Karlsruhe Institute of Technology

# Understanding machine learning:
# How to find out what your neural network is doing

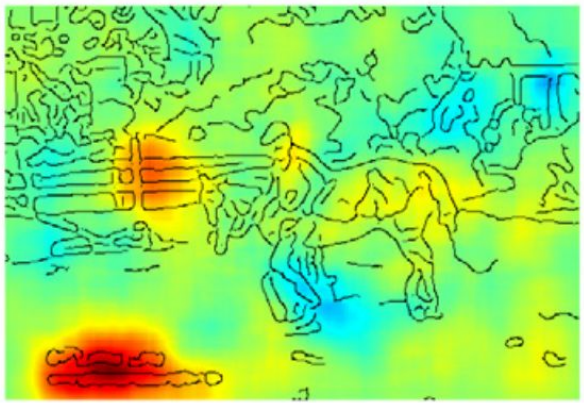
Stefan Wunsch
stefan.wunsch@cern.ch

CERN EP-SFT / KIT ETP

# What is this about?

# Why does the neural network make a prediction?


Input image for the classification



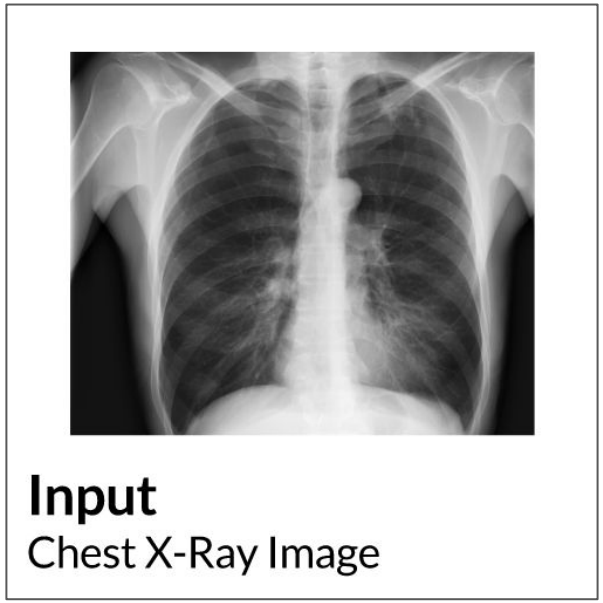
Highlighted are the parts of the image which are primarily used by the NN to classify the image as “horse”

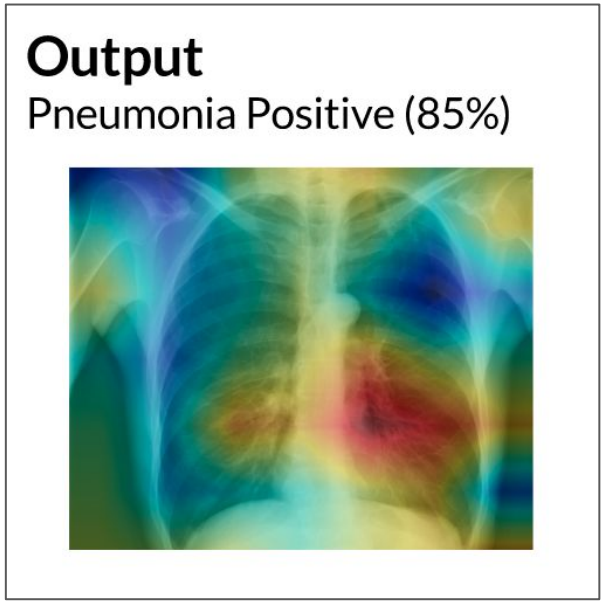
Understanding the internals of neural networks allows to verify predictions



Methods for interpreting and understanding deep neural networks

# Why does the neural network make a prediction?




Understanding the internals of neural networks allows to add additional information to the predictions

[CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning]

# Why does the neural network make a prediction?

BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / A YEAR AGO

## Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin 8 MIN READ

SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

Understanding the internals of neural networks allows to detect biases in the predictions



Amazon scraps secret AI recruiting tool that showed bias against women

# Why does the neural network make a prediction?

**Recital 71**
**EU GDPR**

(71) The data subject should have the right not to be subject to a decision, which may include a measure, evaluating personal aspects relating to him or her which is based solely on automated processing and which produces legal effects concerning him or her or similarly significantly affects him or her, such as automatic refusal of an online credit application or e-recruiting practices without any human intervention.

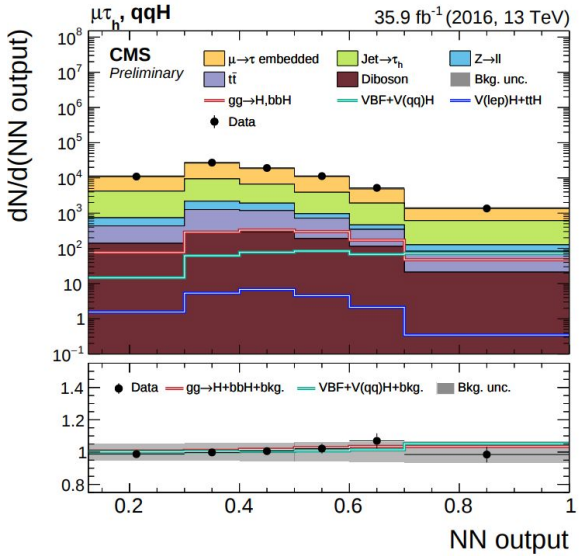
The European Union General Data Protection Regulation states that "[the data subject should have] the right ... to obtain an explanation of the decision reached"

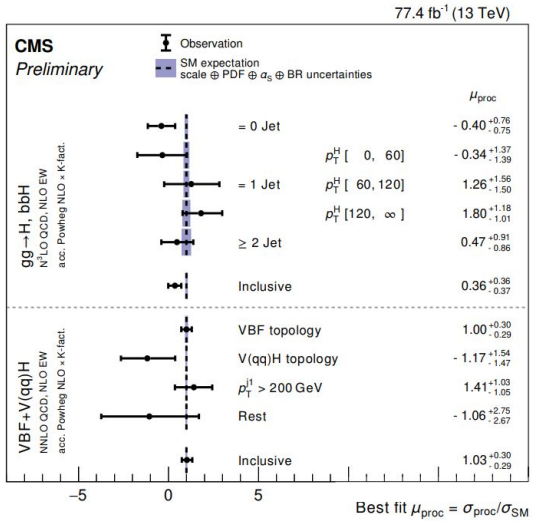
Understanding the internals of neural networks allows to be compliant with the legislation



European Union General Data Protection Regulation

# Why does the neural network make a prediction?




Understanding the internals of neural networks allows to use them safely for scientific measurements

[Measurement of Higgs boson production and decay to the tau tau final state]

# Outline

# Outline

1. **What is a neural network?**
   **“We don’t know what the NN is doing!”**
   Introduction to the problem

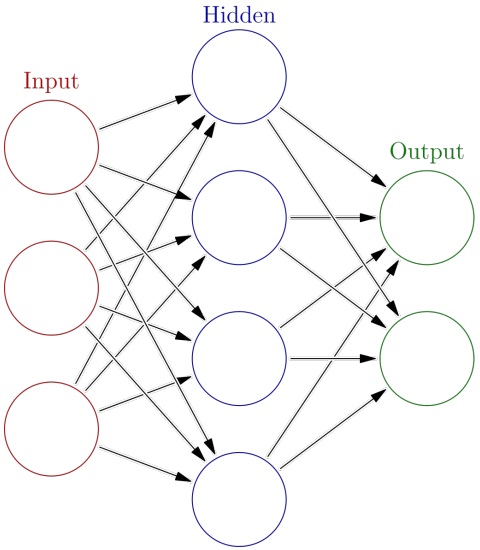
2. **How to study a neural network?**
   Gradient analysis of the neural network function
   Hands-on

3. **Analysis of advanced neural network models**
   Brief introduction to fully convolutional architectures
   Architecture specific methods
   Hands-on

4. **Methods from recent publications**
   Overview over the literature
   Introduction to layerwise relevance propagation
   Testing with Concept Activation Vectors
   Hands-on

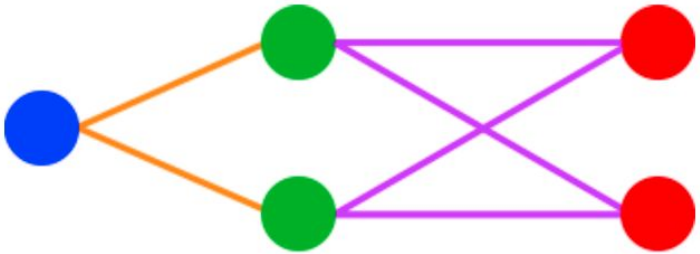
# What is a neural network?

# One step back: What is a neural network?




- What is a neural network? → It's a well defined mathematical function!
- Could you write down the function?

# Brief mathematical introduction to (dense) NNs



$$f_{NN} = \sigma(b_2 + W_2\sigma(b_1 + W_1 x))$$

$$\text{Input} : x = \begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}$$

$$\text{Weight} : W_1 = \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix}$$

$$\text{Bias} : b_1 = \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix}$$

$$\text{Activation} : \sigma(x) = \tanh(x) \quad \text{(as example)}$$

$$f_{\mathrm{NN}} = \sigma_2\left(\begin{bmatrix} b^2_{1,1} \end{bmatrix} + \begin{bmatrix} W^2_{1,1} & W^2_{1,2} \end{bmatrix} \sigma_1\left(\begin{bmatrix} b^1_{1,1} \\ b^1_{2,1} \end{bmatrix} + \begin{bmatrix} W^1_{1,1} & W^1_{1,2} \\ W^1_{2,1} & W^1_{2,2} \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}\right)\right)$$

Key to all methods shown later: Detailed understanding of the mathematics

# Further reading: Deep learning textbook

- Free textbook by Ian Goodfellow et al.
- Written by leading scientists in the field!
- First two parts cover everything you need to know for most applications

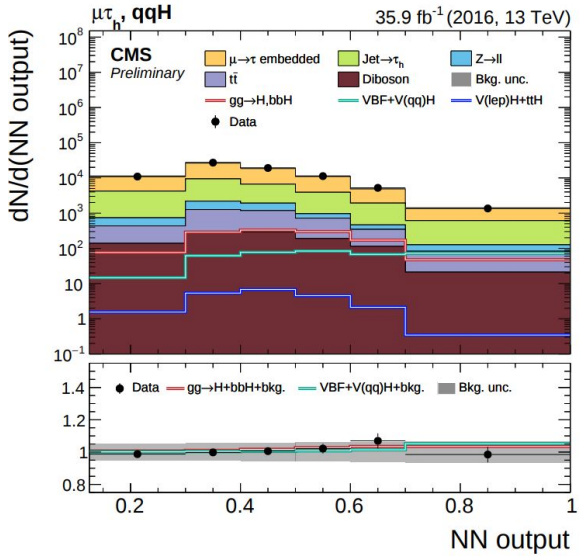




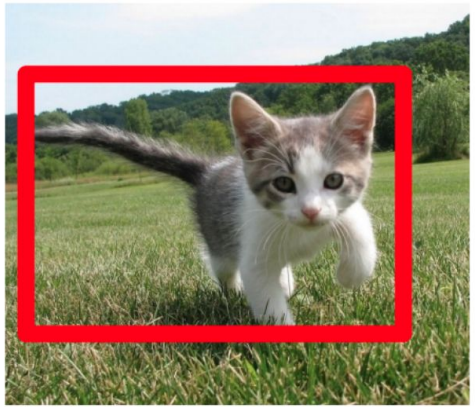
Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

# “We don’t know what the NN is doing!”

# Can we trust the predictions of the NN?




Which are the input variables contributing most to the separation between signal and background?

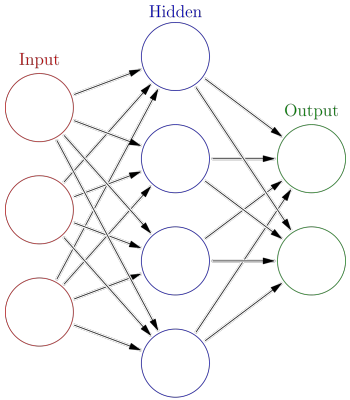


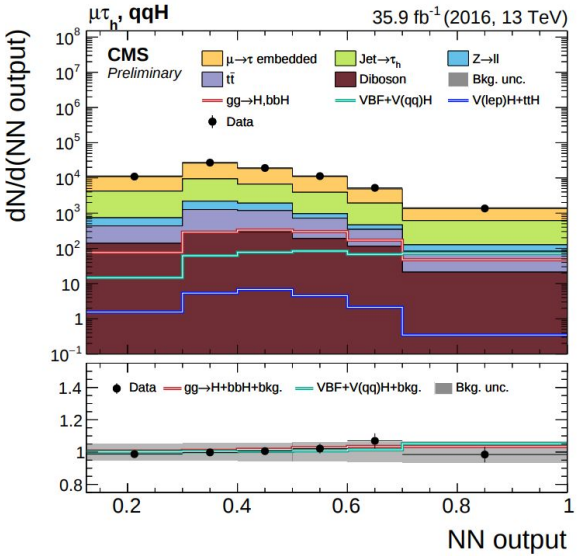
CAT

Which are the input pixels contributing most to the classification “cat”?

# In science: NNs in presence of syst. uncertainties

- Do the predictions of the neural network have an intrinsic systematic uncertainty?
- Does a neural network differ from the computation of any other derived variable of the inputs?
- Since the NN is simply a mathematical function, **all systematic uncertainty comes from the uncertainty of the inputs**
- Still we have to verify for what reason the NN predicted something and why we trust the result.




**A neural network does not have an intrinsic systematic uncertainty.**

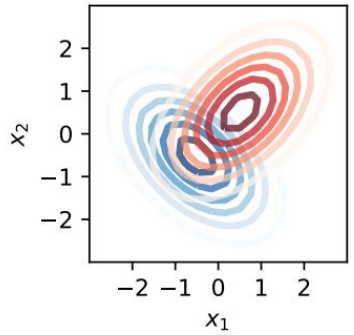
# Which questions should we ask?

- **Only partially correct question:**
  Which are to input variables contributing most to the prediction?

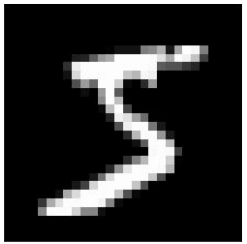
- **More correct question:**
  Which sub-spaces of the N-dimensional input space contribute most to the prediction?

- Problem most often simplified to first-order (dependence on “marginal distributions”)

- Intractable problem in higher orders for high-dimensional inputs (images, …)

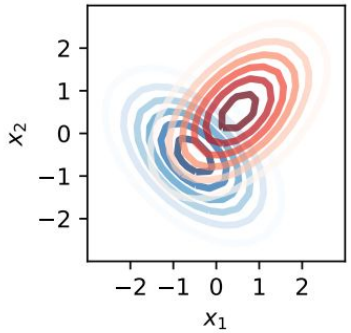



Example of a 2D input space

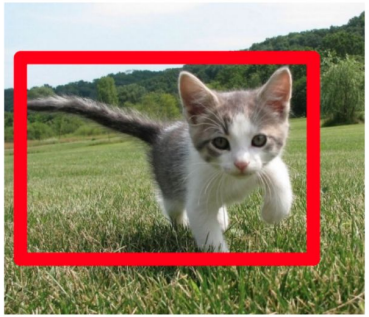


Input image with 28 x 28 x 1 = 784 dimensional input space

# Analysis of a single example vs general measures




What are the features used by the NN to separate the classes red and blue?



**CAT**

What are the features used by the NN to classify this image as a cat?

- **What makes a cat looking like a cat?**
- General measures interesting for NNs applied in data analysis tasks based on summary statistics

- **What makes this specific cat looking like a cat?**
- Individual reasoning interesting for computer vision, natural language processing, ...

# Take away: What is a neural network?

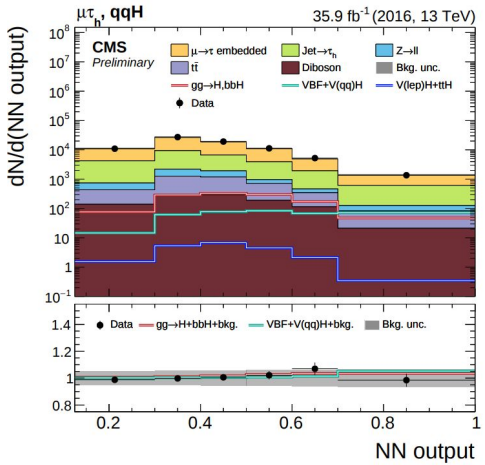
- A neural network is a well defined mathematical function.

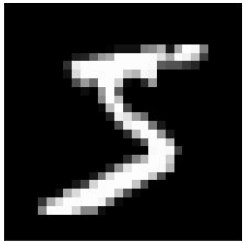
$$f_{\mathrm{NN}} = \sigma_2\left(\begin{bmatrix} b^2_{1,1} \end{bmatrix} + \begin{bmatrix} W^2_{1,1} & W^2_{1,2} \end{bmatrix} \sigma_1\left(\begin{bmatrix} b^1_{1,1} \\ b^1_{2,1} \end{bmatrix} + \begin{bmatrix} W^1_{1,1} & W^1_{1,2} \\ W^1_{2,1} & W^1_{2,2} \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}\right)\right)$$

- A neural network does not have an intrinsic systematic uncertainty.

- The neural network function is sensitive to the N-dimensional input space and not only on the marginal distributions of the input variables.

- Reasoning of the predictions are interesting for a single example and on a full dataset



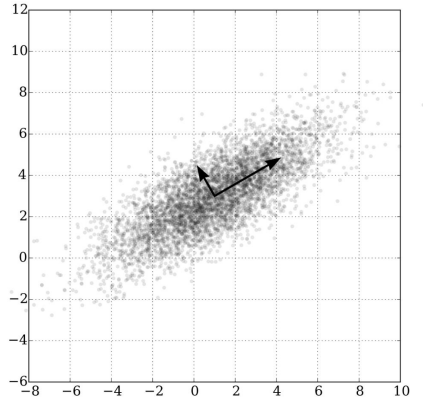
Input image with 28 x 28 x 1 = 784 dimensional input space

# How to study the dependence of the NN function to the input space?

# PCA and the “leave one out” approach

- **Principal component analysis:** Transform the coordinate system of the input space into a linearily uncorrelated one
- Eigenvalues of transformation matrix can be interpreted as measure of information (variance) of the respective axis
- **Problem:** There is no guarantee that the NN will pick up this information during training.

- **“Leave one out” approach:** Retrain the neural network with a subset of input variables and use an appropriate evaluation metric for a ranking (ROC, AUC, …)
- Removing important variables should result in a significant loss of the used evaluation metric
- **Problem:** Each training with changed parameters is unique, no guarantee that the NN will always converge to the same function

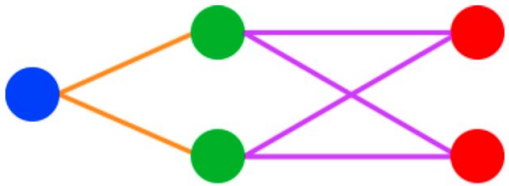



Visualization of the PCA

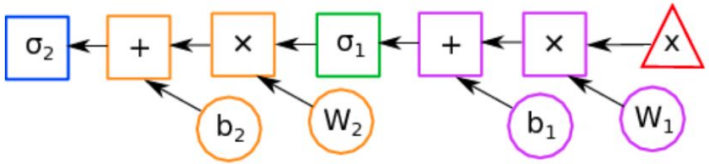
**The trainable parameters must be static for the analysis of the neural network function.**

# Gradient analysis of the neural network function

- **Training a neural network:** Compute gradient from the loss function to the weights
- **Gradient analysis:** Compute gradient from the NN function to the inputs
- **Which inputs make the NN function to change most?**
- **Easy to implement:**
  Modern ML libraries are computational graphs libraries →
  Computing gradients is an essential part of TensorFlow, PyTorch, ...



$f_{NN} = \sigma(b_2 + W_2\sigma(b_1 + W_1 x))$




$$f_{\mathrm{NN}} = \sigma_2\left(\begin{bmatrix} b^2_{1,1} \end{bmatrix} + \begin{bmatrix} W^2_{1,1} & W^2_{1,2} \end{bmatrix} \sigma_1\left(\begin{bmatrix} b^1_{1,1} \\ b^1_{2,1} \end{bmatrix} + \begin{bmatrix} W^1_{1,1} & W^1_{1,2} \\ W^1_{2,1} & W^1_{2,2} \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}\right)\right)$$

# Taylor expansion of the neural network function

Taylor expansion as generalization of the idea of a gradient analysis for the N-dimensional input space:

$$T(x,y) = f(a,b) + (x-a)f_x(a,b) + (y-b)f_y(a,b) + \frac{1}{2!}\left((x-a)^2 f_{xx}(a,b) + 2(x-a)(y-b)f_{xy}(a,b) + (y-b)^2 f_{yy}(a,b)\right) + \cdots$$

Neural network response

1D input space ("marginal distributions")

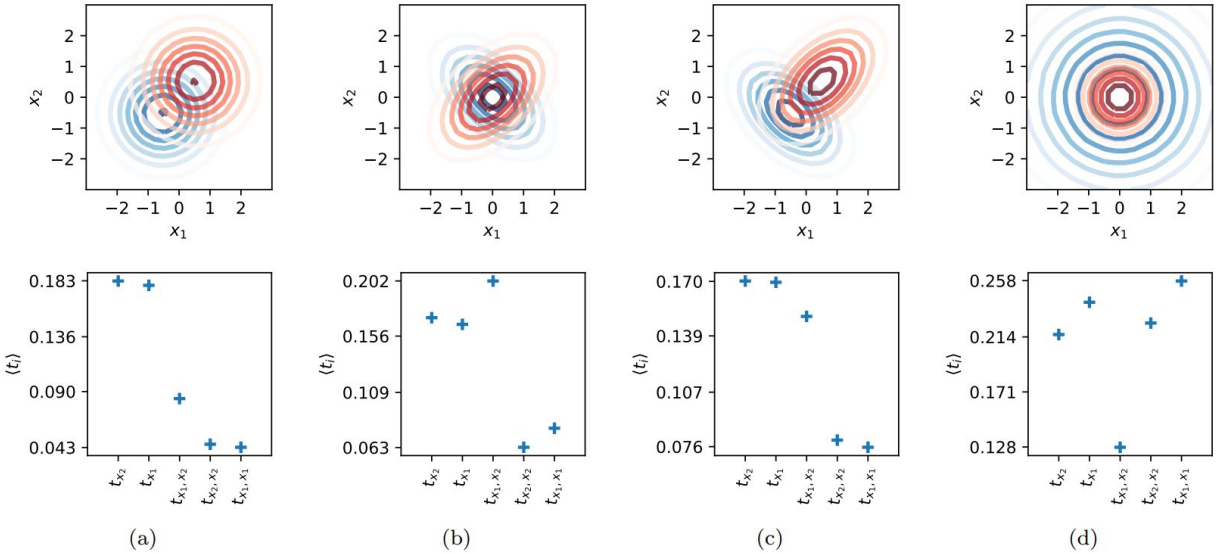
2D input space ("correlations")

Compute mean of absolute values of Taylor coefficients as metric for the sensitivity of the NN to the respective input space

# Toy examples

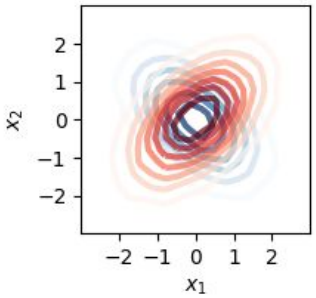
- In each example, a NN is trained to separate the red from the blue class
- $t_i$ refers to the Taylor coefficients

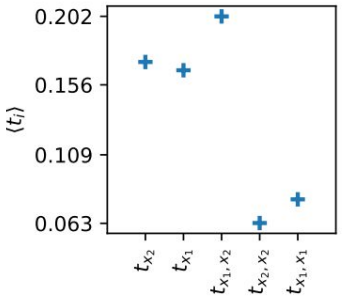
| Task | Mean value | | | | Covariance matrix | |
|---|---|---|---|---|---|---|
| | Signal $(x_1, x_2)$ | | Background $(x_1, x_2)$ | | Signal | Background |
| Fig. 1a | 0.5 | 0.5 | −0.5 | −0.5 | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ |
| Fig. 1b | 0 | 0 | 0 | 0 | $\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$ |
| Fig. 1c | 0.5 | 0.5 | −0.5 | −0.5 | $\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$ |
| Fig. 1d | 0 | 0 | 0 | 0 | $\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$ | $\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$ |



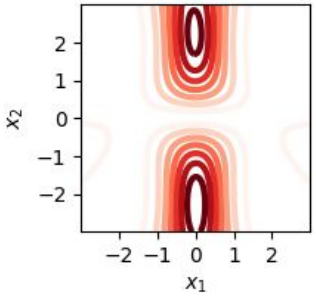

- Why has the NN in example (a) a dependence on the “correlation” of $x_1$ and $x_2$? And why is example (b) strongly dependent on the 1D input space?
- How would you interpret the coefficients $t(x_i, x_i)$?

# Dependence of the NN function to the input space

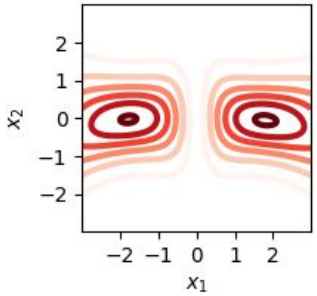


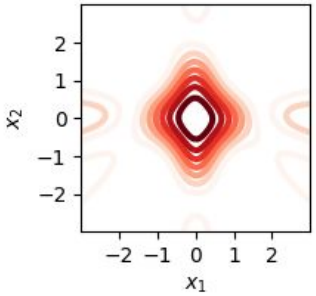

Toy example (b) with distribution of the Taylor coefficients in the input space



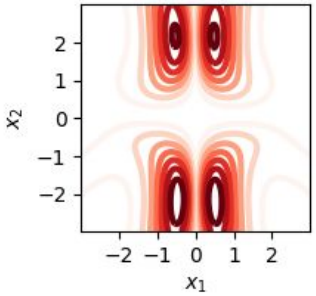

$t(x_1)$



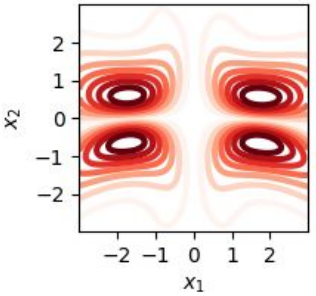

$t(x_2)$




$t(x_1,x_2)$




$t(x_1,x_1)$

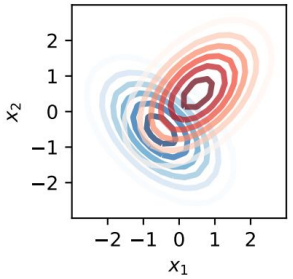



$t(x_2,x_2)$

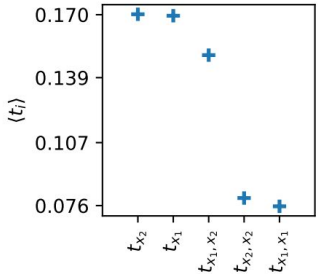
**The sensitivity of the NN function to the input variables varies significantly in the input space.**

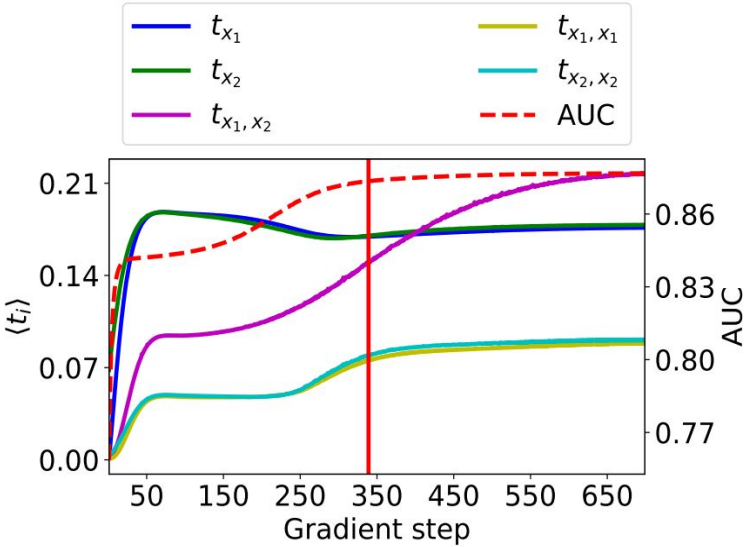
# Visualizing the learning progress

Toy scenario (c):




Taylor coefficient after converged training:




Sensitivity analysis allow to monitor the learning progress

# Take away: (Gradient) analysis of the NN function

- If the analysis needs retraining the neural network, you study the dataset and not the neural network function (similar to PCA).
- Gradient analysis and Taylor expansion allows to study the neural network function statically
- Computation of gradients is an essential part of modern graph computation frameworks and can be done with minimal effort
- Dependence of the neural network function on the inputs in general varies strongly in the inputs space

# Hands on: Gradient analysis of the NN function

**GitHub repository with the notebooks:**
https://github.com/stwunsch/kseta-topical-understanding-ml

**Notebook of interest:**
`kseta001-gradient-analysis.ipynb`

**How do I get the files locally?**

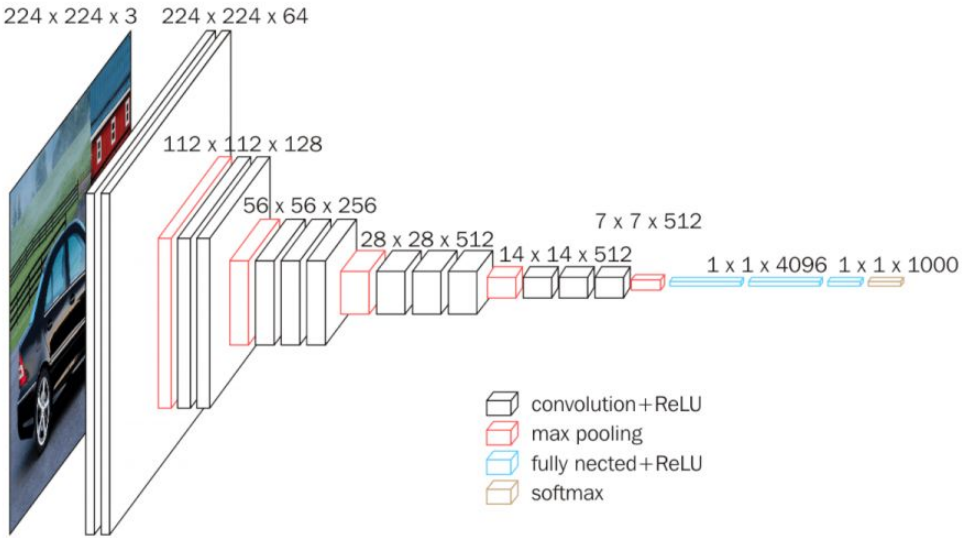
```
git clone https://github.com/stwunsch/kseta-topical-understanding-ml
```

**Access to the JupyterLab server:**
See the e-mail I've sent you.

# Analysis of advanced NN models

# Analysis of advanced NN models

- How to adapt the analysis of the neural network function to more complex NN models?
- Which kind of analysis of the neural network function is of interest?
- Can we explicitly exploit properties of modern NN architectures to study the model?

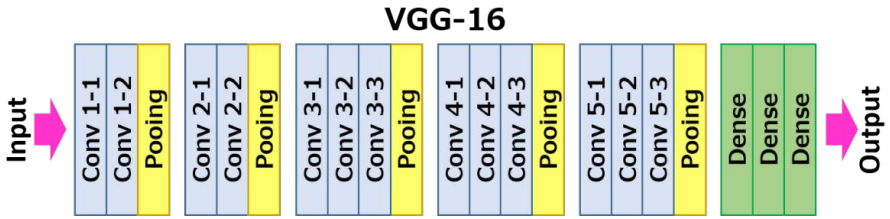


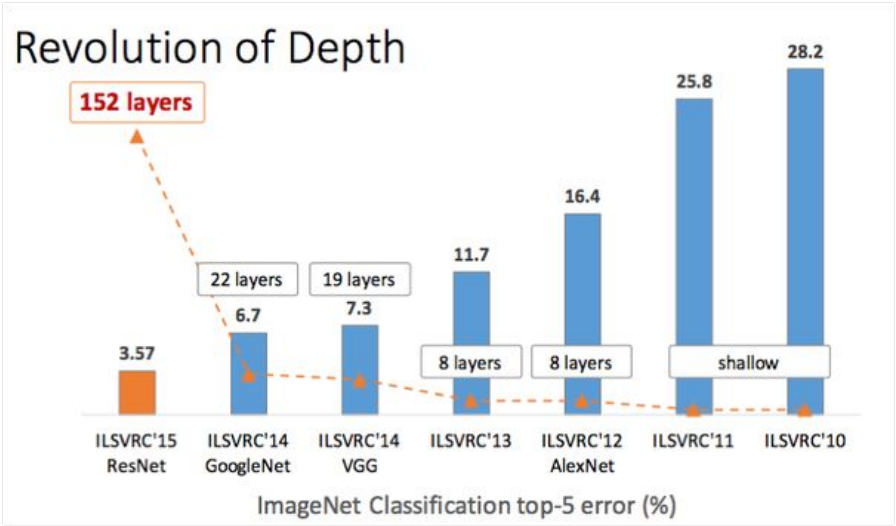

Typical fully convolutional architecture used for image classification tasks

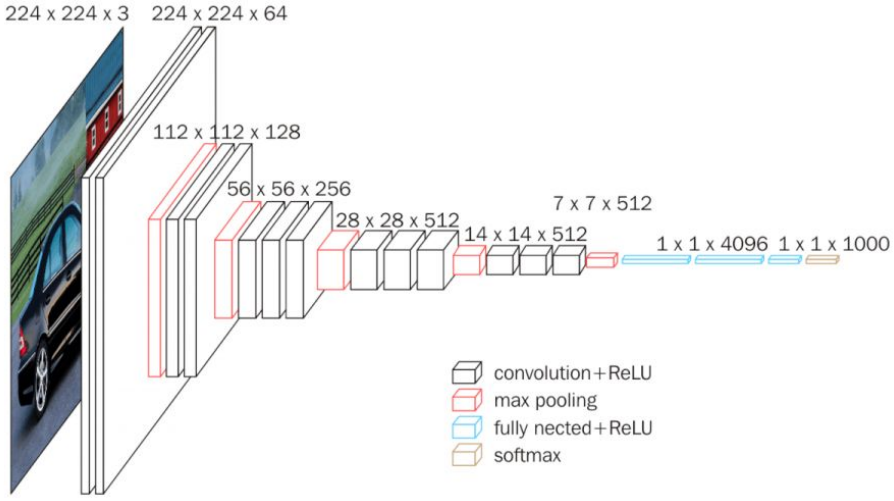
# The VGG16 architecture

IMAGENET

- Famous model from the Large Scale Visual Recognition Challenge 2014
- Achieved a 7% top-5 error for image classification



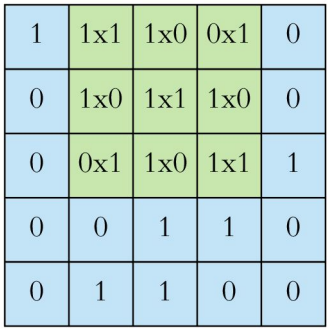



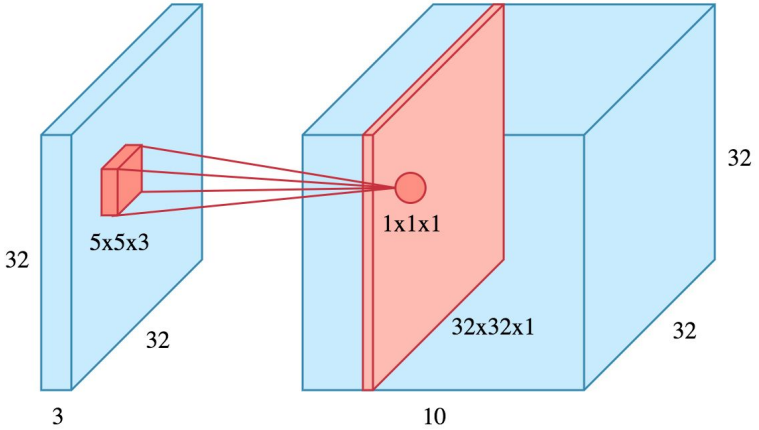




Source, Source

# Reminder: Convolutional layers

- **Input:** Image or any other tensor, e.g., color image with (height, width, channels)
- **Filters:** Weights applied as a sliding window to a spatially restricted part of the input, each filter is usually applied on all channels of the input
- **Feature maps:** Each channel of the output tensor is a single feature map with an own filter




Source

# Class activation maps

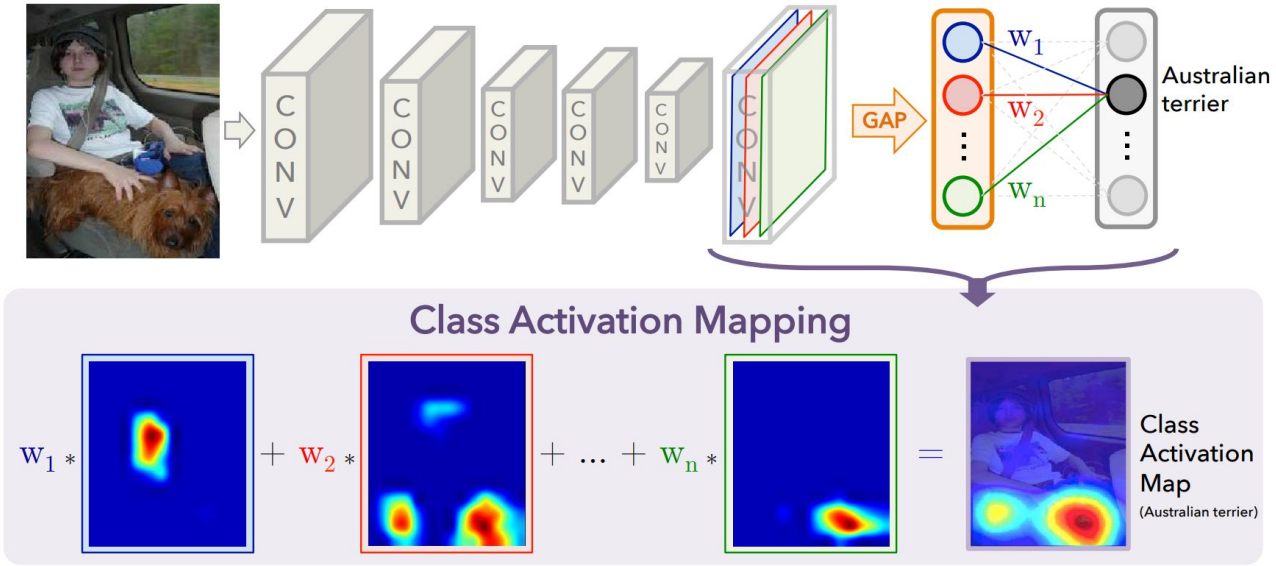


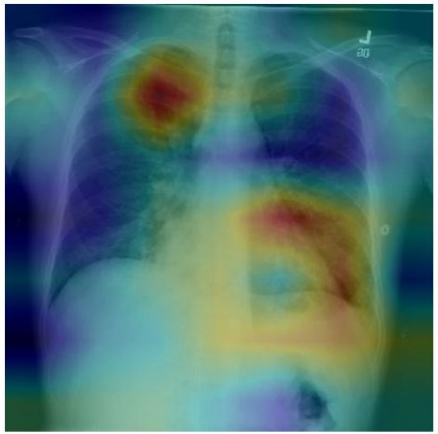

Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

Feature maps of last convolutional layer should contain high-level features associated with the location in the image

Learning Deep Features for Discriminative Localization

# Class activation maps examples


CAM for prediction of a scene in an image from the Stanford 40 Actions dataset



(a) Patient with multifocal community acquired pneumonia. The model correctly detects the airspace disease in the left lower and right upper lobes to arrive at the pneumonia diagnosis.

CAM for detection of pneumonia

Learning Deep Features for Discriminative Localization
CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

# Take away: Analysis of advanced NN models

- N-dimensional input space not explicitly analyzable for models with many inputs due to exploding complexity
  →Study only in first order or use architecture dependent methods

- Architecture dependent methods:
  Convolutional networks allow to visualize the feature maps of the convolutional layers

- Field focuses on interpretation of single predictions due to popular applications of complex neural network models such as computer vision

# Hands on: Analyze the VGG16 model

**GitHub repository with the notebooks:**
https://github.com/stwunsch/kseta-topical-understanding-ml

**Notebook of interest:**
`kseta002-cnn-vgg16.ipynb`

**How do I get the files locally?**

```
git clone https://github.com/stwunsch/kseta-topical-understanding-ml
```

**Access to the JupyterLab server:**
See the e-mail I've sent you.

# Methods from recent publications

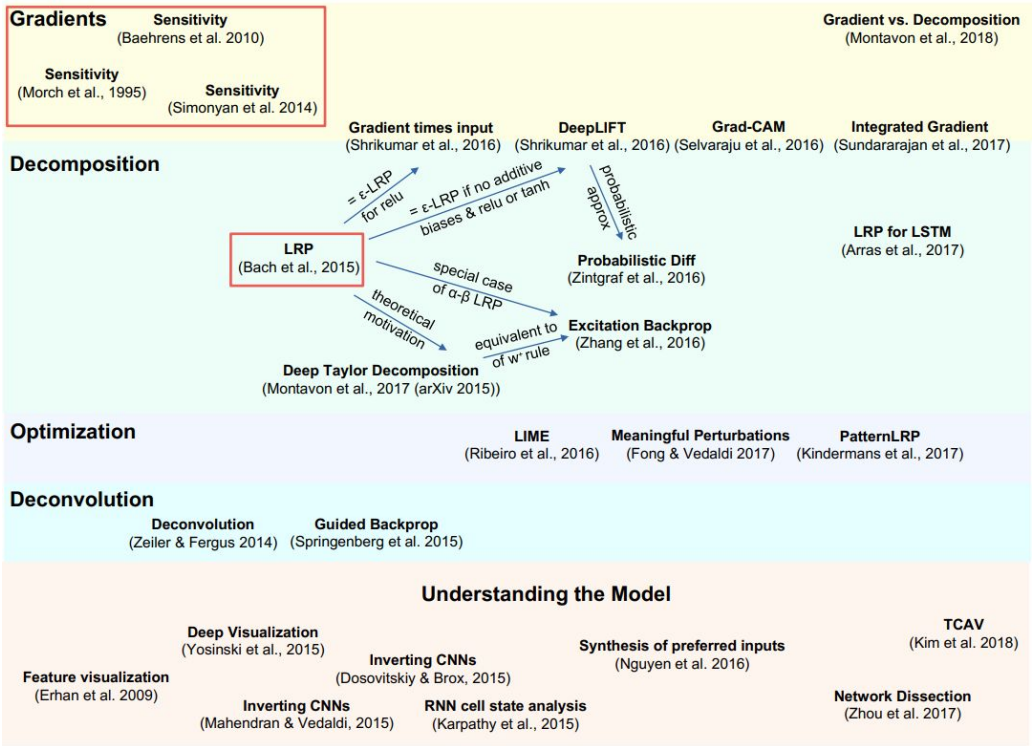
# Methods from recent publications

- Recent publications focus on interpretation of single predictions
- Group around Wojciech Samek highly visible in the field
- (IMHO) Field far away from any satisfactory generic solution / framework / theory




heatmapping.org

# (Biased) overview over the current literature

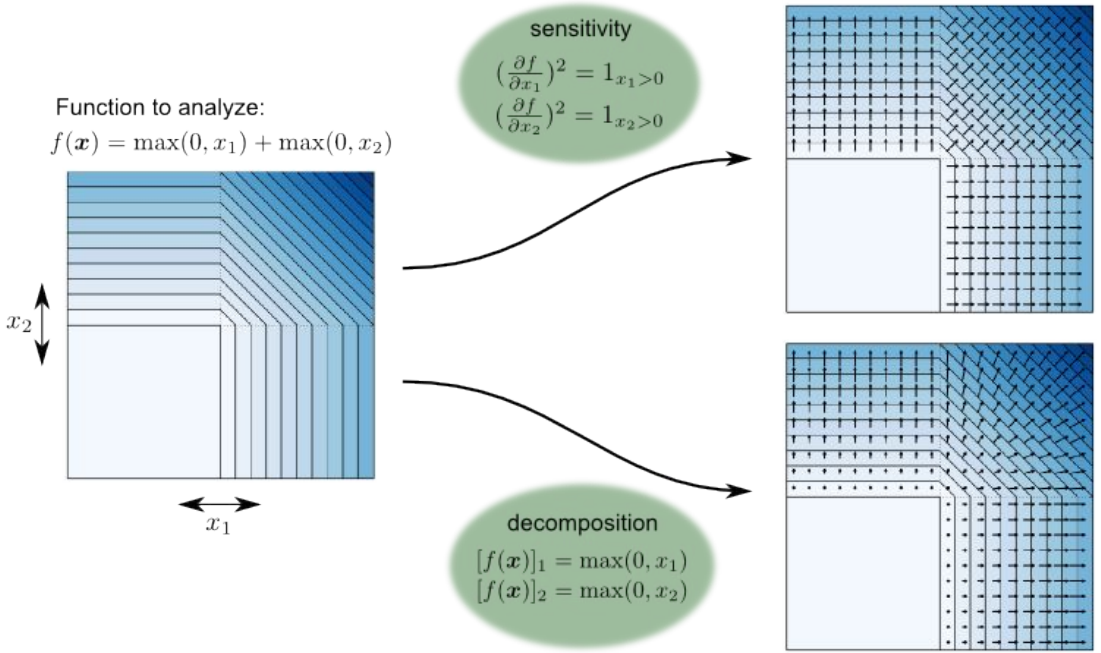


# Sensitivity vs decomposition

- **Sensitivity:** Two points in the input space could have a completely different explanation, measure has no conservation property for the output score

- **Decomposition:** Continuous in the full input space, redistributes the output score fully on the inputs

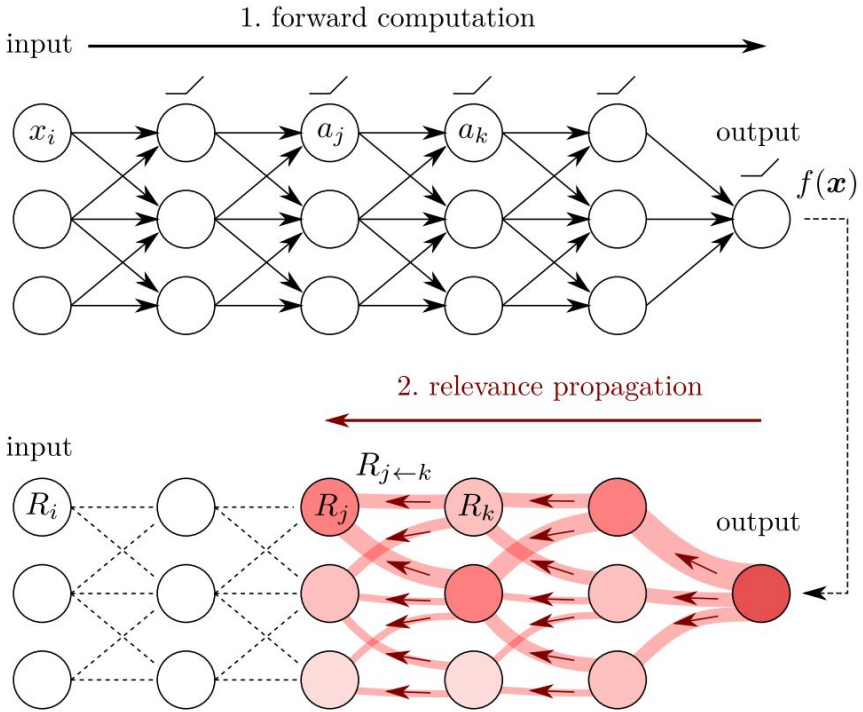
$$\forall\, \boldsymbol{x}\colon f(\boldsymbol{x}) = \sum_{p} R_p(\boldsymbol{x})$$




Source

# Layerwise relevance propagation

- Output score is propagated backwards via redistributions rules conserving the relevance
- Cannot tell dependence on the N-dimensional input space only on the plain input variables
- Can be applied to many architectures almost generically






On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation

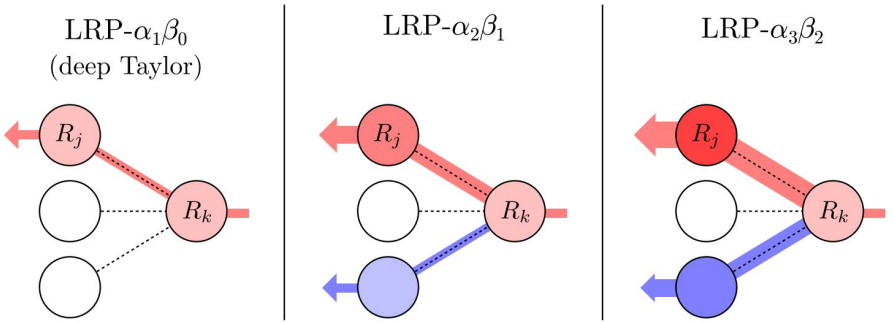
# Redistribution rules for LRP

- Various redistribution rules defined in the paper based on general formula

$$R_j = \sum_k \left( \alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} \right) R_k$$

- Redistribution separated into positive and negative part reflecting “attracting and repelling” features for the inspected output

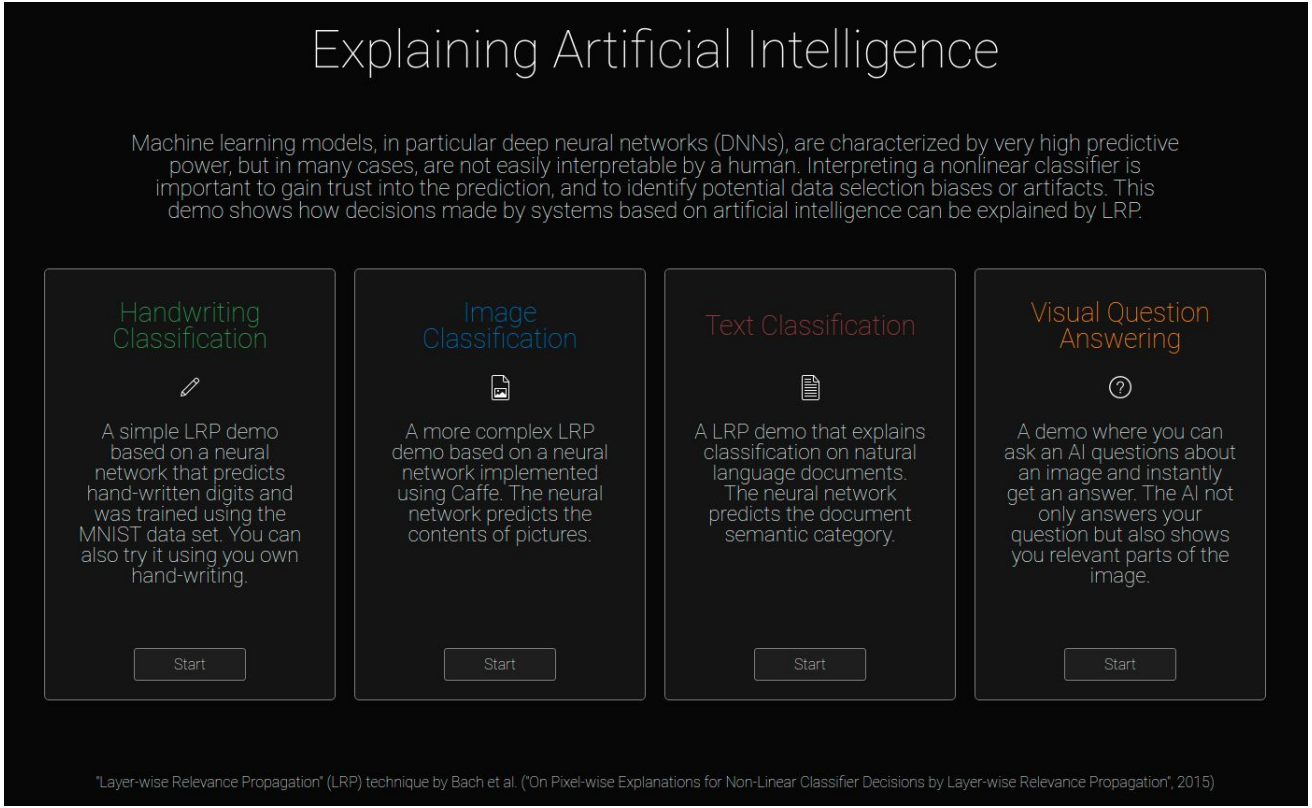
- Indices of alpha and beta denote value in formula





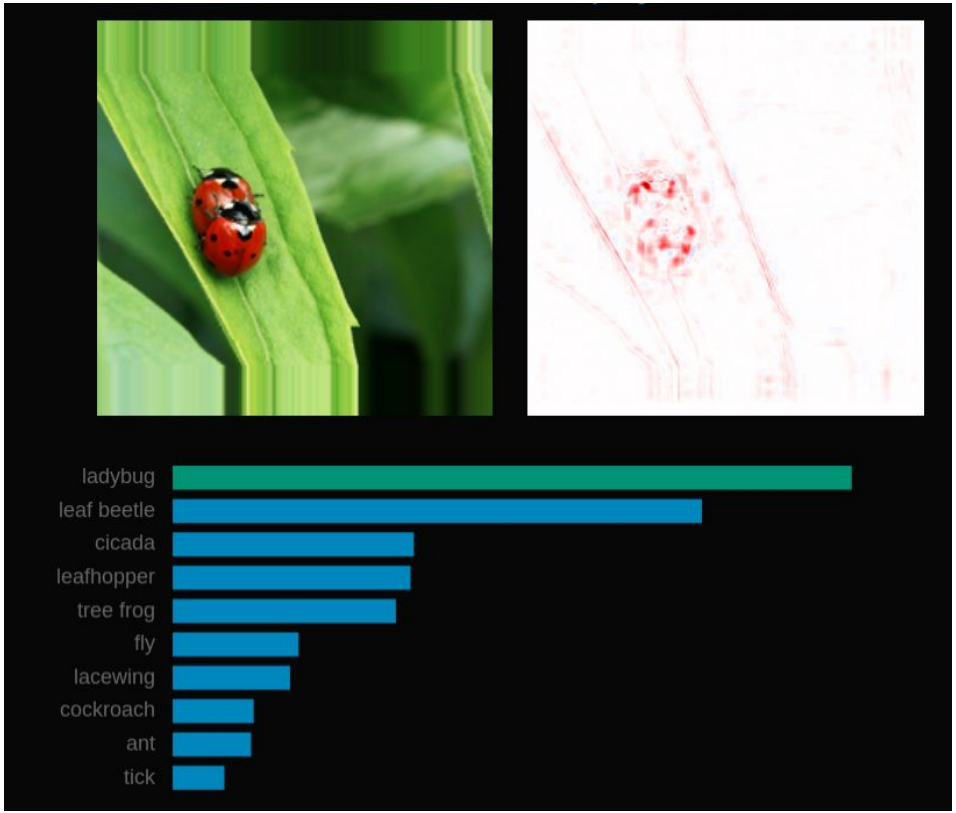

Methods for interpreting and understanding deep neural networks

# Explainable AI demos




Link

# LRP for image classification




- LRP most often applied for computer vision models
- Highly varying informative value of the interpretation
- Still very relevant publication in the field

| TITLE | CITED BY | YEAR |
| --- | --- | --- |
| On Pixel-wise Explanations for Non-Linear Classifier Decisions by Layer-wise Relevance Propagation<br>S Bach, A Binder, G Montavon, F Klauschen, KR Müller, W Samek<br>PLOS ONE 10 (7), e0130140 | 544 | 2015 |

[Methods for interpreting and understanding deep neural networks]

# LRP for sentiment analysis

| true | predicted | N° | Notation: -- very negative, - negative, 0 neutral, + positive, ++ very positive |
|---|---|---|---|
| | | 1. | do n't waste your money . |
| | | 2. | neither funny nor suspenseful nor particularly well-drawn . |
| | | 3. | it 's not horrible , just horribly mediocre . |
| | | 4. | ... too slow , too boring , and occasionally annoying . |
| | | 5. | it 's neither as romantic nor as thrilling as it should be . |
| | -- | 6. | the master of disaster - it 's a piece of dreck disguised as comedy . |
| | | 7. | so stupid , so ill-conceived , so badly drawn , it created whole new levels of ugly . |
| | | 8. | a film so tedious that it is impossible to care whether that boast is true or not . |
| | | 9. | choppy editing and too many repetitive scenes spoil what could have been an important documentary about stand-up comedy . |
| -- | | 10. | this idea has lost its originality ... and neither star appears very excited at rehashing what was basically a one-joke picture . |
| | ++ | 11. | ecks this one off your must-see list . |
| | - | 12. | this is n't a `` friday '' worth waiting for . |
| | - | 13. | there is not an ounce of honesty in the entire production . |
| | - | 14. | do n't expect any surprises in this checklist of teamwork cliches ... |
| | - | 15. | he has not learnt that storytelling is what the movies are about . |
| | - | 16. | but here 's the real damn : it is n't funny , either . |
| | + | 17. | these are names to remember , in order to avoid them in the future . |
| | - | 18. | the cartoon that is n't really good enough to be on afternoon tv is now a movie that is n't really good enough to be in theaters . |
| | ++ | 19. | a worthy entry into a very difficult genre . |
| ++ | | 20. | it 's a good film -- not a classic , but odd , entertaining and authentic . |
| | -- | 21. | it never fails to engage us . |

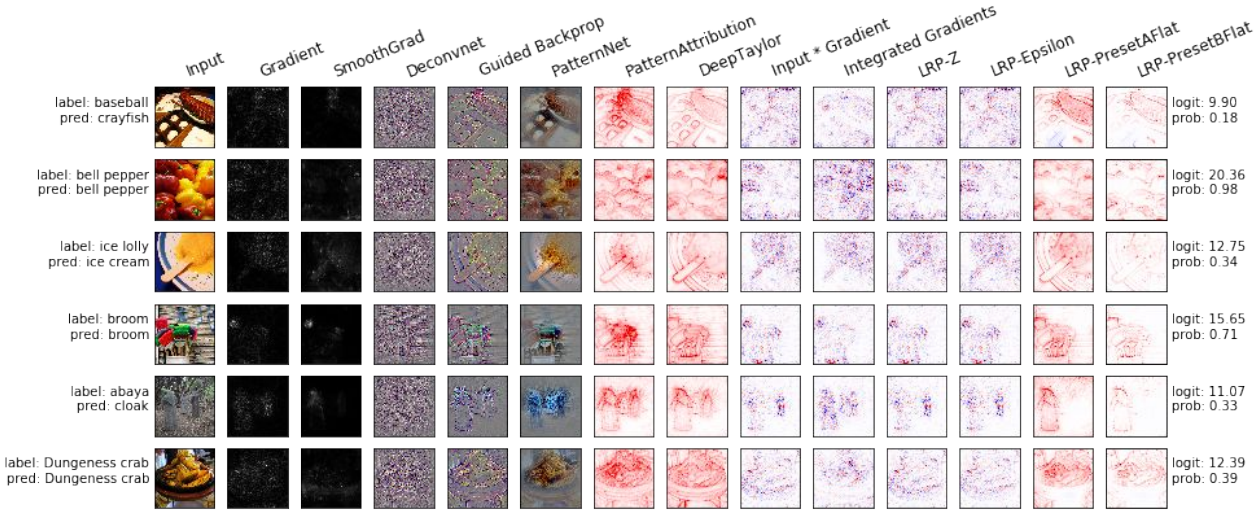
Figure 2: LRP heatmaps of exemplary test sentences, using as target class the *true* sentence class. Positive relevance is mapped to red, negative to blue, and the color intensity is normalized to the maximum absolute relevance per sentence. The true sentence class, and the classifier's predicted class, are indicated on the left.

- Dedicated paper for application on recurrent model using LSTM layers
- **Sentiment analysis:** Is a text positive or negative? Highly important for automated analysis of customer reviews.

[Explaining Recurrent Neural Network Predictions in Sentiment Analysis]

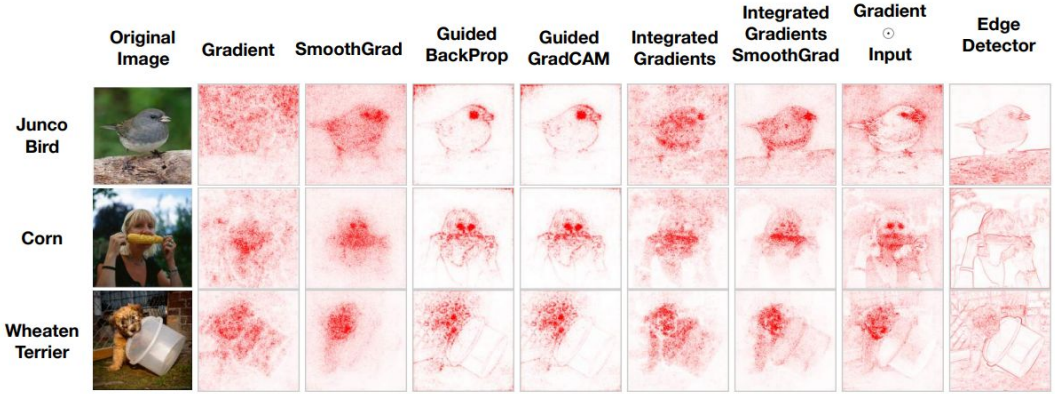
# Reference implementations

- **GitHub repository:** [iNNvestigate]
- Reference implementations covering many methods of the recent literature




[GitHub repository]

# Sanity checks for gradient based visualizations




Do the methods we have seen before actually not fully keep their promise?

Proposal for systematic tests of such methods:

**The model parameter randomization test** compares the output of a saliency method on a trained model with the output of the saliency method on a randomly initialized untrained network of the same architecture. If the saliency method depends on the learned parameters of the model, we should expect its output to differ substantially between the two cases. Should the outputs be similar, however, we can infer that the saliency map is insensitive to properties of the model, in this case, the model parameters. In particular, the output of the saliency map would not be helpful for tasks such as *model debugging* that inevitably depend on the model parameters.

**The data randomization test** compares a given saliency method applied to a model trained on a labeled data set with the method applied to the same model architecture but trained on a copy of the data set in which we randomly permuted all labels. If a saliency method depends on the labeling of the data, we should again expect its outputs to differ significantly in the two cases. An insensitivity to the permuted labels, however, reveals that the method does not depend on the relationship between instances (e.g. images) and labels that exists in the original data.

Sanity Checks for Saliency Maps

# Model parameter randomization test

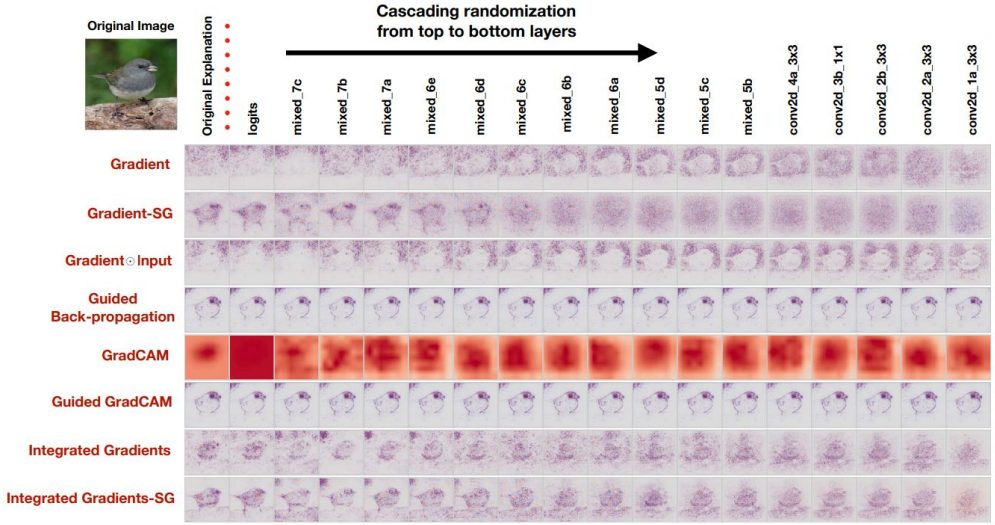
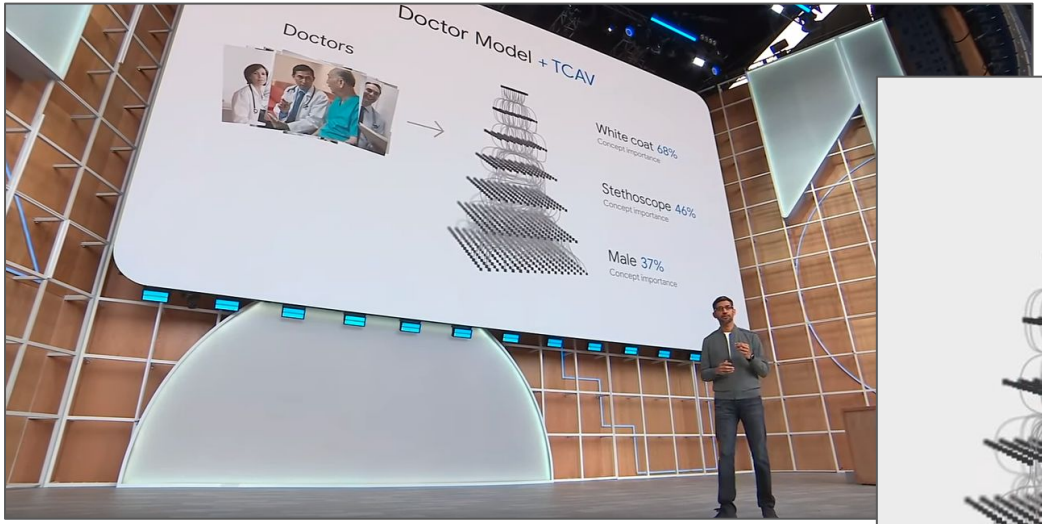


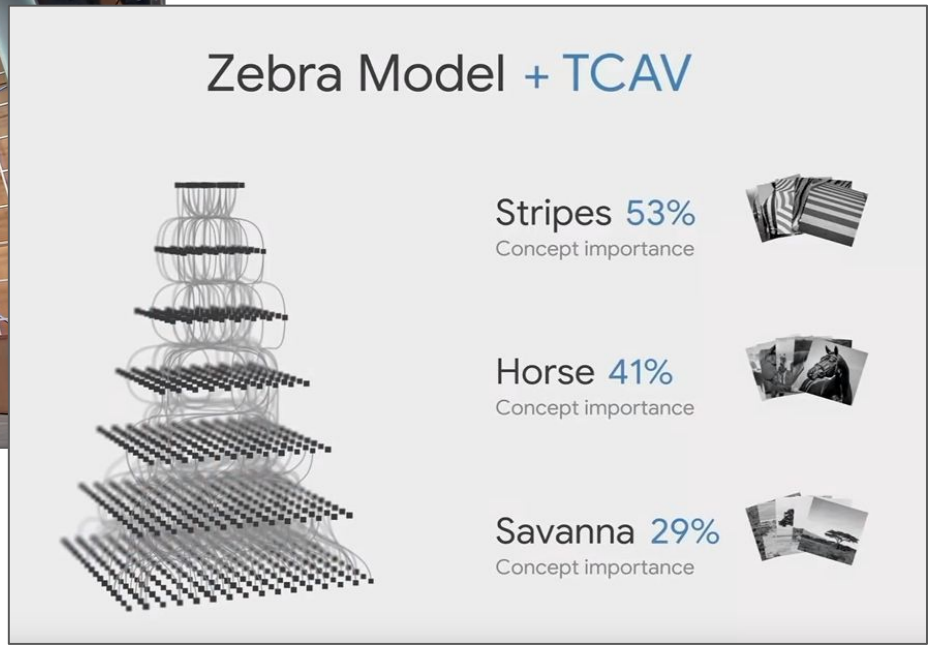

Figure 2: **Cascading randomization on Inception v3 (ImageNet).** Figure shows the original explanations (first column) for the Junco bird. Progression from left to right indicates complete randomization of network weights (and other trainable variables) up to that 'block' inclusive. We show images for 17 blocks of randomization. Coordinate (Gradient, mixed_7b) shows the gradient explanation for the network in which the top layers starting from Logits up to mixed_7b have been reinitialized. The last column corresponds to a network with completely reinitialized weights.

Do the methods we have seen before actually not fully keep their promise?

[Sanity Checks for Saliency Maps]

# Google Keynote (Google I/O ‘19)






Google Keynote (Google I/O'19)

# Testing with Concept Activation Vectors

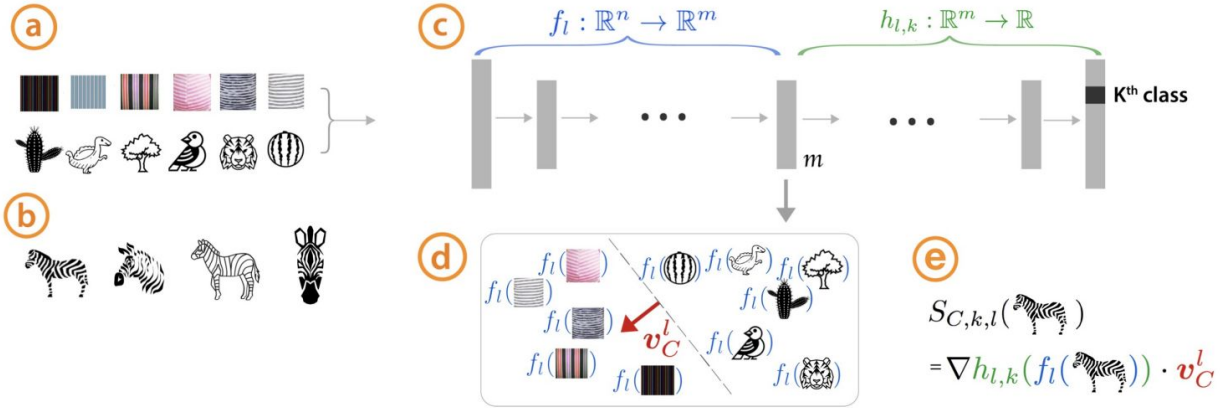



*Figure 1.* **Testing with Concept Activation Vectors:** Given a user-defined set of examples for a concept (e.g., 'striped'), and random examples ⓐ, labeled training-data examples for the studied class (zebras) ⓑ, and a trained network ⓒ, TCAV can quantify the model's sensitivity to the concept for that class. CAVs are learned by training a linear classifier to distinguish between the activations produced by a concept's examples and examples in any layer ⓓ. The CAV is the vector orthogonal to the classification boundary ($v_C^l$, red arrow). For the class of interest (zebras), TCAV uses the directional derivative $S_{C,k,l}(\boldsymbol{x})$ to quantify conceptual sensitivity ⓔ.



Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)

# Take away: Methods from recent publications

- Many methods developed during the last years, most of them gradient/backpropagation based
- Gradient based methods show deficits in systematic tests
- New algorithmic approach: Concept activation vectors
- Very dynamic field of research!

# Hands on: Methods from recent publications

**GitHub repository with the notebooks:**
https://github.com/stwunsch/kseta-topical-understanding-ml

**Notebook of interest:**
`kseta003-innvestigate.ipynb`

**How do I get the files locally?**

```
git clone https://github.com/stwunsch/kseta-topical-understanding-ml
```

**Access to the JupyterLab server:**
See the e-mail I've sent you.

# Thanks!