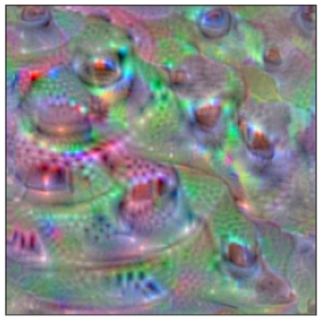
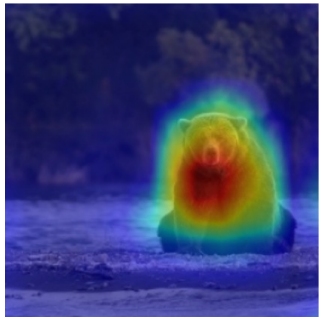
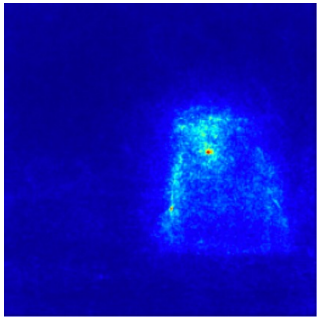
Introspection of Neural Networks

Deep Learning School ‚Basic Concepts‘

Credits: Yasuhiro Kubota, tf-keras-vis

Caroline Heneka, UHH
Haus Nordhelle, Meinerzhagen

Outline of this course

Block 1: Introduction & Visualisation

1) Overview of concepts
2) Feature Visualisation as a tool

Block 2: Attention & Reaction

1) Class Activation Maps (CAM)
2) Saliency
3) From sensitivity to attribtion

Block 3: Further concepts for introspection

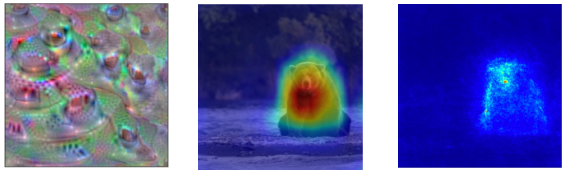
Wrap-up & summary

In total: ca. 1/3 lecture, 2/3 hands-on

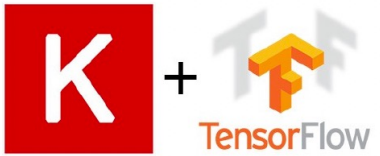
Our setup for the hands-on tutorials

The tutorials provide examples of neural network introspection written in Python, using the Keras* library and TensorFlow tensor ordering convention. We will also use additional toolkits such as tf-keras-vis.**

*Keras provides a high level API to create deep neural networks and train them using numerical tensor libraries (backends) such as TensorFlow, CNTK or Theano.



** https://keisen.github.io/tf-keras-vis-docs/index.html#



Credit: A. Boucaud

For our hands-on tutorial on

Introspection of neural networks:

You can start a binder image here:
https://mybinder.org/v2/gh/csheneka/introspection-tutorial/HEAD
Please start with: visualisation_3D-21cmPIE-Net.ipynb

OR via google Colab here: https://colab.research.google.com for path
https://github.com/csheneka/introspection-tutorial

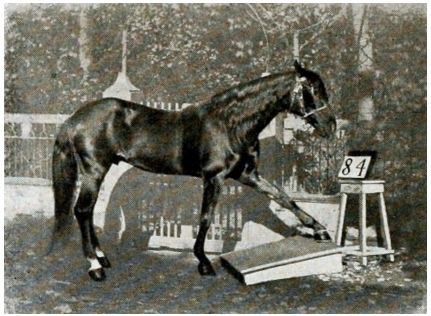
Extra data: https://cloud.hs.uni-hamburg.de/s/gYWW4PQG2b57XK3
(larger file version)

Part 1: Why introspection

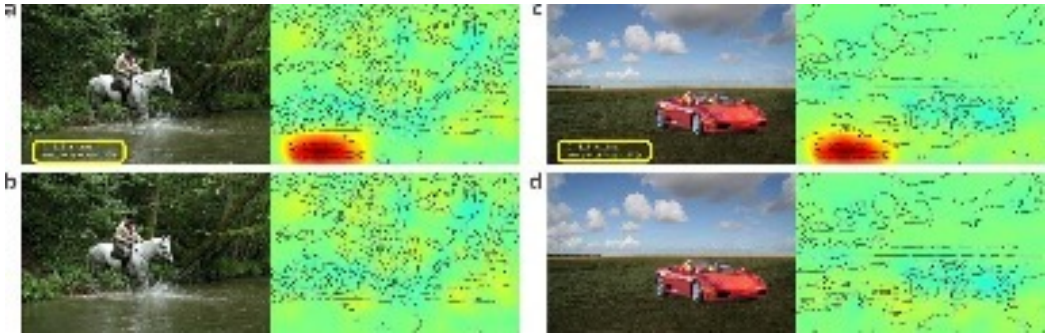
Network models are data driven.

Need introspection to:
- Evaluate training data
- Debug, detect ‘anomalies’
- Understand and verify your model
- Understand predictions

Famous examples:
‘Clever Hans’ effect
(wrong cue – ‘overfitting’)
Over-confidence
(e.g. miss-classifying)



Hans am Tretbrett (1909)



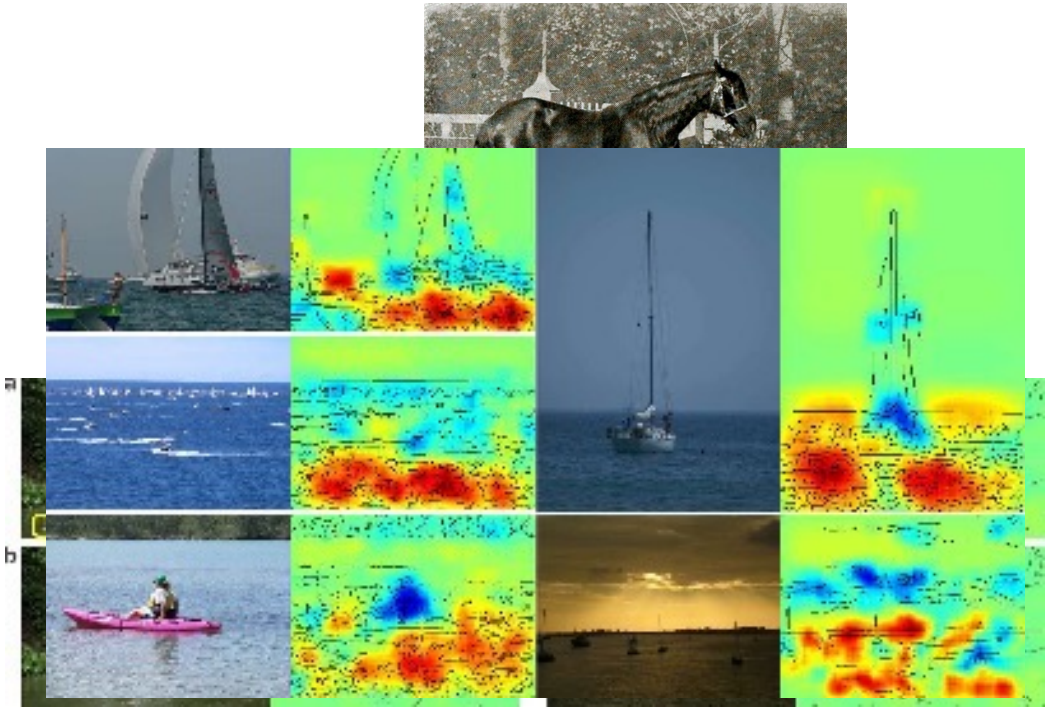
Lapuschkin et al. 2019, arXiv: 1902.10178

Part 1: Why introspection

Network models are data driven.

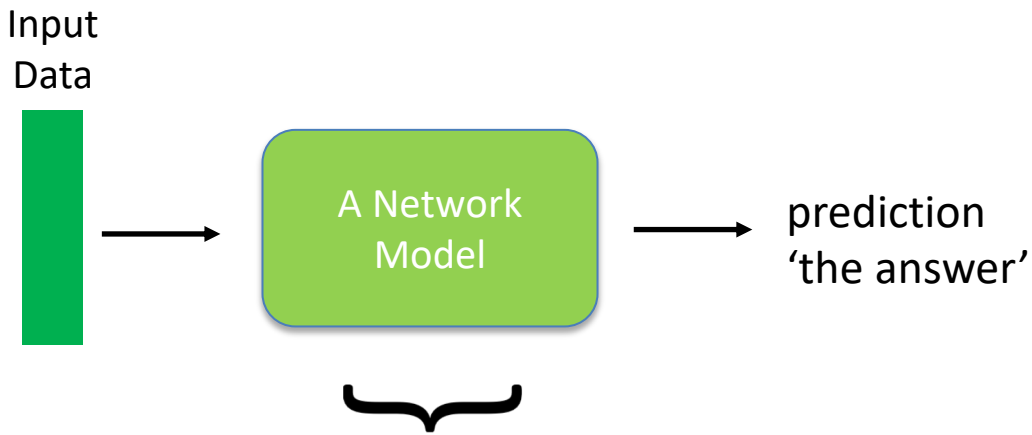
Need introspection to:
- Evaluate training data
- Debug, detect ‘anomalies’
- Understand and verify your model
- Understand predictions

Famous examples:
‘Clever Hans’ effect
(wrong cue, ‘overfitting’)
Over-confidence
(e.g. miss-classifying)



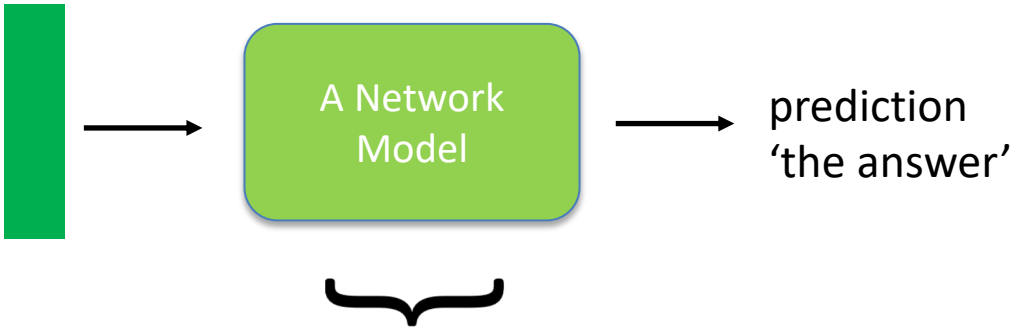
Lapuschkin et al. 2019, arXiv: 1902.10178

Part 1: Why introspection



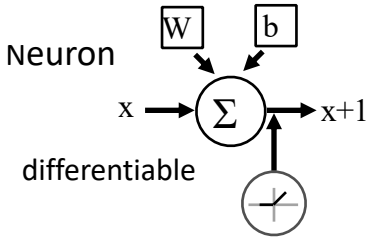
Part 1: Why introspection

Input Data



Easily order(million) parameters

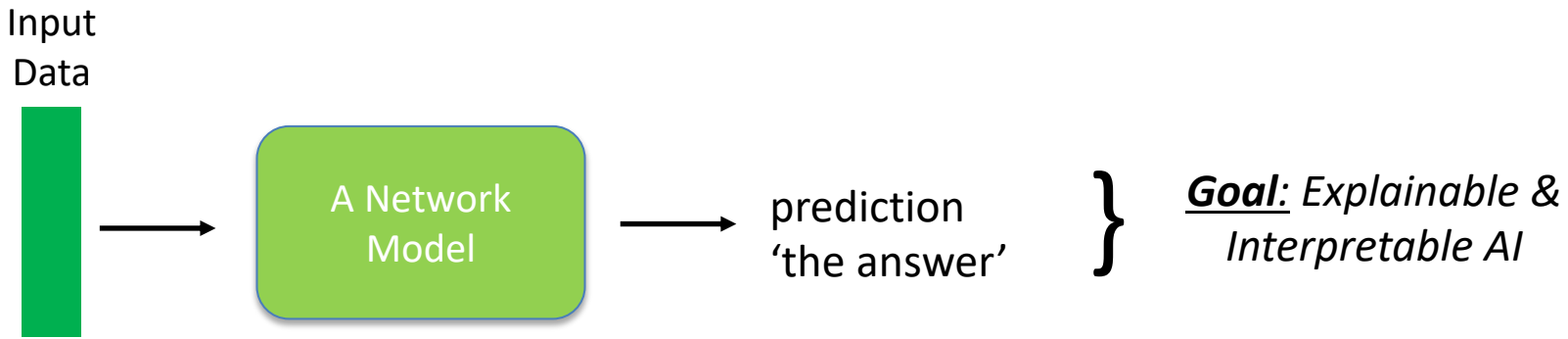
~~'black box'~~



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

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

Part 1: Why introspection



To learn about the model and its reaction to data, one can:

- Visualise feature space [Block 1]
- Analyse attention & reaction (CAM, saliency) [Block 2]
- Be model-agnostic (SHAP, clustering) [Block 3]
- Many more: Put errors/probabilities, regularise, unsupervised … [Block 3]
  … very active field of development!

Part 1: Why introspection

To learn about the model and its reaction to data, one can:

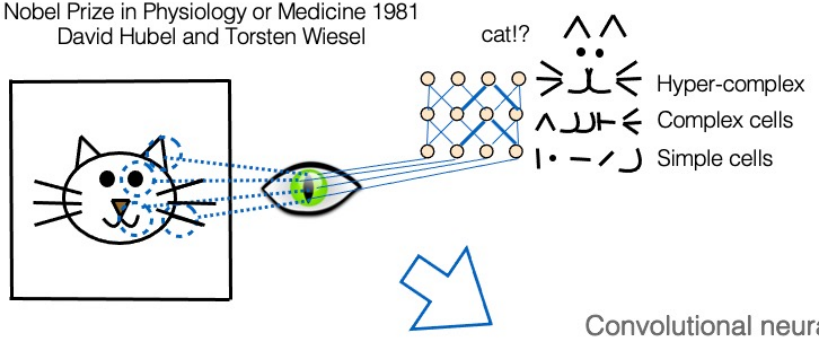
- **Visualise feature space** [Block 1]
- Analyse attention & reaction (CAM, saliency) [Block 2]
- Be model-agnostic (e.g. SHAP) [Block 3]
- Many more: Put errors/probabilities, regularise, unsupervised … [Block 3]
  … active field of development!

⟶ Visualise structure of weights (dense), filters (convolutional)
Example we will learn about with hands-on exercises:
Filter and activation of CNN, (activation maximisation)

Reminder: Convolutional Neural Networks

Neuron response to visual stimuli

Nobel Prize in Physiology or Medicine 1981
David Hubel and Torsten Wiesel

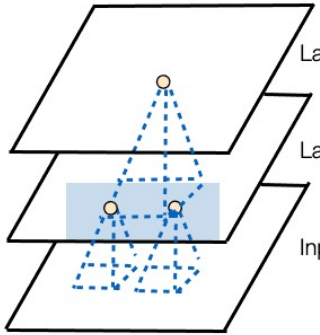


Representation learning

- Feature detection and assembly
- Local detection
- Overlapping fields

Convolutional neural network

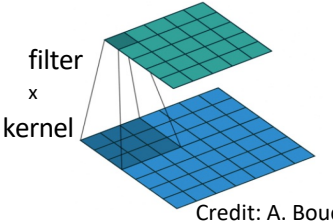
'Neocogniton' Kunihiko Fukushima 1979



Understand
High-level features
↑
Low-level features

Feature Visualisation

Understand
High-level features



Low-level features

How do weights/biases/filters look like?
How do networks ‘see’?
i.e. how do they learn representations?

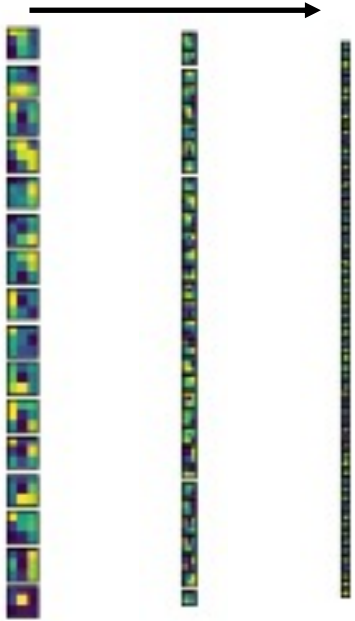
Interpretability:

Are shapes, that we expect to be important, there?

Feature Visualisation layer by layer

Visualisation of filters

Detect:
Shapes
Structures



Visualisation of feature maps, or activation

Input as treated by model (filters)

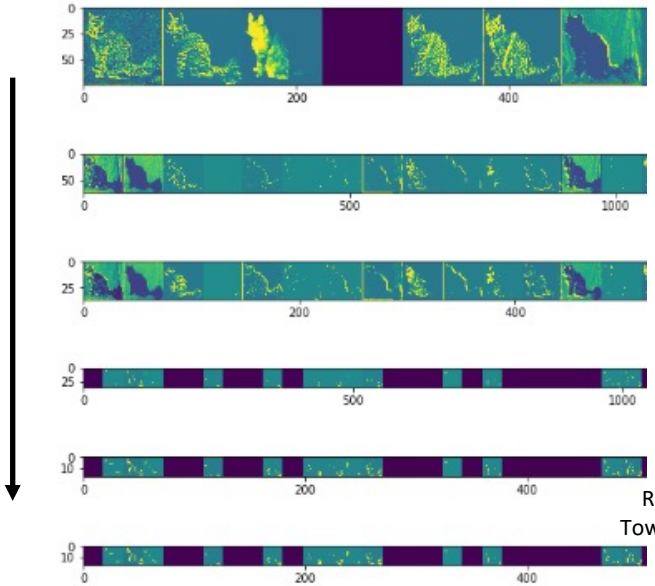
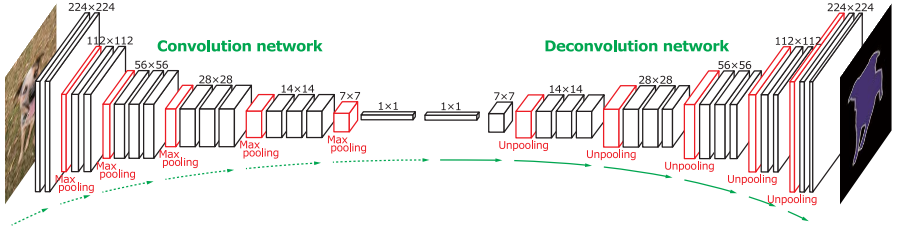


Image Credit:
Renu Khandelwal
Towards Data Science

Transposed Convolution: Feature ‘Deconvolution’

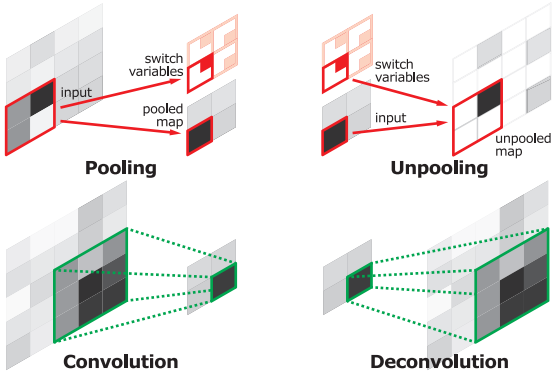
Useful for:

- Segmentation, localisation, super-resolution through upsampling
- Relating input patterns to feature maps
  See Zeiler et al., 2013



Goal: Map intermediate features to input shape

Transposed Convolution consists of unpooling to create a larger, but sparse, activation map, that then is via ‘fractional’ convolution connects for every input activation multiple outputs.



Noh, Hong, Han 2015
arXiv: 1505.04366

Transposed Convolution: Visualisation

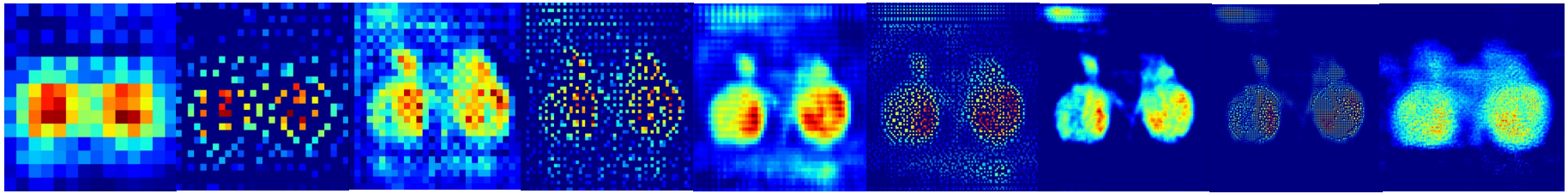
For visualisation we are interested in mapping what caused the highest activation in feature space:

- Model: transposed trained model (weights, filters)
- Upsample only highest activation
- Up to input space dimension

+ Further examples arXiv:1804.11191

Noh, Hong, Han 2015
arXiv: 1505.04366

Suppression of background, specific representation, complexity



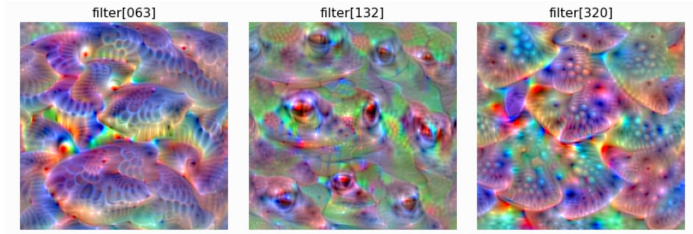
Activation Maximisation

Useful to visualise both dense and convolutional networks parts.

An input pattern image x* is synthesized that maximises the activation of a feature, i.e. that finds preferred input for a neuron, filter, or layer (deep dream):

$$x^* = \underset{x}{\operatorname{argmax}}\, a_{i,l}(\theta, x)$$

for a pretrained model with parameters θ, and input x.



https://keisen.github.io/tf-keras-vis-docs/

Steps:

(1) Start with input noise image x0

(2) Calculate activation gradients via backpropagation at fixed θ

(3) Iteratively maximise the activation, until x*

$$x \leftarrow x + \eta \cdot \frac{\partial a_{i,l}(\theta, x)}{\partial x}$$

Summary Block 1: Introduction & Visualisation

- ✓ Introspection important to verify and debug your model
- ✓ Deep Learning is data-driven: need to understand data representation
- ✓ Goal is interpretable / explainable machine learning

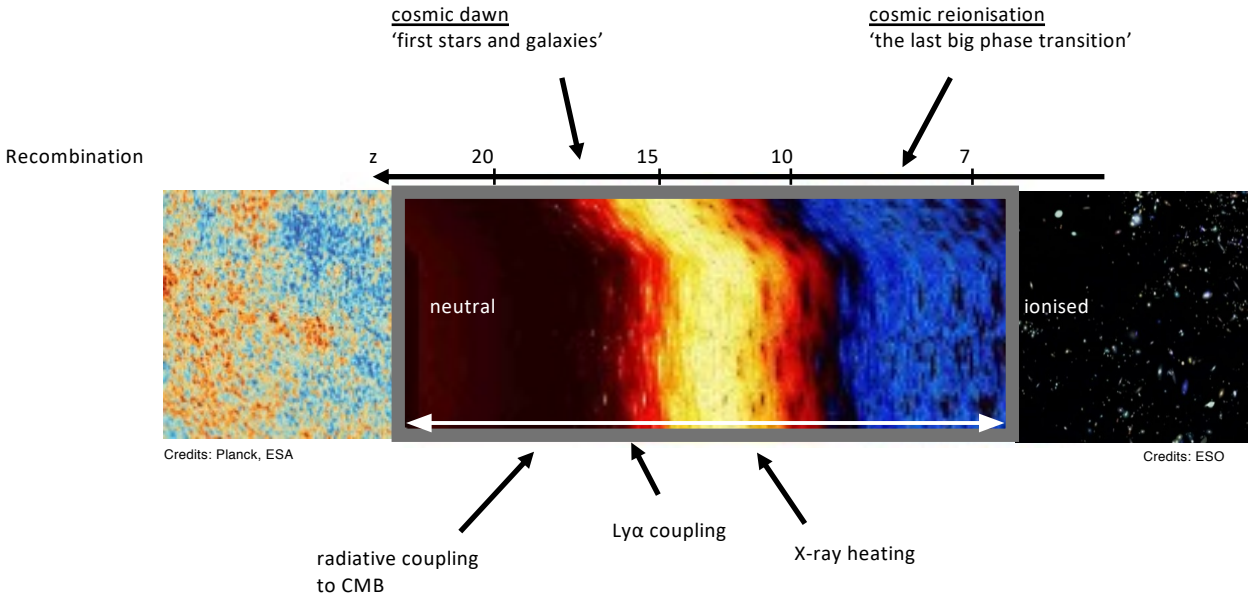
- ✓ Introspection is possible via visualisation of trained properties
  - Visualisation of filter and feature maps
  - Transposed Convolution upsamples and identifies ‘important’ input shapes
  - Activation Maximisation finds shapes that a specific neuron/filter/layer is sensitive to
  - and more: network inversion, network dissection, pruning, …

→ We can track feature hierarchy, representation learning in a probabilistic manner.

Hands-on background: Visualisation of an ‘Astrophysical’ CNN

What is the cosmology at high redshifts? ‘gap’ between CMB and galaxy surveys

What properties do the very first stars and galaxies have?



21-cm lightcone*
generated with 21cmFAST

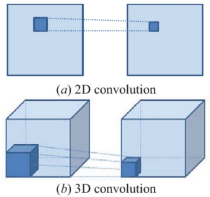
*lightcone = 2D spatial +
1D frequency information

blue = emission
red / yellow = absorption
against the CMB

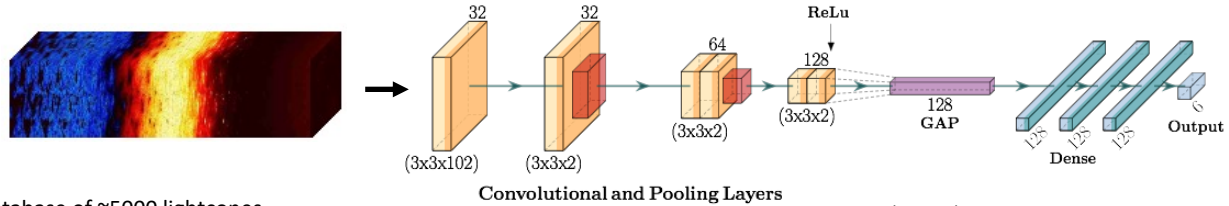
Hands-on background: Visualisation of an 'Astrophysical' CNN

Task to infer astrophysical and cosmological key parameters directly from intensity lightcones $(\Omega_\mathrm{m},\, \zeta,\, T_\mathrm{vir},\, L_\mathrm{X},\, E_0,\, m_\mathrm{WDM})$

Moving from **2D to full 3D convolution**



Best-performing: simple Conv3D architecture



Database of ~5000 lightcones
140x140x2350 pix, 1.4 Mpc resolution
Training (K80 GPU): ~20min/epoch, ~30 epochs

Neutsch, Heneka, Brüggen 2022, arXiv: 2201.07587

Hands-on: Visualisation of an ‘Astrophysical’ CNN

For our hands-on tutorial on

Introspection of neural networks:

You can start a binder image here:
https://mybinder.org/v2/gh/csheneka/introspection-tutorial/HEAD
Please start with: visualisation_3D-21cmPIE-Net.ipynb

OR via google Colab here: https://colab.research.google.com for path
https://github.com/csheneka/introspection-tutorial

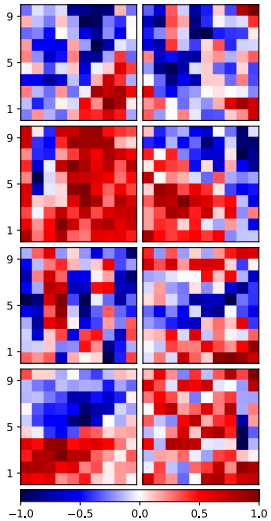
Extra data: https://cloud.hs.uni-hamburg.de/s/gYWW4PQG2b57XK3
(larger file version)

Visualisation of an ‘Astrophysical’ CNN

Hands-on: Visualisation of filters layer by layer

A) Raw filters (spatial dim):
example from Gillet et al. 2019, arXiv:1805.02699

Question: Could we do something with the filters, knowing the variation in 2D (spatial) is very different from the +1D temporal dimension?

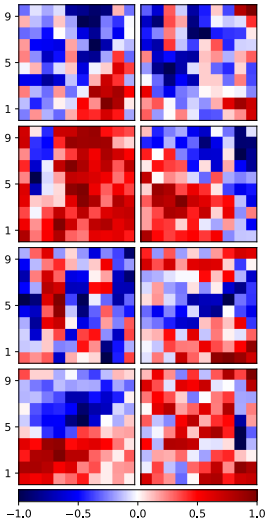


Visualisation of an ‘Astrophysical’ CNN

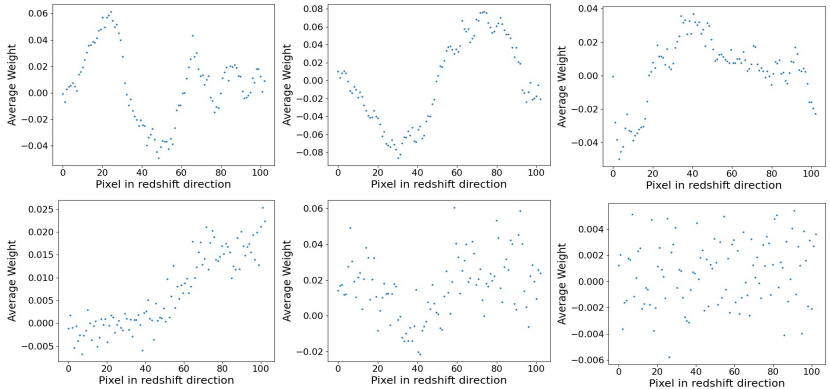
Hands-on: Visualisation of filters layer by layer

A) Raw filters (spatial dim):
example from Gillet et al. 2019, arXiv:1805.02699

Question: Could we do something with the filters, knowing the variation in 2D (spatial) is very different from the +1D temporal dimension?



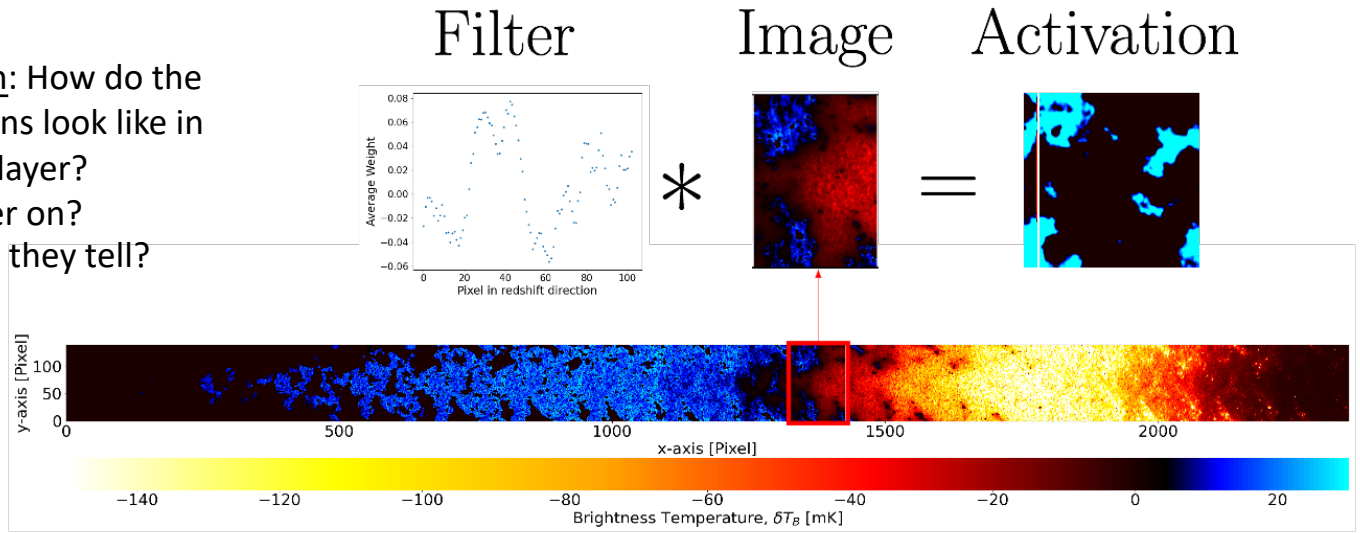
B) Average in redshift (temporal) direction:
Note the filters for ‘slow transitions’ and fluctuations (example our code)



Visualisation of an ‘Astrophysical’ CNN

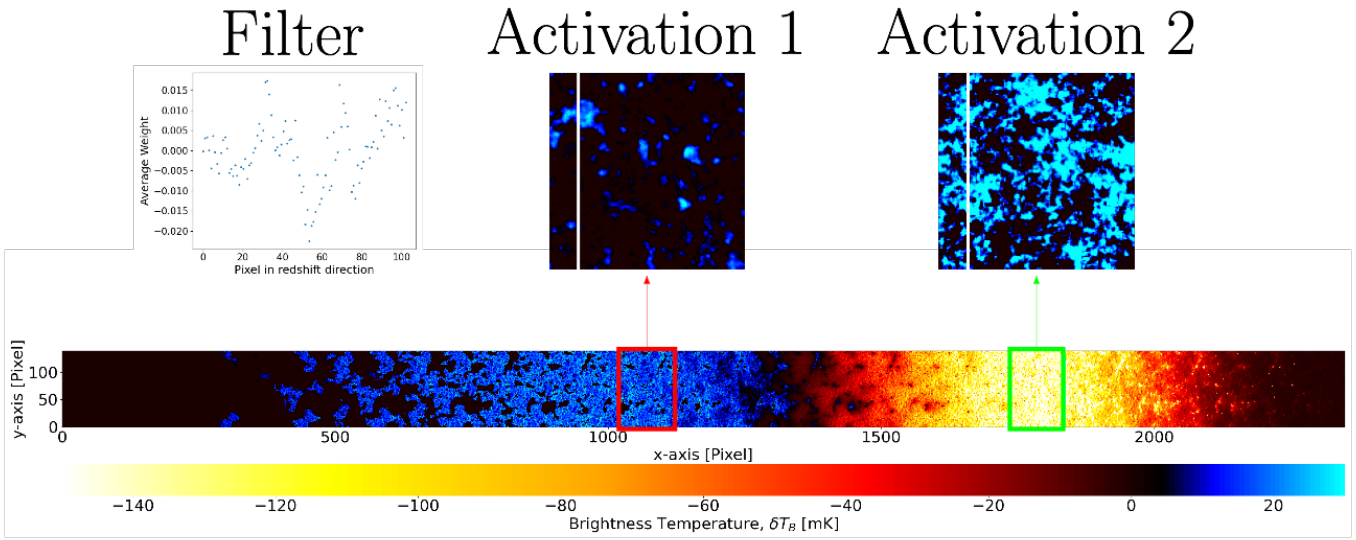
Hands-on: Visualisation of feature maps, or activation, layer by layer

Question: How do the activations look like in the first layer?
How later on?
What do they tell?



Visualisation of an ‘Astrophysical’ CNN

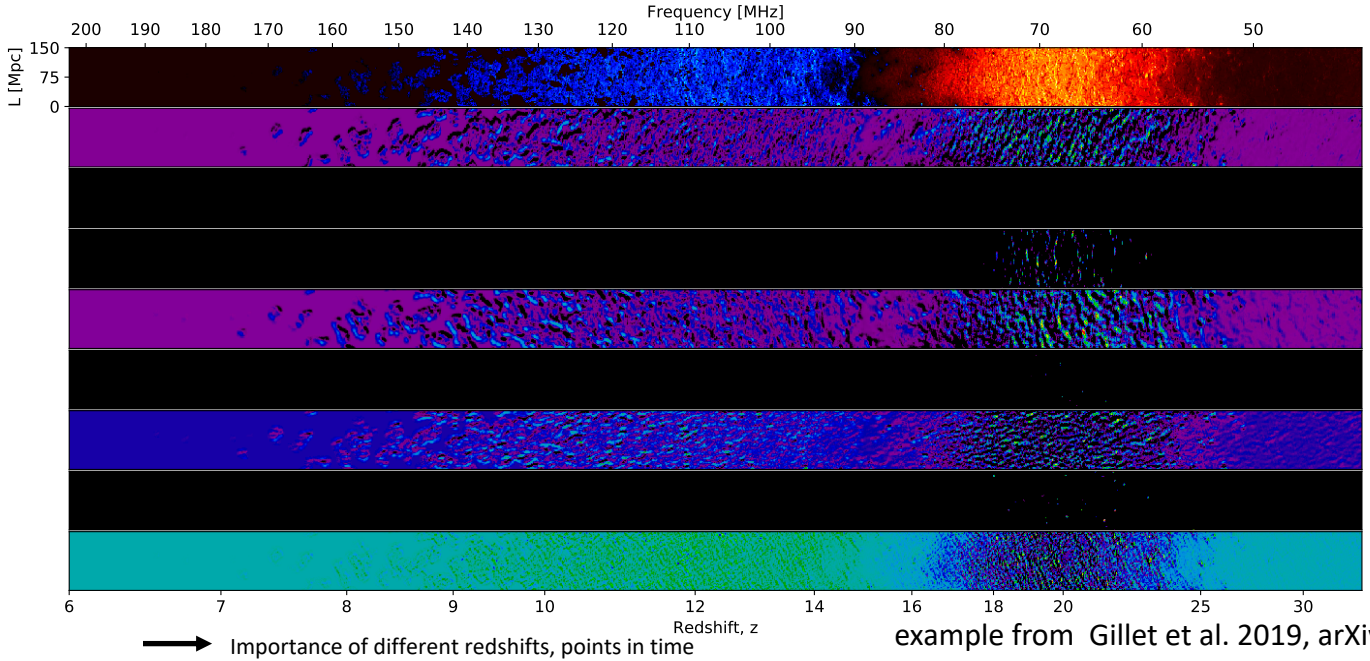
Hands-on: Visualisation of feature maps, or activation, layer by layer



reflects typical fluctuation scales

Visualisation of an ‘Astrophysical’ CNN

Hands-on: Visualisation of feature maps, or activation, layer by layer



Importance of different redshifts, points in time

example from Gillet et al. 2019, arXiv:1805.02699

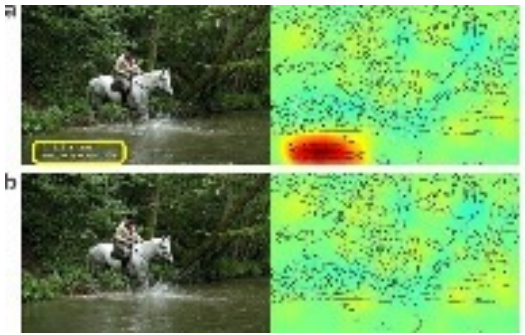
Part 2: Attention & Reaction

To learn about the model and its reaction to data, one can:

- Visualise feature space [Block 1]
- **Analyse attention & reaction** (CAM, saliency) [Block 2]
- Be Model-agnostic (e.g. SHAP) [Block 3]
- Many more: Put errors/probabilities, regularise, unsupervised … [Block 3]
  … active field of development!

→ Analyse the attention over input for the prediction
i.e. sensitivity or attribution of the prediction to input

Example we will learn about with hands-on exercises:
CAM, saliency map



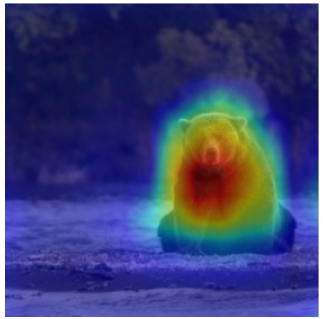
Lapuschkin et al. 2019, arXiv: 1902.10178

Discriminative localisation with CAM

CAM = Class Activation Map

Specific architecture:
GAP* output of the last convolutional layer before dense layers.

* Global Average Pooling

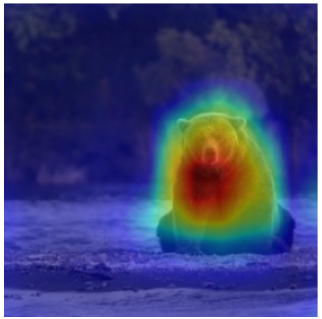


Credits: Yasuhiro Kubota, tf-keras-vis

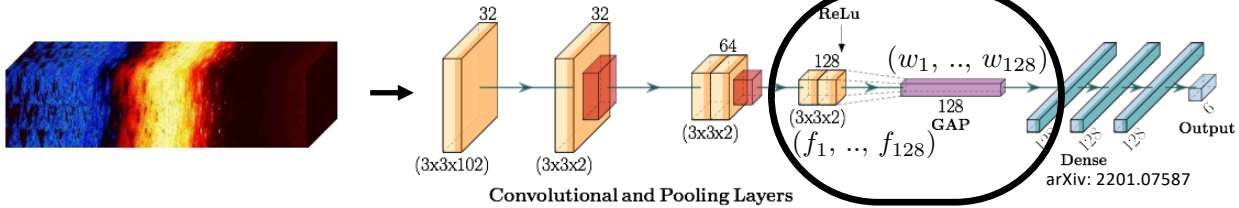
CAM = (Weighted) sum of GAP weights and activations of last CNN layer,
upsampled via interpolation to input data shape

Discriminative localisation with CAM

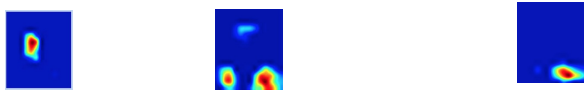
CAM = (Weighted) sum of GAP weights and activations of last CNN layer, upsampled via interpolation to input data shape



Credits: Yasuhiro Kubota, tf-keras-vis

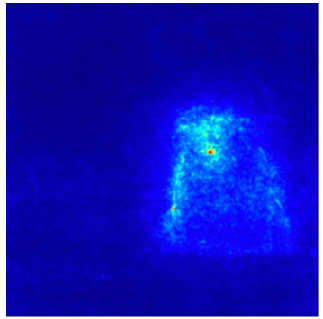


$$w_1 \times f_1 + w_2 \times f_2 + ... + w_N \times f_N$$ + up-sample = CAM



Saliency Maps

Gradient-based approach to localisation
“Most important input features (pixels) cause the highest gradient”
Flexible wrt architecture
Map ranks importance



Credits: Yasuhiro Kubota, tf-keras-vis

Principle: Backpropagation to input (training uses backpropagation to 1st layer)

(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):

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

From sensitivity to attribution

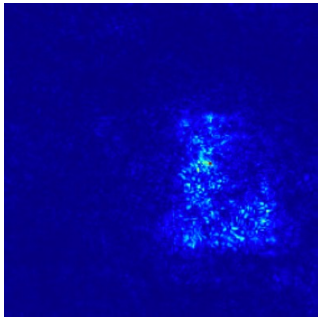
Improvements of sensitivity,
e.g. to reduce noise and achieve better localisation:

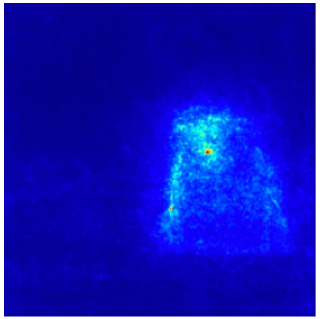
- SmoothGrad: add noise to input for ensemble of maps, average map of attention

- CAM: different algorithms for calculation of class activation (GradCAM vs. ScoreCAM, ++)

Integrated Gradients (Sundararajan, Taly, Yan 2017 arXiv:1703.01365)

Attribution methods should satisfy
Sensitivity and Implementation Invariance





Credits: Yasuhiro Kubota, tf-keras-vis

From sensitivity to attribution

Attribution methods should satisfy
Sensitivity and Implementation Invariance

Attribution = How predictions are formed

Example methods:
Integrated Gradients, Layer-wise relevance propagation (LRP),
Pixel perturbation analysis

General idea:
Rank input by its relevance for the prediction,
Backward mapping & conservation principle $\sum_i R_i^{(l)} = \sum_j R_j^{(l+1)}$ l = layer, neurons l,j

Layer-wise relevance propagation (LRP)

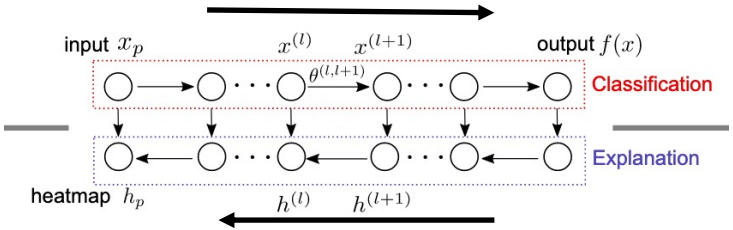
The Principle is based on Taylor decomposition & can in principle be applied to any architecture.

- Perform a standard forward pass for predictions
- Then: Propagate the model output backward, layer-wise, using decomposition as:

$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon\,\mathrm{sign}(\sum_{i'} z_{i'j})} R_j^{(l+1)}$$

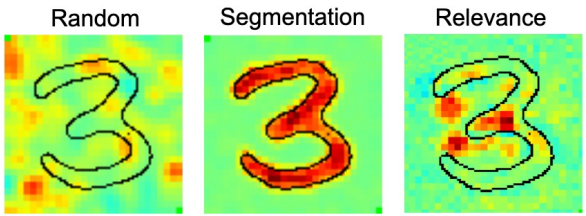
activation

$$z_{ij} = a_i^{(l)} w_{ij}^{(l,l+1)}$$



Samek et al. 2015 arXiv:1509.06321

- Final product: Relevance heatmap of input shape



Summary Block 2: Attention

✓ Introspection is possible via sensitivity analyses and localisation
- Class Activation Maps upsample last CNN-layer to input data shape
- Saliency is a gradient-based approach to rank importance of input pixels

✓ Introspection is possible by analysing how predictions are formed – attribution
- Importance of sensitivity and implementation invariance
- Example of layer-wise relevance propagation for input-shaped relevance heatmaps

→ Go beyond visualization with both sensitivity and attribution to understand the input – model – prediction relations

Hands-on: Attention of an ‘Astrophysical’ CNN

This exercise will create and analyse saliency maps for the trained 3D-21cmPIE-Net here:

https://github.com/stef-neu/3D-21cmPIE-Net/blob/main/paper_plots/saliency_maps.py
This uses tf_keras_vis.saliency

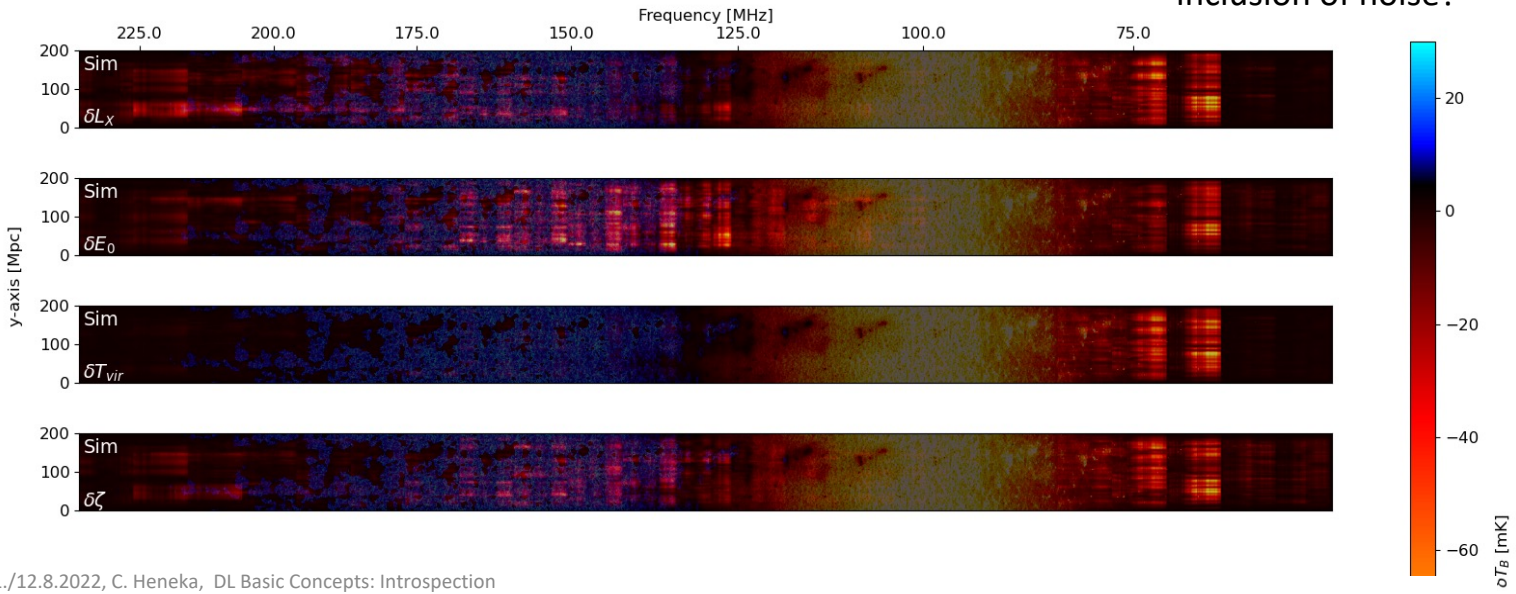
We will also briefly compare saliency maps with CAM generated as well with tf_keras_vis

(a github with a pedagogical walk-through as a jupyter-notebook of this will follow)

Hands-on: Attention of an ‘Astrophysical’ CNN

Hands-on: Creation of saliency maps and their analysis

A) Simulations (no noise)



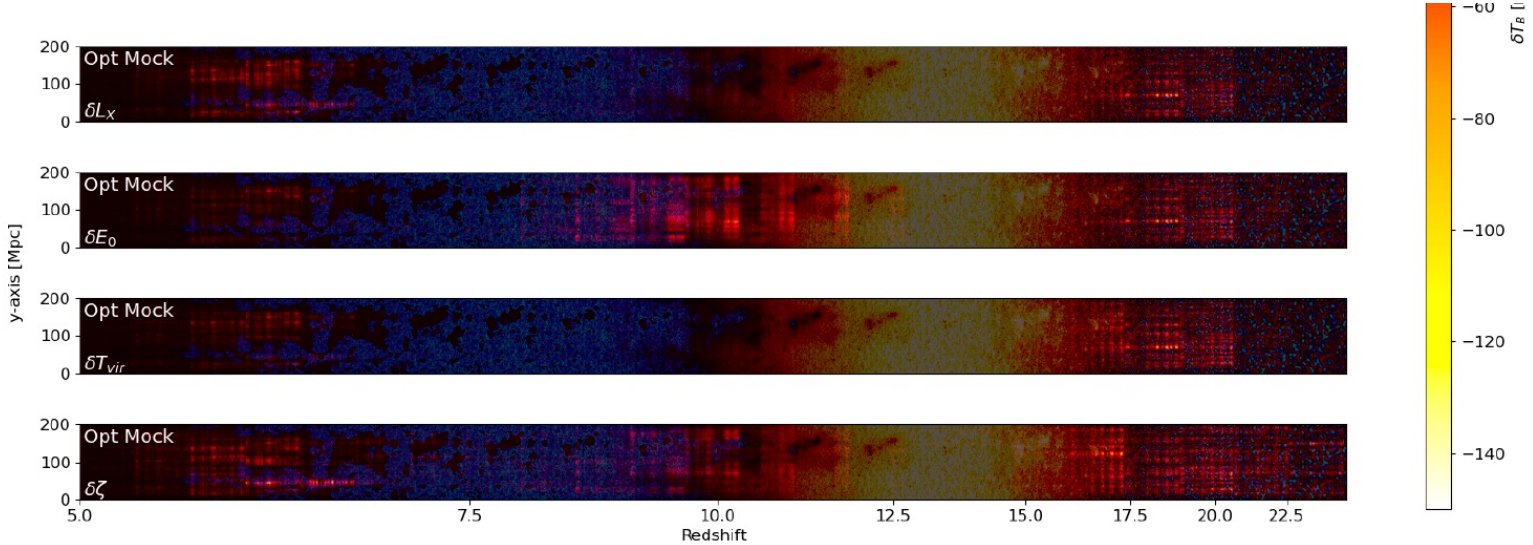
Question: What do you notice, which areas spatially and temporally are important?
How does attention shift with the inclusion of noise?

Hands-on: Attention of an ‘Astrophysical’ CNN

Hands-on: Creation of saliency maps and their analysis

B) Mock data (incl. noise)

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



Part 3: Model-agnostic

To learn about the model and its reaction to data, one can:

- Visualise feature space [Block 1]
- Analyse attention & reaction (CAM, saliency) [Block 2]
- Be **Model-agnostic** (e.g. SHAP) [Block 3]
- Many more: Put errors/probabilities, regularise, unsupervised … [Block 3]
  … active field of development!

→ Only take into account relation input to output

Example we will learn about with hands-on exercises:
SHAP

Part 3: Model-agnostic

Only take into account relation input & predictions for attribution.

Examples:
- SHAP values
- LIME
- QII

SHAP = SHapley Additive exPlanations

Idea:
Show importance of feature on predictions
(without any information on predictive quality).

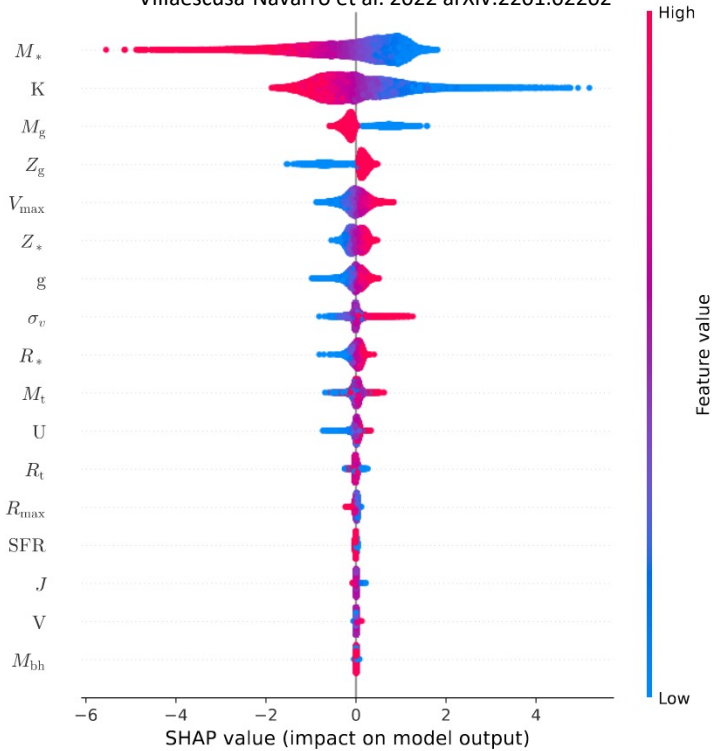
SHAP

SHAP = SHapley Additive exPlanations
Approach from Game Theory

Idea:
Assign to each input feature a value.
A larger value indicates higher importance for the output prediction.

Do so via optimal credit allocation of possible ‘coalitions’.

Villaescusa-Navarro et al. 2022 arXiv:2201.02202



Part 3: Further concepts

To learn about the model and its reaction to data, one can:

- Visualise feature space [Block 1]
- Analyse attention & reaction (CAM, saliency) [Block 2]
- Model-agnostic (e.g. SHAP) [Block 3]
- Many more: Put errors/probabilities, regularise, unsupervised … [Block 3]

→ … active field of research & development!

Part 3: Further concepts

Some concepts interesting for introspection worth to be highlighted:

- Bottlenecks (e.g. in variational autoencoders):
  Disentanglement & physical interpretation

- Generative methods for reconstruction errors

- Regulatory terms & reaction

- Uncertainty estimation ‘I do not know’ ‘Not sure’
  error on predictions

Part 3: Further concepts

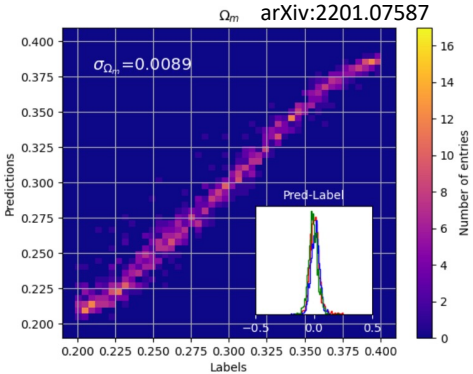
Some concepts interesting for introspection worth to be highlighted:

- Bottlenecks (e.g. in variational autoencoders):
  Disentanglement & physical interpretation

- Generative methods for reconstruction errors

- Regulatory terms & reaction

- Uncertainty estimation ‘I do not know’ ‘Not sure’
  error on predictions



Epistemic (systematic) uncertainty on learner

Aleatoric (statistical) inherent randomness

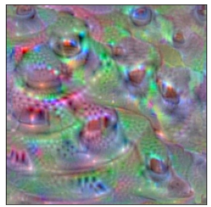
Hands-on Part 3: Further concepts

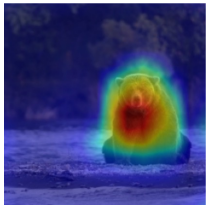
Very brief exercise planned using SHAP
(https://shap.readthedocs.io/en/latest/index.html)

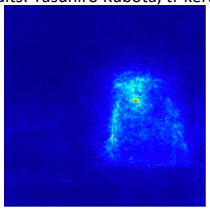
Plus bonus if time permits: Playing e.g. with deep dream

Summary of this lecture

1) We learned that Introspection important to verify and debug your model, together with understanding your data and predictions.
2) Visualisation helps to learn more about the trained model; we can track to hierarchy of features in network models.
3) Sensitivity and attribution relate predictions to input, enable discriminative localisation.
4) A zoo of methods waits to be used and developed for introspection!







Credits: Yasuhiro Kubota, tf-keras-vis