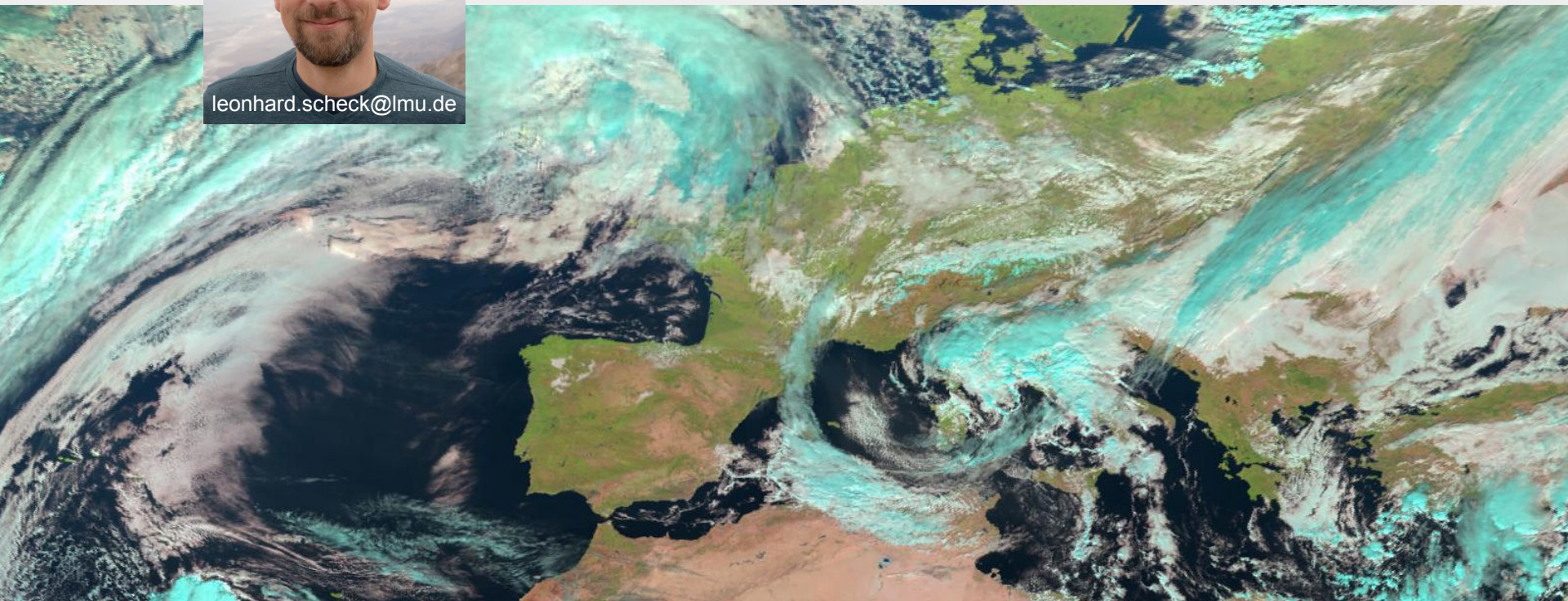
A fast neural network based method to generate synthetic visible and near-infrared satellite images

Leonhard Scheck[1,2], Florian Baur[1,2], Christina Stumpf[2], Christina Köpken-Watts[2]

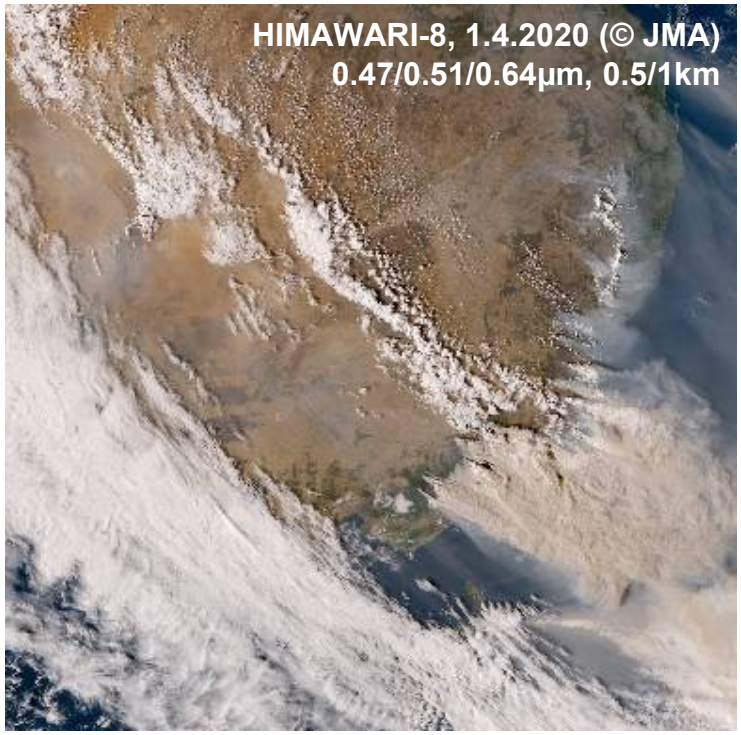
leonhard.scheck@lmu.de

1) Hans-Ertel-Center for Weather Research / LMU Munich, Germany
2) Deutscher Wetterdienst, Offenbach, Germany



Visible and near-infrared satellite images

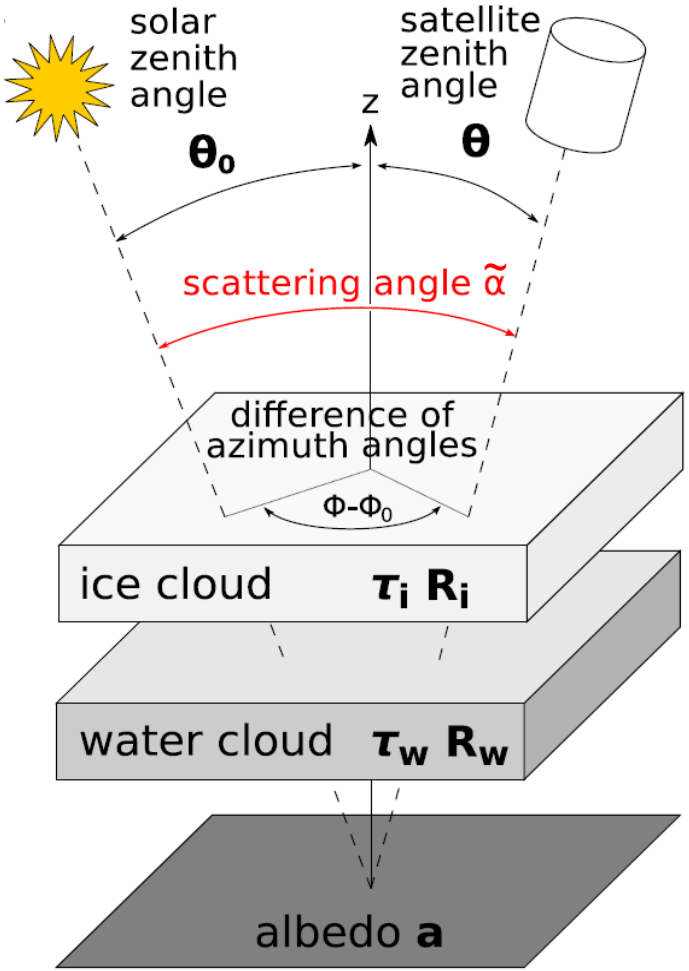
- High-resolution cloud & aerosol observations, well-suited for convective scale data assimilation
- So far mainly thermal channels were considered for data assimilation
- Complementary information from solar channels:
  - Better information on water content (signal saturates later)
  - can distinguish low clouds from ground
  - sensitive to microphysics (e.g. effective radii) and water phase (1.6μm)



- Multiple scattering makes standard radiative transfer methods too expensive for use in operational DA systems
  → development of MFASIS for visible channels (included in RTTOV)
- We investigated (deep) feed forward neural networks for improving the accuracy and extending the capabilities of the forward operator

Efficient generation of synthetic visible images: MFASIS

Method for Fast Satellite Image Synthesis



- **Simplify equation:** 3D → 1D RT (independent columns)
- **Simplify vertical structure:** Cloud water and ice can be separated (→ two homogeneous clouds at fixed heights) without changing reflectance significantly

  → only 4 parameters (optical depth, particle size)

  + 3 angles, albedo → **8 parameters per column**
- **Compute 8-dimensional reflectance look-up table** (LUT) with discrete ordinate method (DOM) for all parameter combinations, interpolate in LUT

  Problem: **Table is huge! (about 8GB)**
  → not suitable for online operator, slow interpolation
- **Compress table to 21MB** using truncated Fourier series (lossy compression, similar to JPEG graphics format)

fast (O(µsec/column)), mean reflectance error < 0.01
implemented in RTTOV by DWD in collab. with MetOffice

Could we replace the LUT by a neural network?

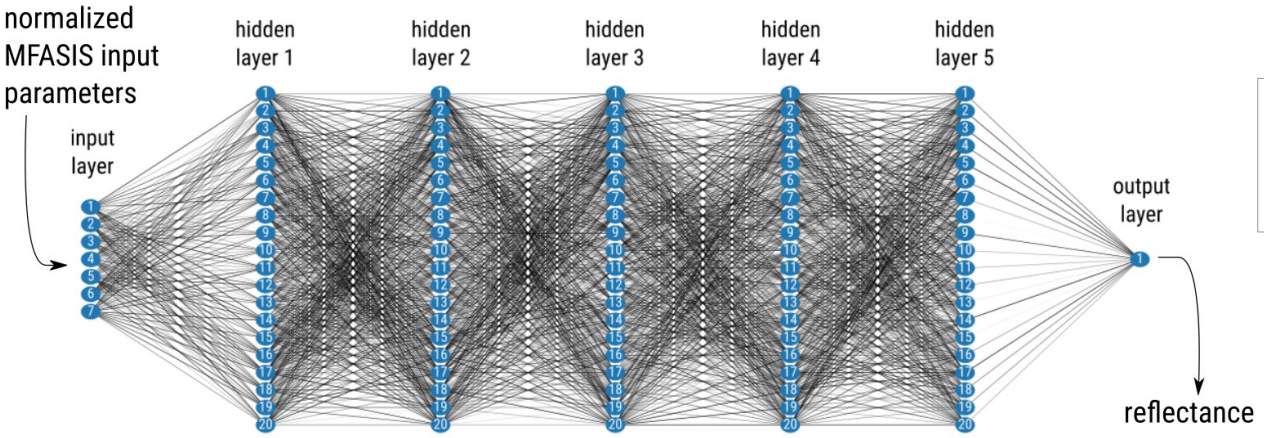
Motivation: MFASIS works well for non-absorbing visible channels and clouds, but it would be very useful to support also aerosols (many species) and absorbing channels (trace gases or clouds) → would require many additional LUT dimension → strongly increased LUT size, problematic… Machine learning methods could be more efficient.

First goal: Replace current MFASIS LUT by a feed forward network (no additional dimensions).

Can we reduce training data (8GB in MFASIS) and memory consumption (21MB in MFASIS)?

We want to match the speed of MFASIS → **can afford only O(10^3) network parameters…**

→ To be investigated: **Can a sufficiently fast (=small) network be sufficiently accurate?**

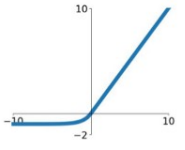


$$s_{l+1,n} = f(b_{l,n} + \sum s_{l,m} * w_{l,n,m})$$

Differentiable activation function

ELU

$$\begin{cases} x & x \geq 0 \\ e^x - 1 & x < 0 \end{cases}$$



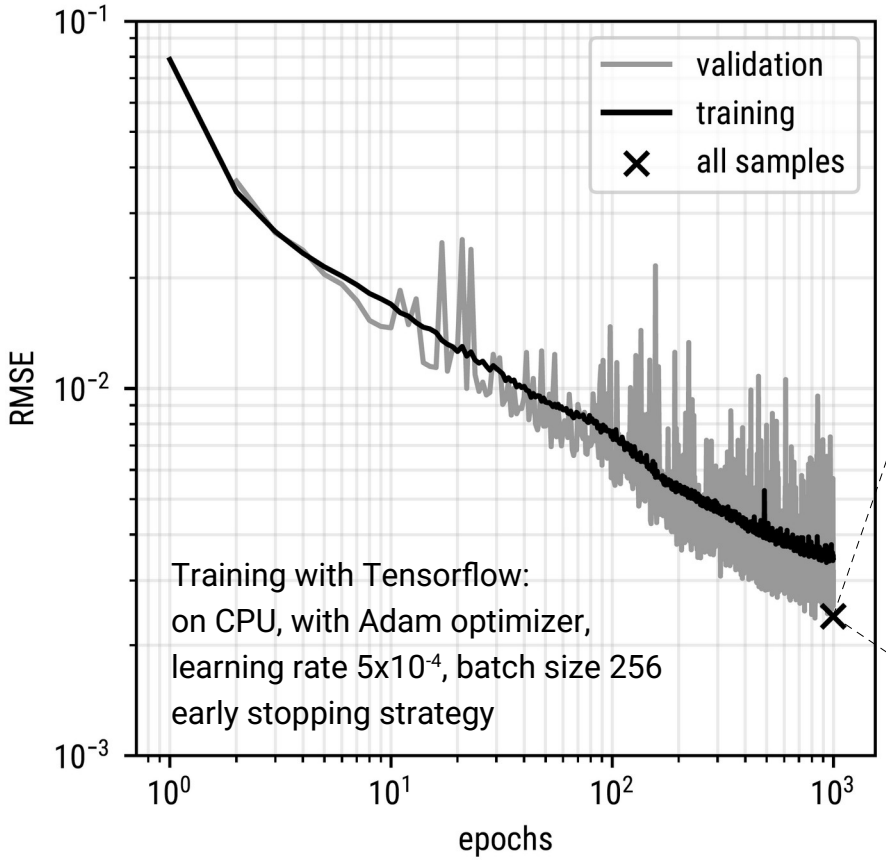
Best results for
4 – 8 hidden layers

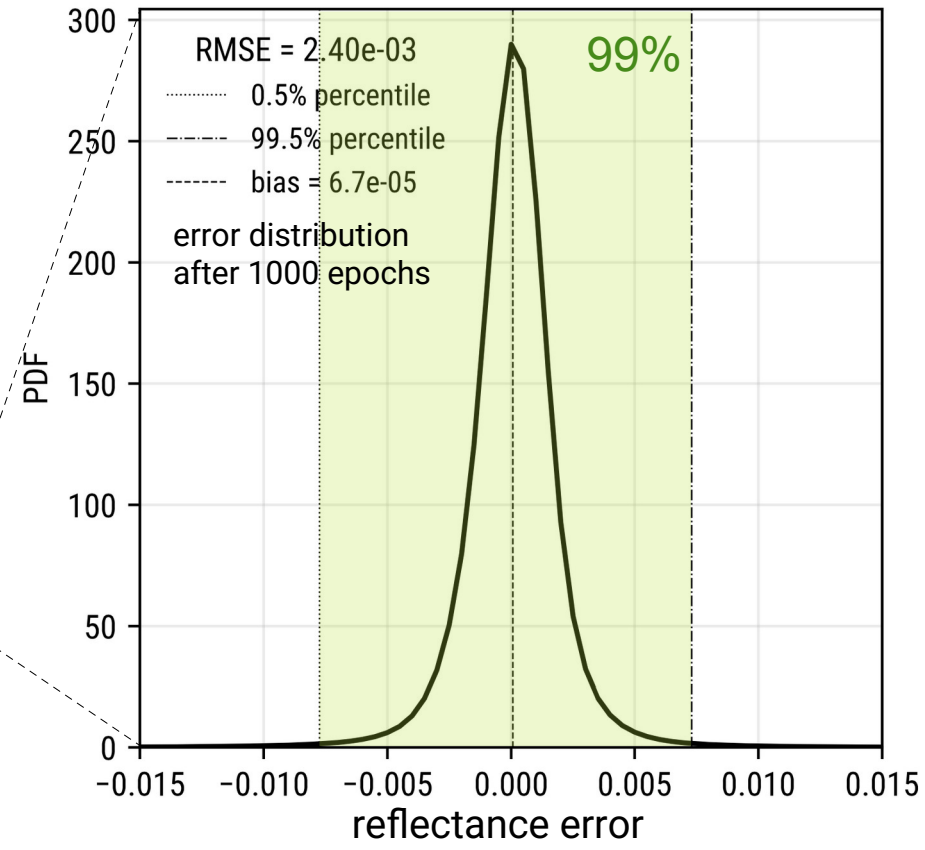
Accuracy and memory requirements for an example

3000 parameters, 6 hidden layers, 23 nodes/layer, ELU activation, 3.4×10^6 samples, trained for 13h

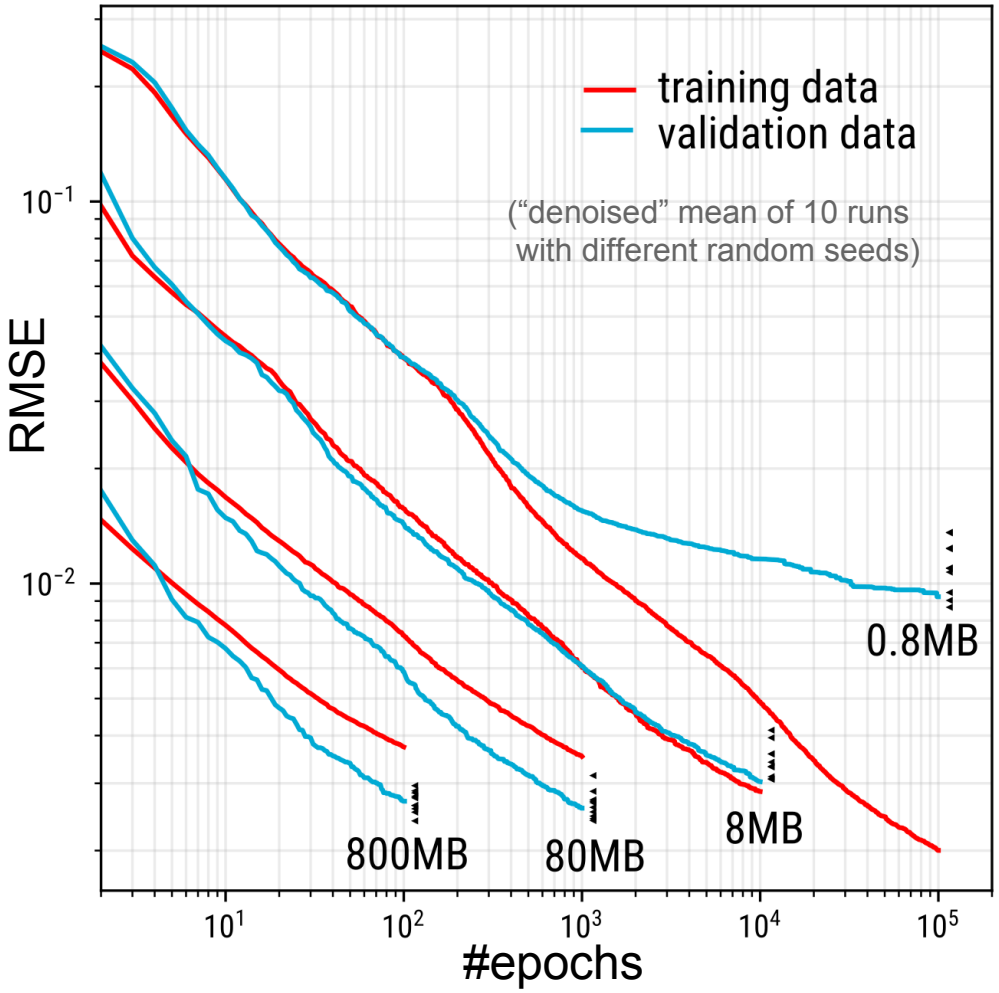
Similar to MFASIS: Reflectance RMSE=2.4×10^{-3}, Error < 8×10^{-3} in 99% of cases

Memory required: 3000 x 8 Byte = **24KB** → **Factor ~1000 less than in LUT based approach**





Required training data size



Training for ~10 hours with different training data sizes (each experiment repeated 10 times with different randomly chosen data):

10%, 1%: no problems, similar results

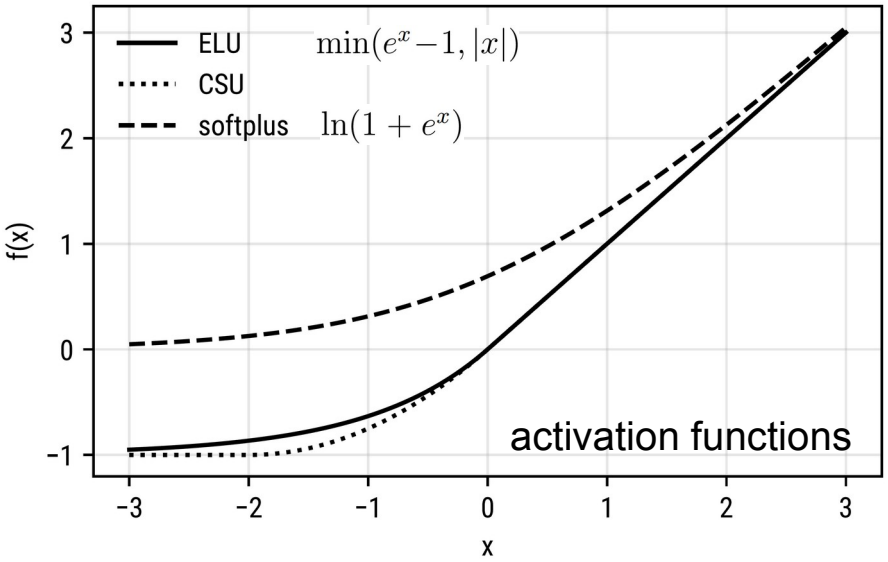
0.1%: first signs of overfitting, but final error is still acceptable

0.01% → "Overfitting": network adapts too strongly to details of the training data set

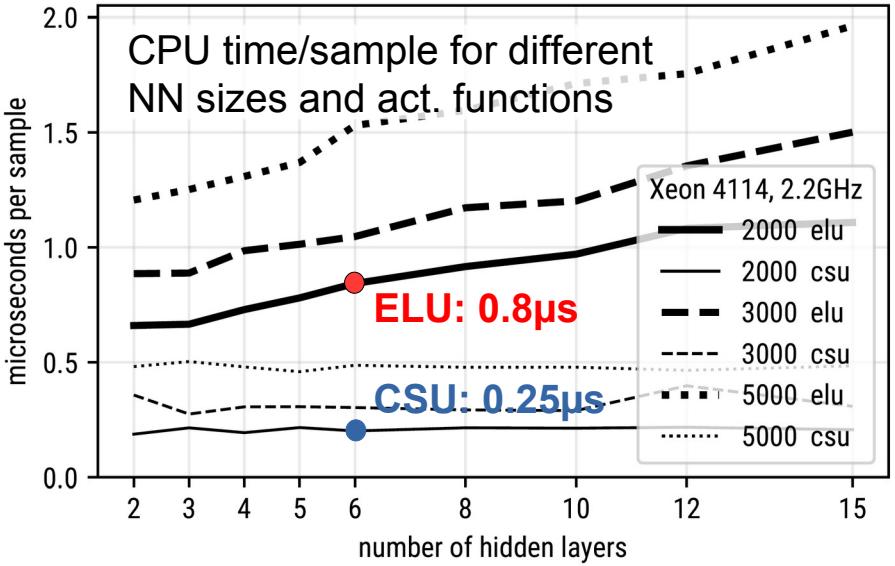
→ Also the training data can be reduced by a factor 1000, compared to the LUT-based approach

Optimizations

- Using **random samples** instead of LUT training data → NN size can be reduced by fact. 2-3
- Transforming input & output variables to reduce nonlinearity and variance → 20% reduction
- Development of **Fortran inference code optimized for small NNs** (<100 nodes/layer) (vectorized, inner loop over samples, not nodes; much faster than Tensorflow predict())
- Using a **activation function without exp()** → inference 3-4 times faster for small NNs



$$f_{\mathrm{CSU}}(x) = \begin{cases} 0, & \text{if } x < -2 \\ -1 + 0.25\,(x+2)^2, & \text{if } x \in [-2, 0] \\ x, & \text{if } x > 0 \end{cases}$$



→ 2000 parameter NN with CSU:
11 x faster than MFASIS, similar errors

Adjoint / tangent linear codes

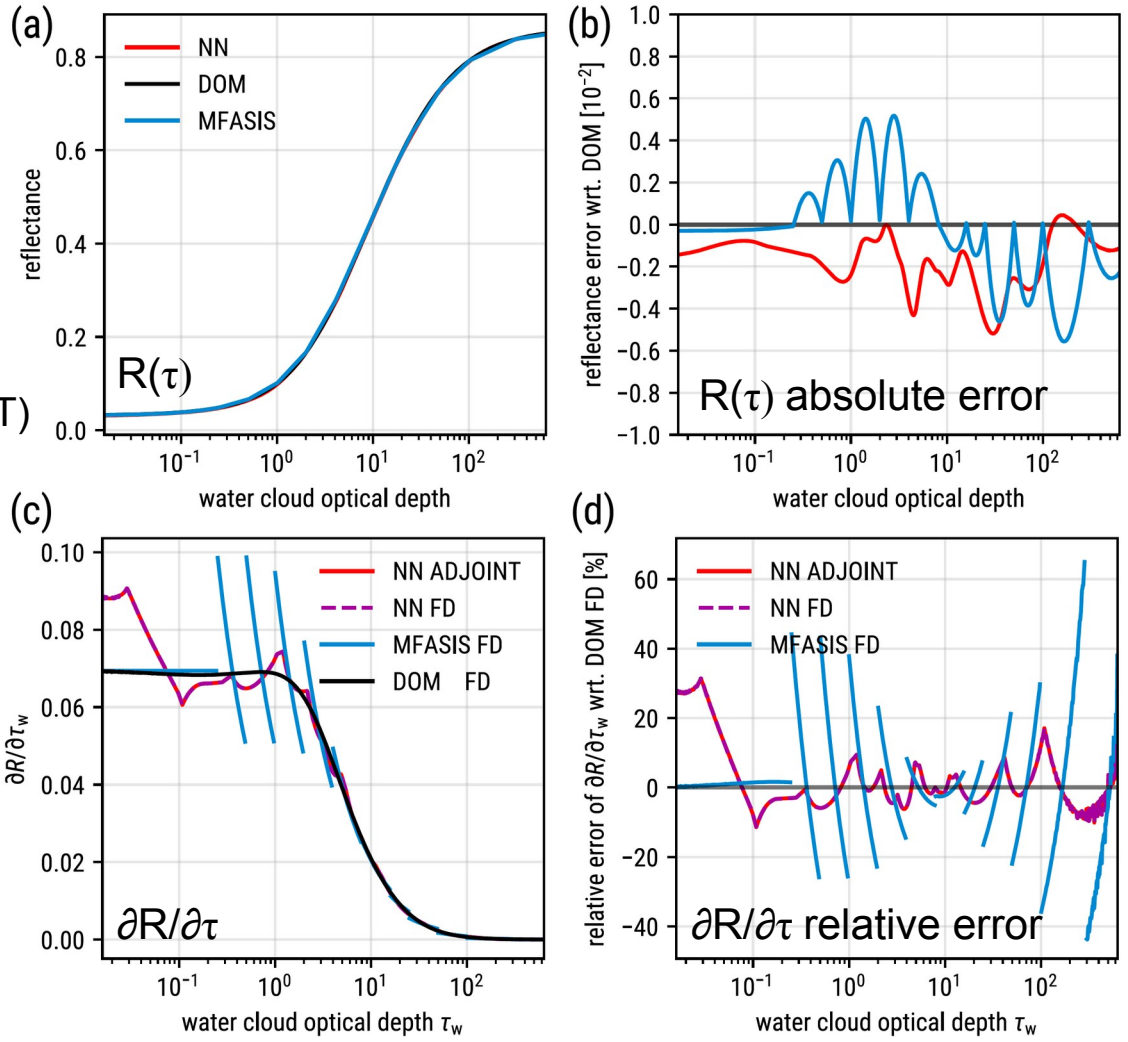
Example: Reflectance R as a function of water cloud optical depth τ

Adjoint (AD) + tangent linear (TL) versions of the nonlinear NN inference code (NL) are required for variational and hybrid DA methods.
→ Implemented for Fortran code

- AD/TL are 30-50% slower than NL (non-vectorized versions)
- output continuous (in contrast to LUT)
- if required, error in derivatives could be reduce with regularization term

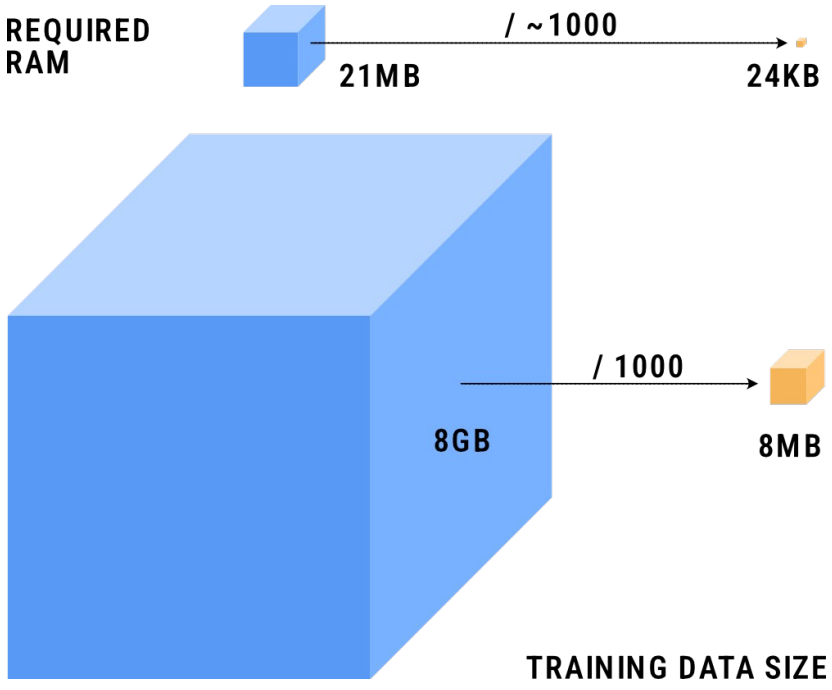
AD/TL codes have to be developed and kept in sync with nonlinear code
→ usually requires quite some effort

AD/TL for neural networks is easy to derive [done] and **does not have to be modified** when training data or network structure is changed.



Summary & Outlook

- Feed forward neural network instead of compressed look-up table: Memory consumption and amount of training data can be reduced by factor 1000, speed can be increased by factor 10 while maintaining accuracy of MFASIS
- Extensions requiring additional input parameters (water vapor, NIR, aerosols, ...) become feasible with the neural network approach
- Fortran inference code optimized for small networks is available (incl. “maintainance free” AD/TL versions)



References:
- Scheck (2021): *A neural network based forward operator for visible satellite images and its adjoint,* JQSRT, submitted.
- Saunders et al. (2020): RTTOV-13 Science and Validation Report
- Scheck, Frerebeau, Buras-Schnell, Mayer (2016): *A fast radiative transfer method for the simulation of visible satellite imagery,* JQSRT

Code is available from https://gitlab.com/LeonhardScheck/fornado