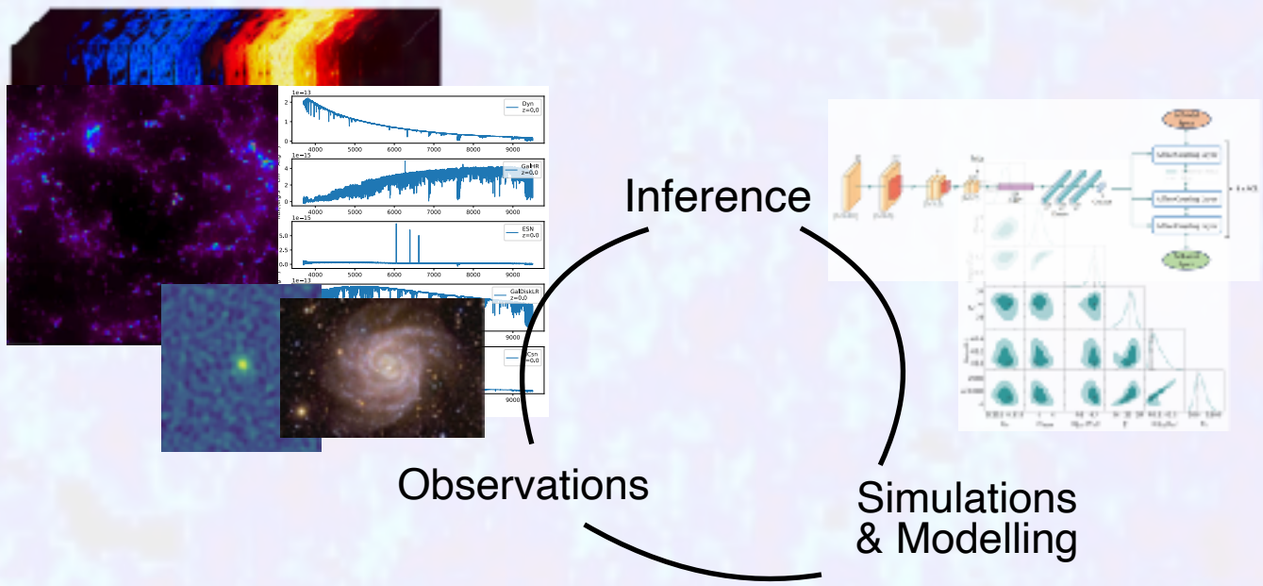
Machine Learning for Astrophysics

-

Tutorial

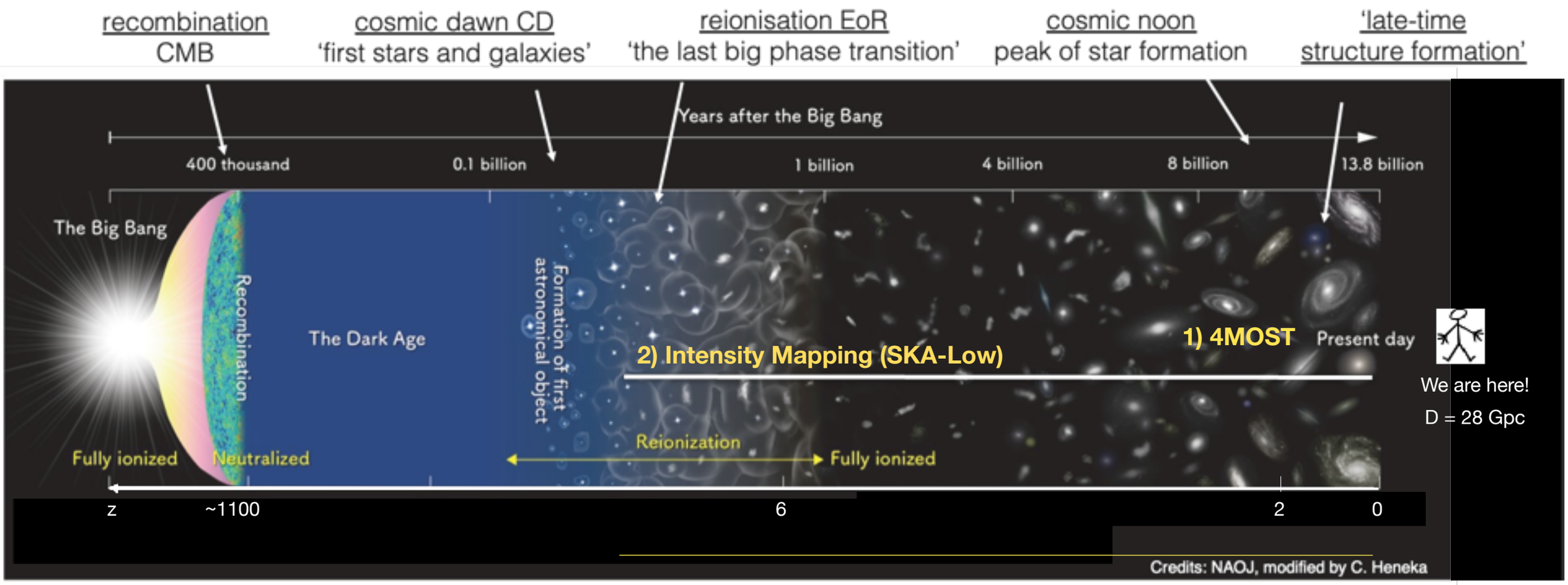


Caroline Heneka, ITP Heidelberg

Group ‘Computer Vision Astrophysics and Cosmology’

Advanced School on Applied Machine Learning, ICTP Trieste, May 28th 2024

Machine Learning for Astrophysics - Tutorial



Two routes for this tutorial:

1) **Low-z: Classification (4MOST)**
2) **High-z: Inference (SKA)**

1) Low-redshift (late time): Classification of 4MOST spectra

4MOST: On-the-fly classification of spectra (1D)

- 5-year survey
- wide-field, fibre-fed, optical spectroscopy
- on ESO's 4-m-class telescope VISTA
- 2.5-degree diameter field-of-view, 2436 fibres
- HRS R ≈ 18000 – 21000, LRS R ≈ 4000 – 7500
- 20mio. (LRS), 3mio. (HRS) sources

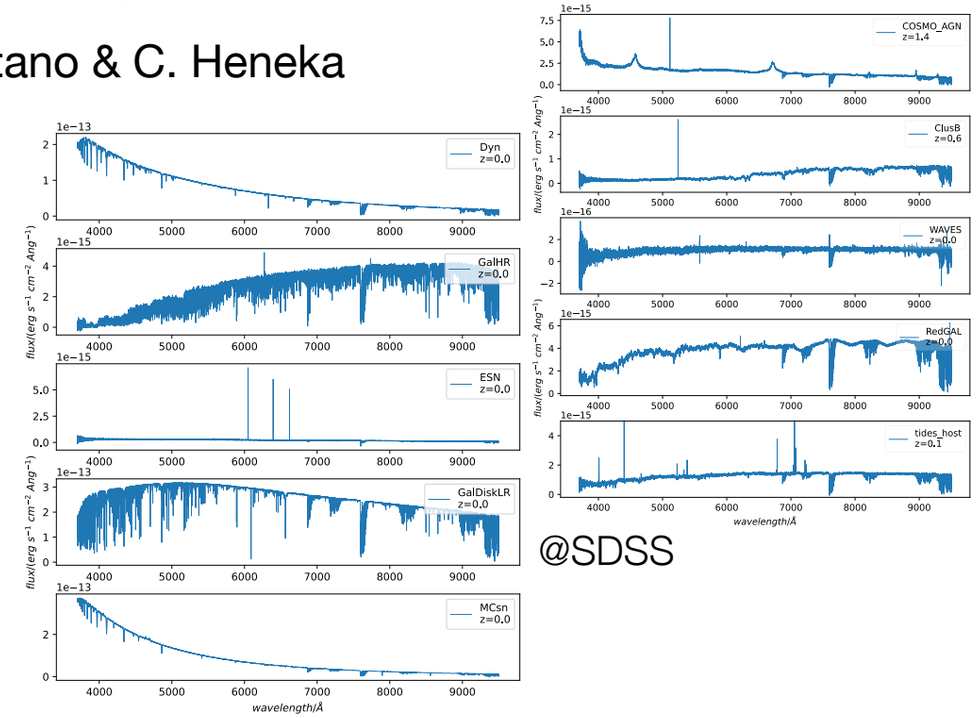
https://www.4most.eu Credit: ESO

<u>Goal:</u> Data-driven classification pipeline layer (galactic & extragalactic sources)

Classification infrastructure working group, led by: N. Napolitano & C. Heneka

→ *Benchmark with SDSS archival spectra:*

See now our tutorial!



@SDSS

1) Low-redshift (late time): Classification of 4MOST spectra

4MOST: On-the-fly classification of spectra (1D)

<u>Goal:</u> Data-driven classification pipeline layer (galactic & extragalactic sources)

Classification infrastructure working group, led by: N. Napolitano & C. Heneka

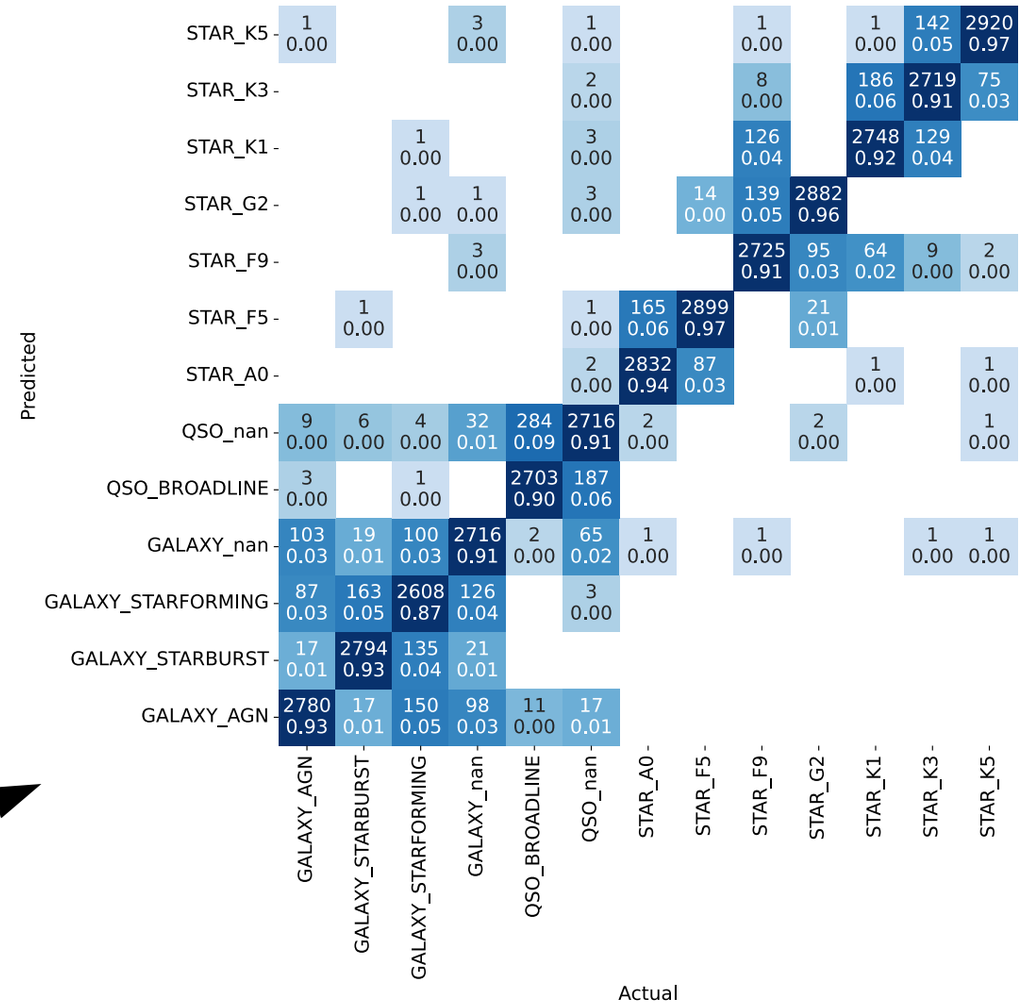
→ Probabilistic multi-classifier

<u>For class:</u>
Convolutional network variants

<u>For class uncertainties:</u>
Bayesian neural networks and contrastive learning

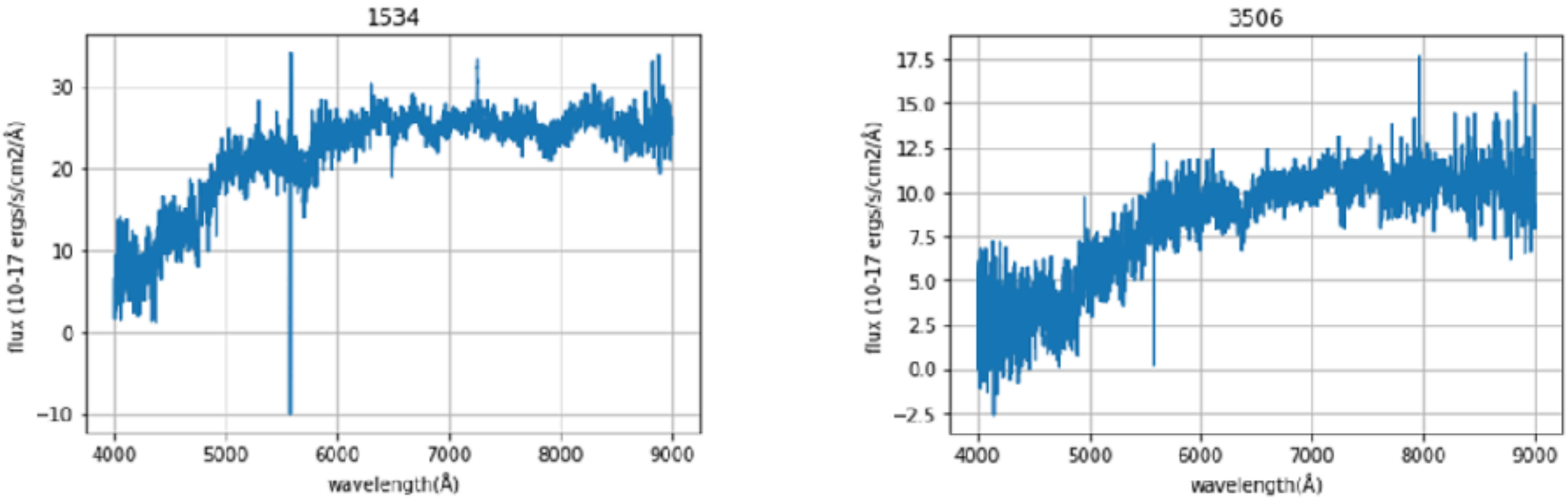
++ competitive with template fitting

Zhong, Napolitano, Heneka+ arXiv:2311.04146



This is what one wants to beat!

1) Tutorial classification challenge: 4MOST spectra



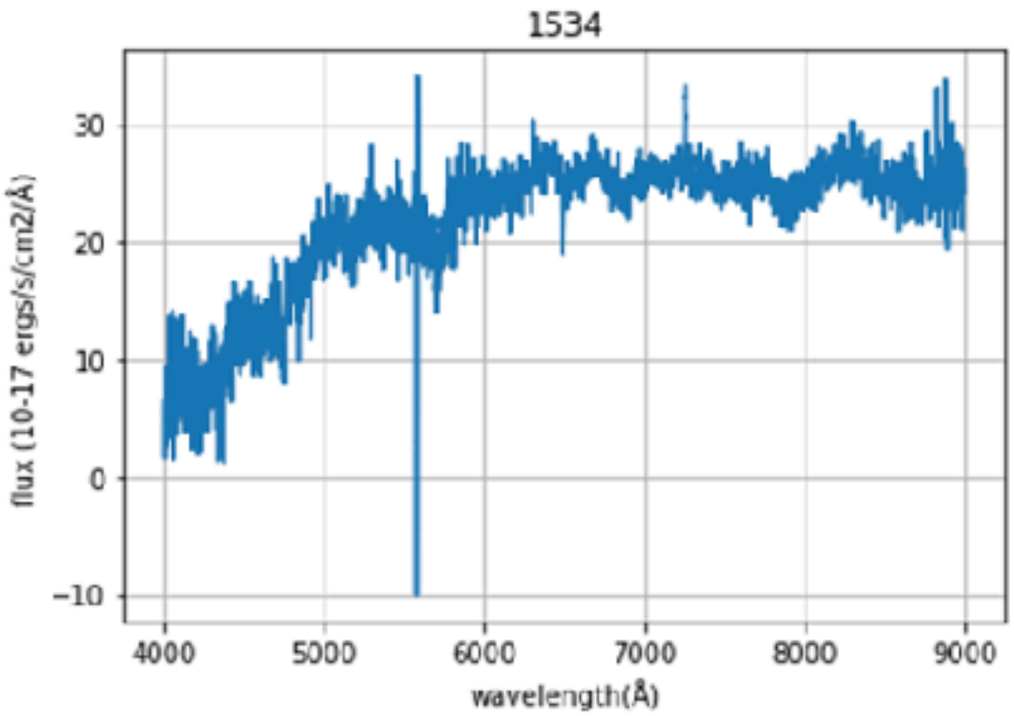
Class: Galaxy or Star? Other? A subtype?

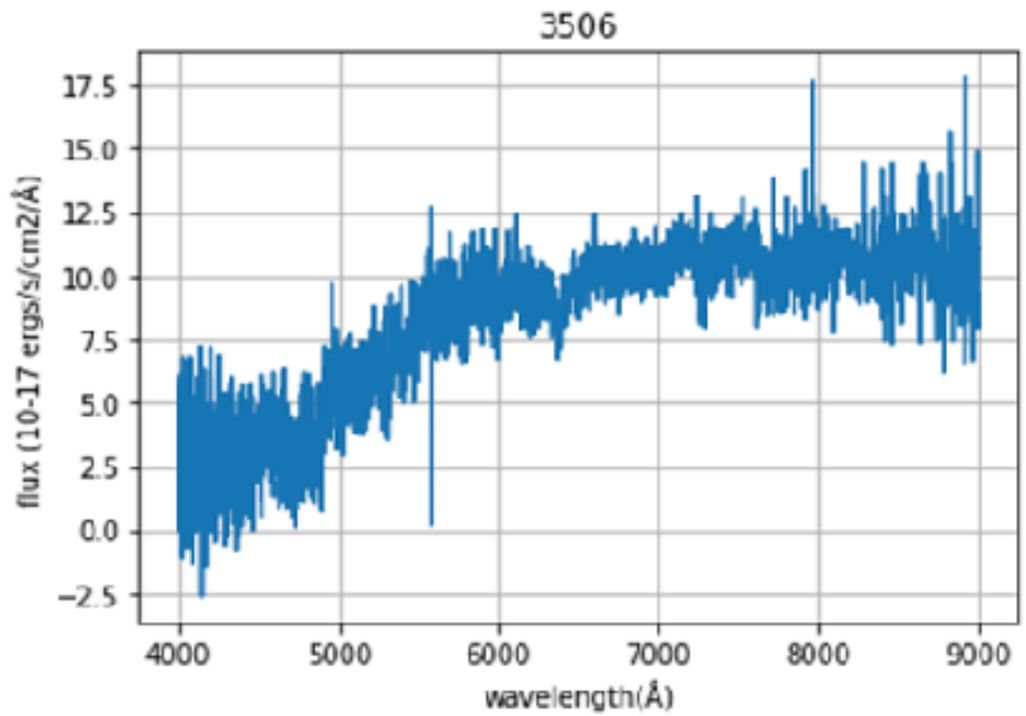
See tutorial part:

https://github.com/csheneka/ML-for-Astro-tutorial

spectral_classifier.ipynb

1) Tutorial classification challenge: 4MOST spectra



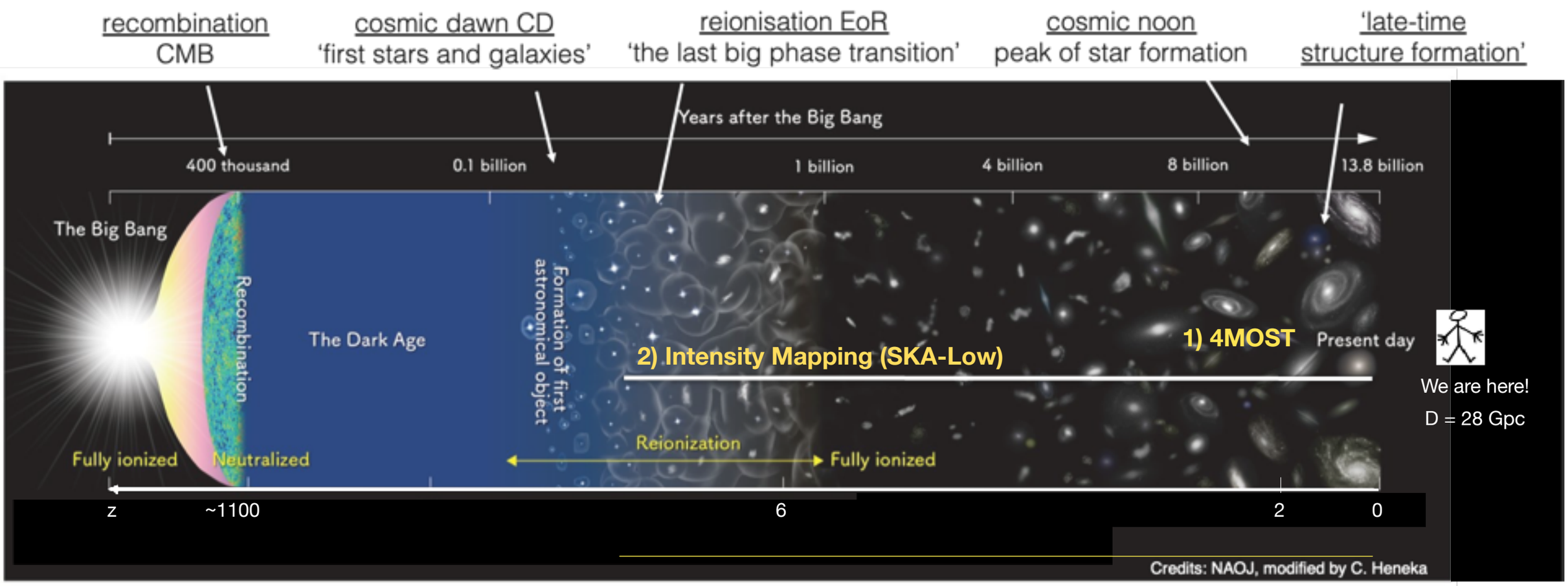


Class: Galaxy or Star? Other? A subtype?

More data (Credit: Fucheng Zhong):

https://huggingface.co/datasets/Fucheng/GaSNet-II-SDSS-dataset/tree/main/train_data

2) High-redshift: Regression for the Square Kilometre Array (SKA)



Two routes for this tutorial:

1) **Low-z: Classification (4MOST)**
2) **High-z: Regression (SKA)**

2) High-redshift: Regression for the Square Kilometre Array (SKA)

Why care?

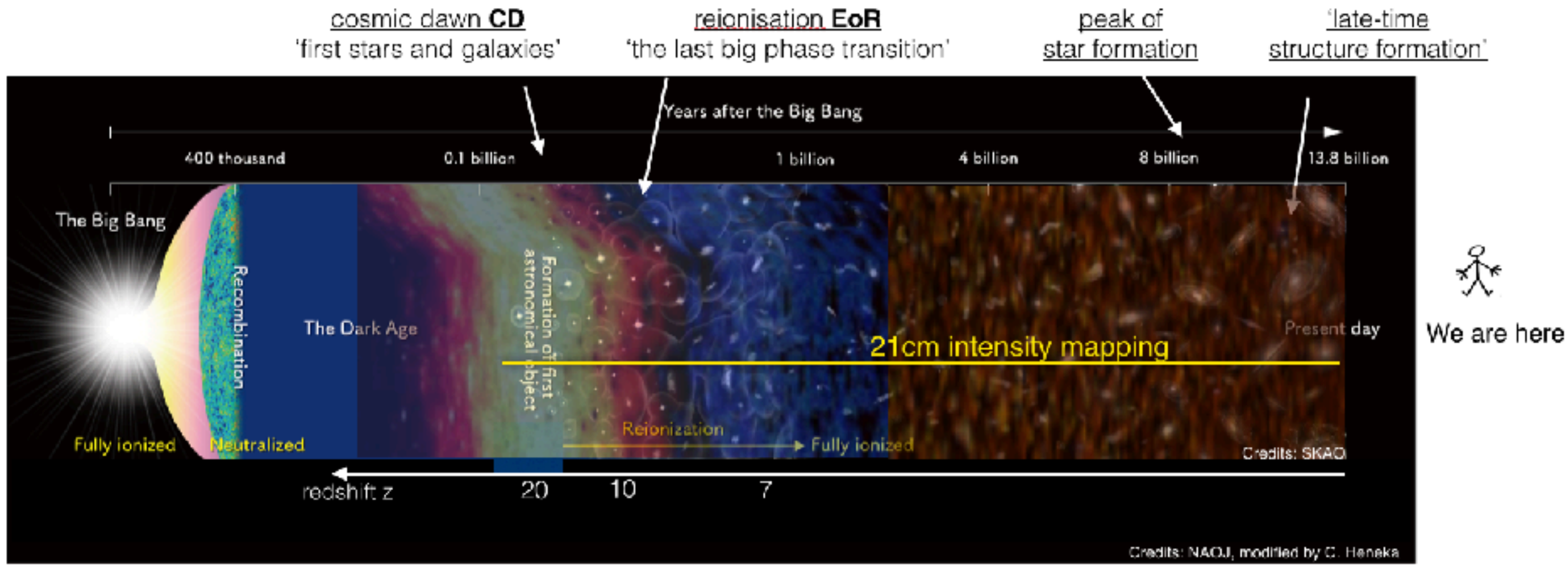
Tomography of >80% of the Universe

Square Kilometre Array - true ‘Big Data’

non-linear, non-Gaussian signal

Expected data SKA rate:
TB/s, few EB/day
Archive: ~700 PB/yr

→ Tutorial: Introspection on regression with networks



We are here

2) High-redshift: Regression for the Square Kilometre Array (SKA)

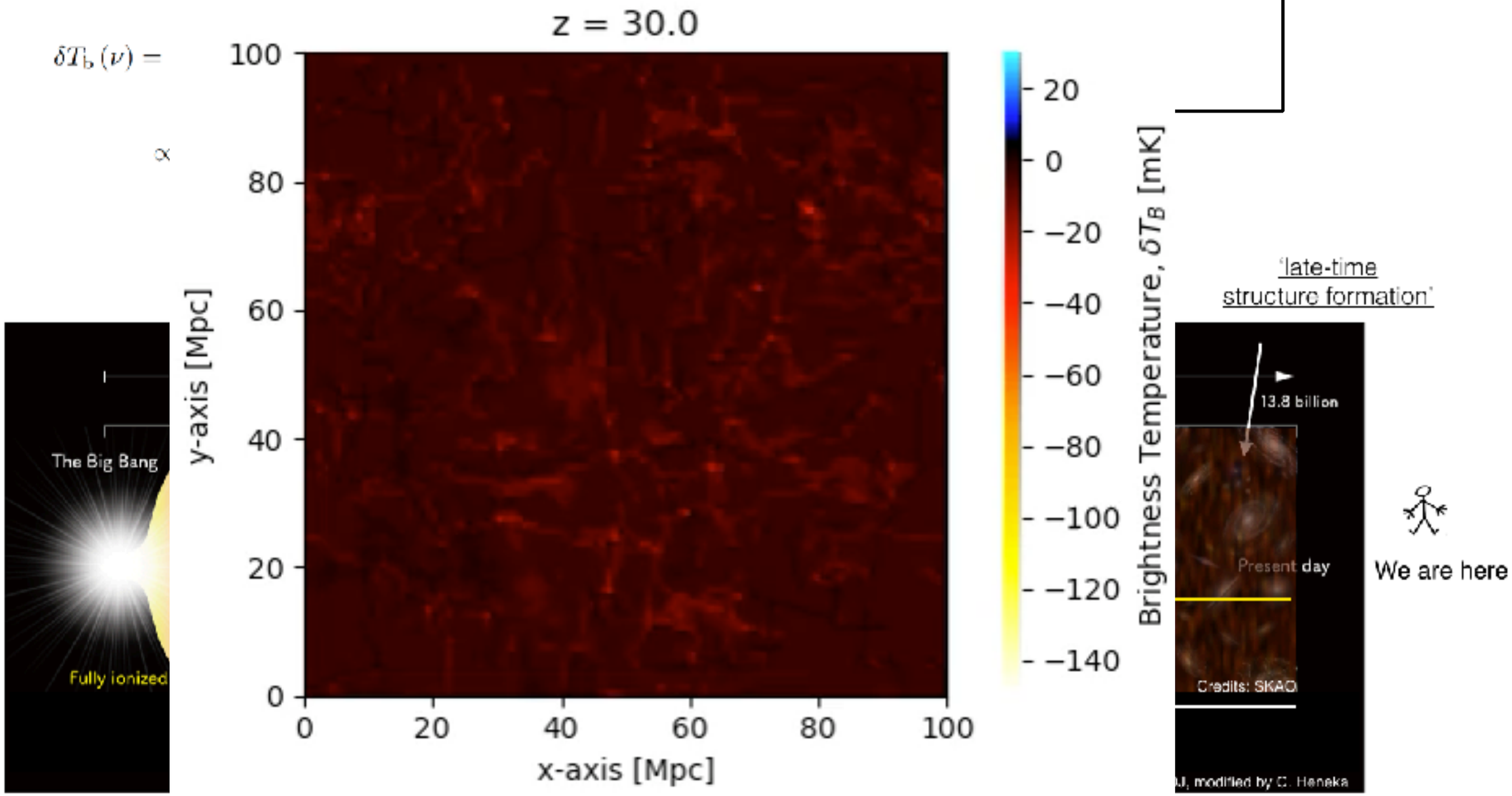
21cm signal

a tracer of neutral hydrogen:

Why care?

Tomography of >80% of the Universe

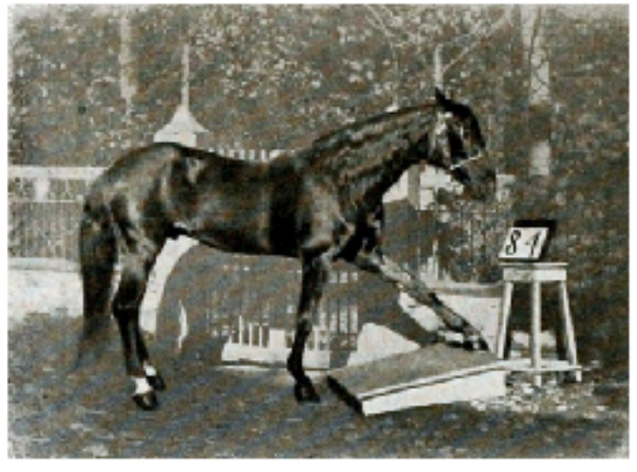
$$\delta T_b(\nu) = \propto$$



Network models are data driven.

Need introspection to:

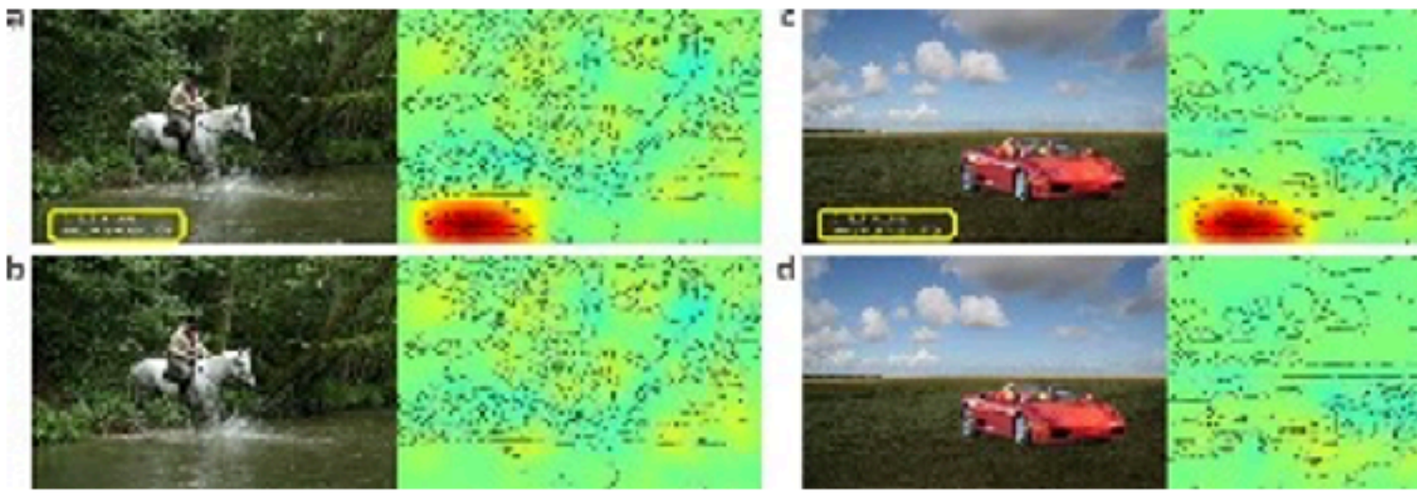
- Evaluate training **data**
- Debug, detect 'anomalies'
- Understand and verify your **model**
- Understand **predictions**



Hans am Tretbrett (1909)

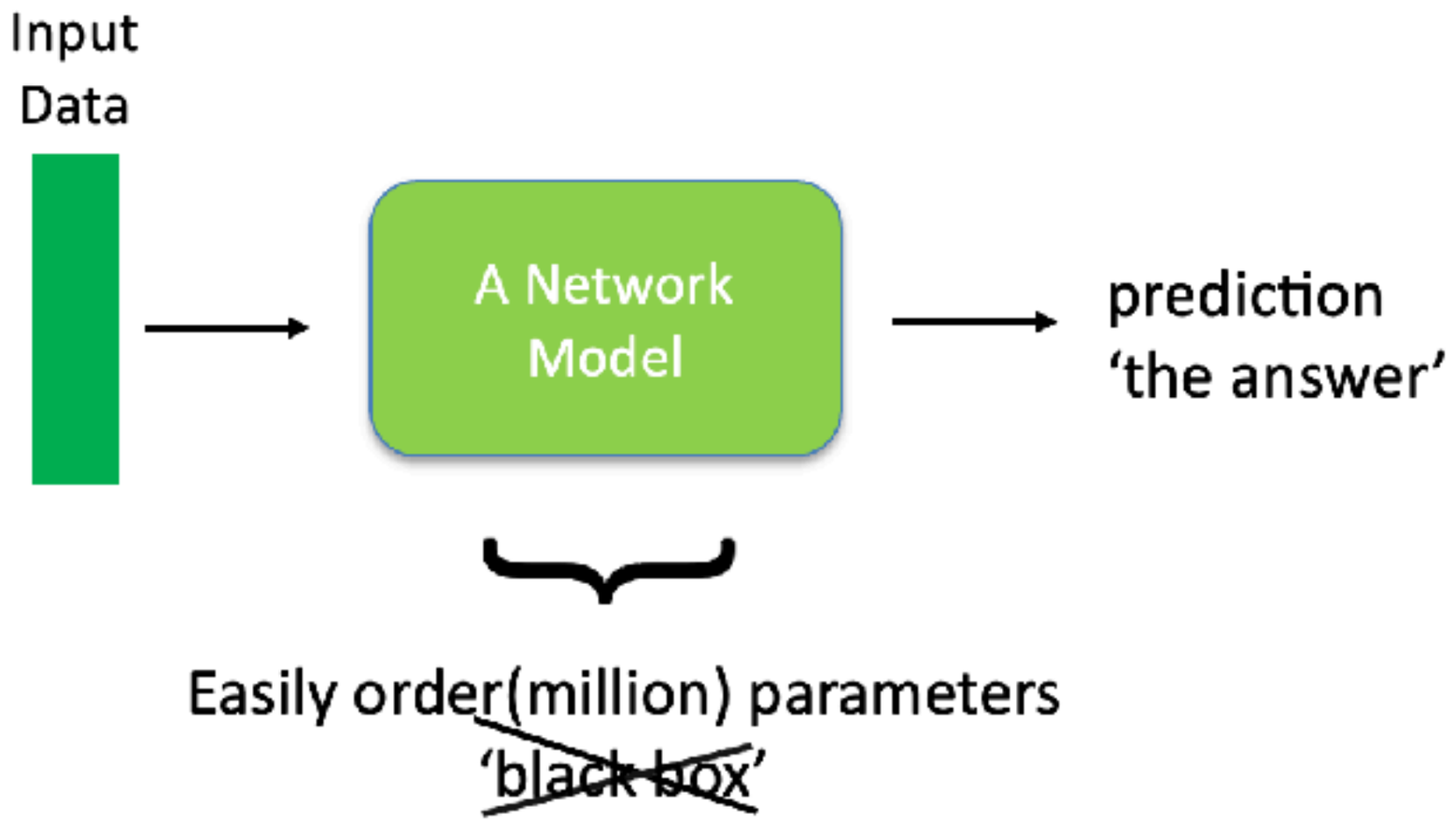
Famous examples:

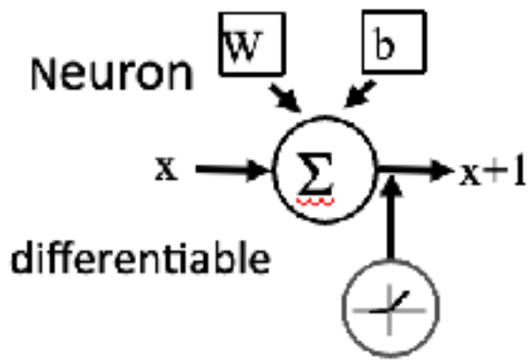
'Clever Hans' effect
(wrong cue, 'overfitting')
Over-confidence
(e.g. miss-classifying)



Lapuschkin et al. 2019, arXiv: 1902.10178

2) Brief Introspection basics for our tutorial

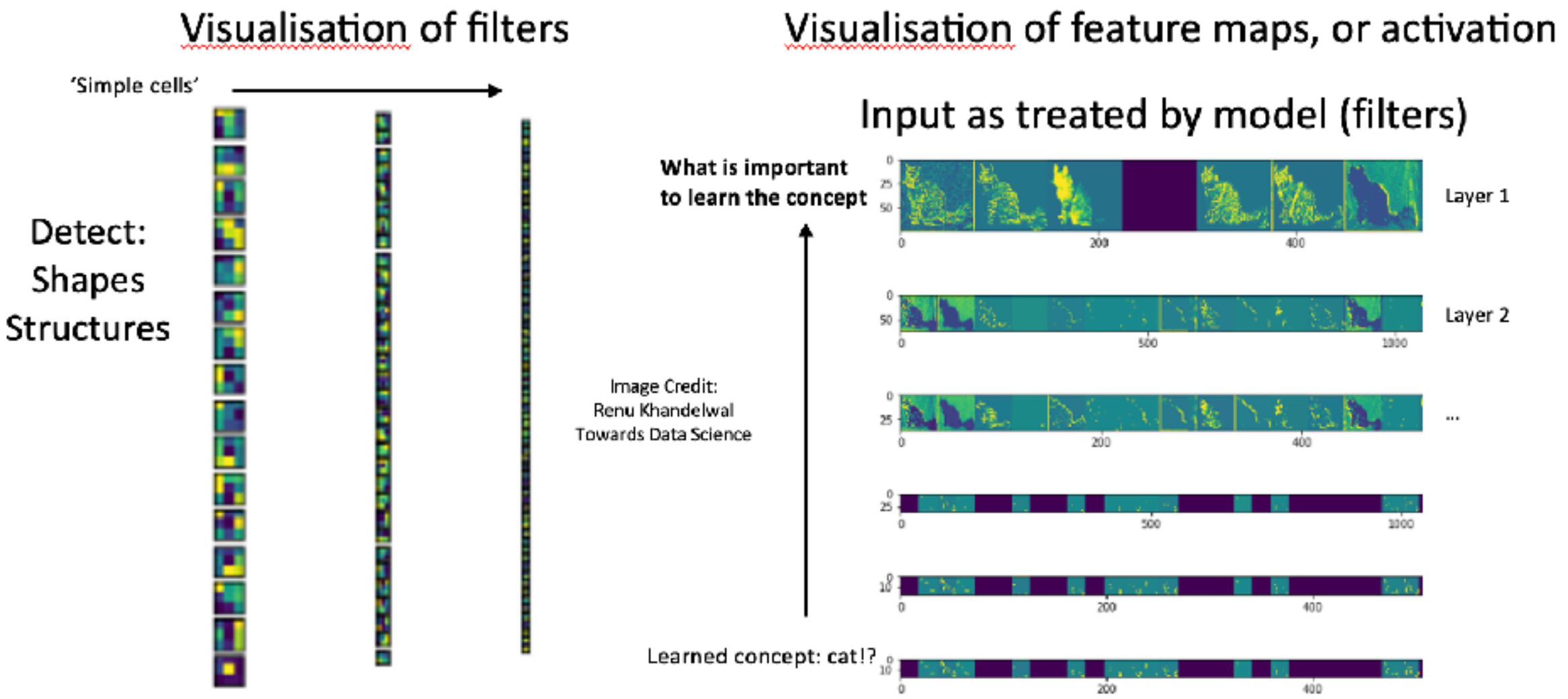




$$x_j^{(l+1)} = g\left(\sum_j w_{j,k}^{(l)} x_k^{(l)} + b_j^{(l)}\right)$$

We do know: architecture (model), best-fitting parameters (weights/bias/filter)
+ how these change during training + reaction to data ⟶ analyse

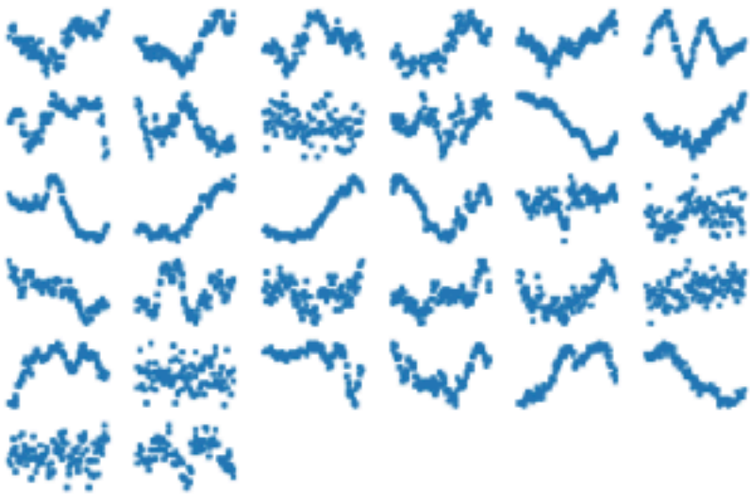
2) Tutorial part 1: Network visualisation



See tutorial part: https://github.com/csheneka/introspection-tutorial/blob/main/**visualisation_3D-21cmPIE-Net.ipynb**

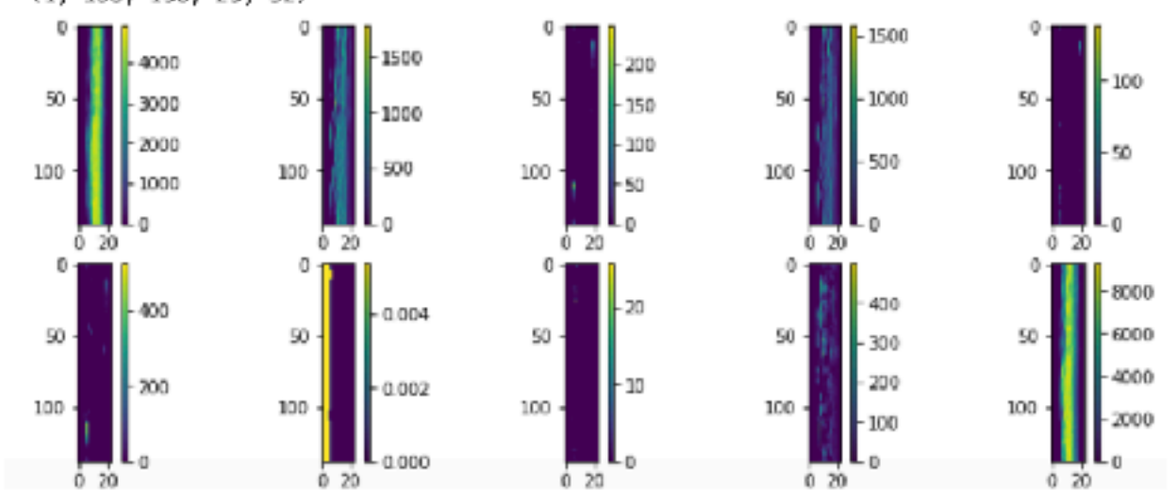
2) Tutorial part 1: Network visualisation

```
# Task 2: Spatially average and plot filters
nfilter = 32
nplots = nfilter
n_axis = 6
for i in range(nplots):
    weights = model.layers[1].weights[0].numpy()
    weights = weights.reshape(3,3,102,nfilter)[:,:,:,i]
    weights = np.mean(weights,axis=(0,1)) # spatial dims
    x = np.linspace(0,len(weights),len(weights))
    plt.subplot( n_axis, n_axis, 1 + i )
    plt.scatter(x,weights,s=4)
    plt.axis('off')
```



```
# plot the activation, or features
nslice = 1
nfilter = 32

lc_predict = data[0,0]
img_tensor = np.expand_dims(lc_predict, axis=0)

# get the activation for example after the first layer
activation_model = keras.Model(inputs=model.inputs, outputs=model.layers[1].output)
activation = activation_model(img_tensor)
print(activation.shape)

plt.figure(figsize=(15,15))
for i in range(32):
    plt.subplot(6,6,i+1)
    feature_map = activation[0,nslice,:,:,i]
    xy = plt.imshow(feature_map)
    plt.colorbar(xy,orientation="vertical",aspect=40)

# alternate approach:
# directly multiply and scan the data with the filters to see the effect of convoluti
```

```
(1, 138, 138, 23, 32)
```



See tutorial part: https://github.com/csheneka/introspection-tutorial

visualisation_3D-21cmPIE-Net.ipynb

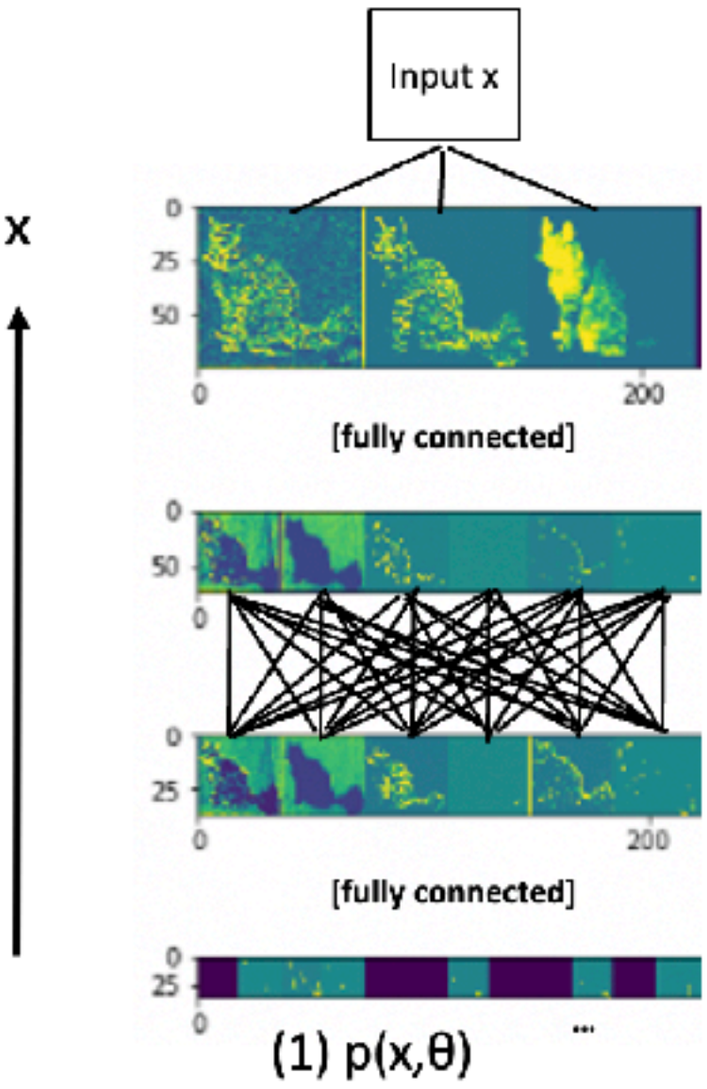
2) Tutorial part 2: Saliency maps

(1) For fixed model parameters θ, obtain for input space x the NN prediction p(x,θ)

(2) Calculate the gradient map (same shape as input):

$$S = \left| \frac{\partial p(x,\theta)}{\partial x} \right|_{x=x_0}$$

(3) Potentially: Same procedure for other class of p



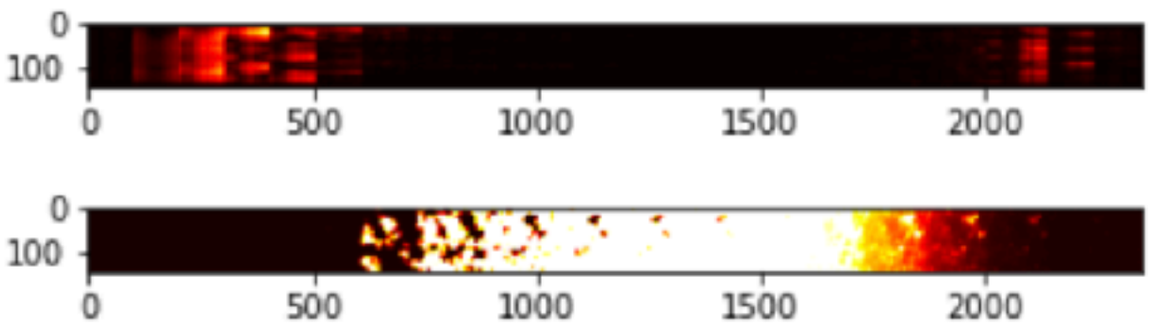
See tutorial part: https://github.com/csheneka/introspection-tutorial

attention_exercise.ipynb

2) Tutorial part 2: Saliency maps

```
# generate saliency maps for parameters i.e. classes
parameters=[0,1,2,3,4,5]
lc = data[0,0]
for para in parameters:
    def loss(output):
        return output[0][para] # shape (samples,classes)
    map_saliency = obj_saliency(loss,lc.reshape(140,140,2350,1))

print(map_saliency.shape)
nslice = 130
plt.figure()
plot_saliency = plt.imshow(map_saliency[0,nslice,:,:], cmap=cm.hot)
plt.figure()
plot_lc = plt.imshow(lc[nslice,:,:], cmap="EoR",vmin=-150,vmax=30)
```

```
(1, 140, 140, 2350)
```



Question: What do you notice, which areas spatially and temporally are important? How does salinecy shift with the inclusion of noise?

See tutorial part: https://github.com/csheneka/introspection-tutorial

attention_exercise.ipynb