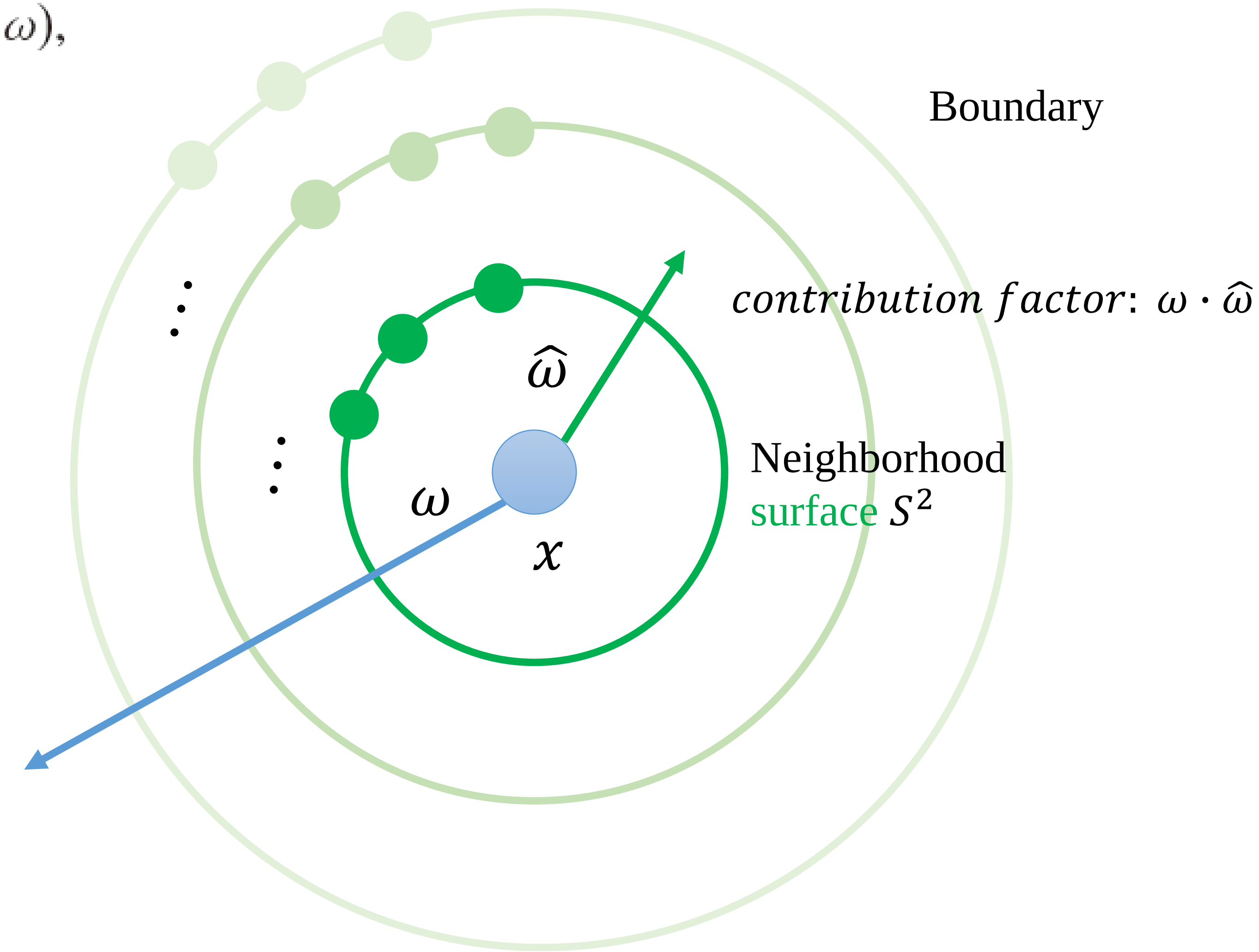
Assuming $\mathbf{x}_b$ is on a hypothetical boundary Ω that encloses the cloud, i.e. $\forall \mathbf{x}_b \in \Omega : \mu_t(\mathbf{x}_b) = 0$, we have

$$L(\mathbf{x}, \omega) = \int_0^b T(\mathbf{x}, \mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega \cdot \widehat{\omega})L(\mathbf{x}_u, \widehat{\omega})\, \mathrm{d}\widehat{\omega}\, \mathrm{d}u + T(\mathbf{x}, \mathbf{x}_b)L(\mathbf{x}_b, \omega),$$

$\mu_t(\mathbf{x})$: extinction coefficient

Radiative Transfer

$$L(\mathbf{x}, \omega) = \int_0^b T(\mathbf{x}, \mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega \cdot \widehat{\omega}) L(\mathbf{x}_u, \widehat{\omega})\, d\widehat{\omega}\, du + T(\mathbf{x}, \mathbf{x}_b) L(\mathbf{x}_b, \omega),$$



RADIANCE-PREDICTING NEURAL NETWORKS

$$L(\mathbf{x}, \omega) = \int_0^b T(\mathbf{x}, \mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega \cdot \widehat{\omega}) L(\mathbf{x}_u, \widehat{\omega}) \,\mathrm{d}\widehat{\omega} \,\mathrm{d}u + T(\mathbf{x}, \mathbf{x}_b) L(\mathbf{x}_b, \omega),$$

The in-scattered radiance

$$L_s(\mathbf{x}, \omega) = \int_{S^2} p(\omega \cdot \widehat{\omega}) L(\mathbf{x}, \widehat{\omega}) \,\mathrm{d}\widehat{\omega}.$$

Rule out **uncollided** radiance
(directly from the sun)

$$L_i(\mathbf{x}, \omega) = \int_{S^2} p(\omega \cdot \widehat{\omega}) (L(\mathbf{x}, \widehat{\omega}) - L_d(\mathbf{x}, \widehat{\omega})) \,\mathrm{d}\widehat{\omega}.$$

This is what the NN **predicts (estimate)**

A **combination** of Monte Carlo integration and neural networks

Monte-Carlo Integration

$$L(\mathbf{x},\omega) = \int_0^b T(\mathbf{x},\mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega\cdot\widehat{\omega})L(\mathbf{x}_u,\widehat{\omega})\,\mathrm{d}\widehat{\omega}\,\mathrm{d}u + T(\mathbf{x},\mathbf{x}_b)L(\mathbf{x}_b,\omega),$$

The in-scattered radiance

$$L_i(\mathbf{x},\omega) = \int_{S^2} p(\omega\cdot\widehat{\omega})(L(\mathbf{x},\widehat{\omega}) - L_d(\mathbf{x},\widehat{\omega}))\,\mathrm{d}\widehat{\omega}.$$

This is what the **NN**
predicts (estimate)

RADIANCE-PREDICTING NEURAL NETWORKS

Want to find (learn) a function

$$g(\mathbf{z}; \boldsymbol{\theta}) : \mathbb{R}^d \longrightarrow \mathbb{R}$$

Such that,

given $\mathbf{z} = \phi(\mathcal{S}) \in \mathbb{R}^d$

S: shading configuration around x, ω

it predicts $L_i(\mathbf{x}, \omega)$

RADIANCE-PREDICTING NEURAL NETWORKS

Want to find (learn) a function

$$g(\mathbf{z};\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$$

via

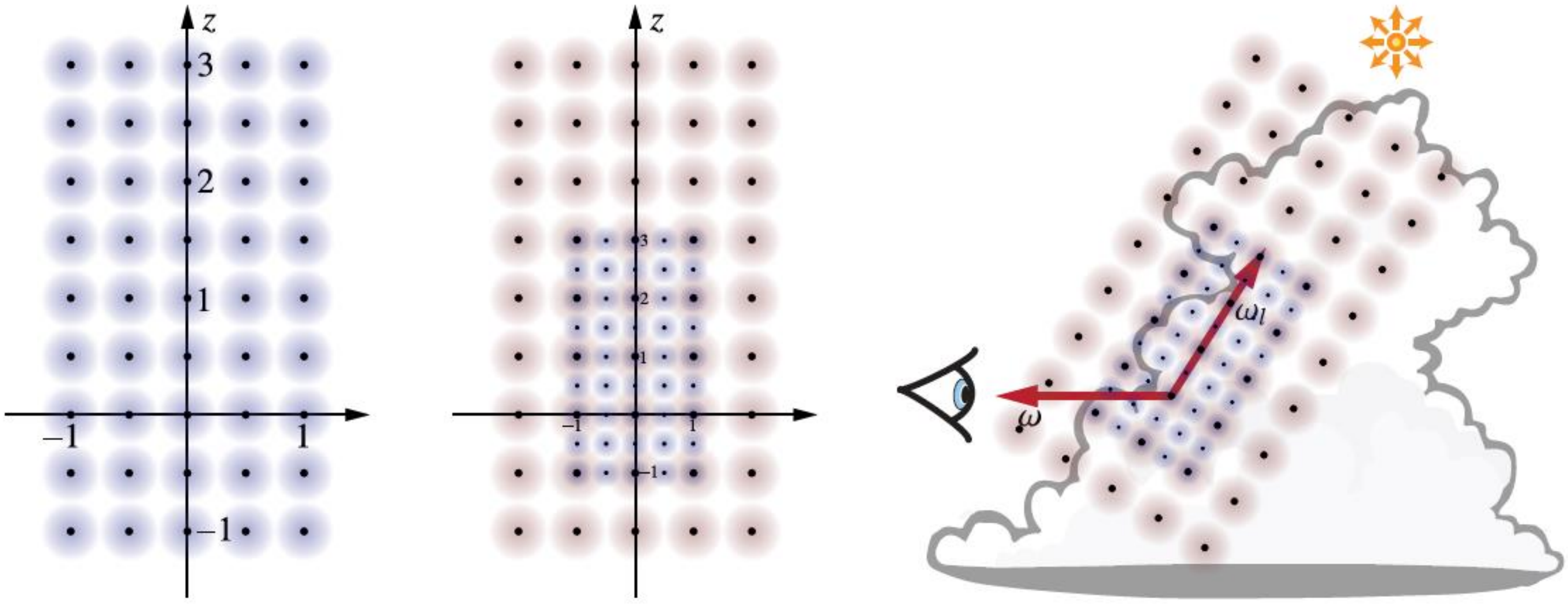
$$\widehat{\boldsymbol{\theta}} \in \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(g(\mathbf{z}_i;\boldsymbol{\theta}), L_i(\mathbf{x}_i, \omega_i)).$$

using

$$\mathcal{D}_N = \{(\mathbf{z}_1, L_i(\mathbf{x}_1, \omega_1)), \ldots, (\mathbf{z}_N, L_i(\mathbf{x}_N, \omega_N))\},$$

$$\ell_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \Big(\log\left(1 + g(\mathbf{z}_i;\boldsymbol{\theta})\right) - \log\left(1 + L_i(\mathbf{x}_i, \omega_i)\right) \Big)^2.$$

The Descriptor at a specific configuration (x, ω)

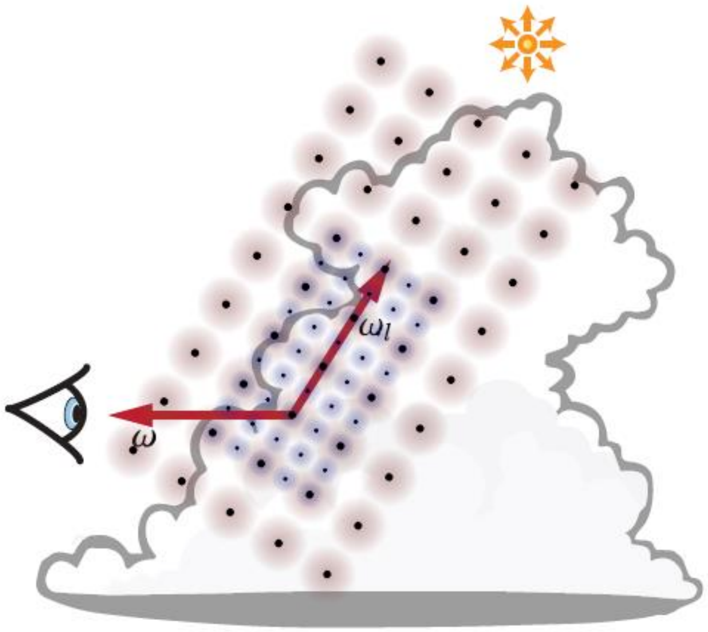


(a) Stencil grid (b) Two levels (c) Local frame

- Each descriptor consists of 5 × 5 × 9 stencils
- The stencil at level k is scaled by 2^{k-1}
- They use K=10 levels (10 stenciles)
- Each stencil is formed by 225 points

- The stencil is oriented towards the light source
- Two levels of the hierarchy are shown here

The Descriptor at a specific configuration (x, ω)



$$\Sigma^k = \left\{\rho(\mathbf{q}_1^k), \rho(\mathbf{q}_2^k), \ldots \rho(\mathbf{q}_{225}^k)\right\}$$

$$\Sigma = \bigcup_{k=1}^{K} \Sigma^k$$

$$\gamma = \cos^{-1}(\omega \cdot \omega_l)$$

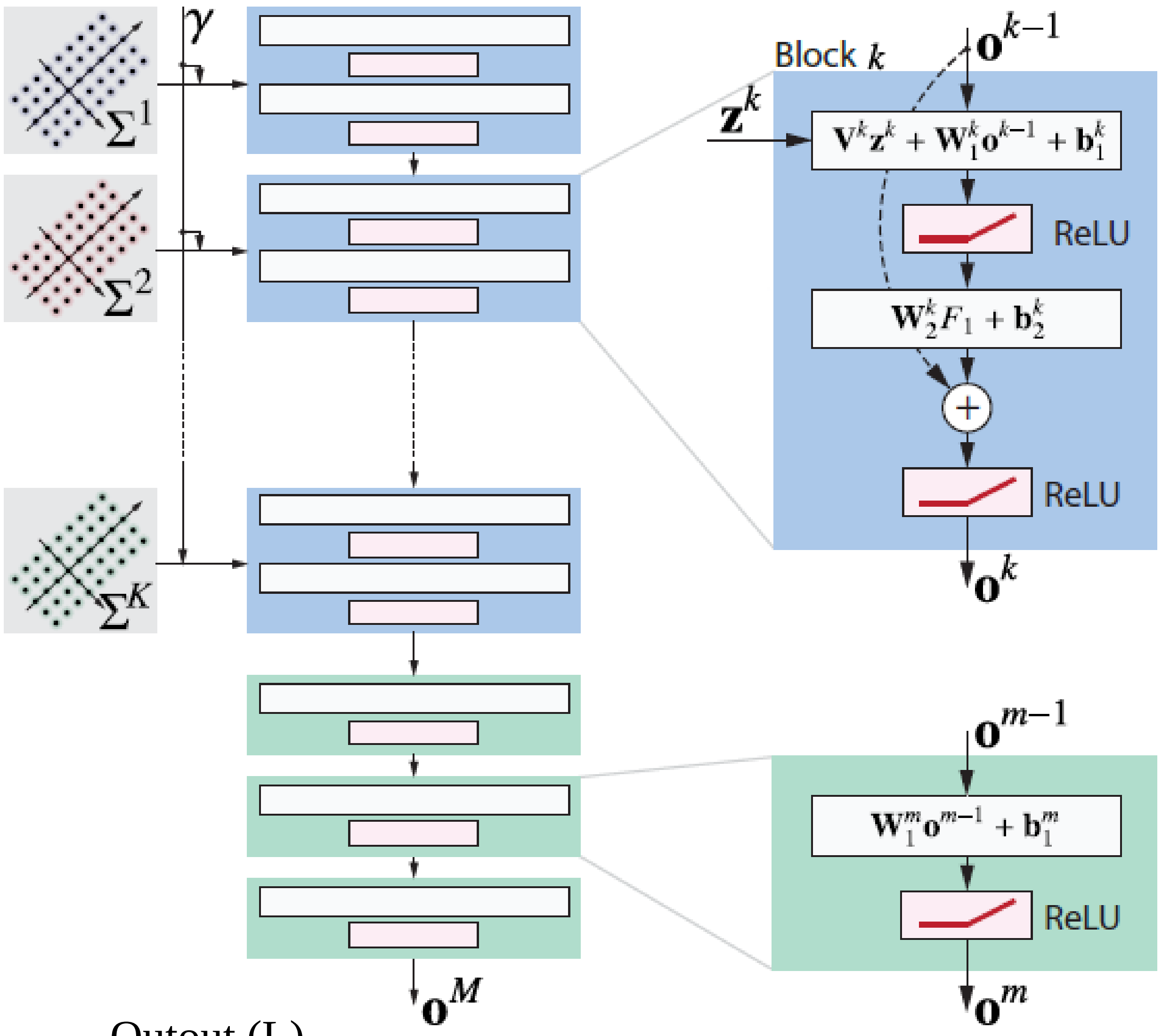
x: *location*
ω: *direction*
ω_l: *direction towards the light source*

The Descriptor: $\mathbf{z} = \{\Sigma, \gamma\}$

Neural Network Architecture (progressive feeding)

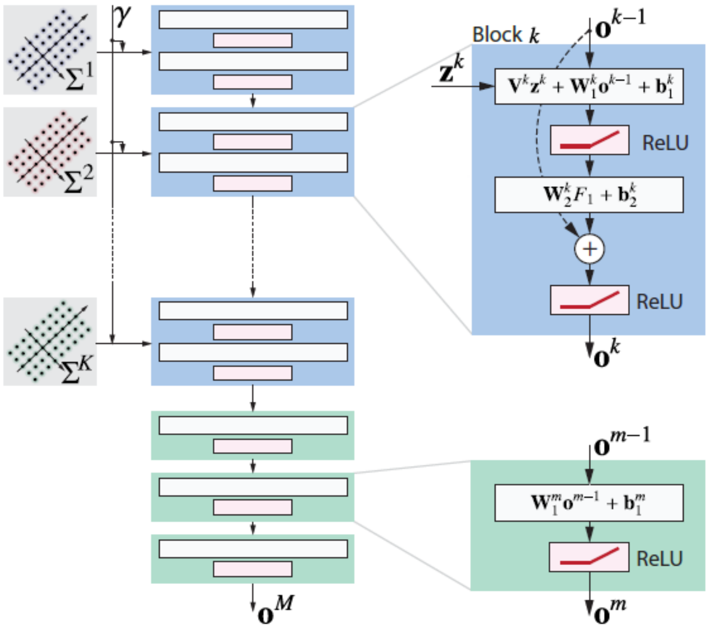
The most **finest** scale stencil

The most **coarse** scale



Outout (L)

Training Configuration



Ground Truth data from **Path Tracing**

N = ~15 million samples

Adam update rule using the default learning rate
The minibatches of size |B| = 1000

It requires **∼12 h of training** on a single GPU

Result (Test Time)

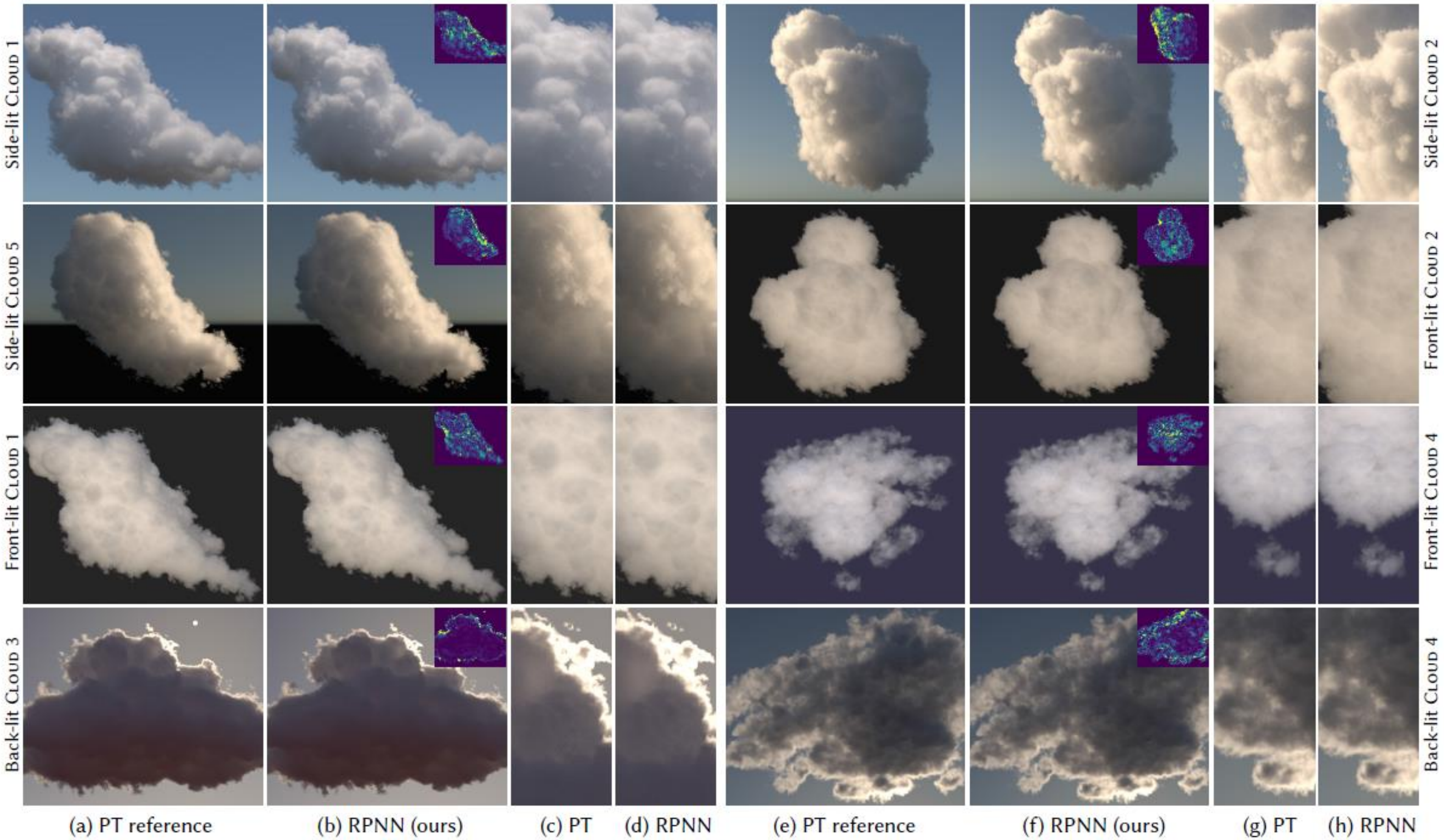
(a) PT reference

(b) RPNN

Path Tracing

Radiance-Predicting Neural Networks (RPNN)

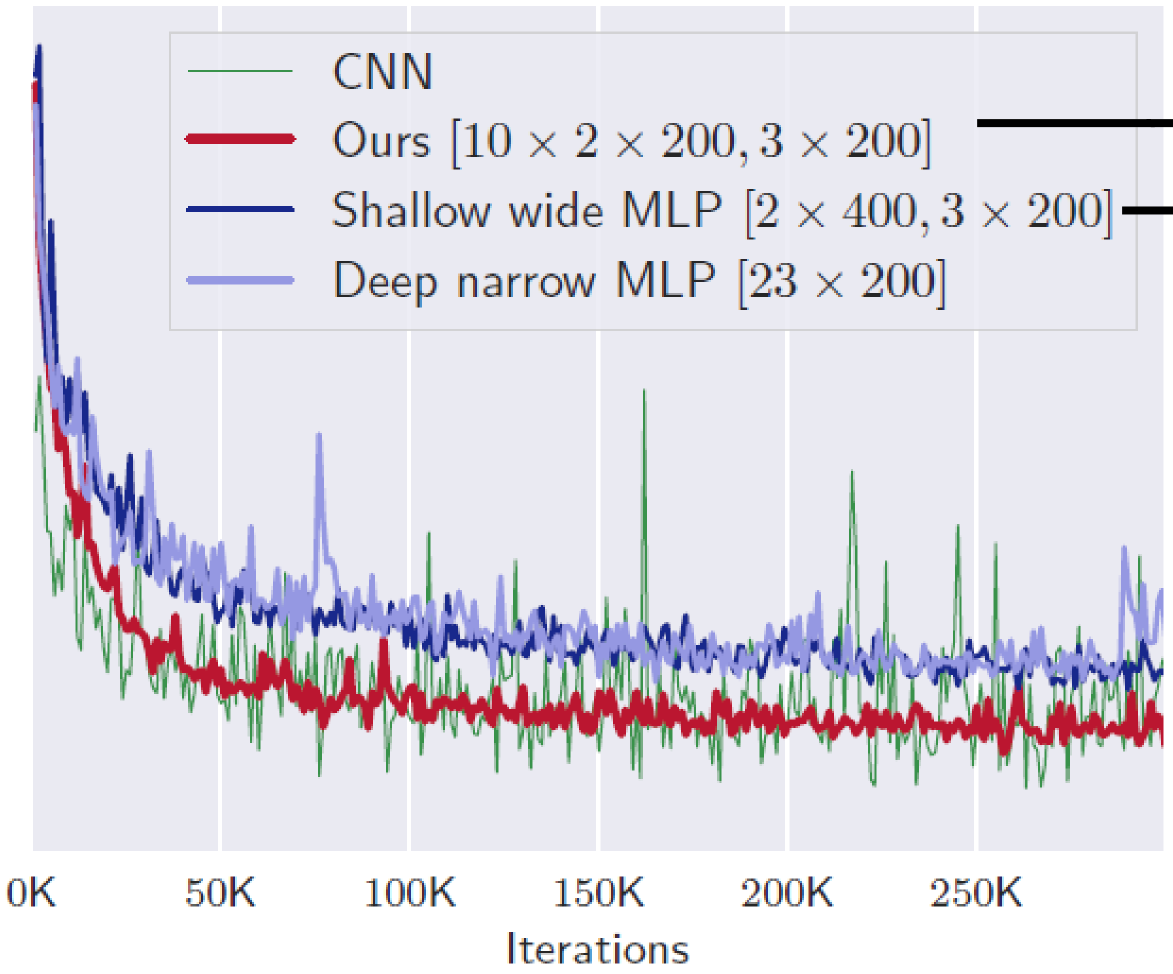
Result (Test Time)



They argued that RPNN (seconds to minutes.) converges 24 times faster than PT

Experiment - Neural Network Architecture

Validation error

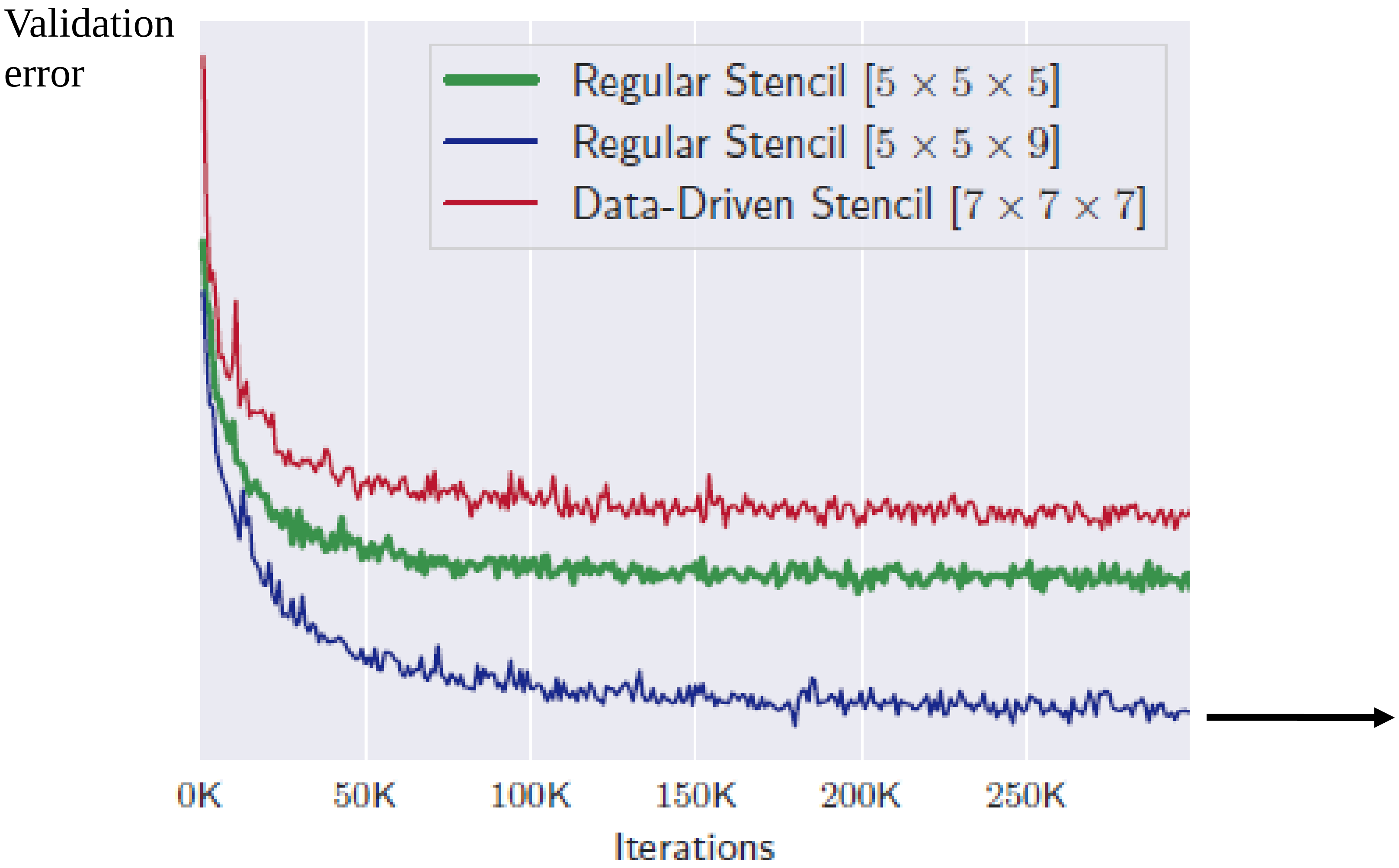


Progressive feeding

The entire stencil hierarchy is input to the first layer

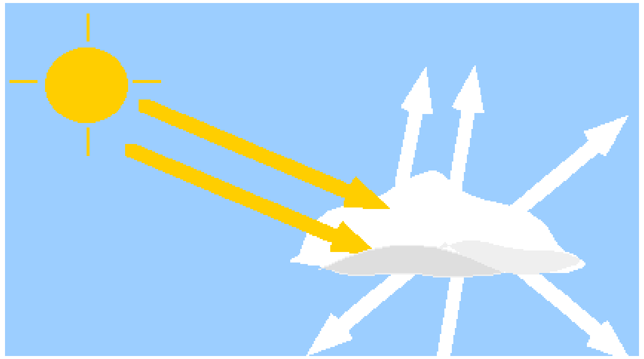
This highlights the benefit of the progressive feeding that provides means to **better adapt to signals** at different frequency scales.

Experiment – Stencil Size



A good balance between accuracy and the cost of querying the density values and number of trainable parameters in the network

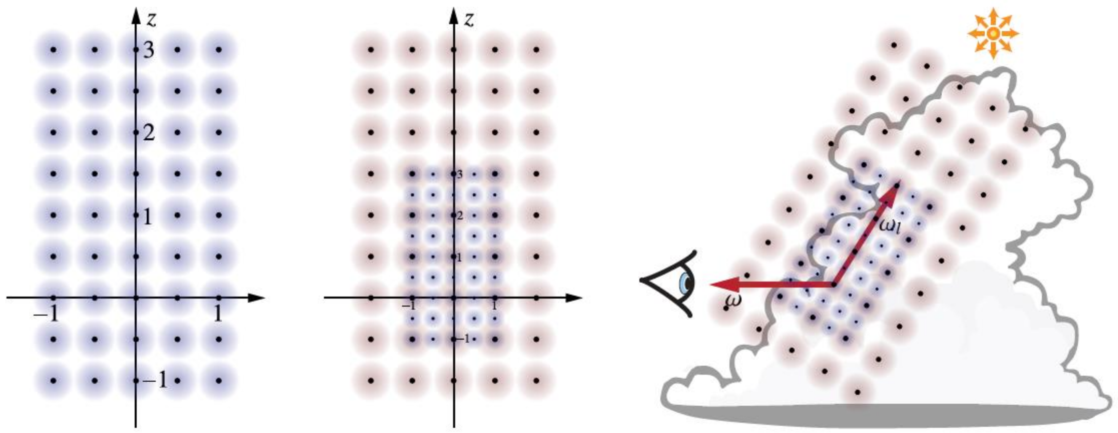
Summary



Radiative Transfer Equation (RTE)

$$L(\mathbf{x},\omega) = \int_0^b T(\mathbf{x},\mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega\cdot\widehat{\omega})L(\mathbf{x}_u,\widehat{\omega})\,\mathrm{d}\widehat{\omega}\,\mathrm{d}u$$
$$+\,T(\mathbf{x},\mathbf{x}_b)L(\mathbf{x}_b,\omega),$$

Hierarchical Stencil Descriptor



(a) Stencil grid (b) Two levels (c) Local frame

Progressive Feeding Neural Network