



Massachusetts  
Institute of  
Technology



# What is the real information in weather and climate data?

**Milan Klöwer<sup>1</sup>, Ayoub Fatihi, Miha Razinger<sup>2</sup>, Juanjo Dominguez<sup>2</sup>, Aaron Spring<sup>3</sup>, Hauke Schulz<sup>4</sup>, Peter Düben<sup>2</sup>, Tim Palmer<sup>1</sup>**

<sup>1</sup>Massachusetts Institute of Technology, Cambridge USA

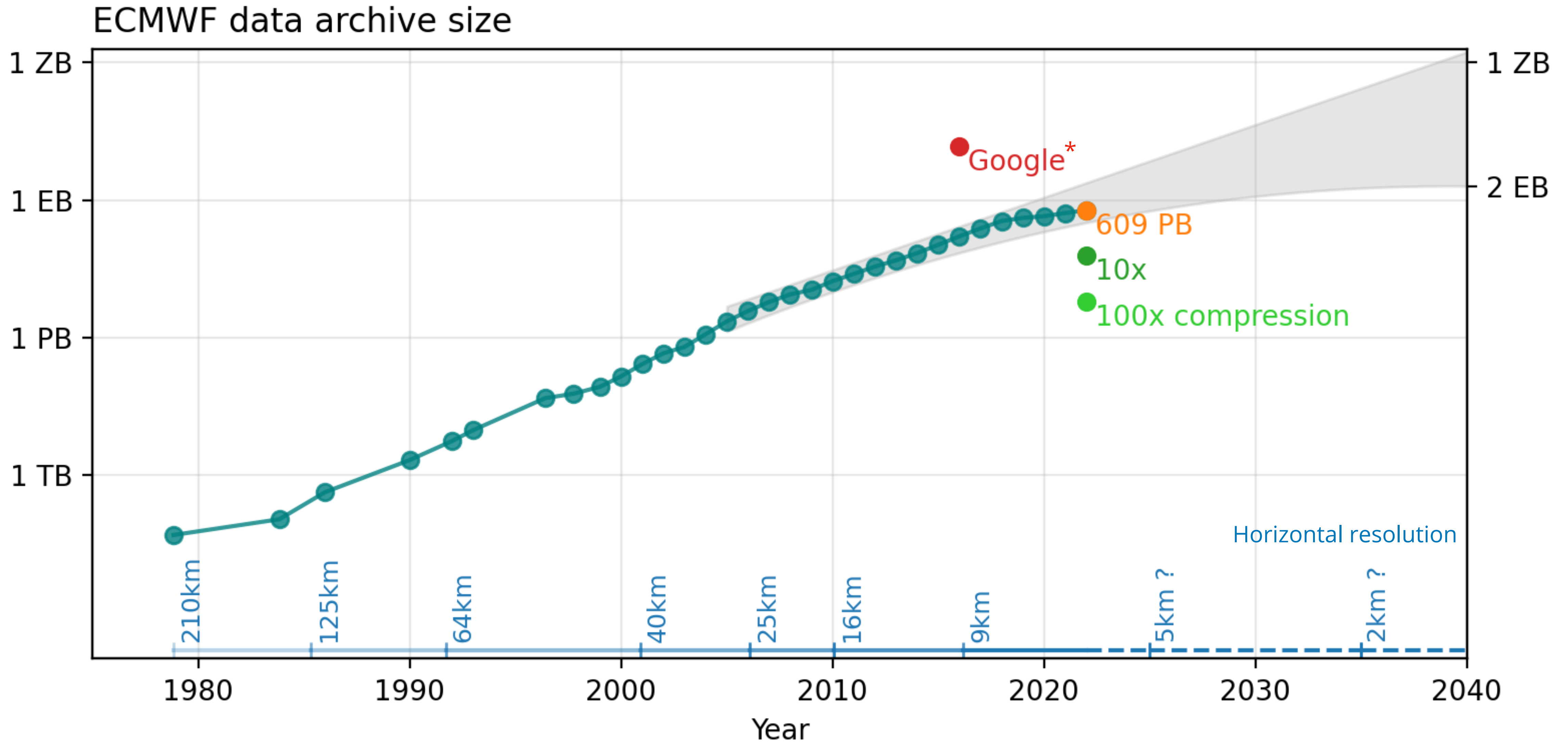
<sup>2</sup>European Centre for Medium-Range Weather Forecasts, UK

<sup>3</sup>Max Planck Institute for Meteorology, Hamburg, Germany

<sup>4</sup>University of Washington, Seattle, USA



# Will we enter the *Google regime*?

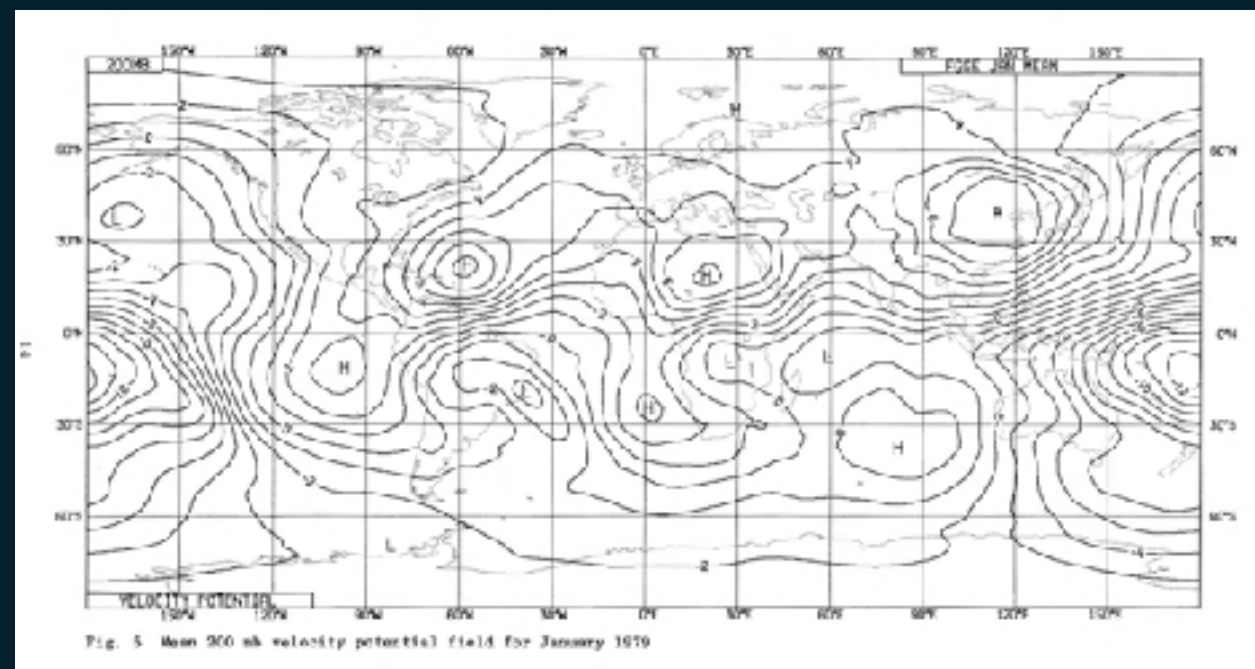


\*estimate from XKCD, 2016

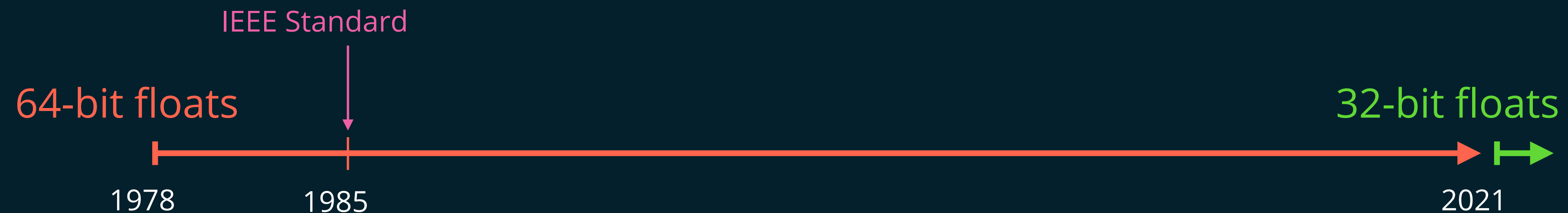
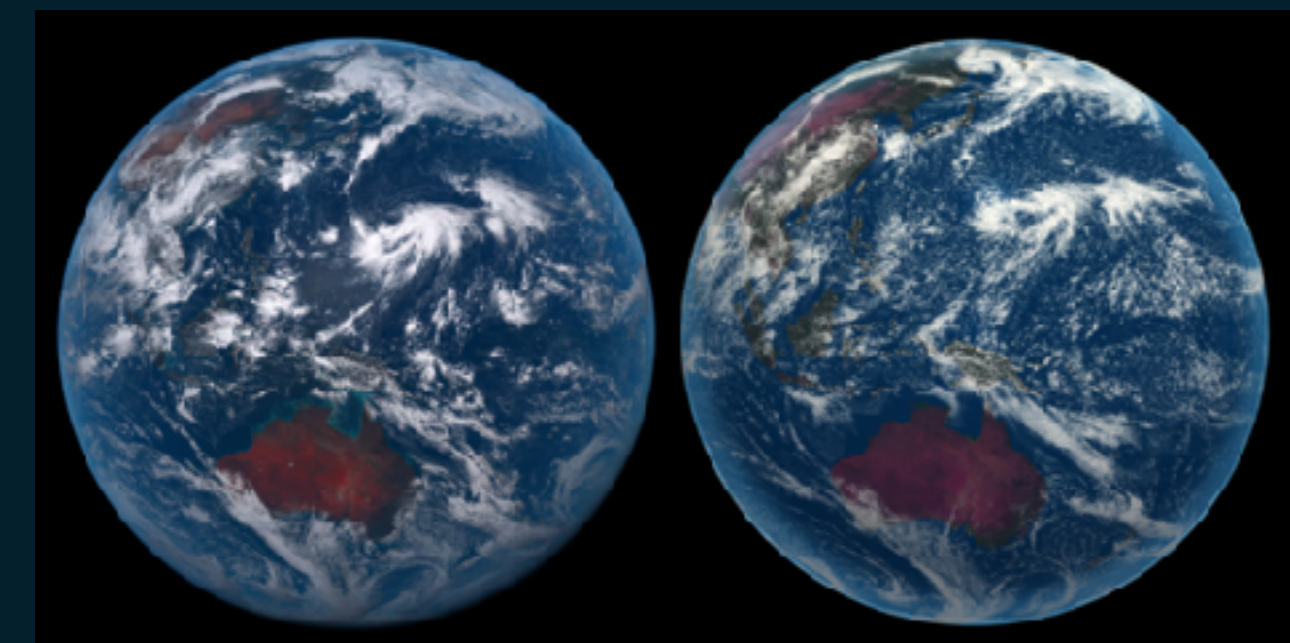
# 40+ years of numerical weather prediction

*How much have we questioned the bitwise representation of our data?*

ECMWF 1979



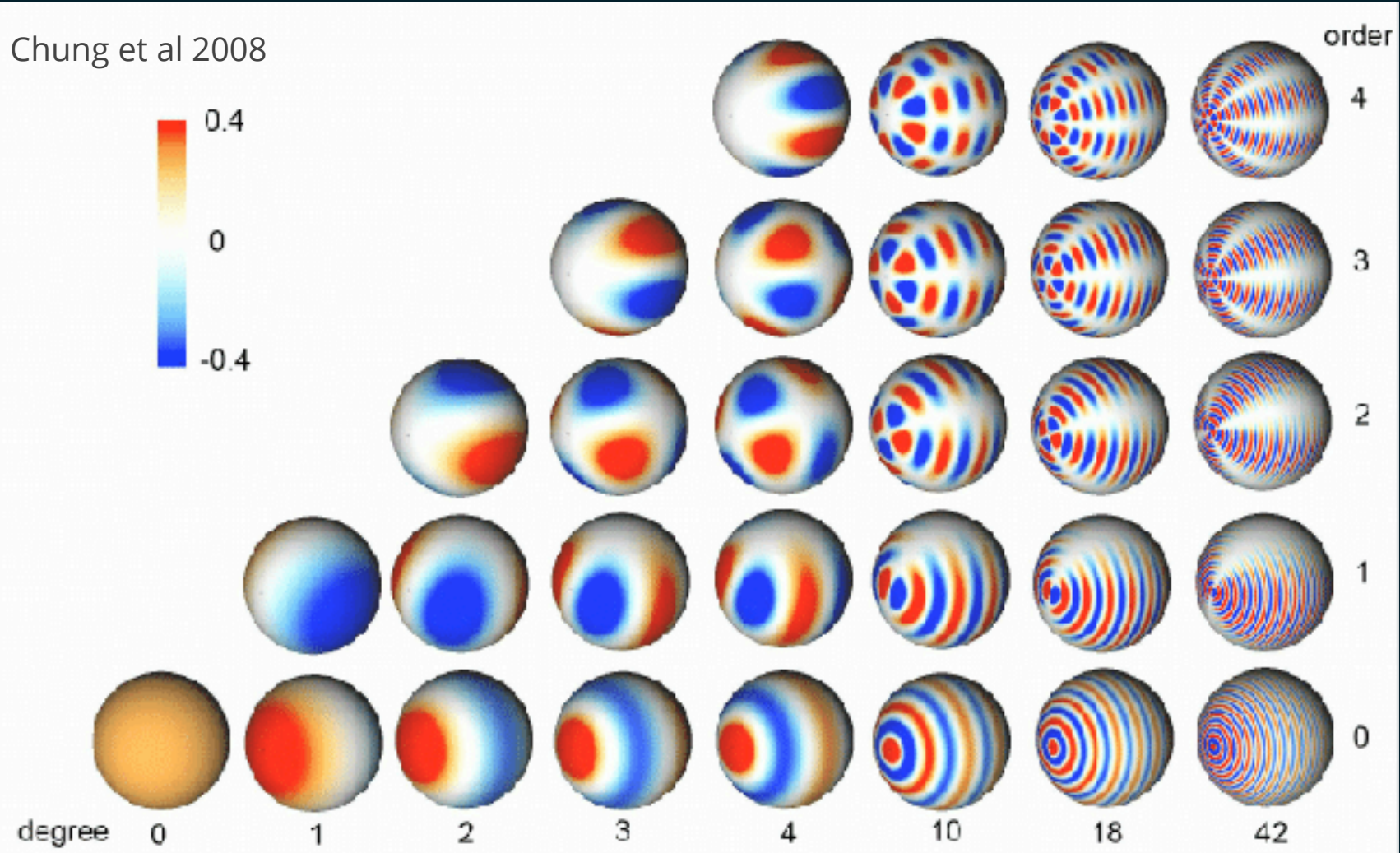
Digital twins, Stevens 2019



# Schools of thoughts: Data compression

## School 1: Transformations *(the physical perspective)*

- Spectral
- EOF
- Tensor trains
- Neural Networks
- Spatial structure
- Compute expensive
- Error bounds difficult
- No random access



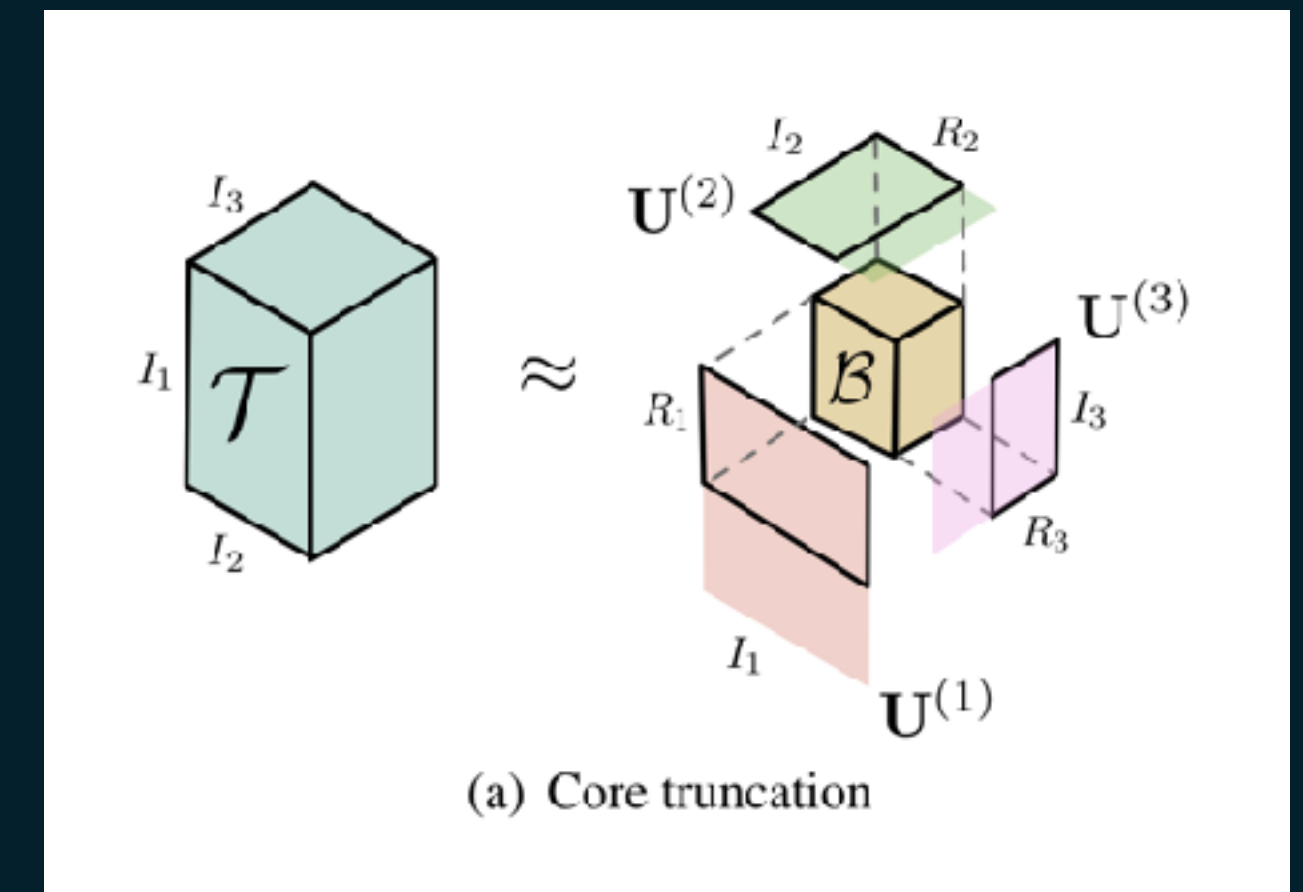
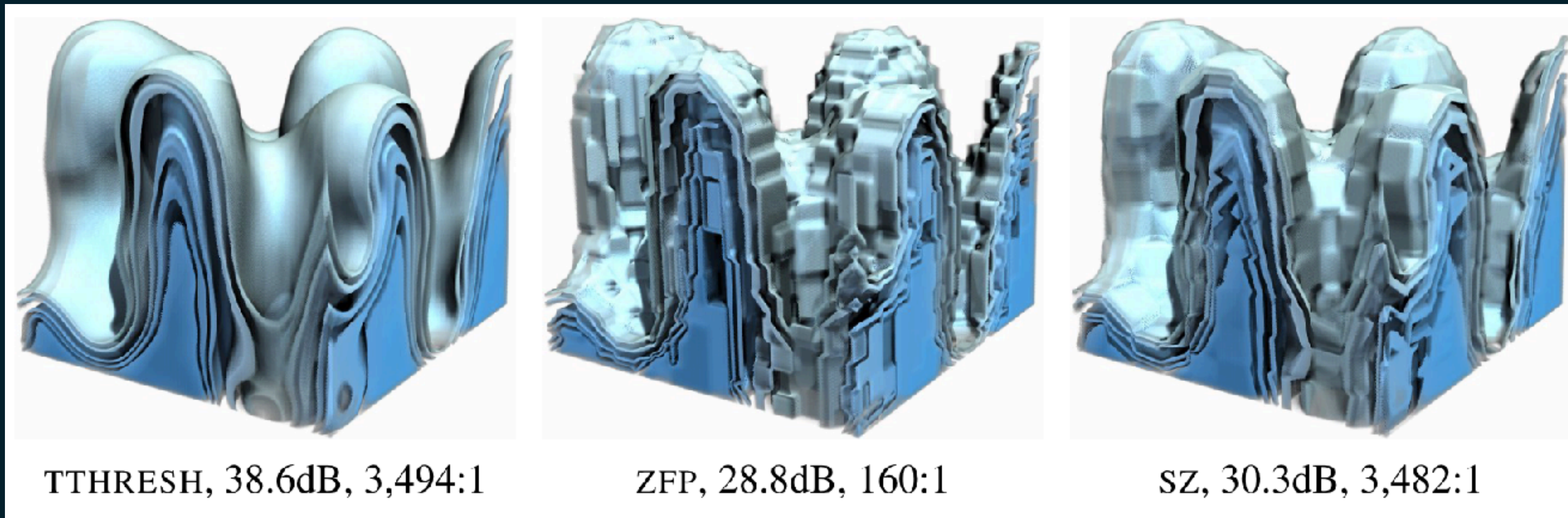
## School 2: Precision and information theory *(the binary perspective)*

- Bitwise encoding
- Floats; linear or logarithmic quantisation
- Entropy coding
- Lossless compression

- Spatial structure (X)
- Compute cheap (✓)
- Rigid error bounds (✓)
- Random access (✓)

# Transformations: TensorTrains

$$T = B \cdot U^{(1)} \cdot U^{(2)} \cdot \dots \cdot U^{(n)}$$



Ballester-Ripoll et al, 2019

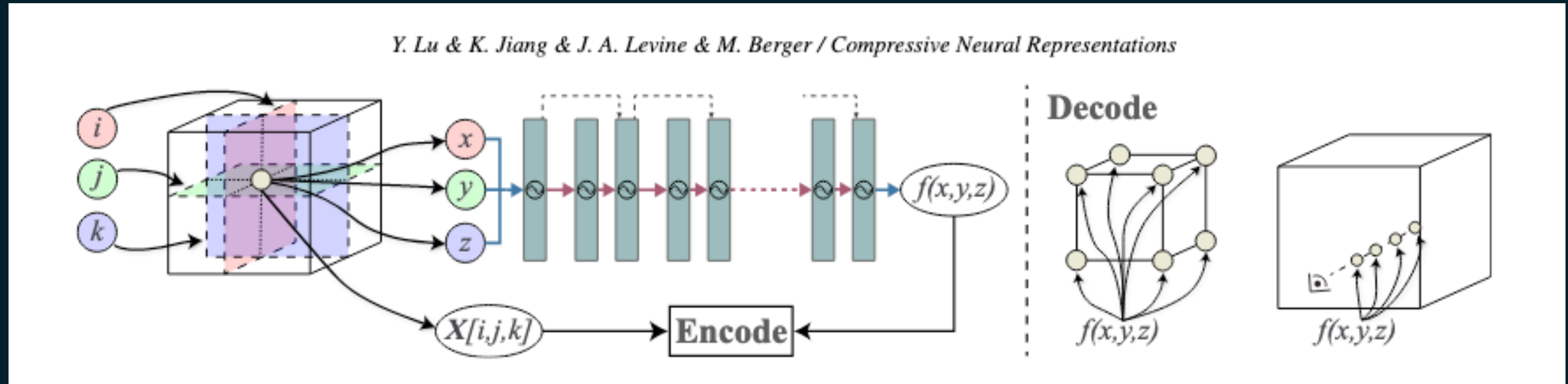
## Advantages

- Approximates well smooth data
- Claim: >1000x compression possible

## Disadvantages

- Expensive (de)compression
- Changing statistics?
- Unstructured grids?

# Neural network compression



## Advantages

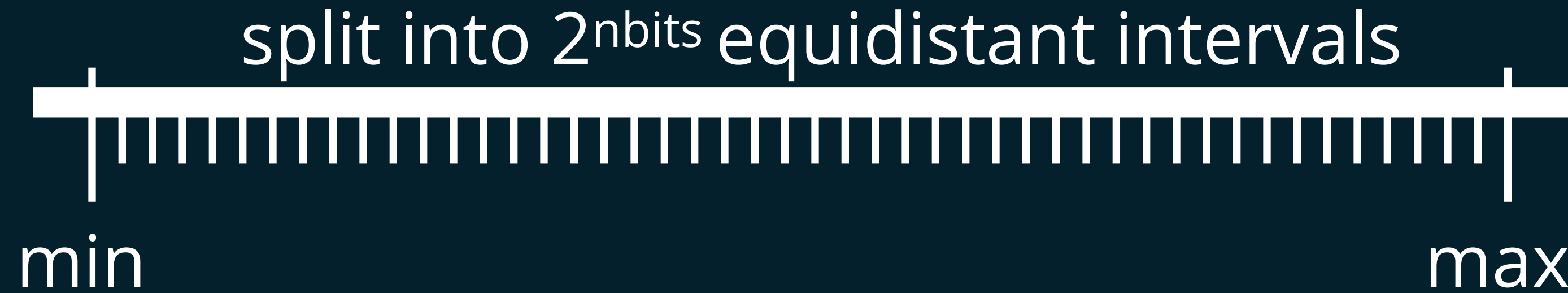
- Interpolates automatically
- Unstructured grids
- Error scales with training, i.e. compute
- ~1000x compression possible

## Disadvantages

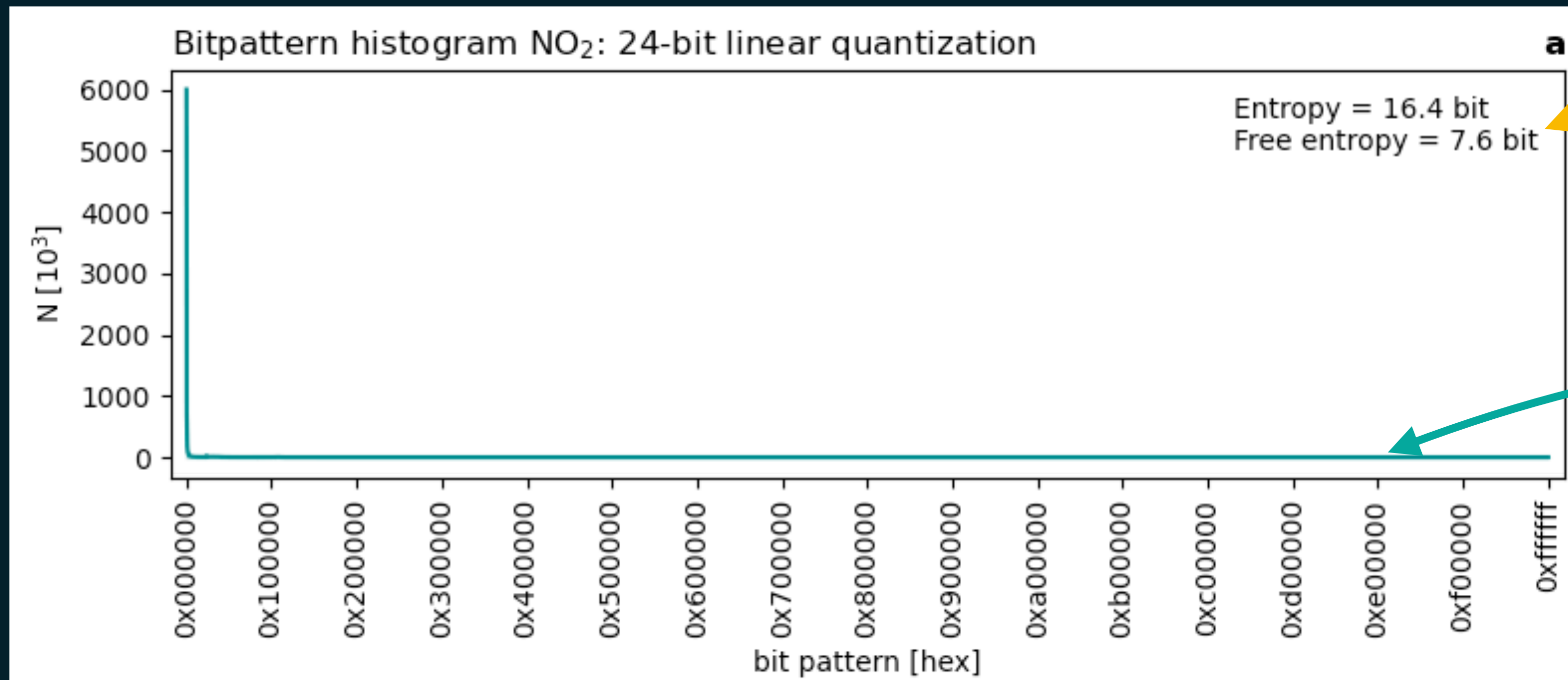
- Difficult to control the error
- Expensive (de)compression
- Size impacts the error

See Huang, Hoefler, 2023: *Compression of weather and climate data into neural networks*

# Current compression methods: 24-bit linear quantization



Use  $n_{\text{bits}} = 24$  instead of 64 bit per number

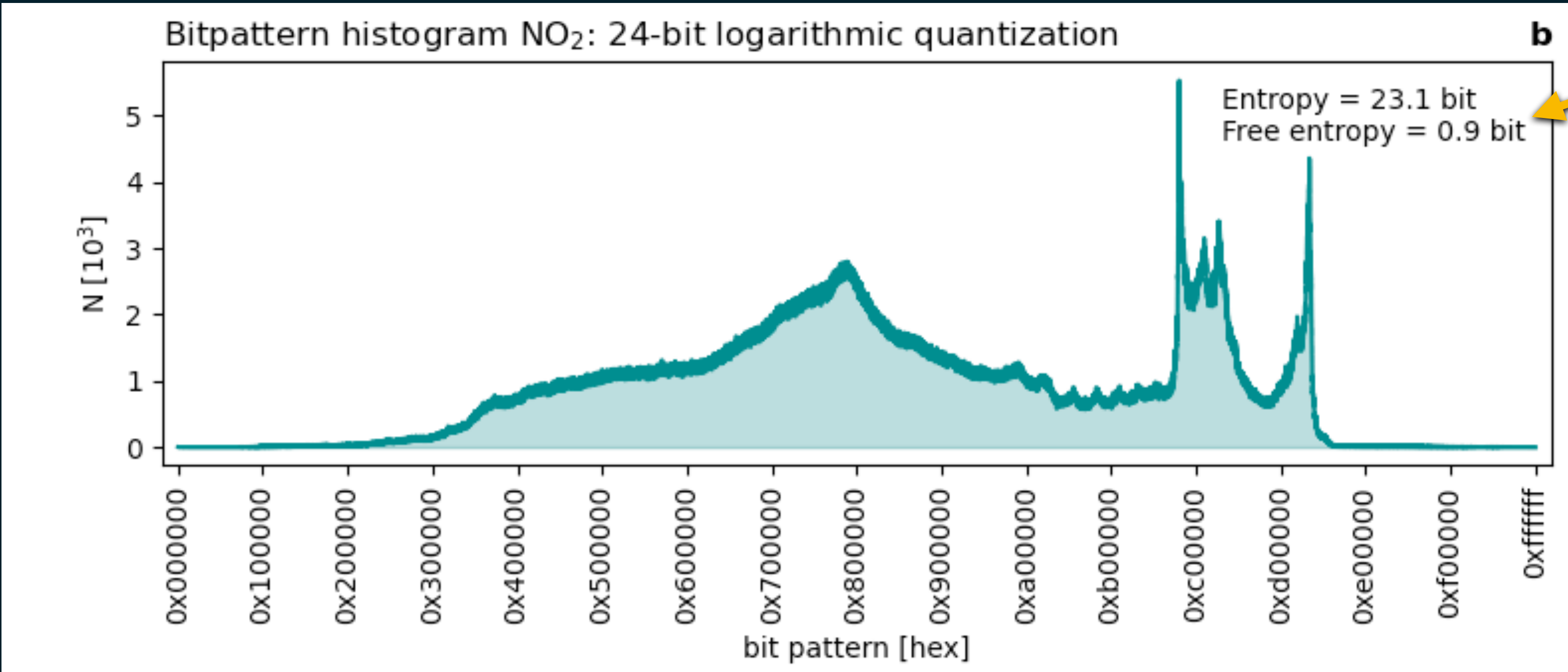


>7 bits are effectively unused

Most bit patterns very rarely used

*How much information is in the first bit?*  
*~0 bit. Information  $\leq$  Entropy.*

# Alternative: Logarithmic quantization



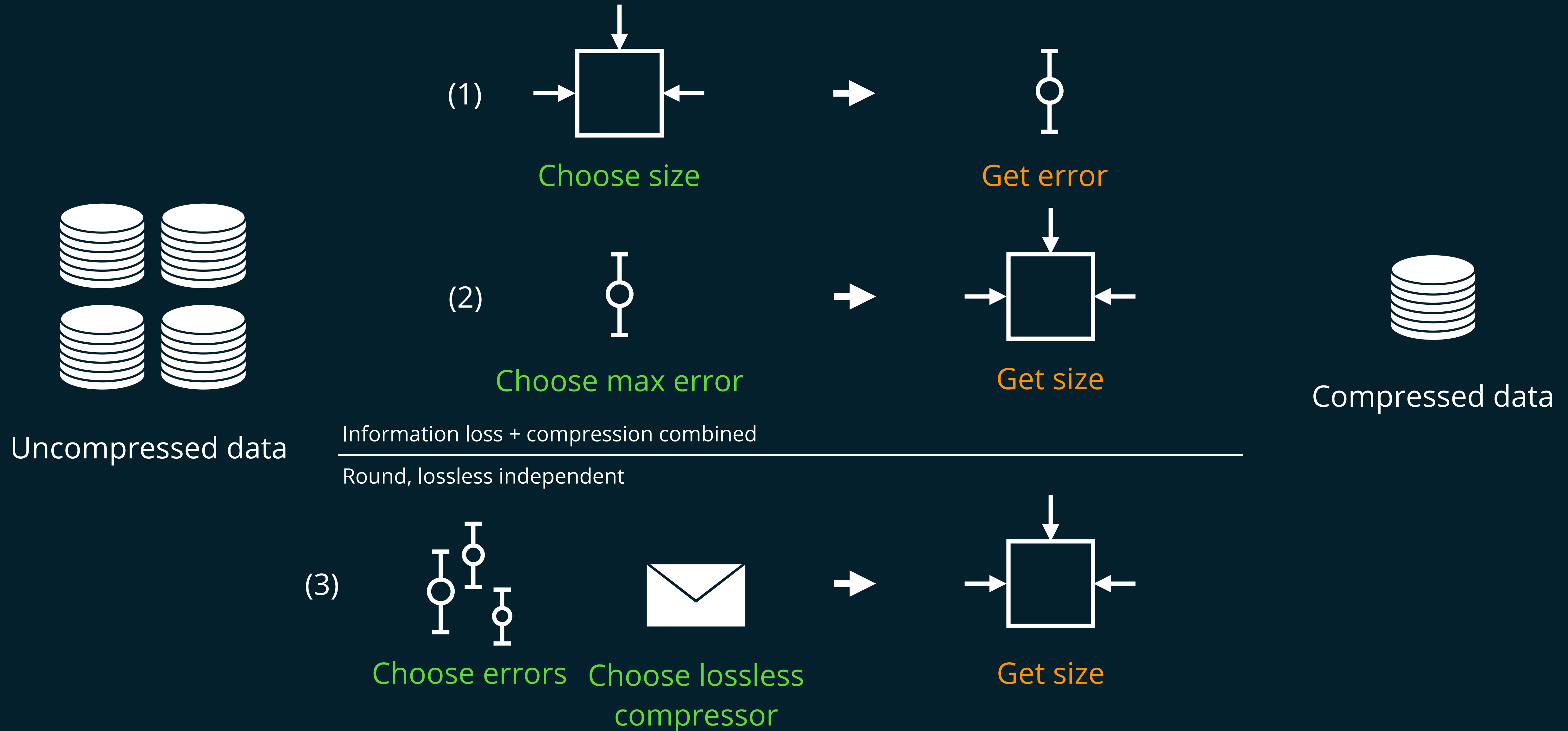
Most bits are effectively used

Floats are also approximately log-distributed

*How much entropy is in the first bit?*  
**~1bit**



# Data compression control knobs



# Data compression: What do we want?

## Case 1: Reanalysis data

### Important

- Small
- Decompression speed
- Portability
- Random access

### Less relevant

- Compression speed

## Case 2: Research simulation

### Important

- (De)compression speed
- Random access

### Less relevant

- Size

## Case 3: Long storage

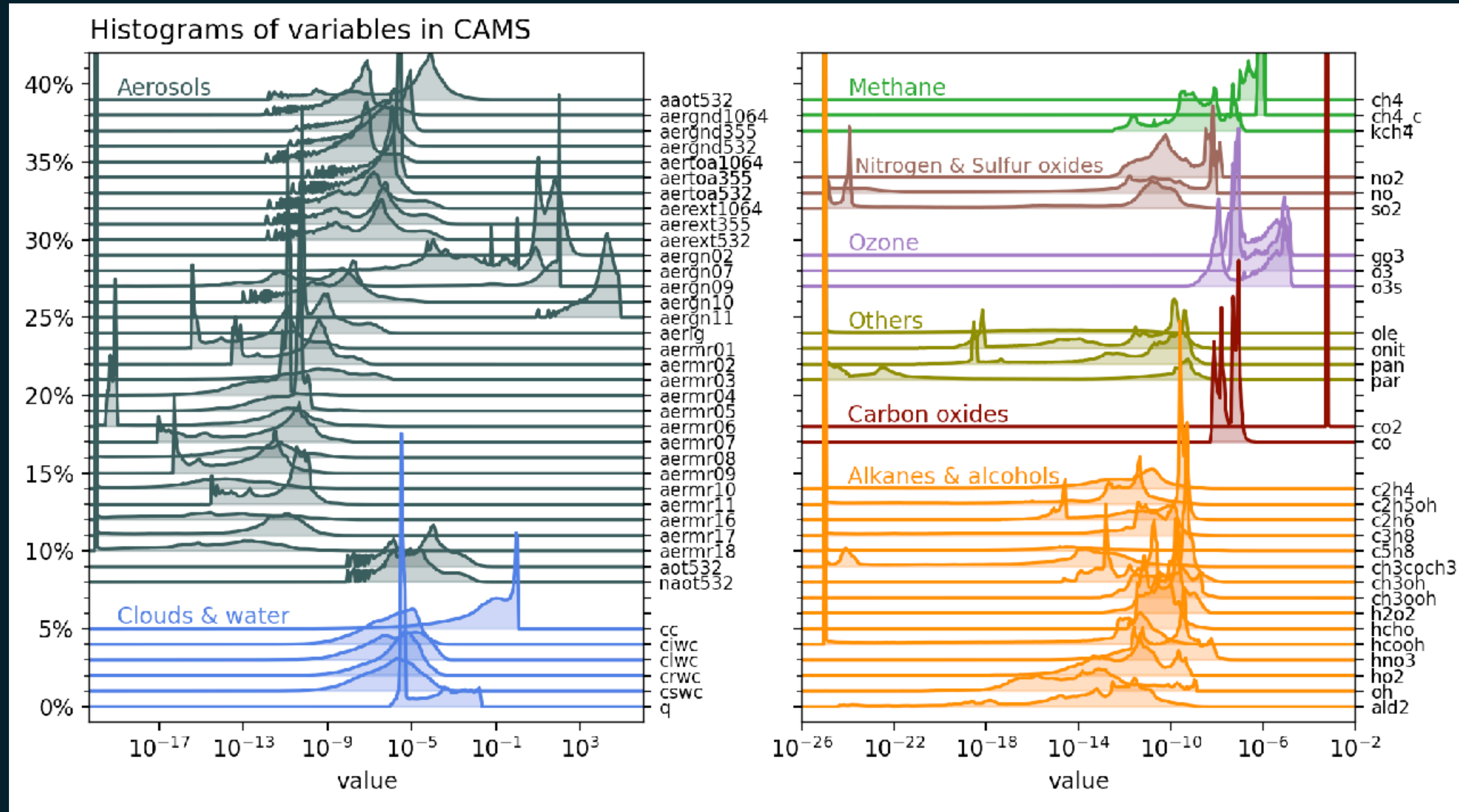
### Important

- Size

### Less relevant

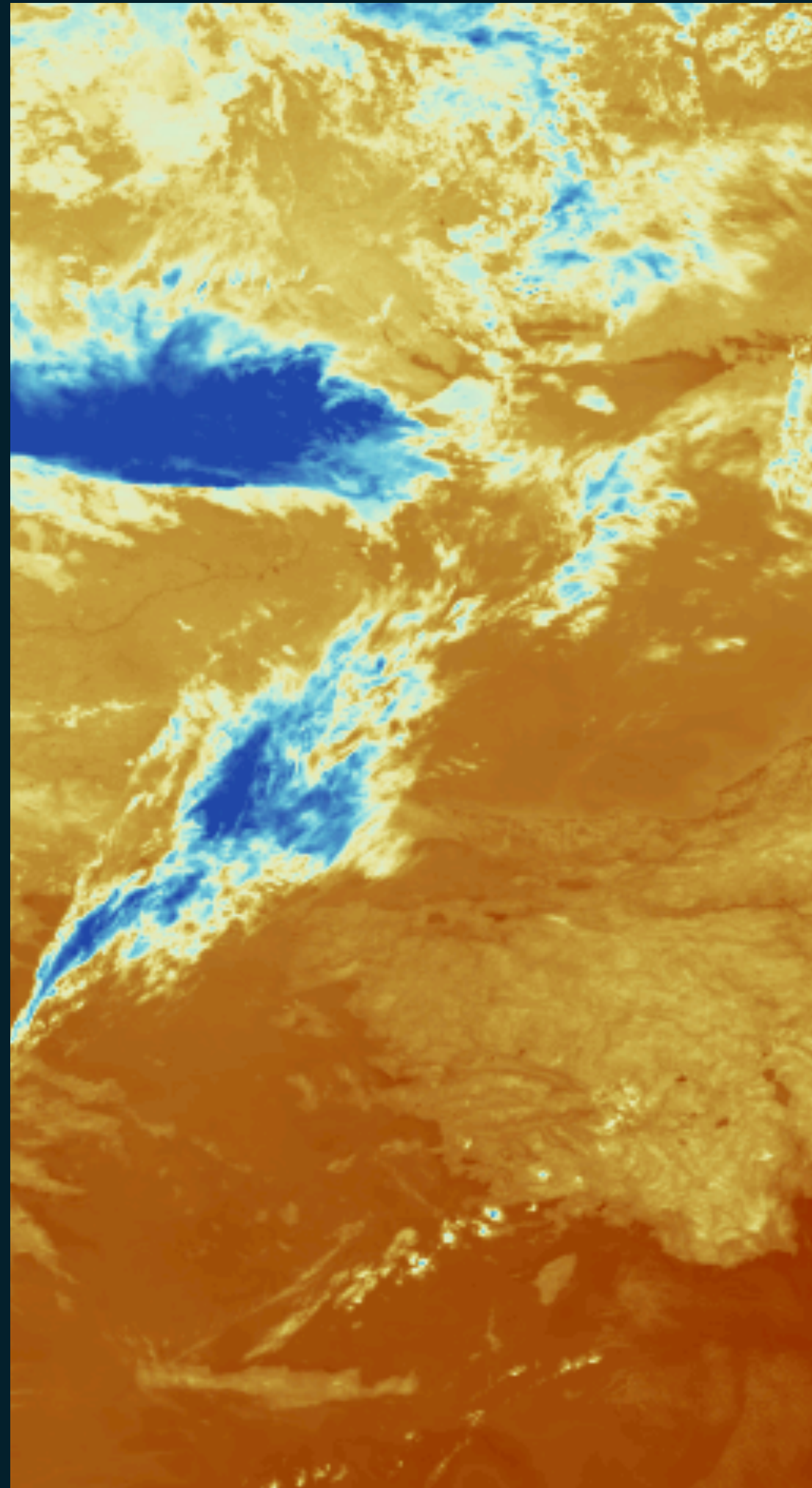
- Portability
- (De)compression speed
- Random access

# What do we compress?

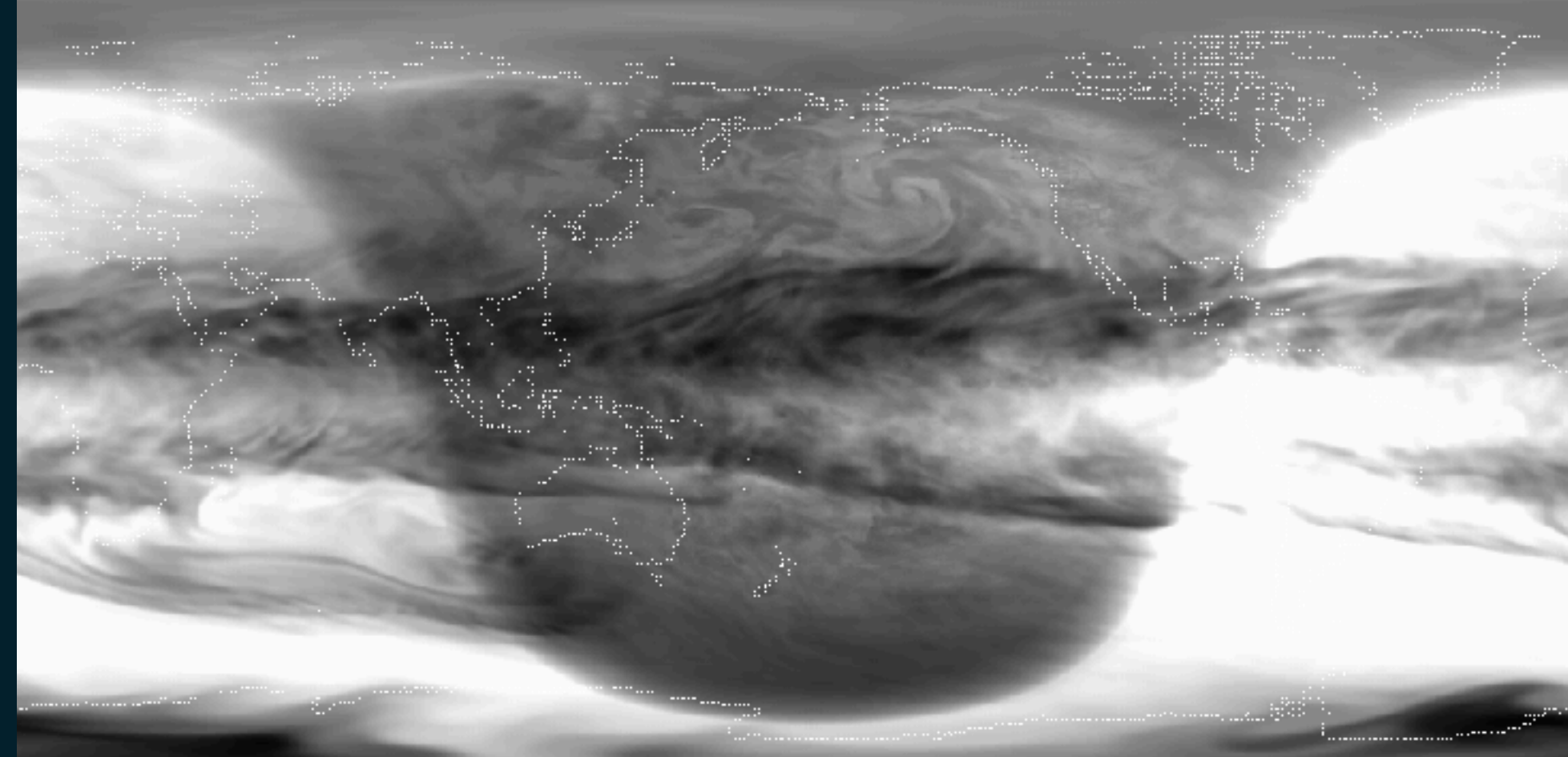


- **Many** different variables
- Varying uncertainties
- Linear or log-distributed
- Possibly many zeros

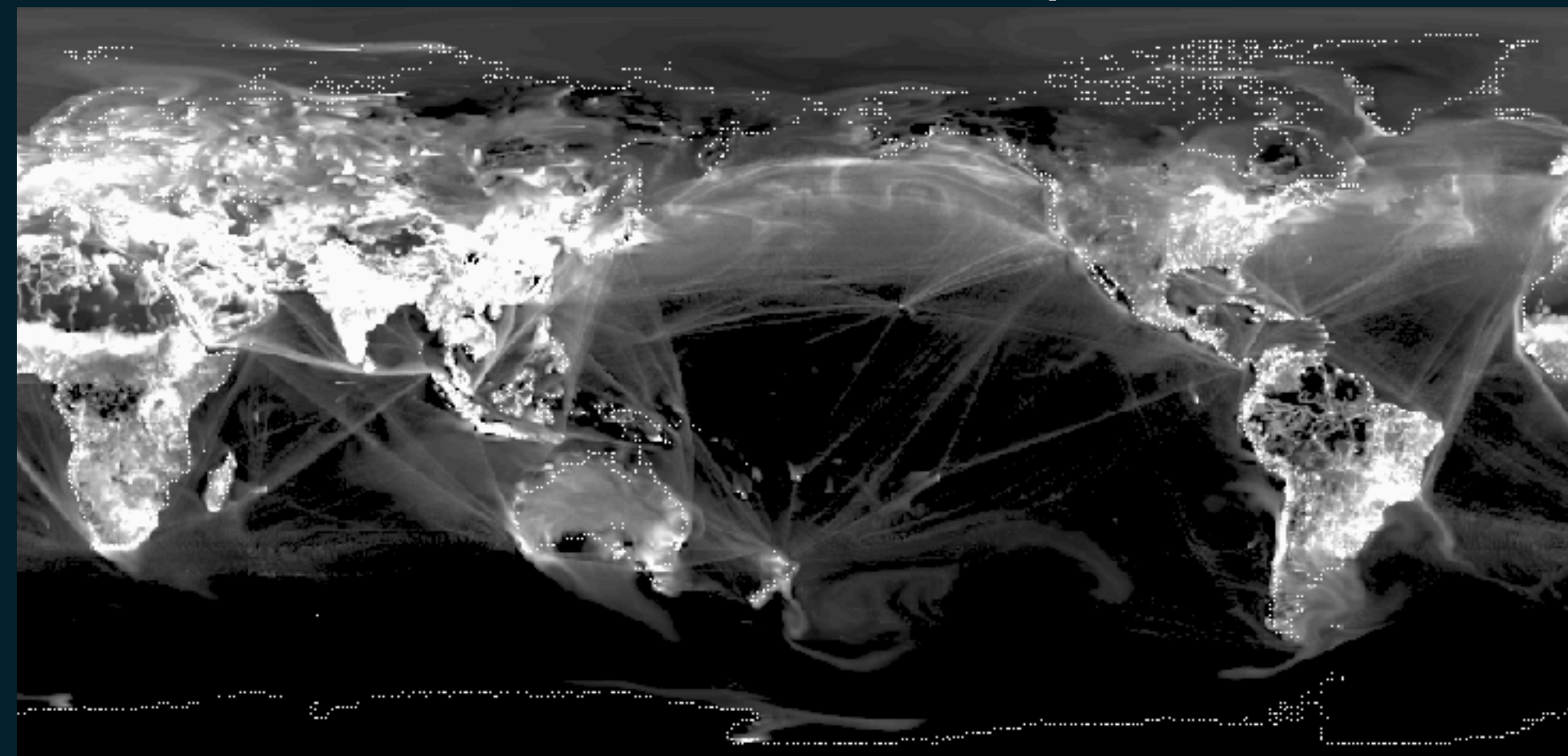
# What do we compress?



Brightness temperature



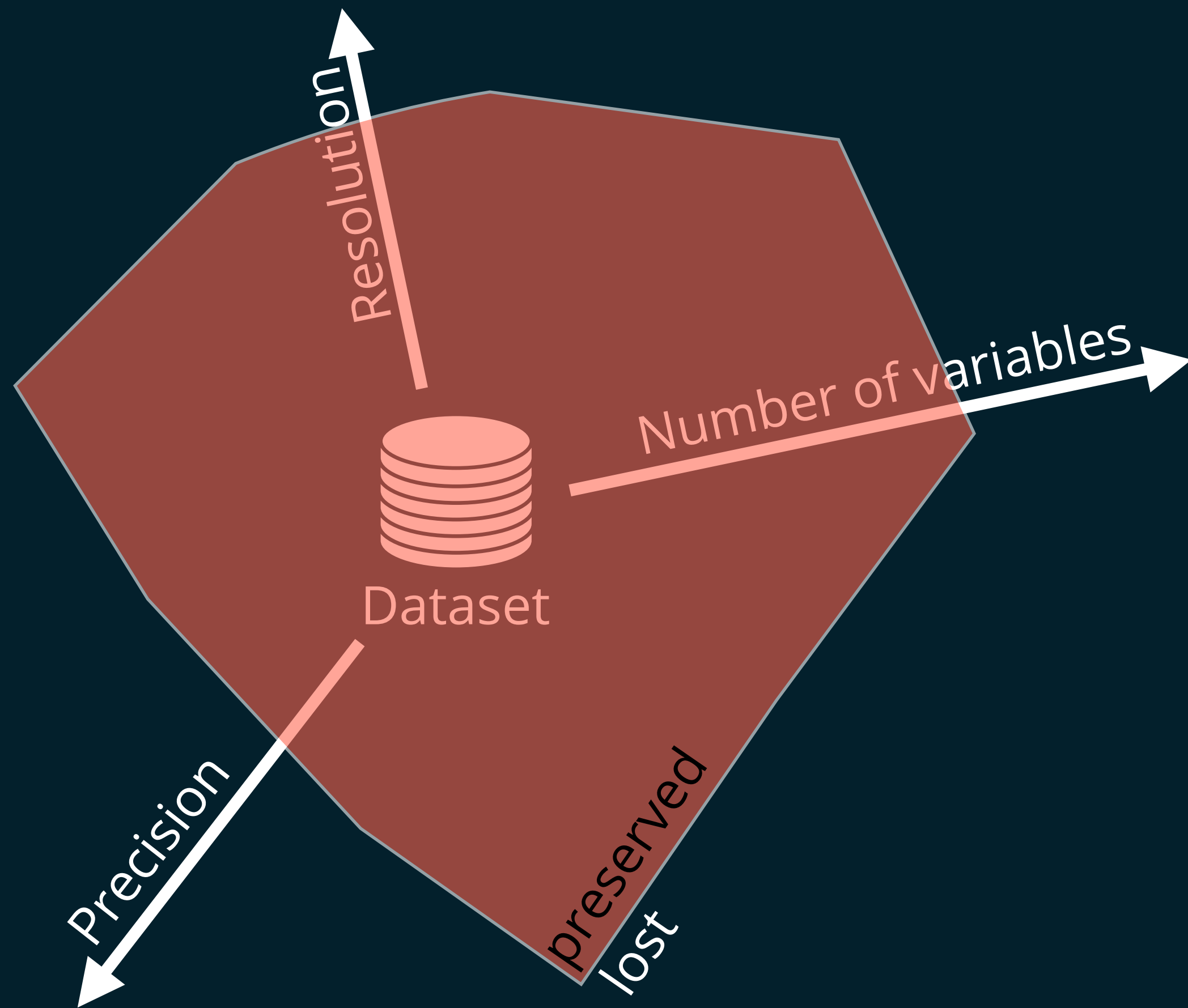
NO<sub>2</sub> in the stratosphere



NO<sub>2</sub> at the surface

- **Many** different variables
- Varying uncertainties
- Linear or log-distributed
- Possibly many zeros
  
- Smoothness varies spatially
- Strong gradients
- Point sources
  
- Unstructured grids
- Spectral coefficients
- Masked data

# What information is there in a dataset?



The *real information problem* of lossy data compression

$$F\left(\begin{array}{c} \text{Smallest} \\ \text{subset of} \\ \text{Dataset} \end{array}\right) = \text{Some answer}$$

Any question  
researchers ask

Qualitatively the same  
compared to full dataset

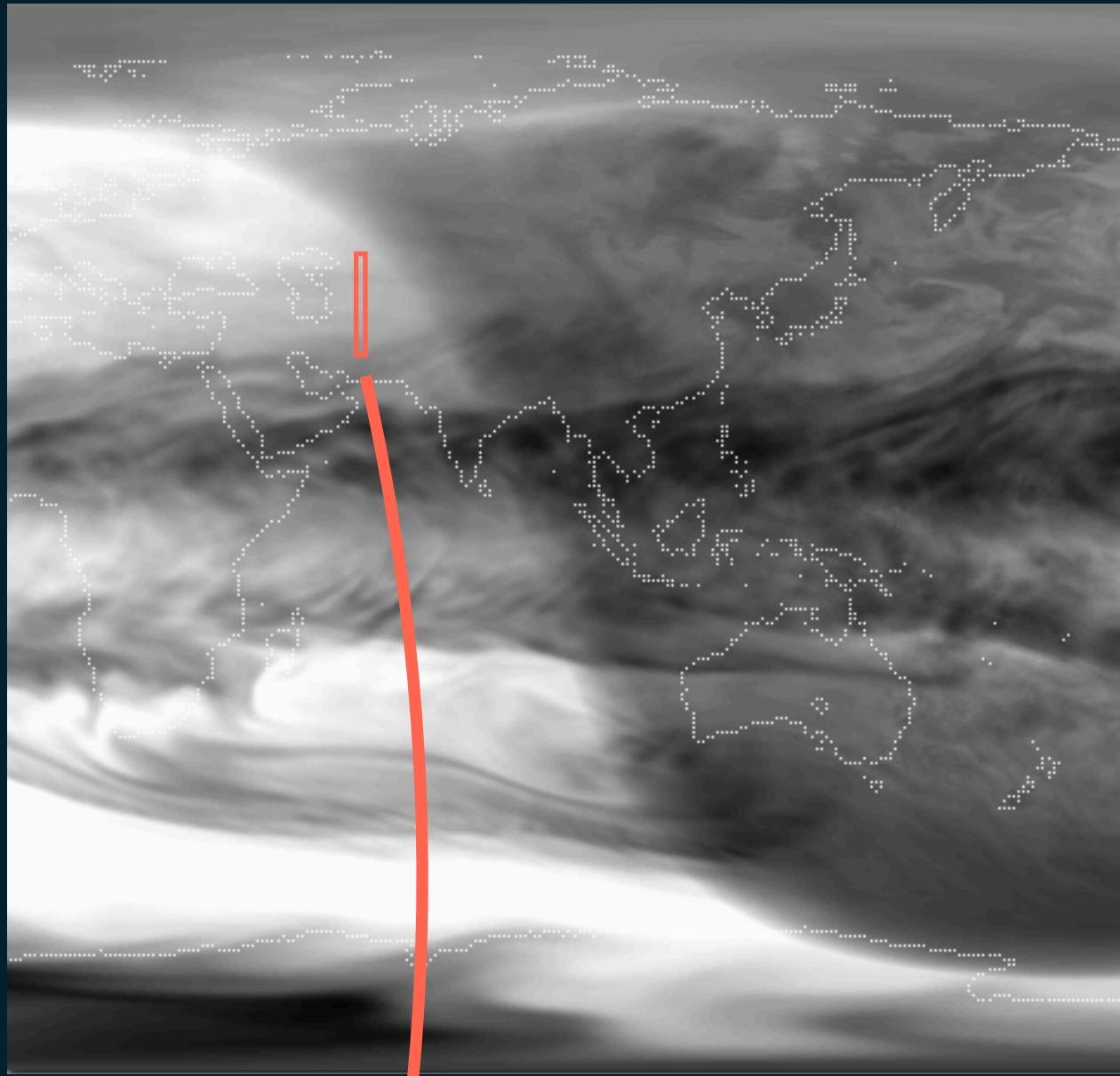
# What compression error is okay?



**The acceptable error** depends on

- Uncertainty, changes with:
- Variable and unit
- Application
- Model and its resolution
- ...

***Problem: What is the uncertainty in data and how can it be estimated if unknown?***



# What is real and false information in data?

*Problem: What is the uncertainty in data and how can it be estimated if unknown?*

Possible solution:

**Find an objective way to separate real and false information!**

0.050386034  
0.050390966  
0.05040059  
0.050441727  
0.05046302  
0.05046855  
0.050488267  
0.05050127  
0.050520953  
0.05052939  
0.050532646

Encoded in bits

Real!?

False!?

```
00111101010011100110000110010110  
00111101010011100110011011000010  
00111101010011100111000011011001  
00111101010011101001101111111100  
00111101010011101011001001010000  
00111101010011101011100000011100  
00111101010011101100110011001001  
00111101010011101101101001101011
```

# Mutual information of adjacent grid points

...1101000000011000001100001000011000000100...

*0 is mostly followed by a 0; 1 either remains 1 or switches back to 0.*

Joint probability matrix: 
$$\begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} = p_{rs} = \begin{pmatrix} 0.6 & 0.1 \\ 0.1 & 0.2 \end{pmatrix}$$

Mutual information:

*What does one bit tells about the next?*

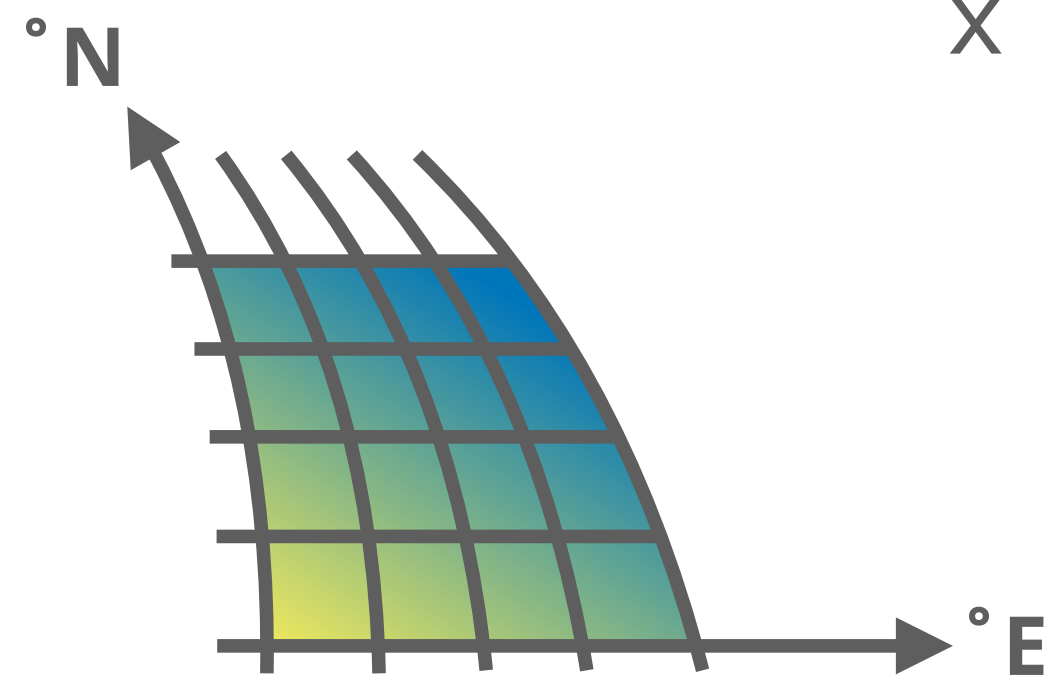
$$M = \sum_r \sum_s p_{rs} \log_2 \left( \frac{p_{rs}}{p_{r=r} p_{s=s}} \right) = 0.2 \text{ bit}$$



# Bitwise real information content

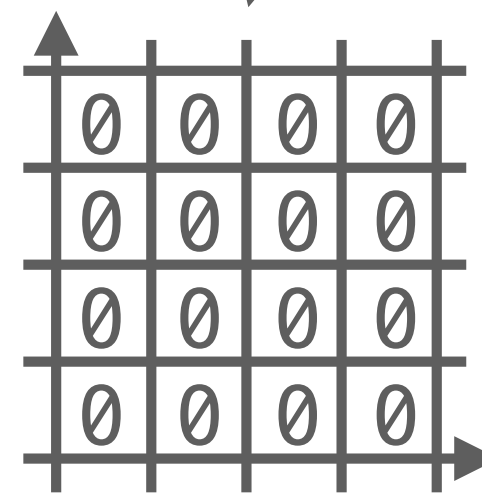
defined here as the **mutual bitwise information in adjacent grid points**

## 1 Gridded data



## 2 Data as binary floating-point numbers

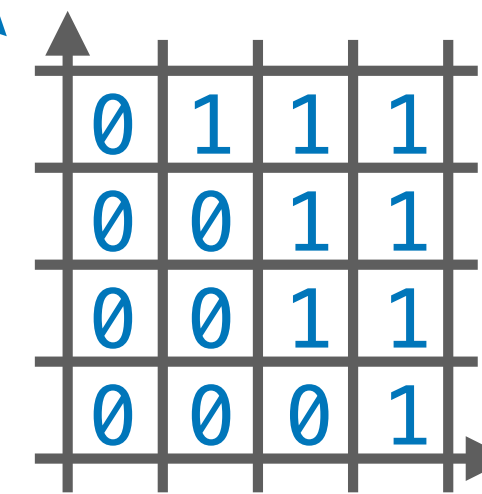
$x = 1.23\dots = 0$  **sign** **exponent** **mantissa bits**  
 $0$   $01111111$   $00110100100011001010001$



sign bits

**M = 0 bits**

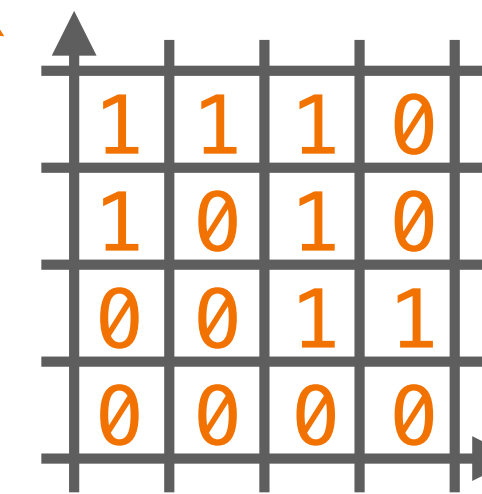
If all bits are identical



5th exponent bits

**M ≈ 1 bit**

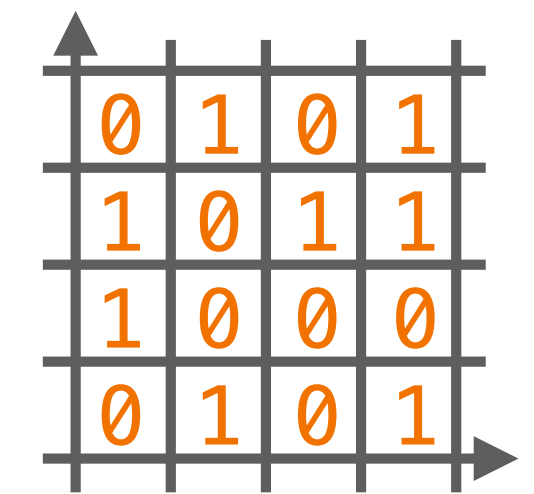
If 0 is **certainly adjacent** to 0; 0 and 1 occur equally frequent



8th mantissa bits

**M > 0 bit**

If 0 is **likely adjacent** to 0 and 1 is likely adjacent to 1



last mantissa bits

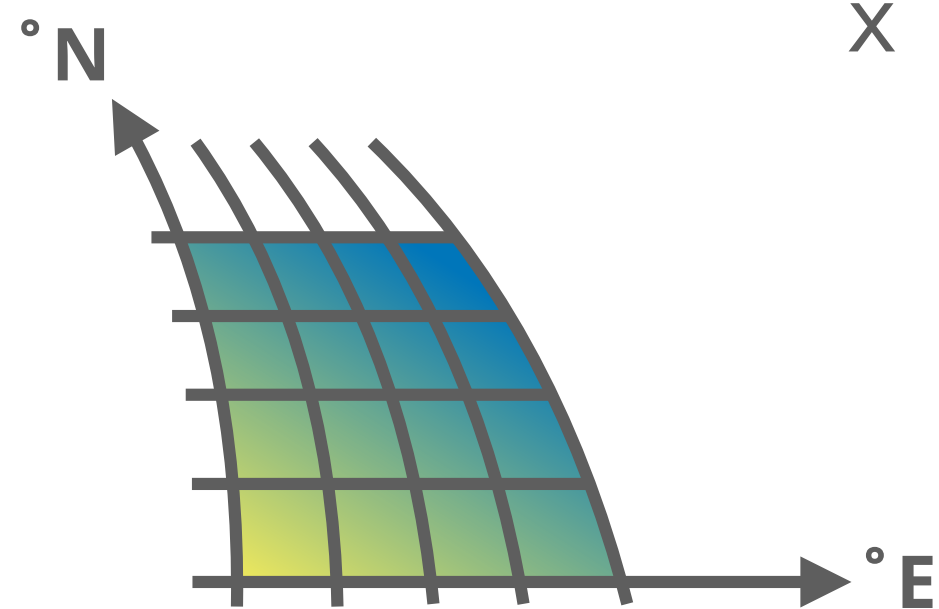
**M = 0 bit**

If adjacent bits are **independent**

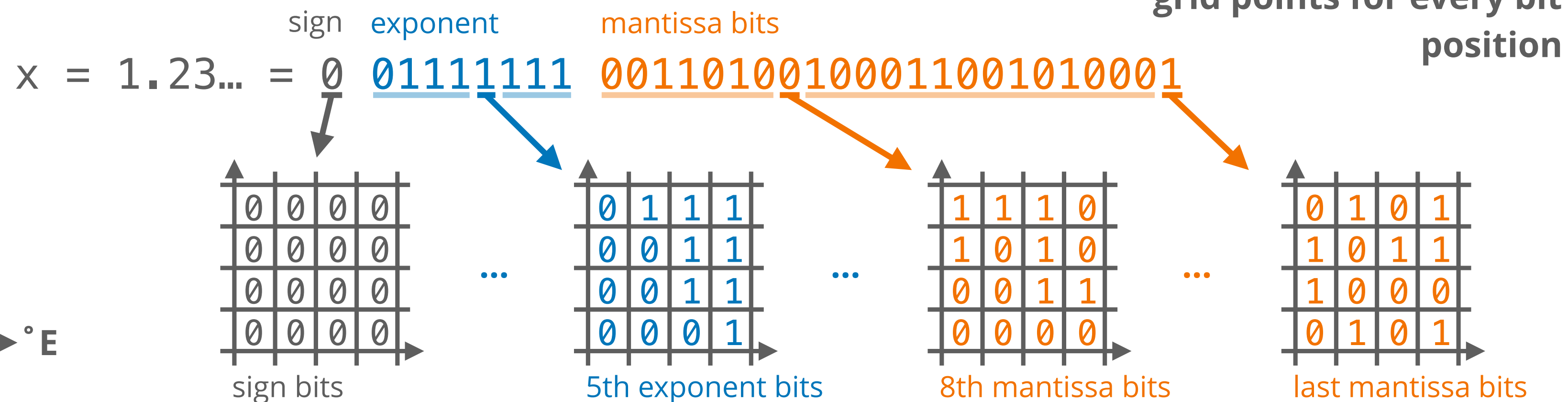
## 3 Analyse bits in adjacent grid points for every bit position

## 4 The mutual information M between bits in adjacent grid points

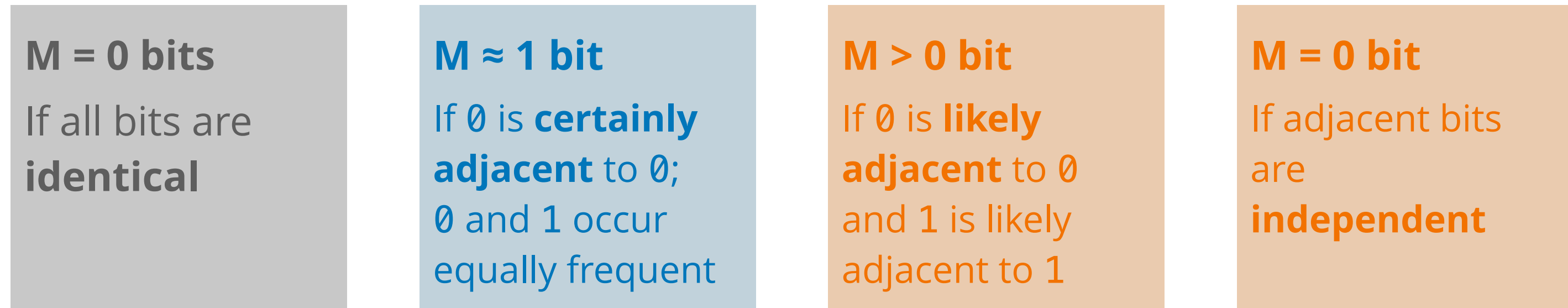
**1** Gridded data



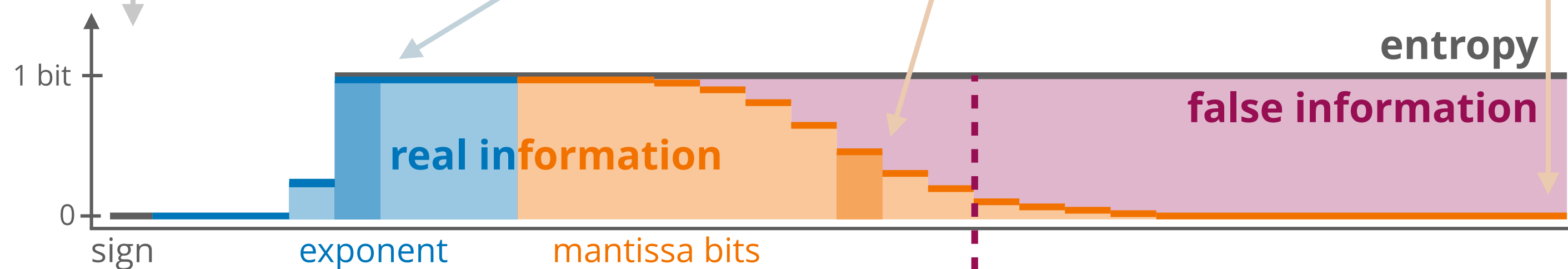
**2** Data as binary floating-point numbers



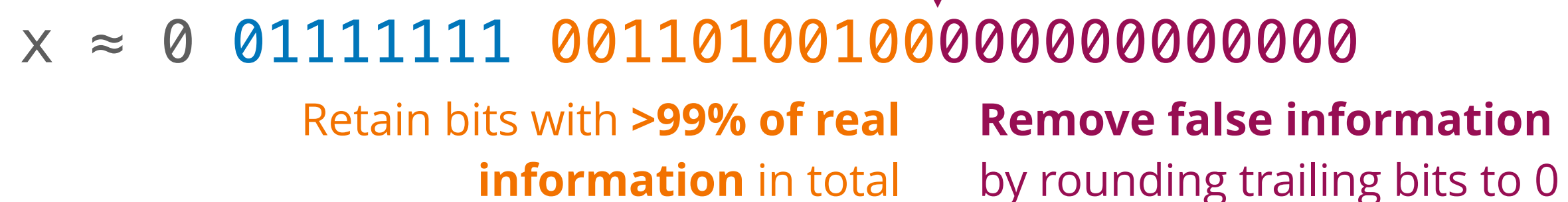
**4** The mutual information  $M$  between bits in adjacent grid points



**5** Bitwise real information is the mutual information between bits

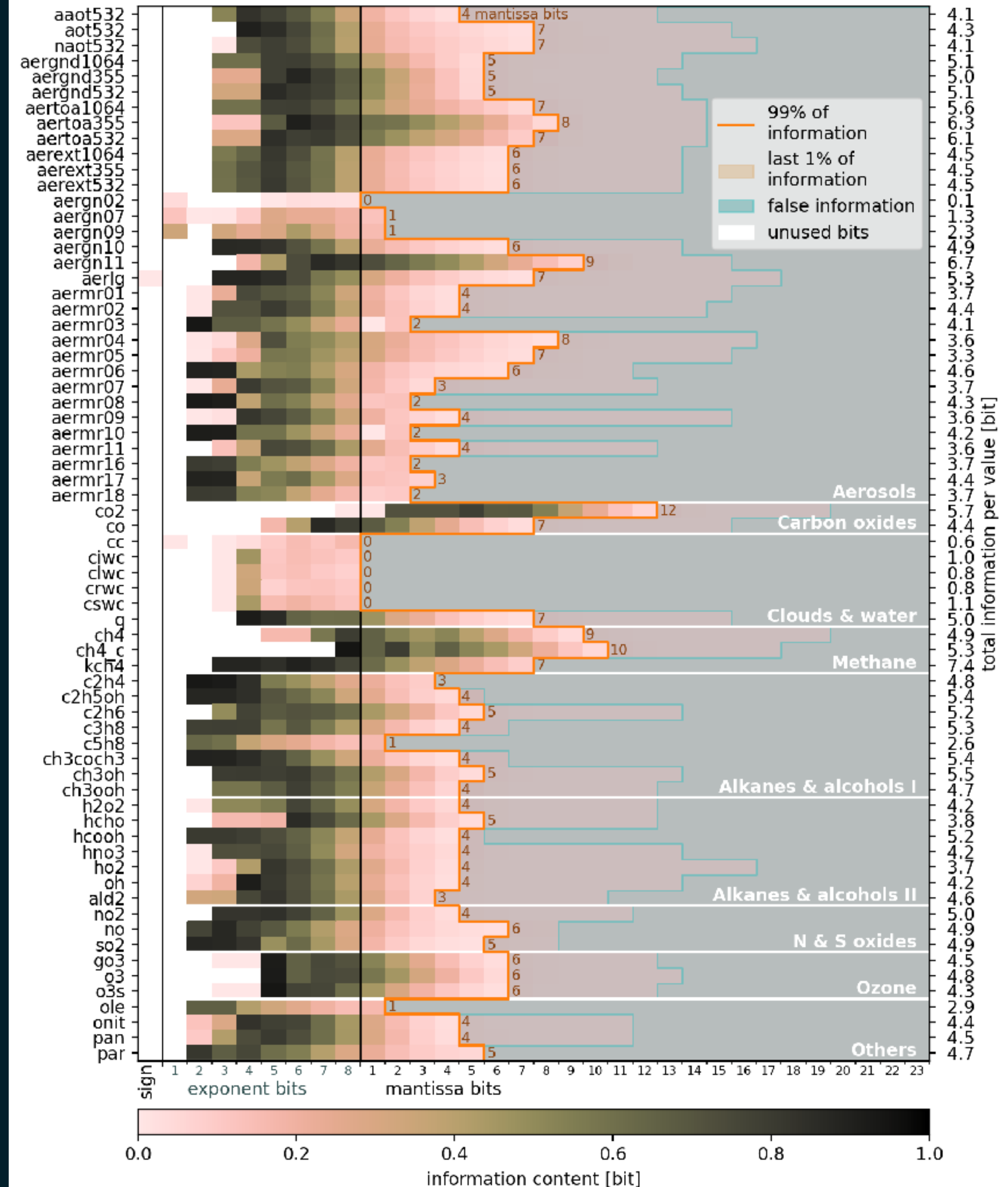


**6** Rounding to remove the false information

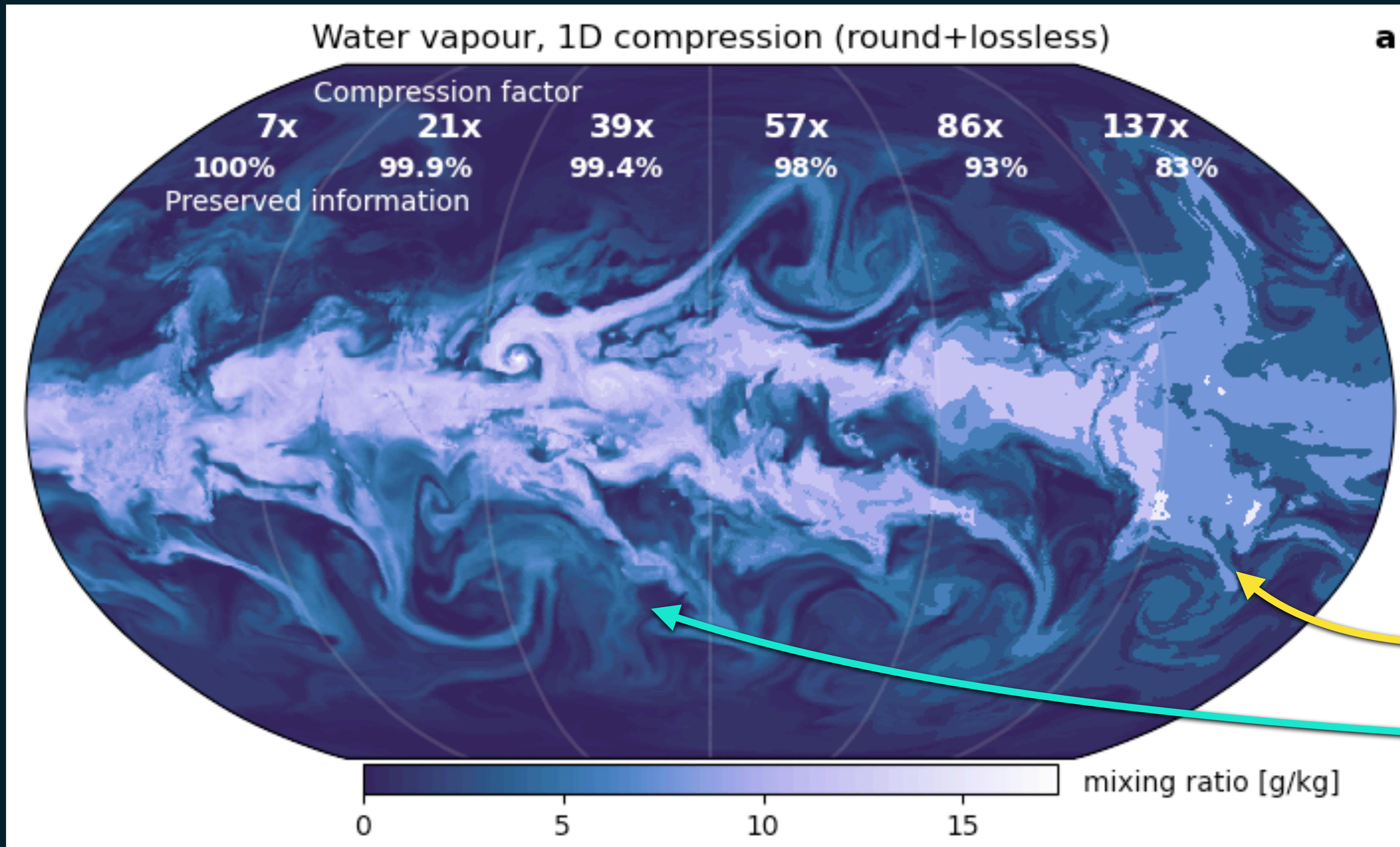
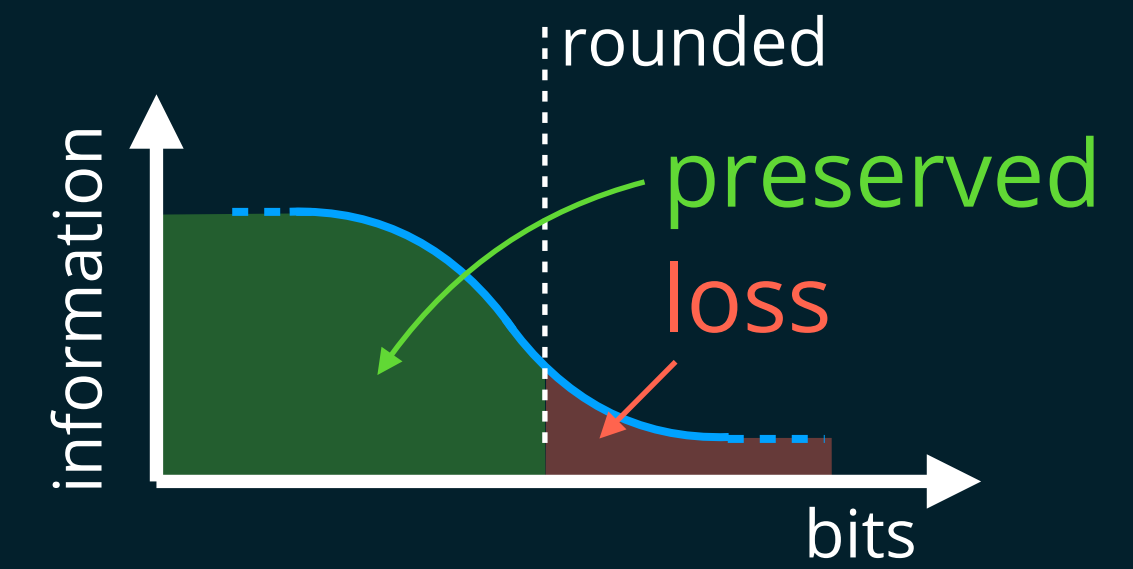


# Bitwise real information content

- Every variable requires a different precision
- Many bits do not contain real information
- Preserve only the bits with real information



# Information-preserving compression

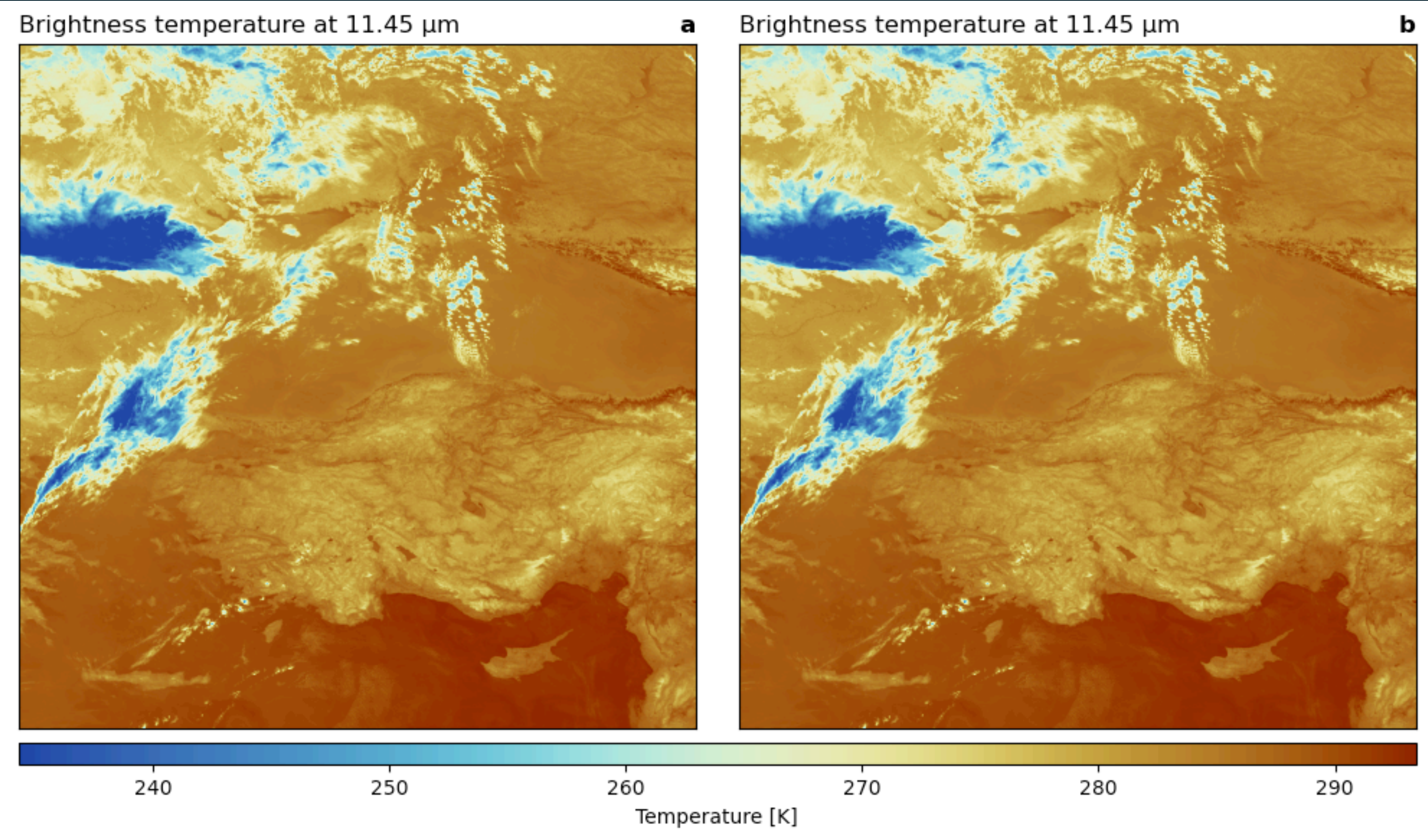


**Visual artefacts**

**99% preserved information suggested as sweet spot**

# Information-preserving compression

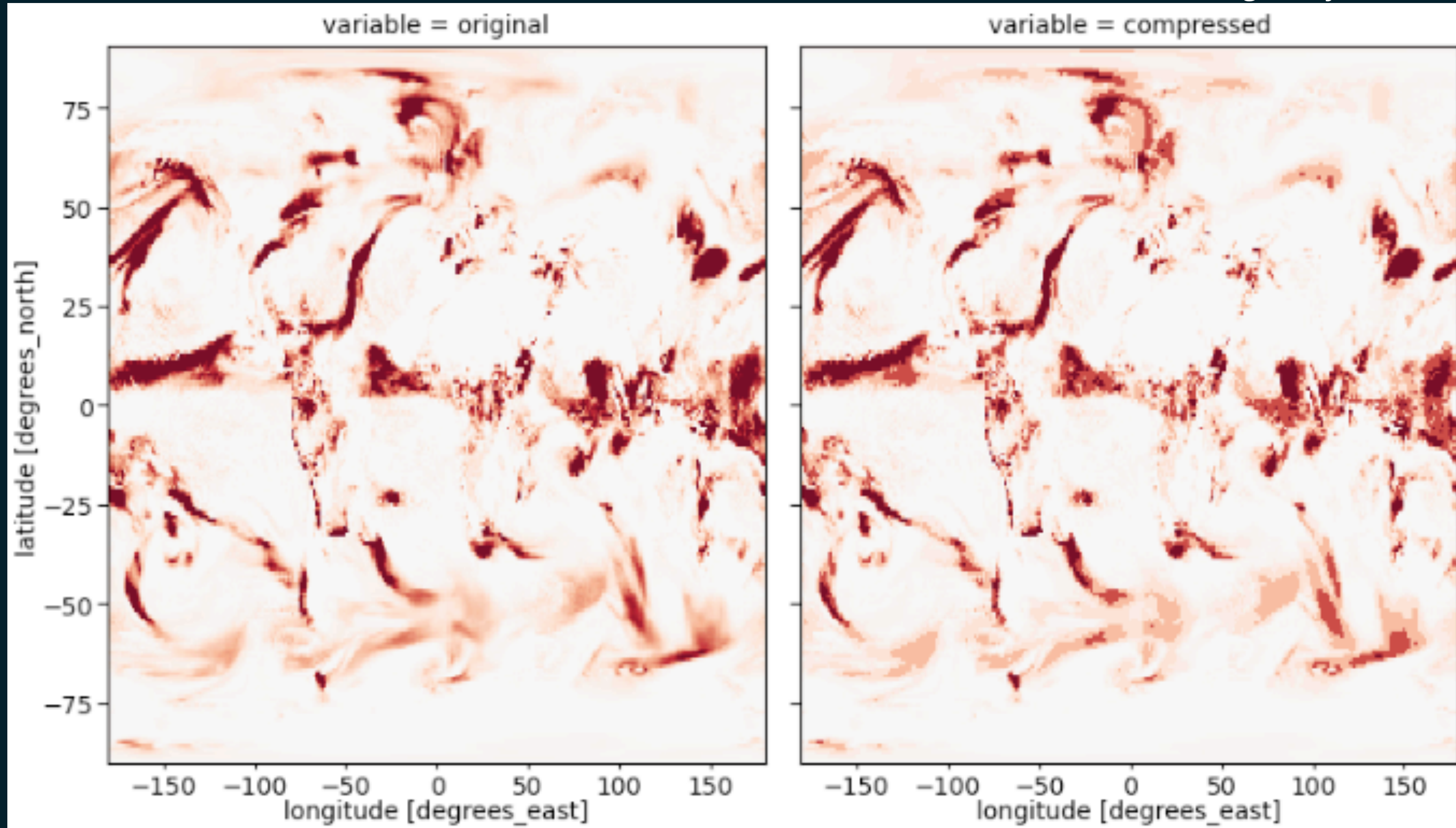
*One is 15-20x smaller than the other, which has been compressed?*



# The ugly?

Precipitation

Aengenheyster 2022

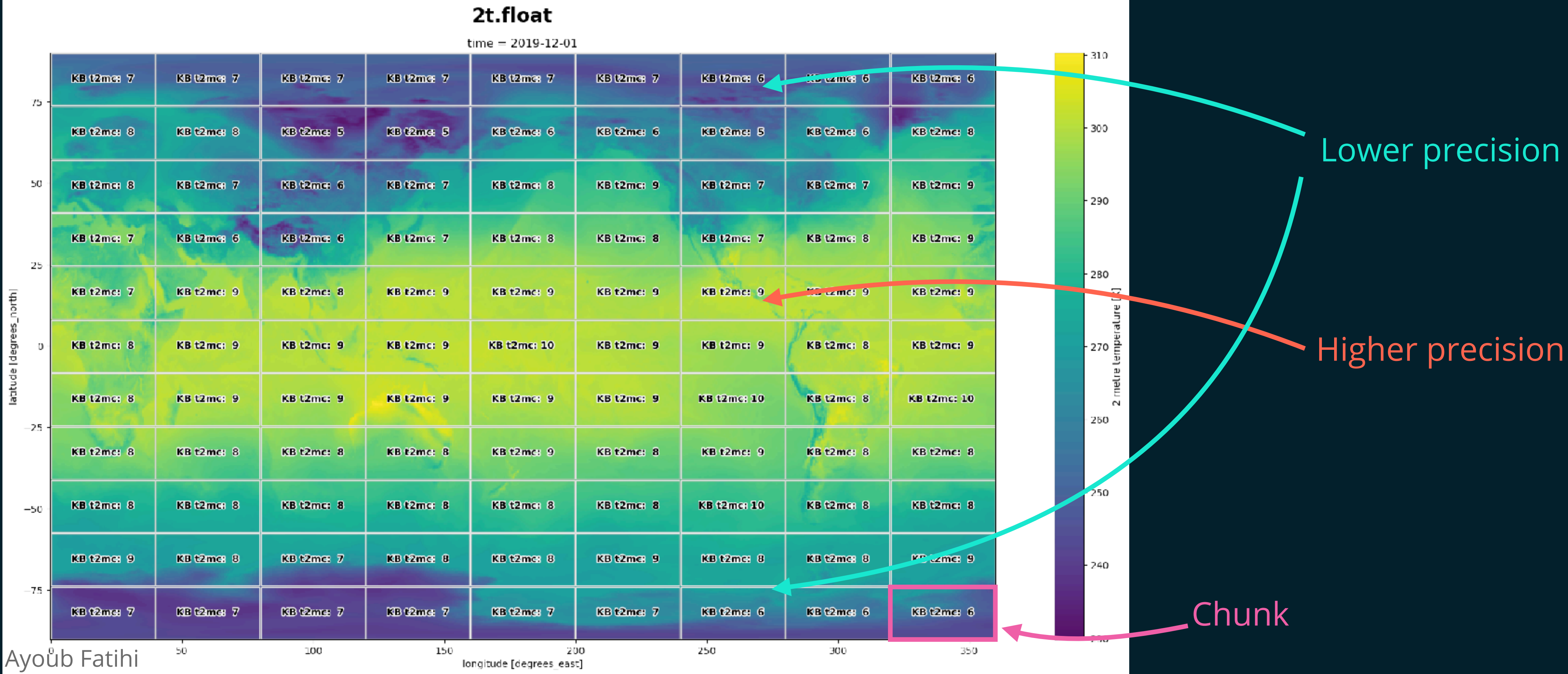


**Problem:** Both smooth and rough data

**Solution:** Independent information analysis and rounding by region

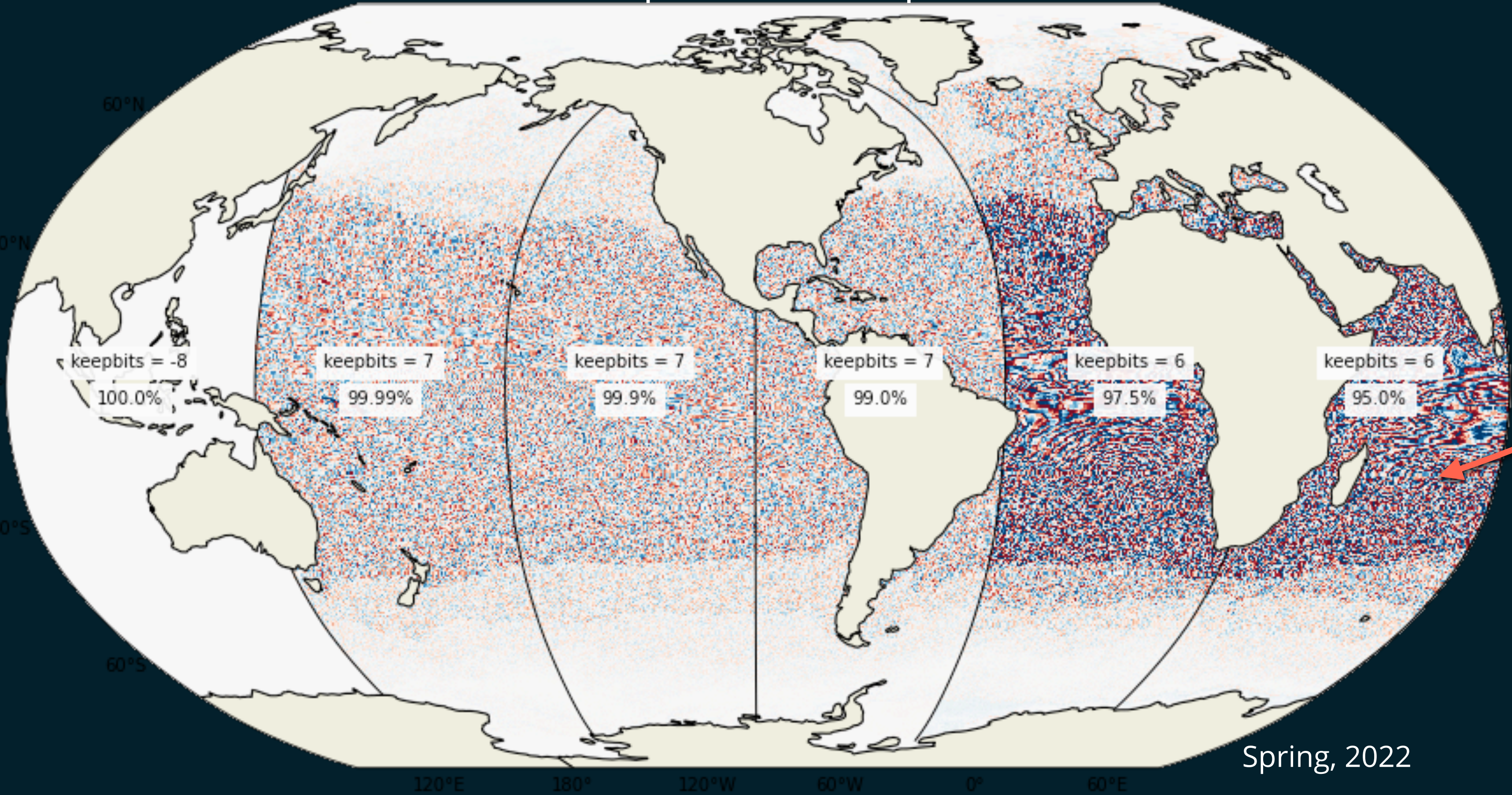
# Variable-precision compression

Higher precision in one region, lower in another?



# The ugly 2?

Sea surface temperature compression error



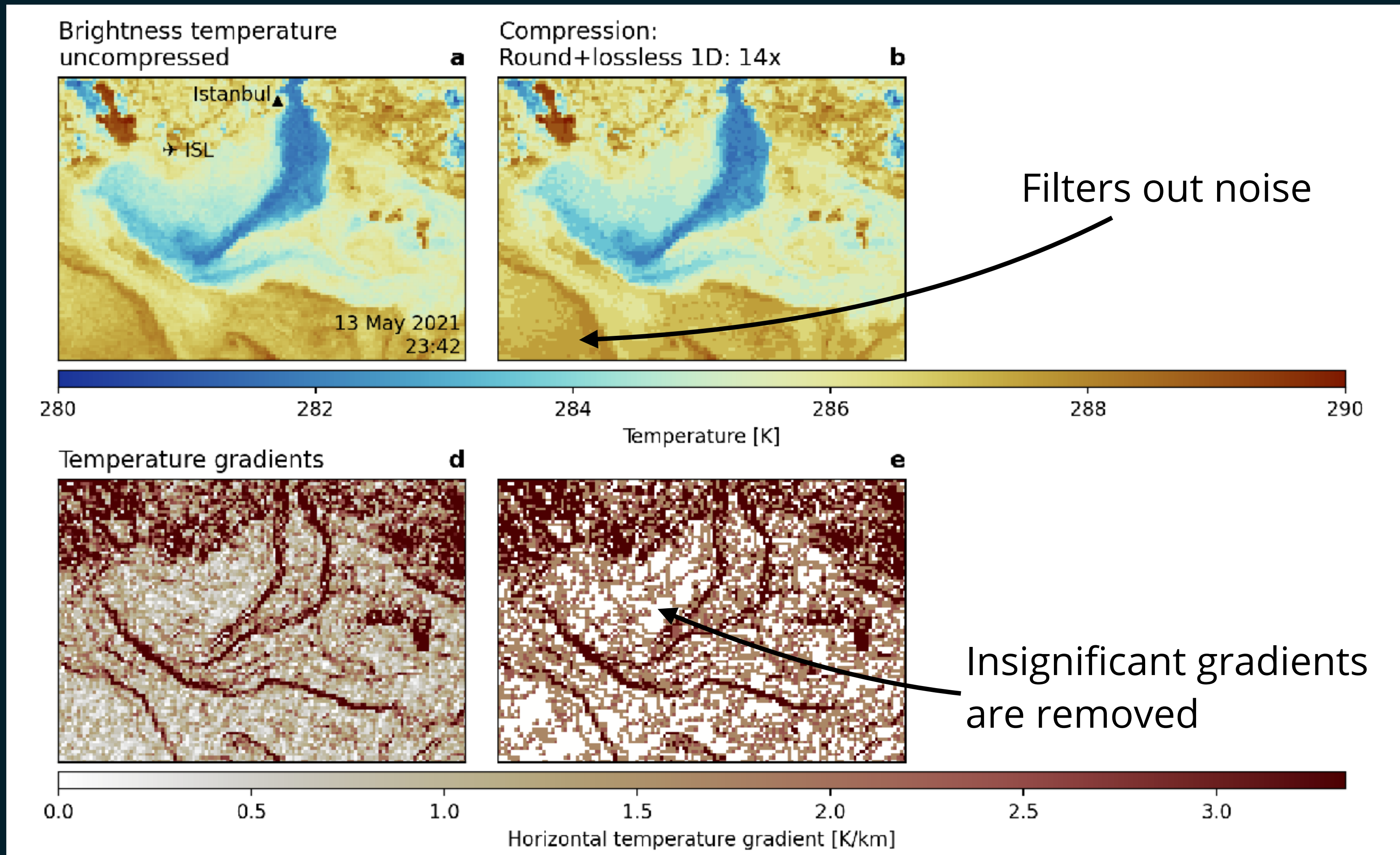
Compression error higher in tropics

**Problem:** Absolute error vs log encoding  
**Solution:** Adapt encoding to data or variable keep bits



# The bad? What about gradients!?

*Filtering out insignificant gradients*



# Workflow

## 1. Information analysis

- Only once offline
- Yields acceptable error bounds

## 2. Rounding

- Very fast
- Variable precision possible

## 3. Lossless compression

- Any codec can be used
- Flexible size/performance trade-off
- Chunking possible

# Implementations

Julia: BitInformation.jl

The screenshot shows the GitHub repository for BitInformation.jl by milank. The repository description is "Information between bits and bytes." It lists 5 stars and 1 fork, with 100% of the code being Julia. The repository name is BitInformation.jl. A brief description states: "BitInformation.jl is a package for bitwise information analysis and manipulation in Julia arrays. Based on counting the occurrences of bits in floats (or generally any bits type) across various dimensions, this package calculates quantities like the bitwise real information content, the mutual information, the redundancy or preserved information between arrays." It also mentions various rounding modes and bitwise operations.

Milan Klöwer

Python & xarray: xbitinfo

The screenshot shows the GitHub repository for xbitinfo by observingClouds. The repository description is "Python wrapper of BitInformation.jl to easily compress xarray datasets based on their information content." It lists 22 stars and 4 forks, with an MIT license. The repository name is xbitinfo. A central graphic features the text "xbitinfo" over a grid of pink and white squares. The main heading is "xbitinfo: Retrieve information content and compress accordingly". It lists supported environments like Binder, JupyterLab, and Conda-Forge.

Schulz, Spring

Python, netCDF+GRIB

The screenshot shows the GitHub repository for field-compression by ecmwf-lab. The repository description is "ECMWF Field Compression Laboratory". It lists 1 star and 0 forks, with an Apache-2.0 license. The repository name is field-compression. The main heading is "Field Compression Laboratory". A table of contents lists: Overview, Prerequisites, Set up, How to use, Example notebooks, How to contribute, Development notes, Copyright and license, and References. The overview text states: "The Field Compression Laboratory aims to evaluate the impact of lossy compression on the accuracy of meteorological quantities used in numerical weather prediction. The current framework includes a Python library (fcpy) and example notebooks. Currently, we support latitude/longitude and Gaussian gridded data in netCDF and GRIB formats."

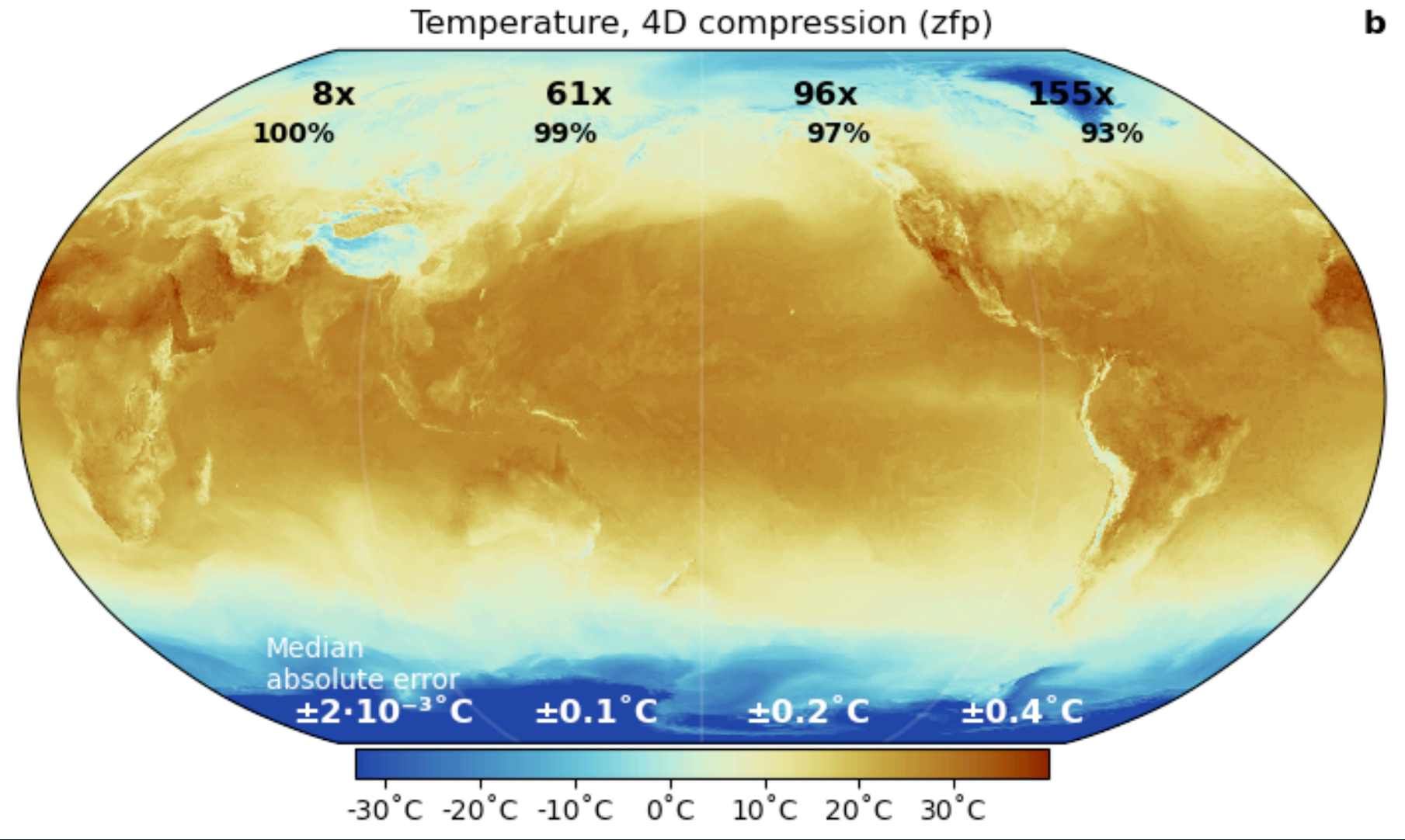
David Meyer

netCDF+HDF: NCO

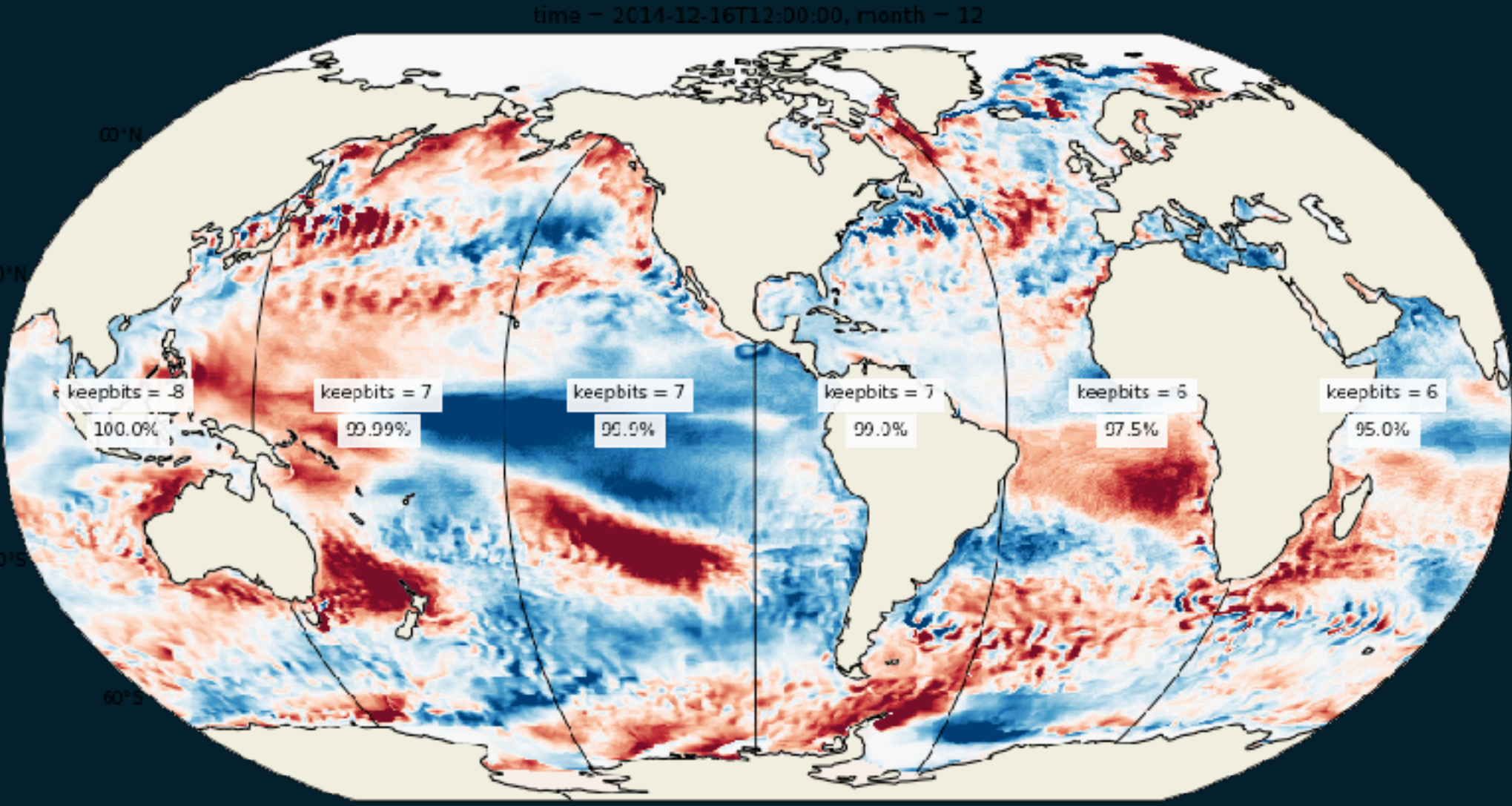
The screenshot shows the GitHub repository for NCO by nco. The repository description is "netCDF Operators". It lists 134 stars and 60 forks, with an Unknown license. The repository name is nco. The main heading is "NCO NetCDF Operators". A large graphic features the text "NCO = Σ" with a spiral and a summation symbol. Below it are logos for netCDF, CF, OPeNDAP, HDF, and GSL. The text "netCDF Operators (NCO) Software Stack" is displayed. A brief description states: "The NCO toolkit manipulates and analyzes data stored in netCDF-accessible formats, including DAP, HDF4, and HDF5. It exploits the geophysical expressivity of many CF (Climate & Forecast) metadata conventions, the flexible description of physical

Zender et al

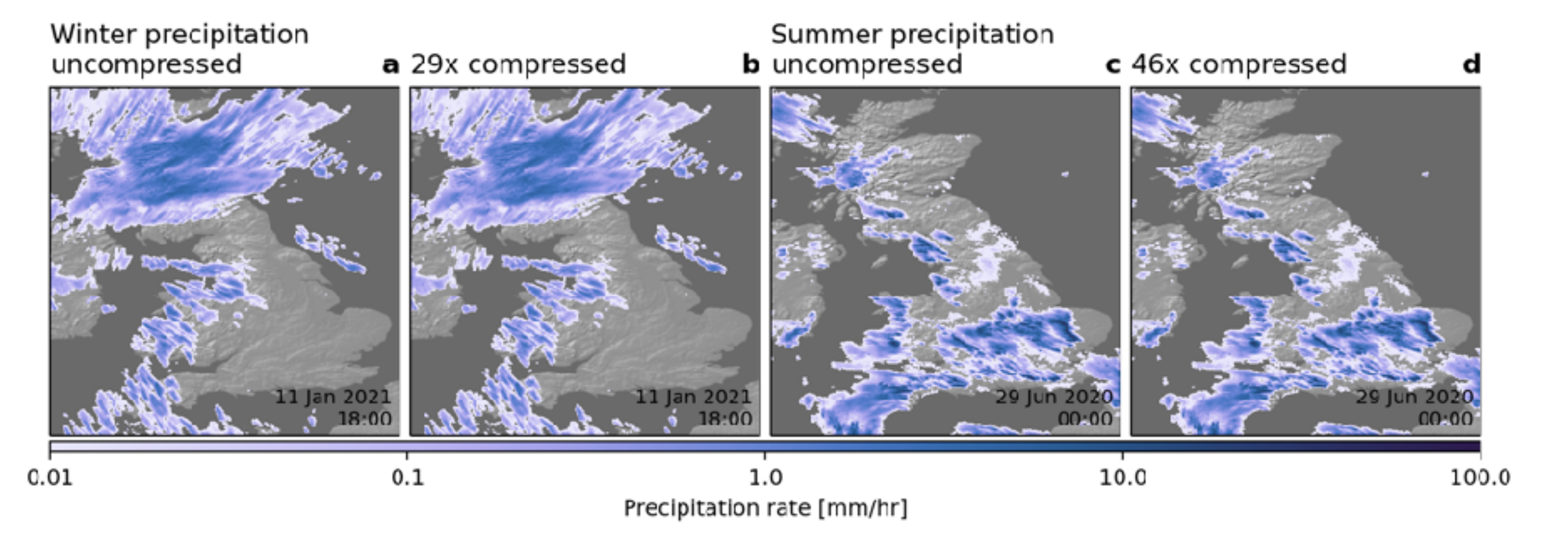
# Application examples



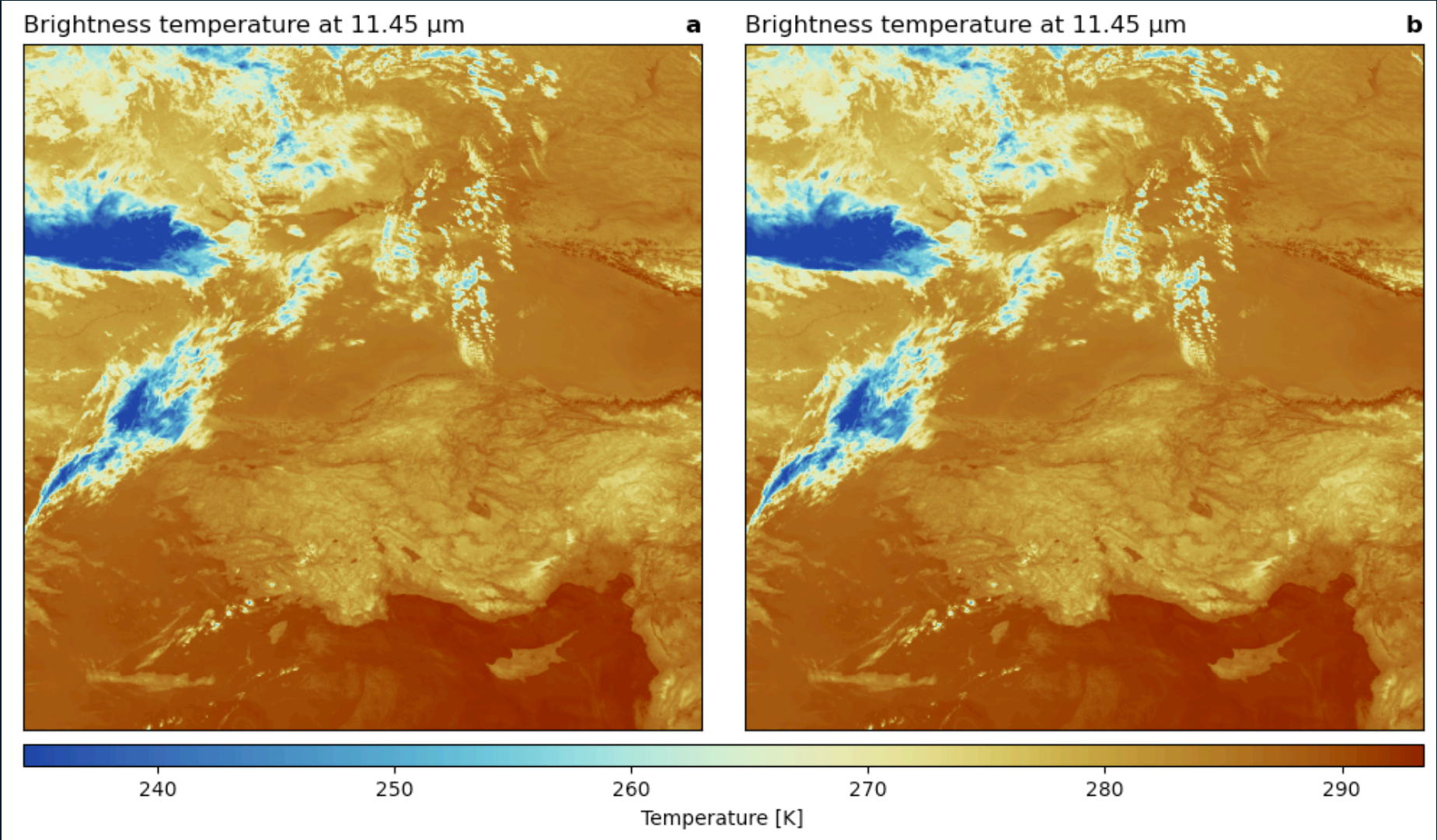
ECMWF's ensemble temperature forecast



CMPI6 sea surface temperatures and ICON-Ocean model output



Precipitation from radar data



Brightness temperature from satellites