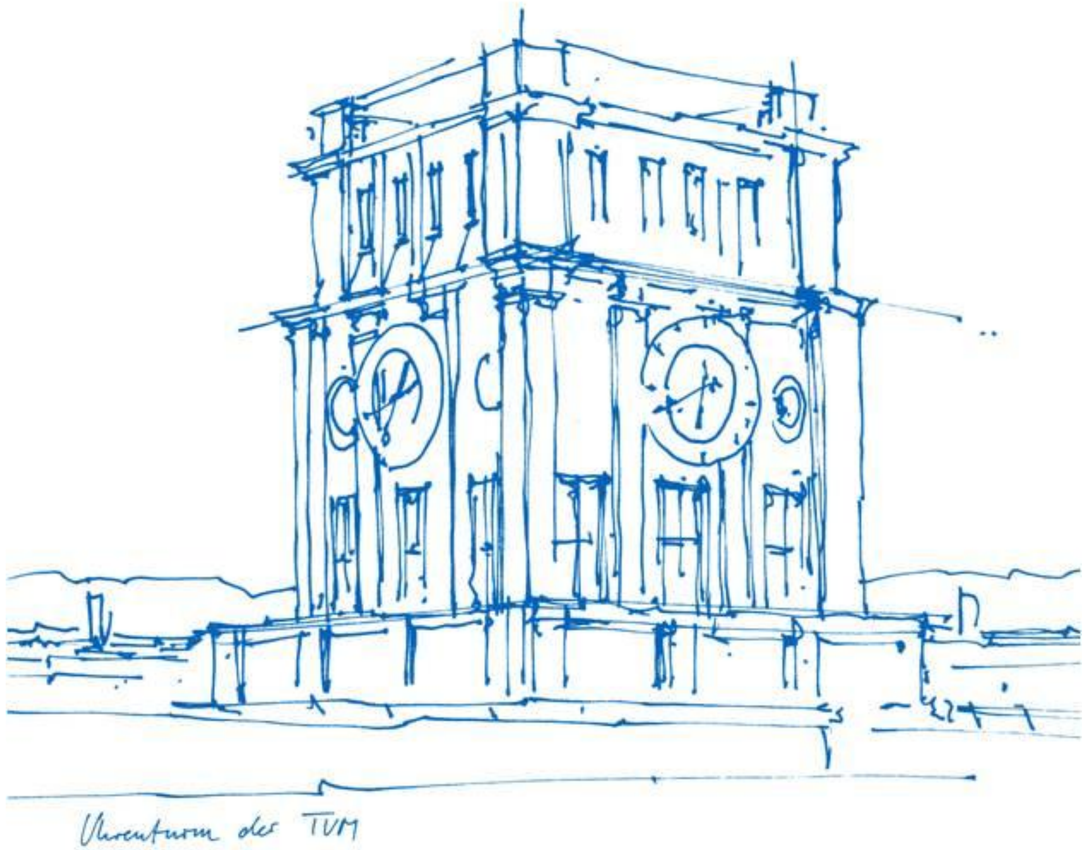
# Trustworthy AI: Explainability, Interpretability and Uncertainty



What are we (and the non-technical audience) looking for in a well-trained model from a performance perspective?

- High utility (given a set of metrics)
- Fair predictions (no subgroup is discriminated against)
- Generalises well (can correctly make predictions on test data)

Data during training

Data during deployment

# Interpretability: Why is it important?

- This is not a new problem, so why now?
  - Complexity and prevalence!
  - Safety is critical
  - Generating knowledge

- Debugging machine learning
  - Why does my model not train?
  - Why does my model exhibit unexpected/unintuitive behaviour?

- To use machine learning responsibly we need to ensure that
  - Our values are aligned
  - Our knowledge is reflected

# How do we evaluate if a particular model is making sensible predictions?

Naive answer: Simply change the features of the data and analyse the response!

(this is actually an example-based explanation)

Try many different inputs and observe how the model behaves (e.g. change your address to Westminster and see how your predicted grades go up [1]).

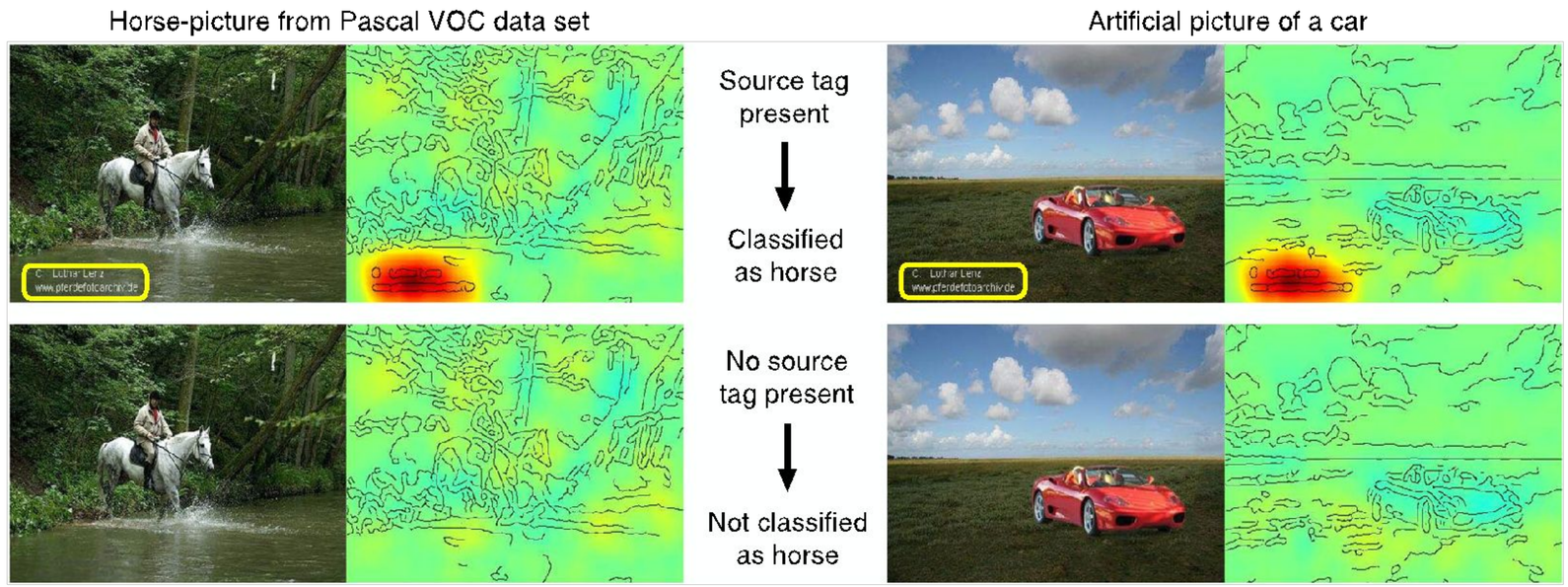
[1] - https://news.sky.com/story/a-levels-thousands-could-miss-out-on-top-university-choices-as-pre-covid-grading-returns-12671738

# Problems with this approach:

- You cannot (effectively) analyse thousands of data points (for certain data types it does not make any sense either)

- It is not clear as to why a specific feature causes a response -> you have to manually evaluate the importance of each feature yourself

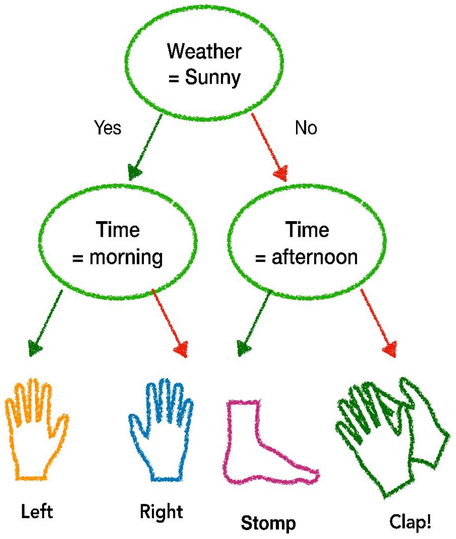
- Thousands of data points with thousands of features (not all of which are observable by humans)

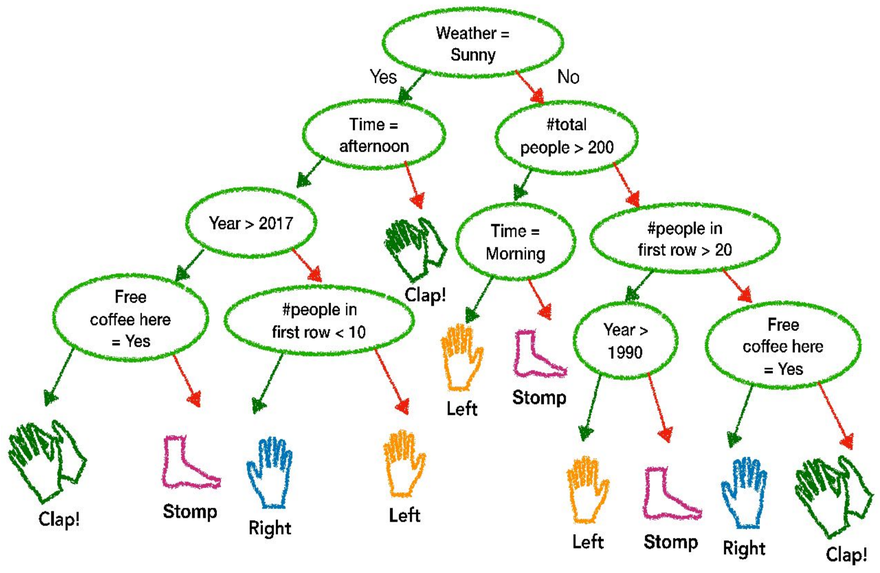


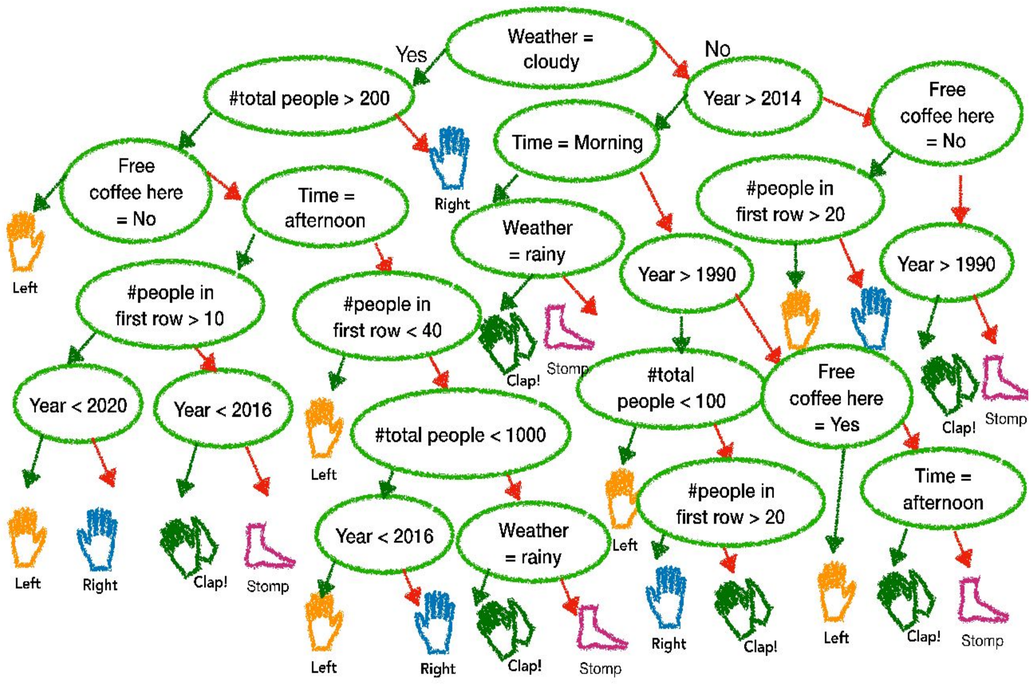
# Interpretable Models

For certain models you can analytically evaluate the relative importance of each individual parameter when it comes to predictions.

Examples of these include *linear* and *logistic regression models, random forests,* etc.



# Model Interpretability Methods

Since most of deep learning approaches rely on complex models that cannot be analysed easily, we need to derive methods that would allow us to interpret their predictions

There exist two broad families of methods to interpret model predictions: The **model-agnostic methods** and the **example-based explanations**.

The model-agnostic methods are further split into **global** and **local** methods.

# Interpretability: Example-Based Explanations

These methods select particular instances of the dataset to explain the behaviour of either the ML model or the underlying data distribution.

Example-based explanations are ‘almost’ model-agnostic:

They can be applied to any ML model, but they explain the prediction by selecting instances of the dataset and not by creating summaries of features

Examples: *adversarial samples, prototypes, influential instances,* etc.

# Some samples can be more challenging than the rest

# Interpretability: Global Model-Agnostic Methods

These describe the *'average'* behaviour of a trained machine learning model.

Global methods are often expressed as the expected values derived from the distribution of the data.

Useful for understanding the general behaviour of the model or general mechanisms in the data.

N.B. Most of these are available in *scikit-learn*

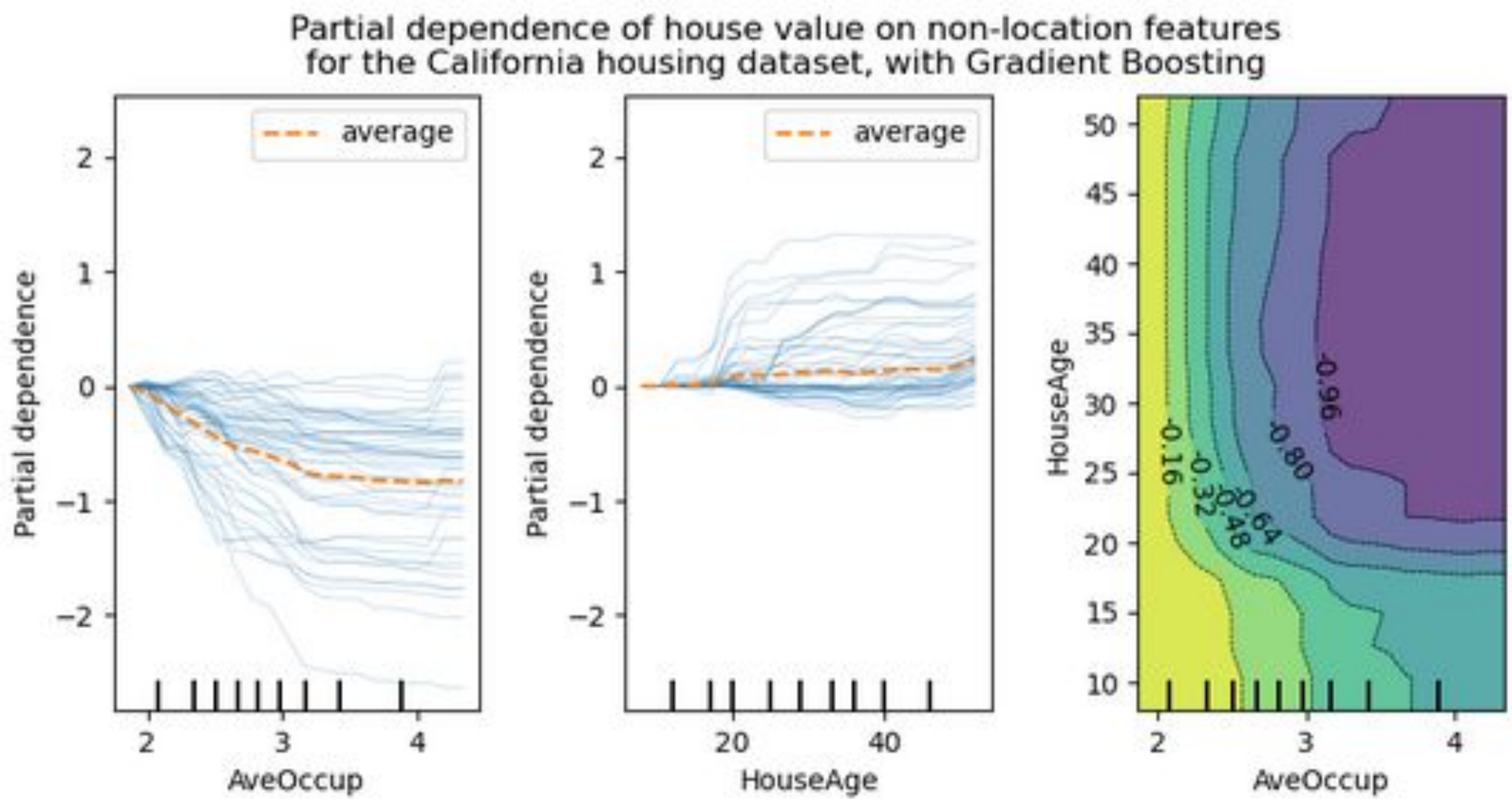
# Interpretability: Partial Dependence Plot

PDP shows the marginal effects one or two features have on the predicted outcome of a model, marginalising over the values of all other input features.

Intuitively (from the interpretability book): *“Let me show you what the model predicts on average when each data instance has a certain value for that feature. I ignore whether this value makes sense for all data instances.”*

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Limited (one or two) number of features |
| Easy to implement | Assumes independence of features |

# Interpretability: Partial Dependence Plot



# Interpretability: Accumulated Local Effects Plot

ALE is similar to PDP, but instead of relying on the average of predictions, this plot relies on the difference in predictions. Additionally, ALE uses the conditional probability instead of the marginal probabilities.

Intuitively (from the interpretability book): *“Let me show you how the model predictions change in a small ”window” of the feature for data instances in that window.”*

| **Advantages** | **Disadvantages** |
|---|---|
| Unbiased | Cannot be used if features are correlated |
| Fast to compute | Interval number trade-off |
| Prediction function can be decomposed | Much more complex implementation (than of PDP) |

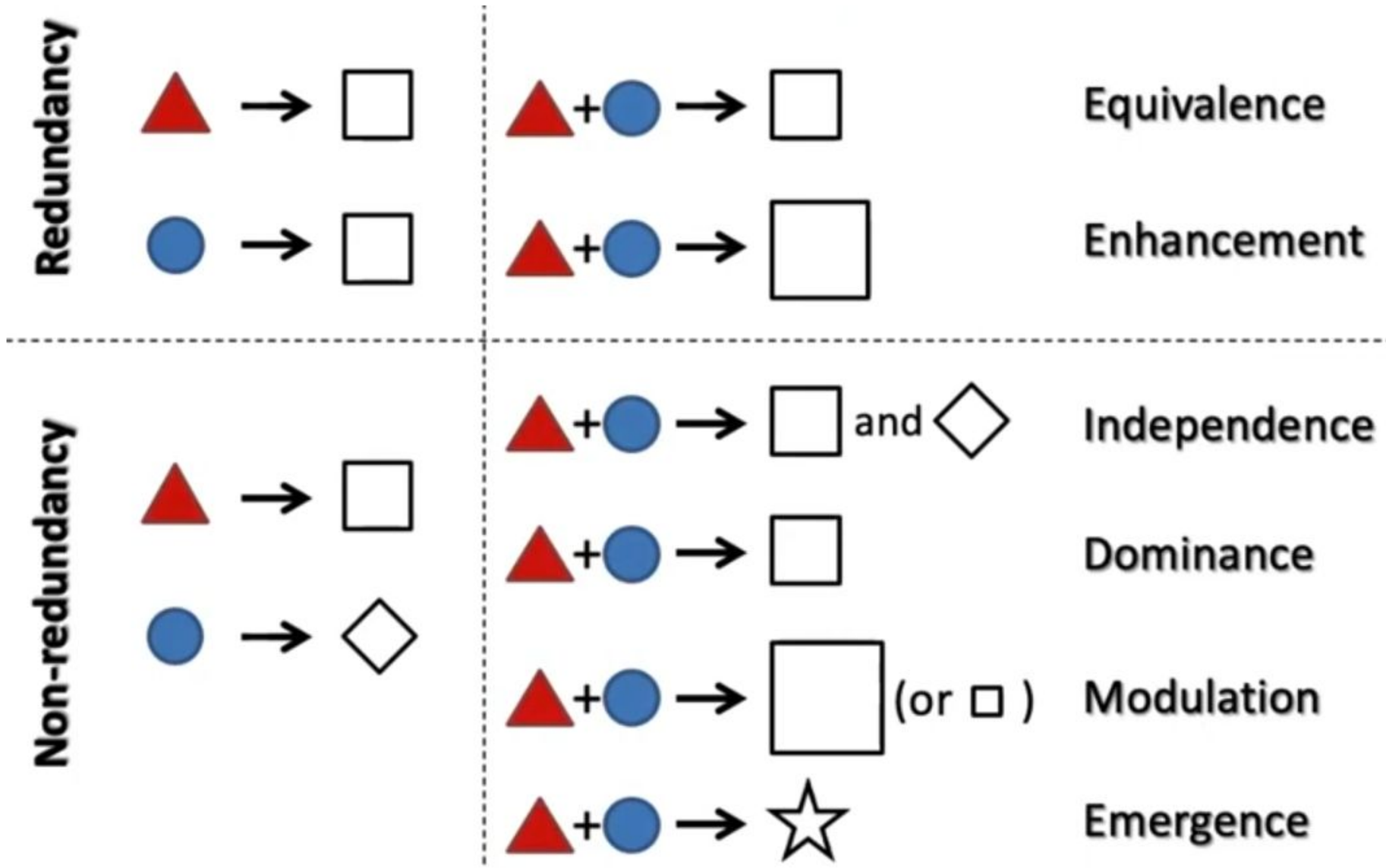
# Interpretability: Feature Interaction

These methods measure the compounded effect of a small number of features on the prediction.

There exist various methods to measure the strength of the interaction between the features, one of the most commonly used ones being the H-statistic, which measures how much variation the prediction exhibits upon the interaction of the features.

| Advantages | Disadvantages |
| --- | --- |
| Meaningful interpretation | Computationally expensive |
| Deimensionless, so can be compared across features or models | Computation involves estimating marginal distributions, resulting in large variance for small samples |
| Detects all kinds of interactions | |

# Interpretability: Feature Interaction



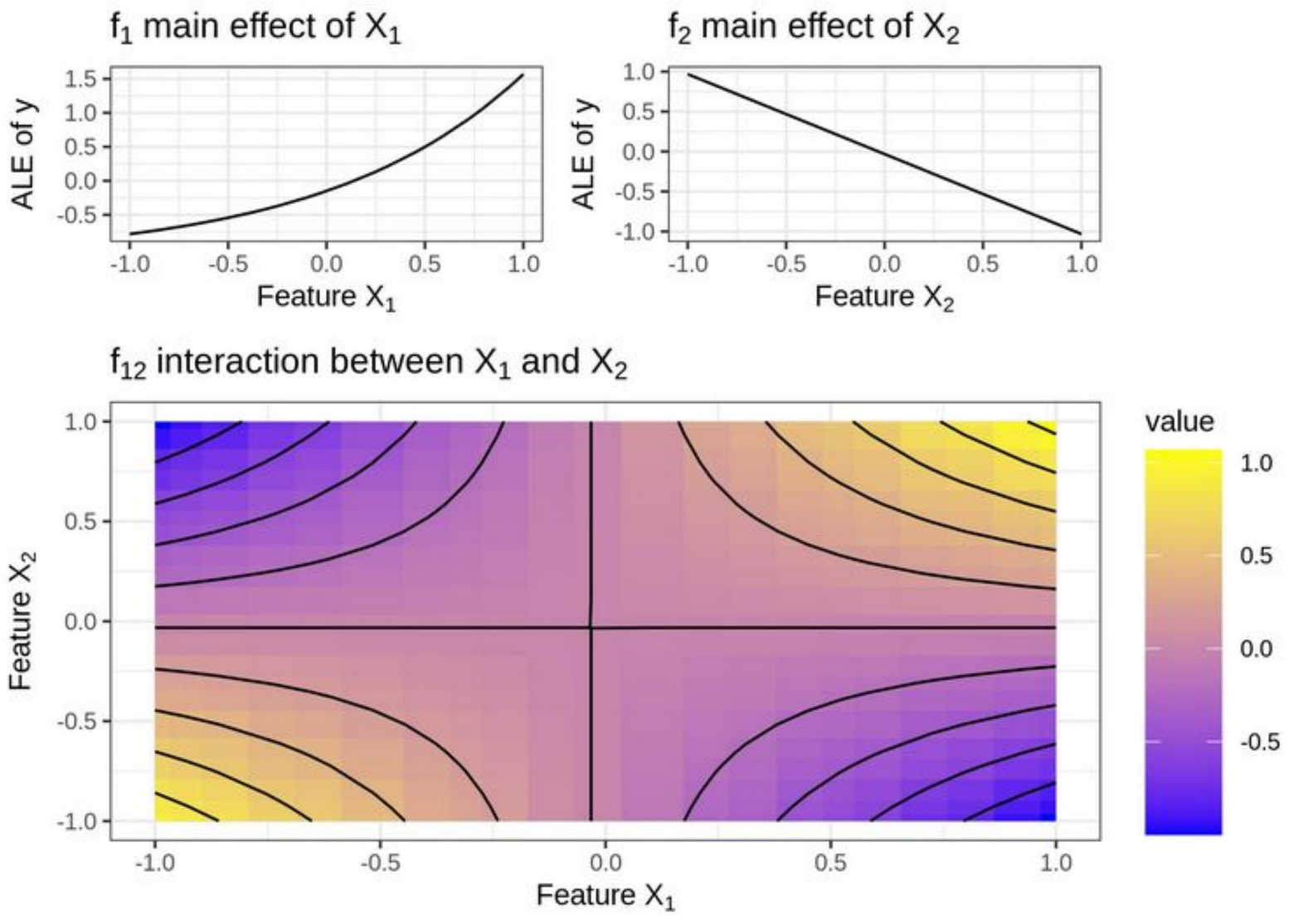
# Interpretability: Functional Decomposition

The approach involves breaking down more complex functions into a number of simpler individual functions and analysing them separately such that the initial function can be reconstructed exactly from its constituent parts.

The complication here is similar to the one of feature interaction: It is often the case that the effects of individual function components are not additive.

| Advantages | Disadvantages |
|---|---|
| Gives theoretical justification for decomposing high-dimensional models into individual interactions | Difficult to visualise high-dimensional components |
| Provides a better understanding of other interpretability methods | More appropriate for tabular data |
| | If features are correlated, the results could be misleading |

# Interpretability: Functional Decomposition



# Interpretability: Permutation Feature Importance

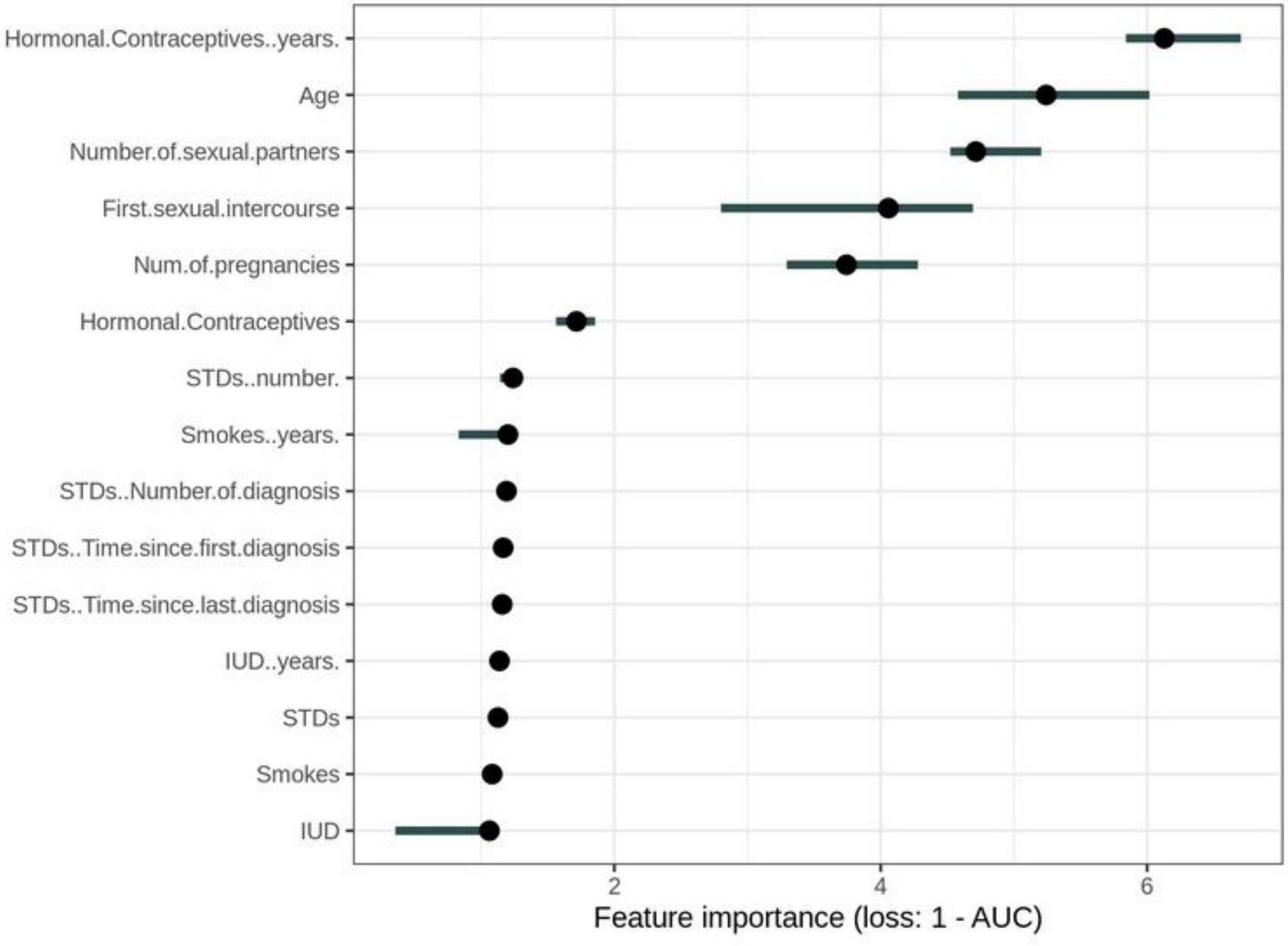
We measure the importance of a feature by calculating the increase in the model's prediction error after permuting the feature.

A feature is "important" if perturbing its values increases the model error, because in this case the model relied on the feature for the prediction.

A feature is "unimportant" if perturbing its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction.

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Linked to the error of the model |
| Considers all interactions with other features | Results may vary because of feature shuffling |
| Does not require re-training | Adding a correlated feature can decrease the importance of the associated feature |

# Interpretability: Permutation Feature Importance



# Interpretability: Surrogate Model

This method involves creating a new (often interpretable) model, which would approximate the predictions of the target (often a black-box) model.

This allows to analyse the target behaviour of a much simpler model type that approximates the behaviour of the more complex model.

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Only draws conclusions about the model and not the data |
| Very flexible with a variety of appropriate approximations | Surrogate model can approximate specific behaviour behaviours rather than the overall one |
| Can identify a ‘cheaper’ alternative for the model | Not always easy to select an appropriate surrogate |

# Interpretability: Local Model-Agnostic Methods

These methods explain individual predictions, which can (but do not have to) be extrapolated to the overall behaviour of the target model.

Can often be used for valuation of data.

# Interpretability: Local Surrogate (LIME)

This method explains a prediction by replacing the complex non-convex model with a locally interpretable surrogate.

Surrogate models are trained to approximate the predictions of the underlying model. Instead of training a global surrogate model, LIME focuses on training local surrogate models to explain individual predictions.

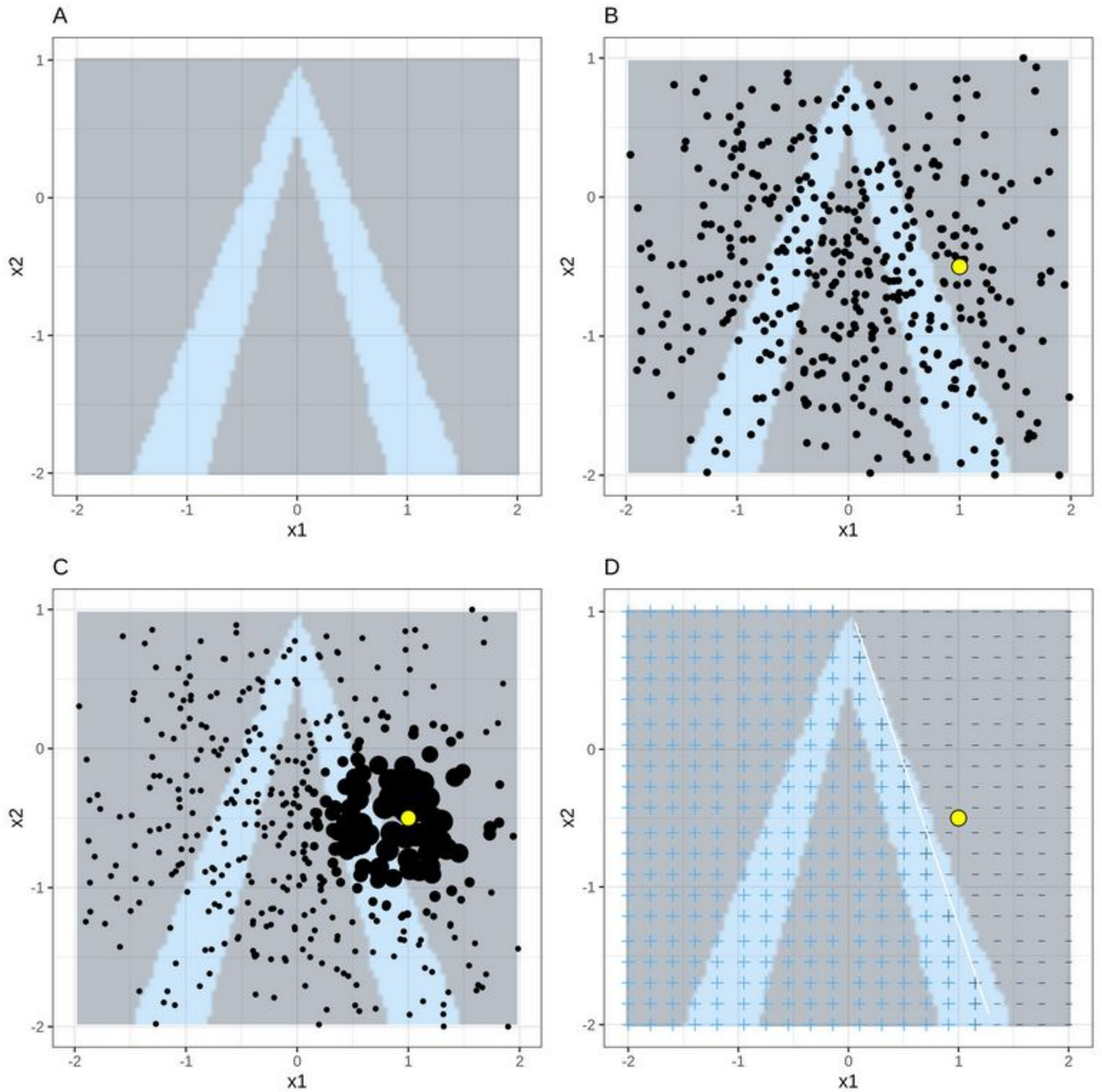
LIME generates a new dataset consisting of perturbed samples and the corresponding predictions of the black box model. On this new dataset LIME then trains an interpretable model, which is weighted by the proximity of the sampled instances to the instance of interest.

| **Advantages** | **Disadvantages** |
|---|---|
| Works for tabular, text and image data | Neighborhood is very hard to define |
| Easy to use | Can be unstable |

# Interpretability: Local Surrogate (LIME)

The recipe for training local surrogate models:

- Select your instance of interest for which you want to have an explanation of its black box prediction.
- Perturb your dataset and get the black box predictions for these new points.
- Weight the new samples according to their proximity to the instance of interest.
- Train a weighted, interpretable model on the dataset with the variations.
- Explain the prediction by interpreting the local model.



# Interpretability: Counterfactual Explanations

Prediction is explained by examining which features need to be changed to achieve a desired prediction.

In interpretable machine learning, counterfactual explanations can be used to explain predictions of individual instances. The “event” is the predicted outcome of an instance, the “causes” are the particular feature values of this instance that were input to the model and “caused” a certain prediction.

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Rashomon effect: multiple counterfactuals can exist for a single instance |
| Does not require access to the model or the data | |
| Easy to implement | |

# Interpretability: Individual Conditional Expectations

ICE (similarly to PDP) shows the dependence between the target function and an input feature of interest.

An ICE plot shows the dependence of the prediction on a feature for each sample separately (unlike PDP, which shows the average effect of an input feature).

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Limited (one or two) number of features |
| Easy to implement | Assumes independence of features |

# Interpretability: Scoped Rules

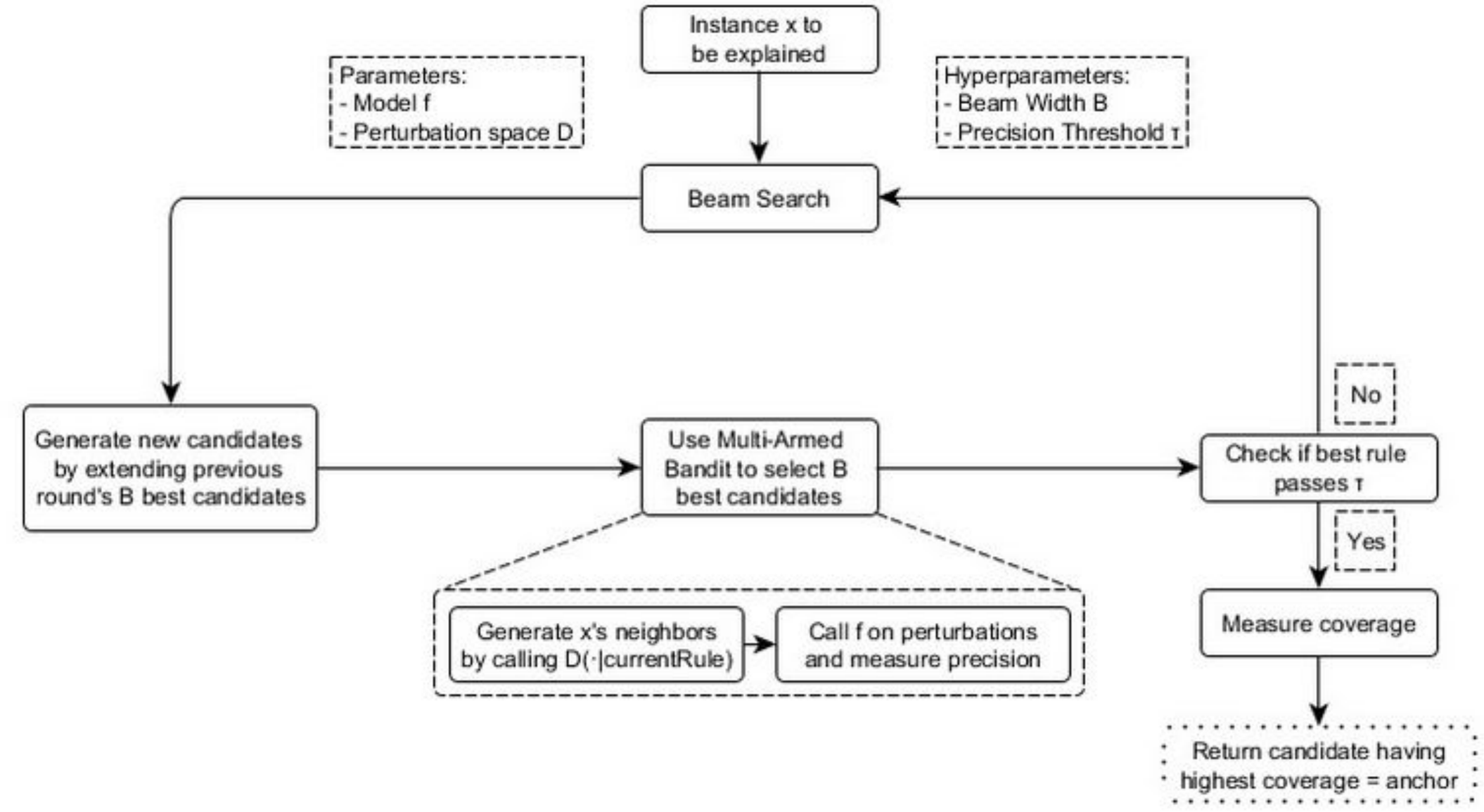
This method imposes ‘rules’, whose feature values anchor a prediction.

The anchors approach deploys a *perturbation-based* strategy to generate *local* explanations for predictions of black box machine learning models.

However, instead of surrogate models used by LIME, the resulting explanations are expressed as easy-to-understand *IF-THEN* rules, called *anchors*.

| **Advantages** | **Disadvantages** |
|---|---|
| Very intuitive | Can be very sensitive to the initialisation |
| Works when model predictions are non-linear | Requires many calls to the model |

# Interpretability: Scoped Rules



# Interpretability: Shapley Additive Explanations (SHAP)

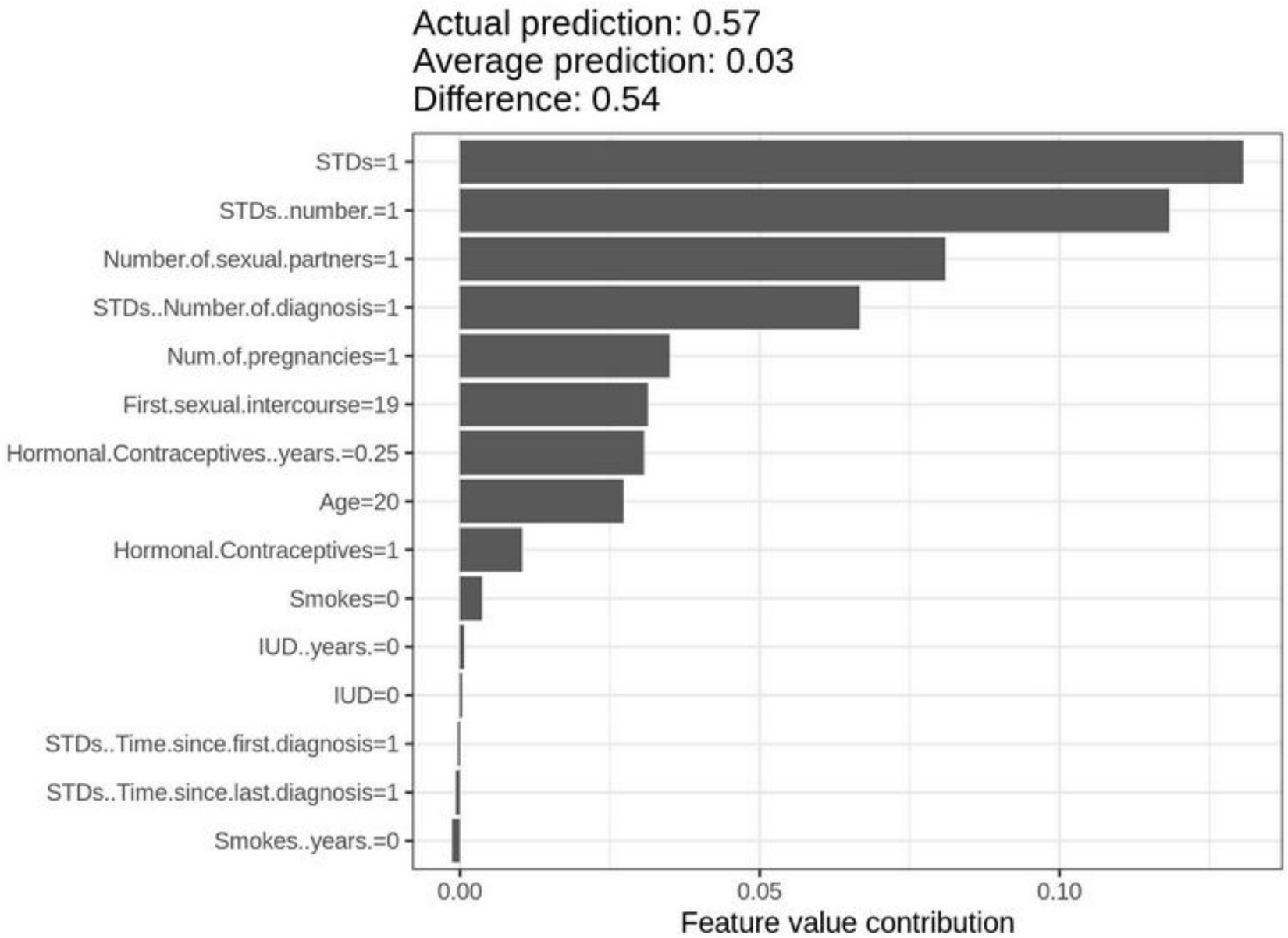
This is derived from the notion of Shapley Values, which assigns the relative importance on the prediction outcome to individual features.

SV is a method for assigning payouts to players depending on their contribution to the total payout. Players cooperate in a coalition and receive a certain profit from this cooperation.

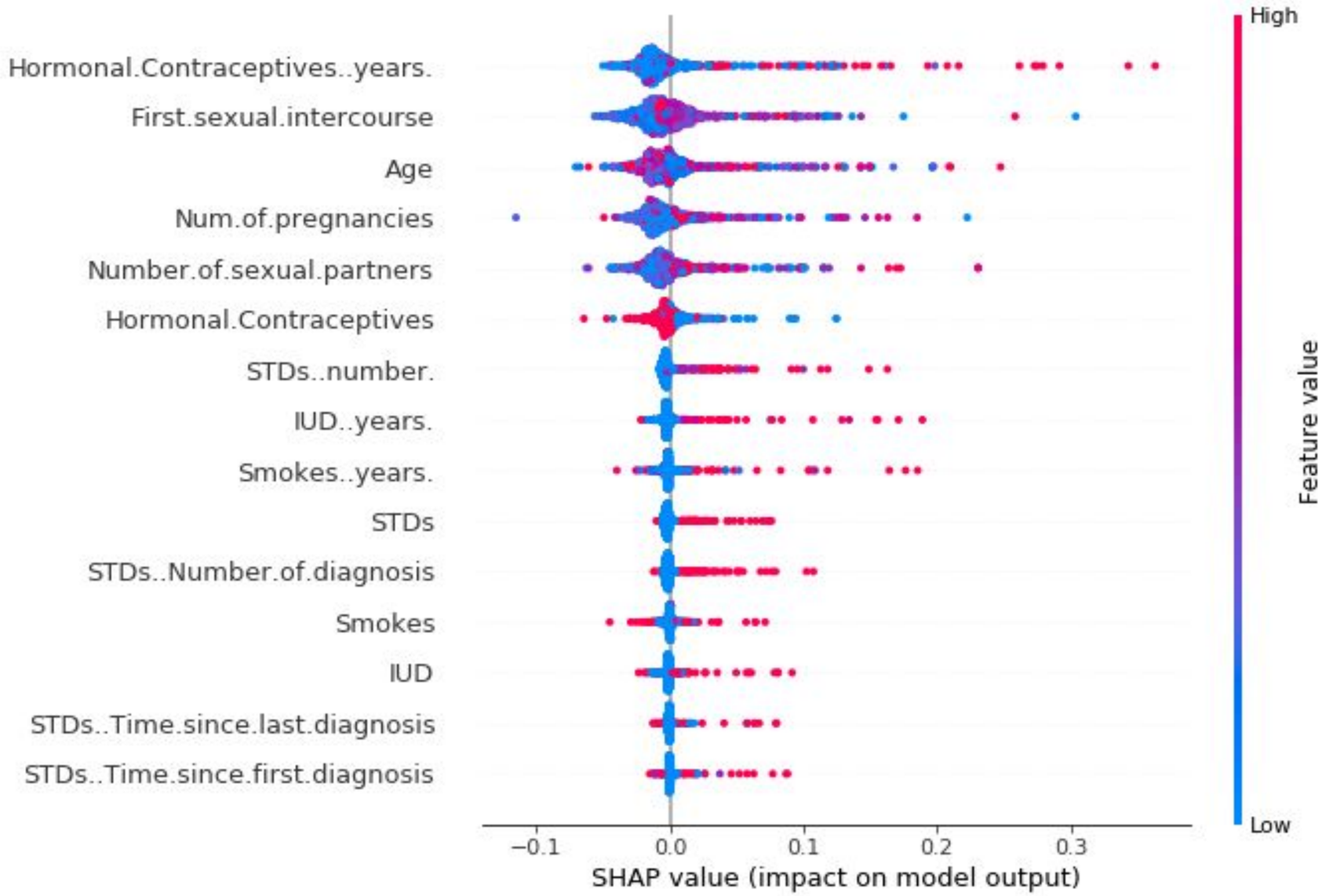
The “game” is the prediction task for a single instance of the dataset. The “gain” is the actual prediction for this instance minus the average prediction for all instances. The “players” are the feature values of the instance that collaborate to receive the gain (= predict a certain value)

| **Advantages** | **Disadvantages** |
|---|---|
| The difference between the prediction and the average prediction is fairly distributed among the features of the instance | Very expensive to compute<br>Requires access to the entire dataset |
| Allows contrastive explanations | Can be misinterpreted |

# Interpretability: Shapley Additive Explanations (SHAP)



# Interpretability: Shapley Additive Explanations (SHAP)



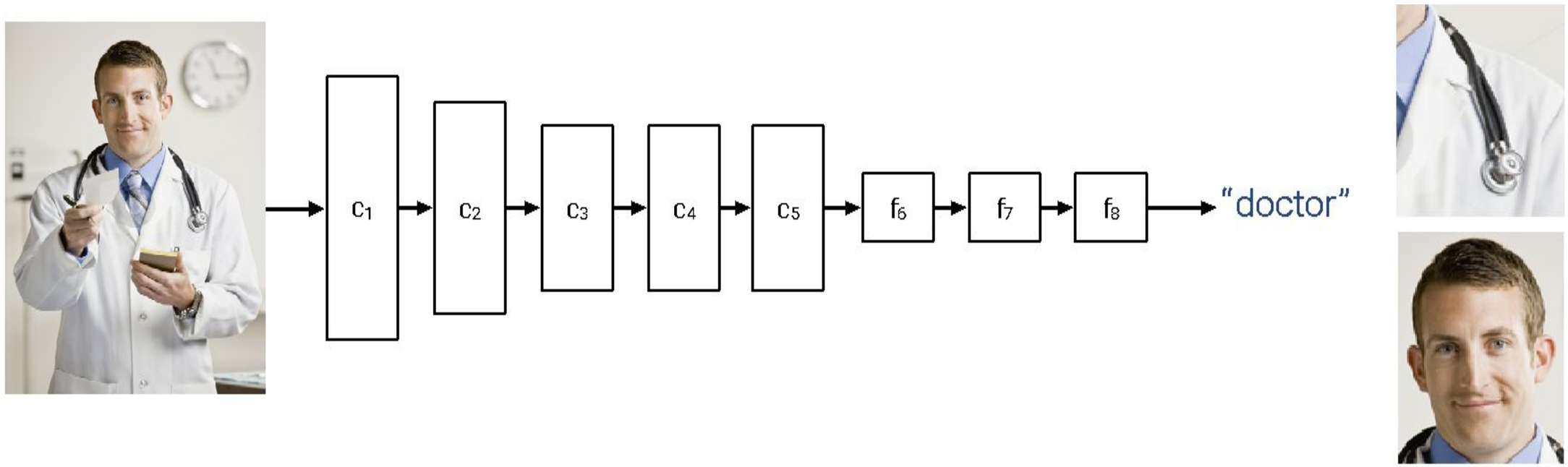
# Issues with most of these methods

These can be used really effectively across many architectures and datasets, however:

- Very often they do not scale: Either computationally (e.g. SV) or in terms of their explanatory powers (e.g. LIME).

- Certain architectures have millions or even billions of individual parameters

- Additionally, neural networks learn features and concepts in their hidden layers and we need special tools to uncover them.



# Explainability: Gradient-Based Methods

In contrast to methods above, in DL we are able to utilise the gradient of the model to implement interpretation methods that are more computationally efficient than model-agnostic methods that look at the model “from the outside”.

E.g. *learned features, saliency maps, adversarial samples* etc.

# Explainability: Learned Features

There are two popular methods that can be used: Feature visualisation and Network dissection.

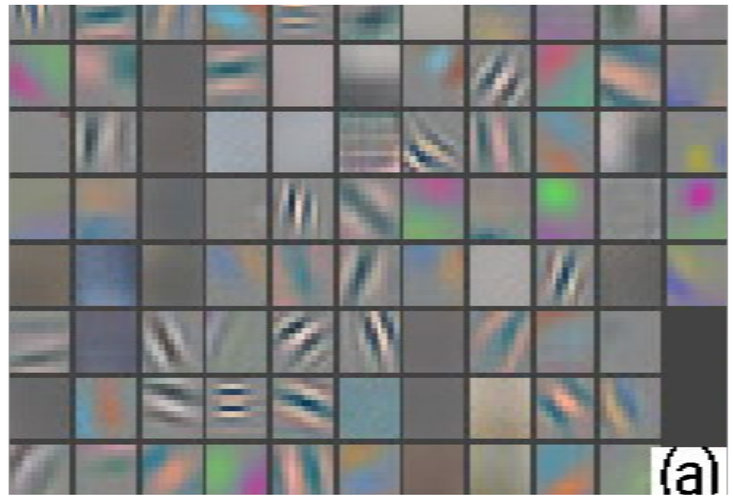
- FV visualises the learned features through activation maximisation.
- ND links highly activated areas of CNN channels with human concepts (e.g. texture extraction).

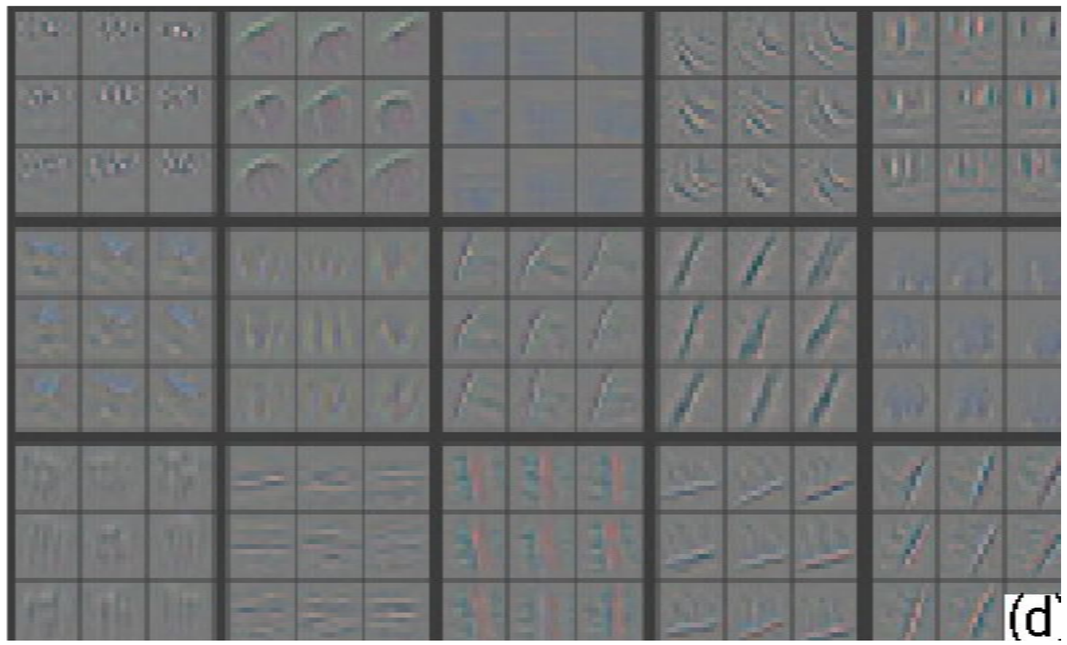
FV aims to find a specific input that maximises the activation of a network unit responsible for a feature(s) of interest.

There is a trade-off: Visualising individual neurons is infeasible for million-parameter networks, so feature maps or layers are preferred.

# Explainability: Feature Visualisation

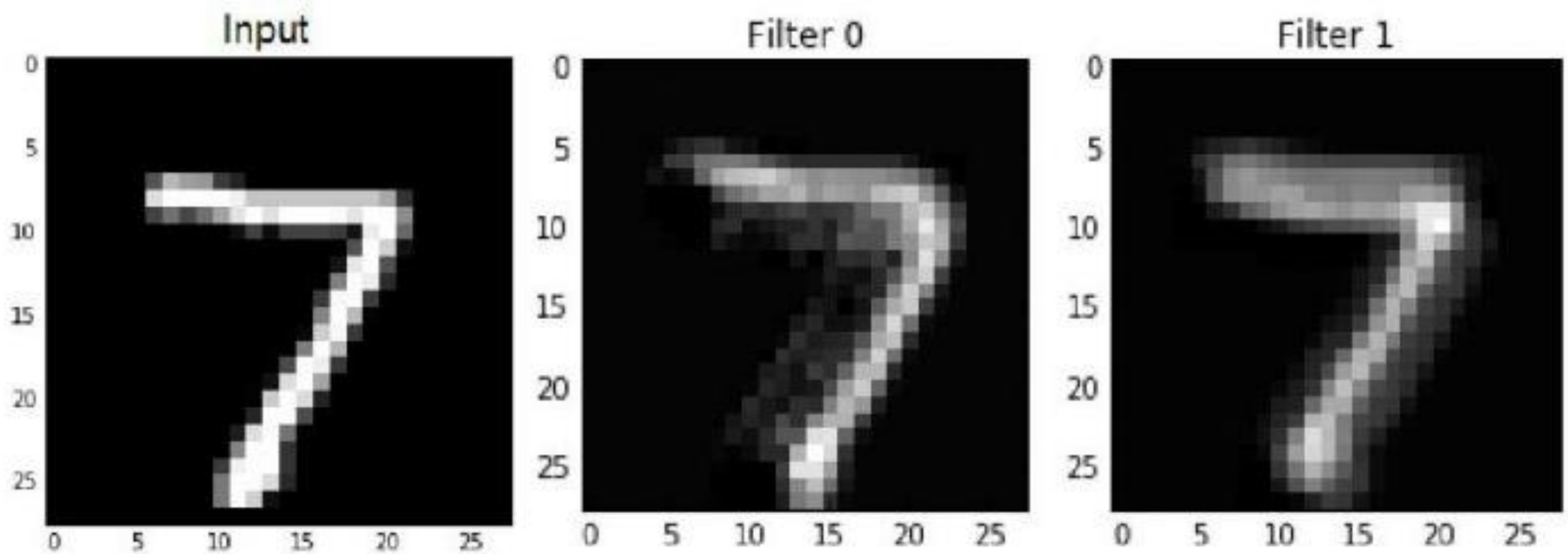
- Direct visualization of filters
- Easy to implement
- Limited practical value
  - First layers are easy to interpret (mostly low-level features)
  - Higher layers are more difficult to interpret (non-interpretable features)



# Explainability: Feature Visualisation

- Problem: Visualization of filters has limited value
- Solution: Instead visualize activations generated by kernels
  - Strong response: Feature is present
  - Weak response: Feature is not present
  - Easy to implement
  - Easy to interpret for early layers

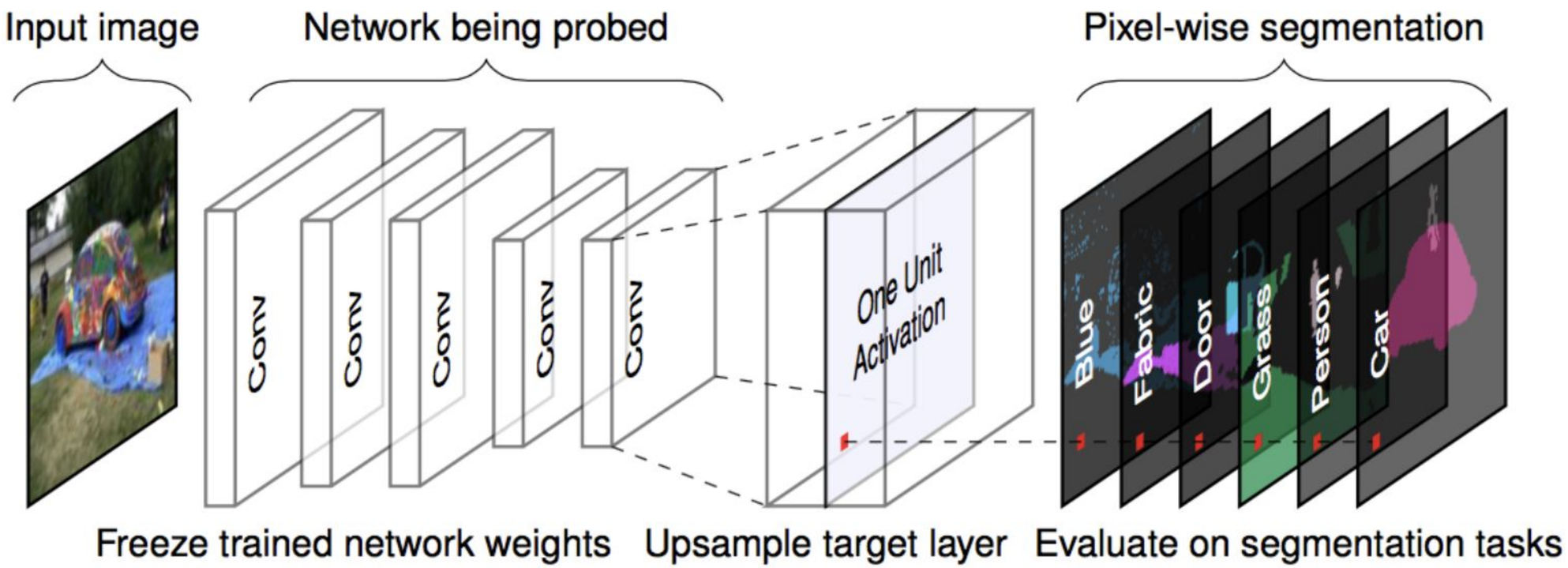


# Explainability: Learned Features

## Summary of the network dissection framework:

1. Get images with human-labeled visual concepts, from stripes to skyscrapers.
2. Measure the CNN channel activations for these images.
3. Quantify the alignment of activations and labeled concepts.

The following figure visualizes how an image is forwarded to a channel and matched with the labeled concepts.



# Explainability: Saliency Maps

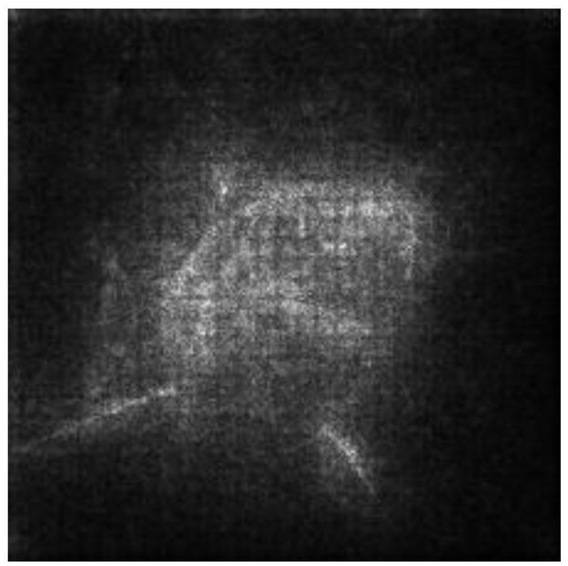
This is a special case of a more general feature attribution method (such as SHAP or  LIME) specifically designed for easier interpretability of the image features.

These highlight the regions of interest of the image which are similar to those a human would concentrate their attention on.

Multiple maps can be constructed wrt each individual area of interest (color, orientation etc.)

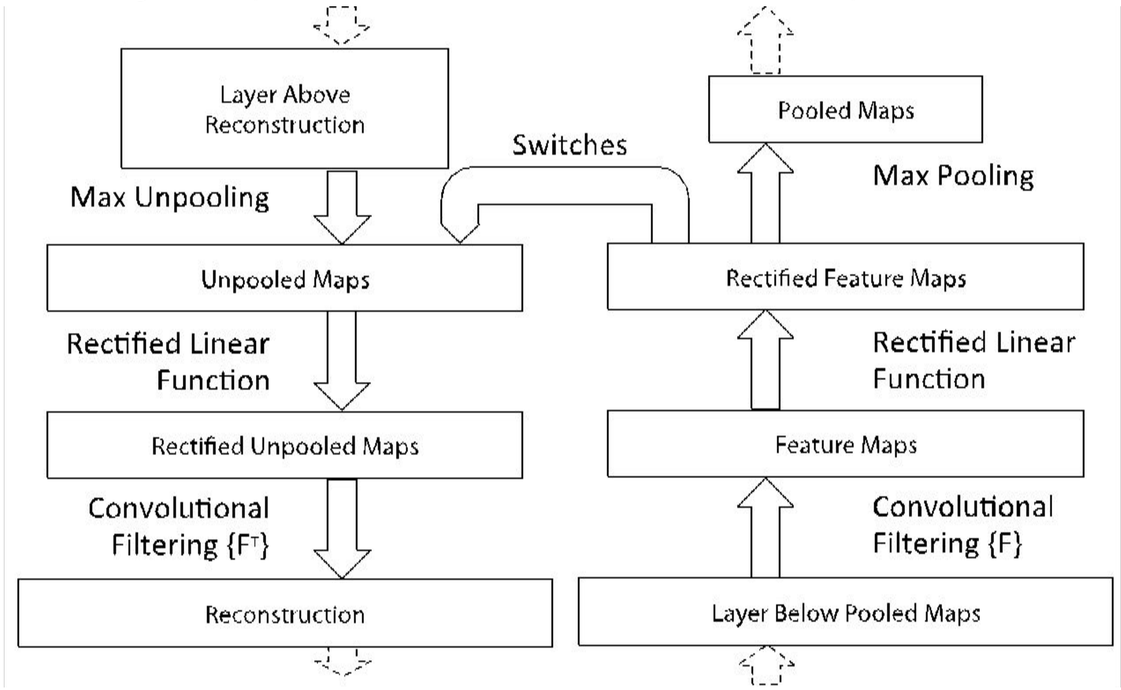
There exist various implementations for saliency maps:

- Deconvolutional approach
- Gradient-based approach
- Guided backpropagation approach

| ImageNet original image (great white shark) | Corresponding saliency map (great white shark) |
| --- | --- |
|  |  |

# Explainability: Saliency Maps

- **DeconvNet**
  - Given a trained network and an image
  - Chose activation at one layer (set all others to zero)
  - Invert network

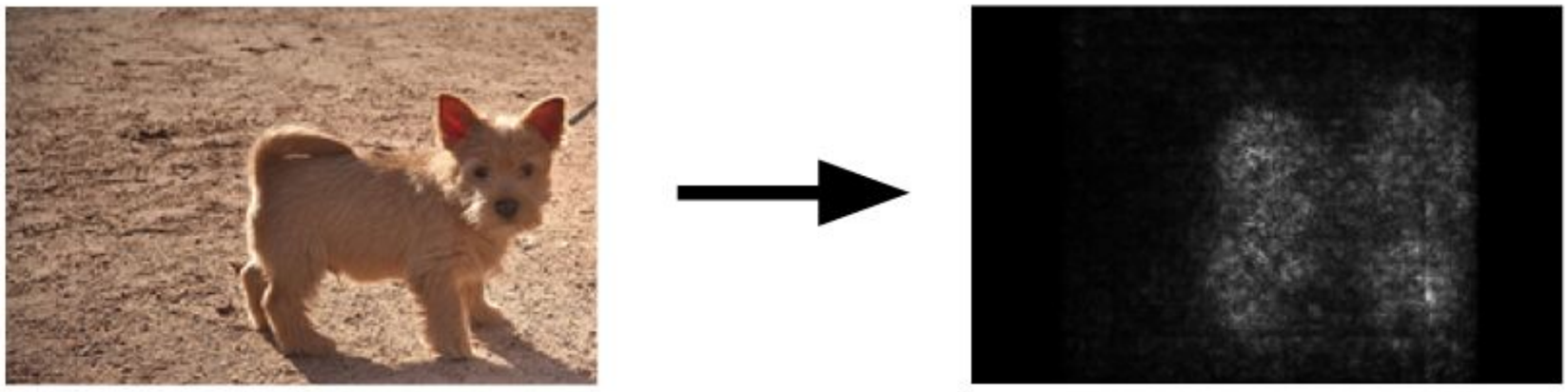



  - No training involved
  - Backward pass in network is almost identical to backpropagation (apart from ReLUs)

# Explainability: Saliency Maps

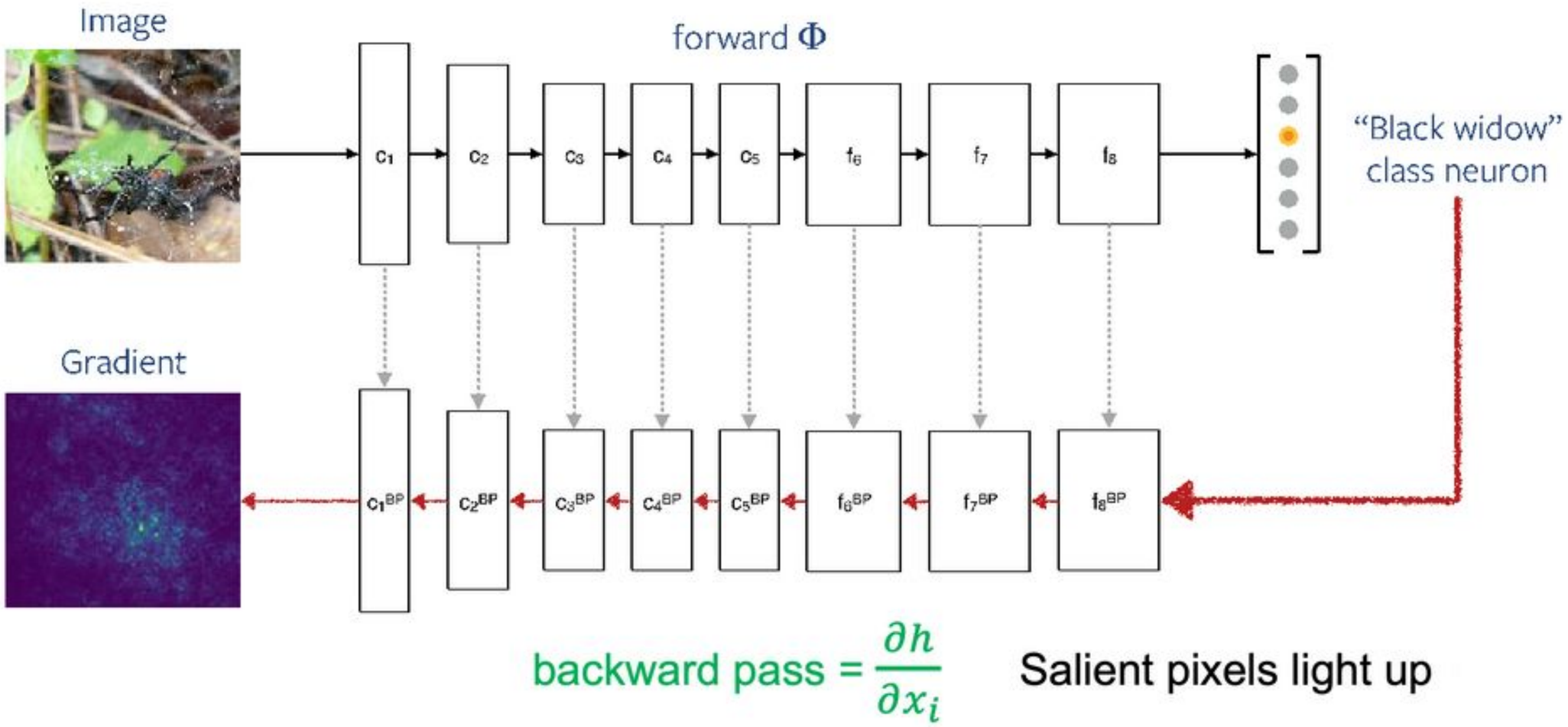
- Question:
  - Which pixels are most significant to a neuron?
  - How would they need to change to most affect the activation of the neuron?
- Solution:
  - Use back propagation but differentiate activation with respect to input pixels, not weights

weights of the network are fixed $\frac{\partial h}{\partial x_i}$ ~~$\frac{\partial f}{\partial \theta_i}$~~



# Explainability: Saliency Maps

- **Gradient (backprogagation)**
  – Define loss as activation of arbitrary neuron in any layer (last layer is most interesting)



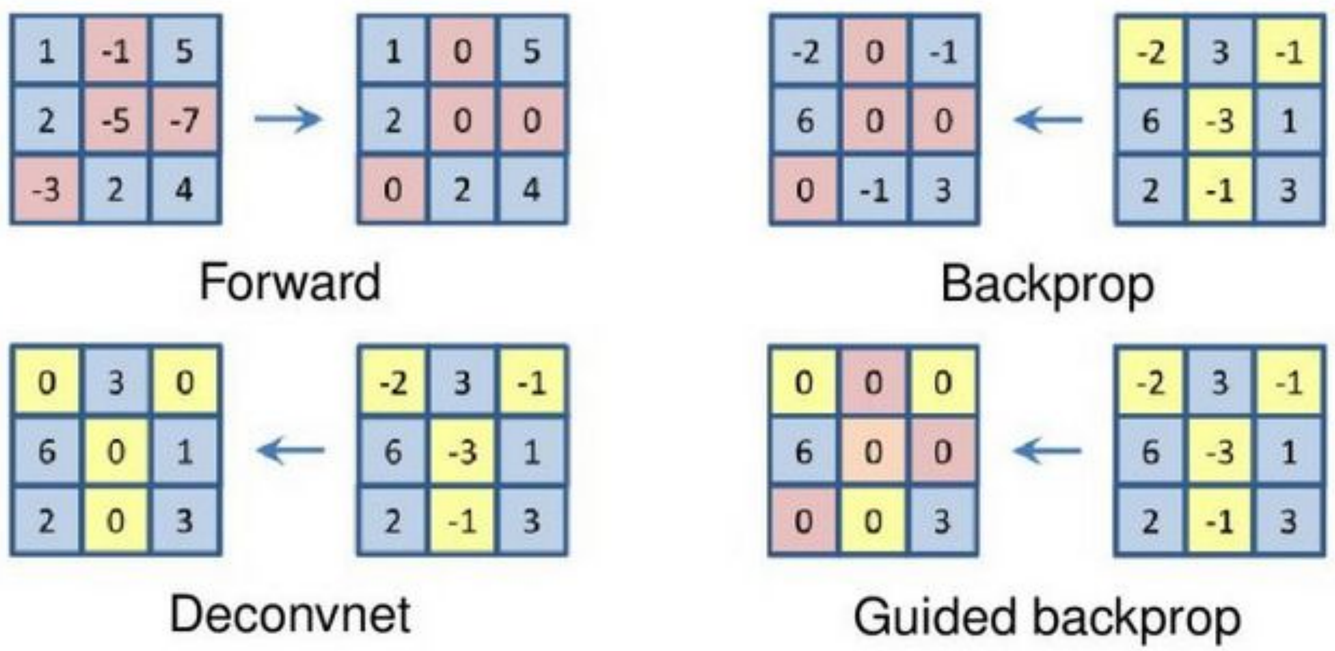

backward pass = $\frac{\partial h}{\partial x_i}$ Salient pixels light up

# Explainability: Saliency Maps

- **Guided backpropagation**
  - Improve results by “guiding” the backpropagation process
  - Idea:
    - Positive gradients = features the neuron is interested in
    - Negative gradients = features the neuron is not interested in

➜ Set all negative gradients in the backpropagation to zero

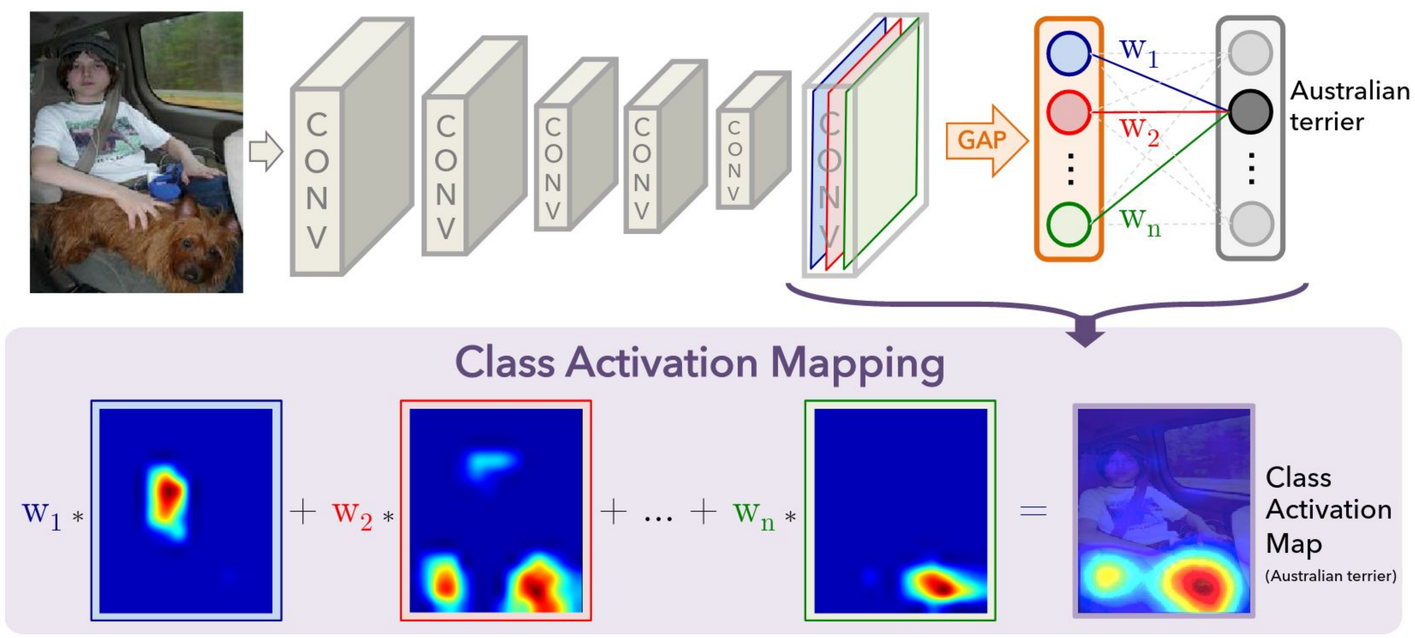
- Propagating through the ReLU:



# Explainability: CAM (and its derivatives)

A newer version of saliency attribution was proposed, namely the Class Activation Maps (CAM), which, as the name suggests, highlight the areas of the image responsible for the prediction of a specific class. This allows to localise the individual parts of the input image that lead to a specific prediction.

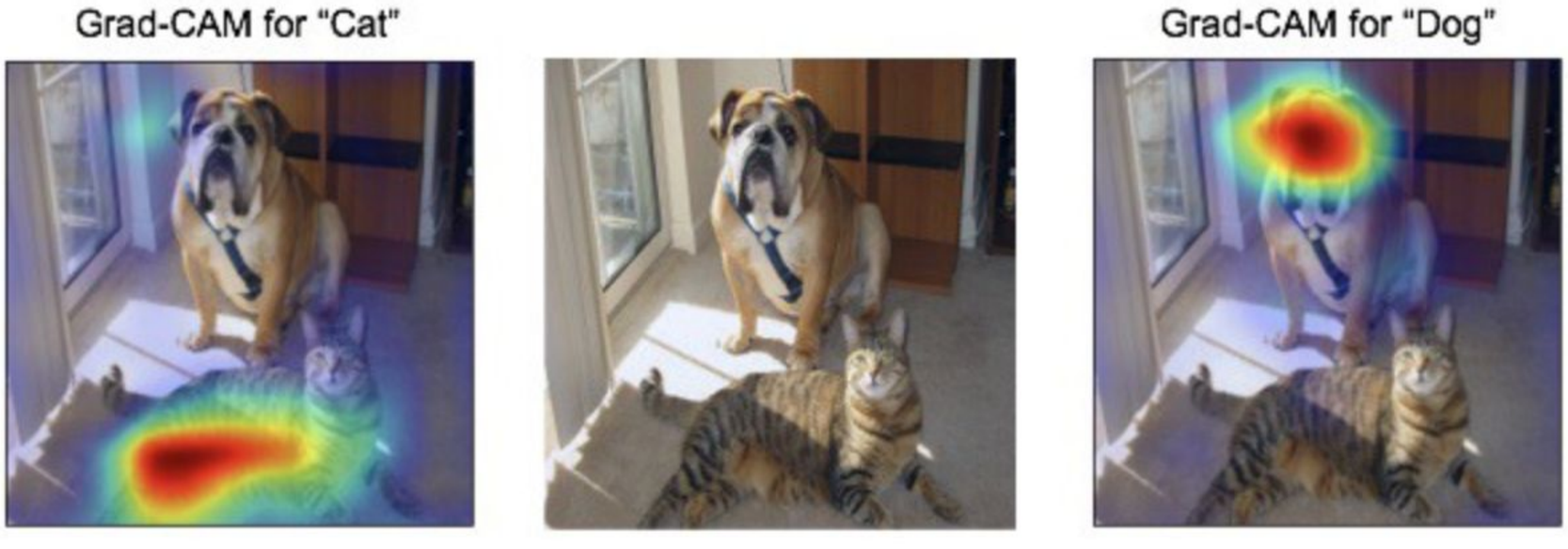
This is done by employing the global average pooling on the convolutional feature maps and feeding this output as the features into the fully-connected layers.



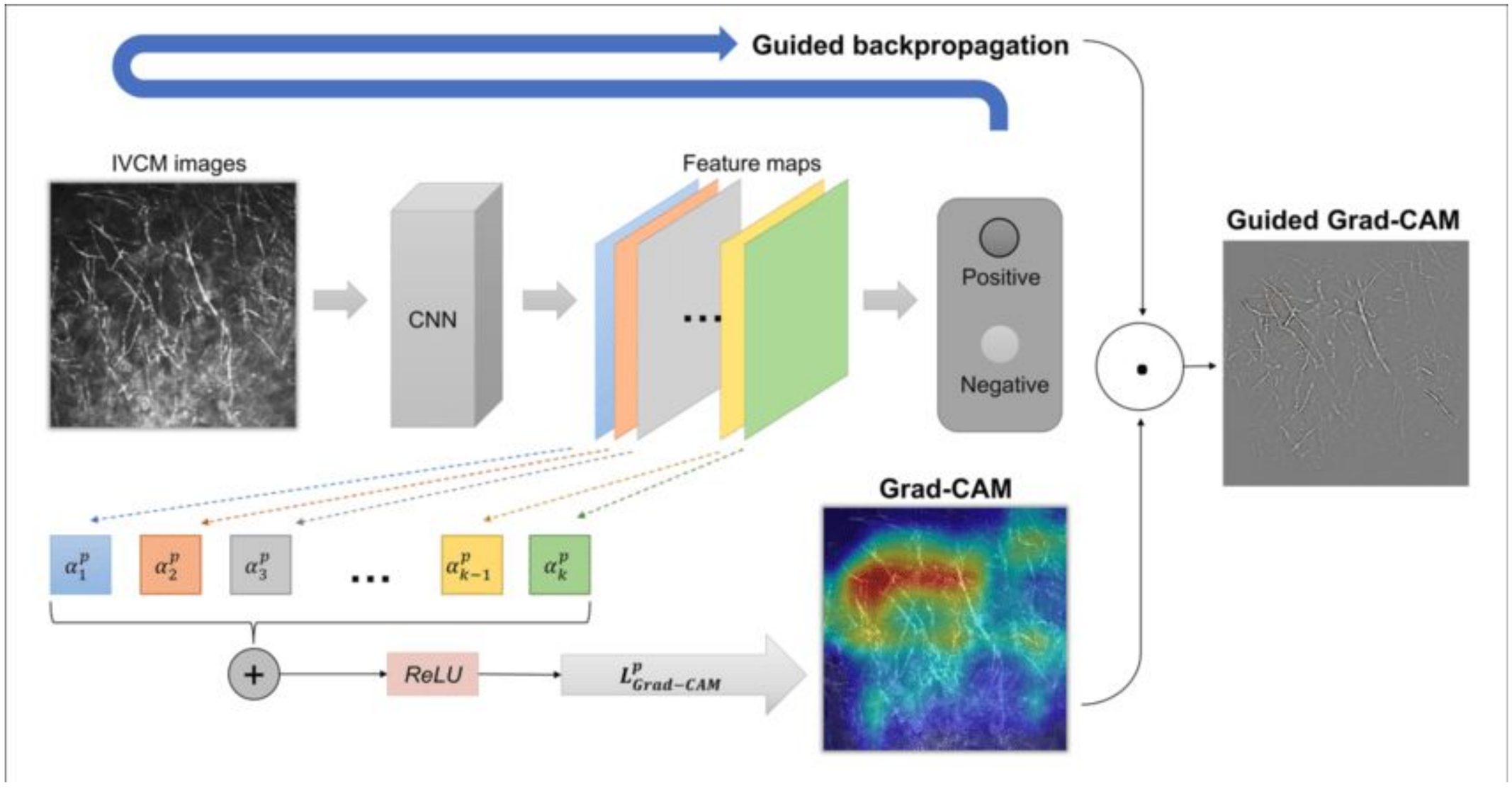
# Explainability: CAM (and its derivatives)

A gradient-based improvement was later proposed, namely the Gradient-weighted Class Activation Mapping (GradCAM). Unlike the original method, this relies on the gradient information to attribute the ROI in the image.

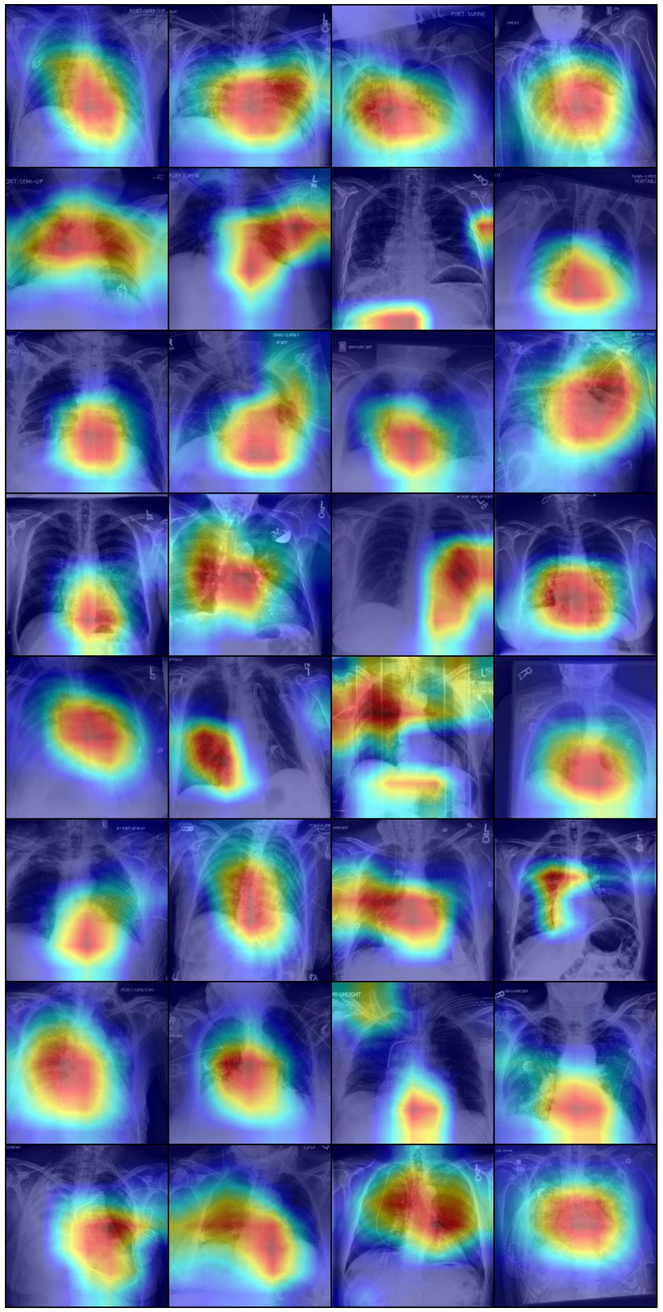
Specifically, GradCAM uses the gradient information flowing into the last convolutional layer (wrt to a feature map activations of a desired class). These gradients are then global-average-pooled to obtain the neuron importance weights. More detailed explanation can be found in the paper.

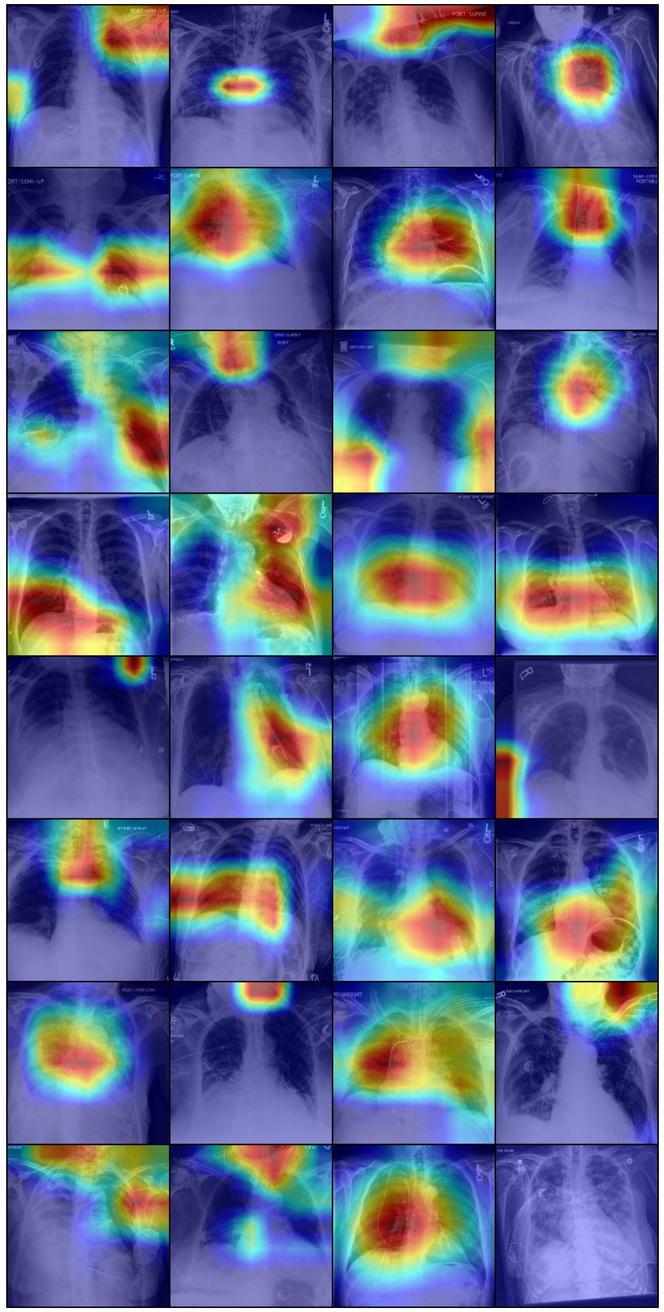


# Explainability: CAM (and its derivatives)



# Explainability: CAM (and its derivatives)





# Explainability: Adversarial Samples

This method loosely describes a number of techniques, which aim at producing samples that are ‘problematic’ for the target model.

There are various methods that allow generation of such samples, but the overall idea is the addition of invisible perturbations to the target image, such that the image ‘looks’ identical to a human observer, yet the target model makes an incorrect prediction on that data point.

These are using the same idea as the learnt features, but try to maximise the activation of the unit for an incorrect class (which can be random or chosen).

# Explainability: Adversarial Samples (FGSM)

Key idea:

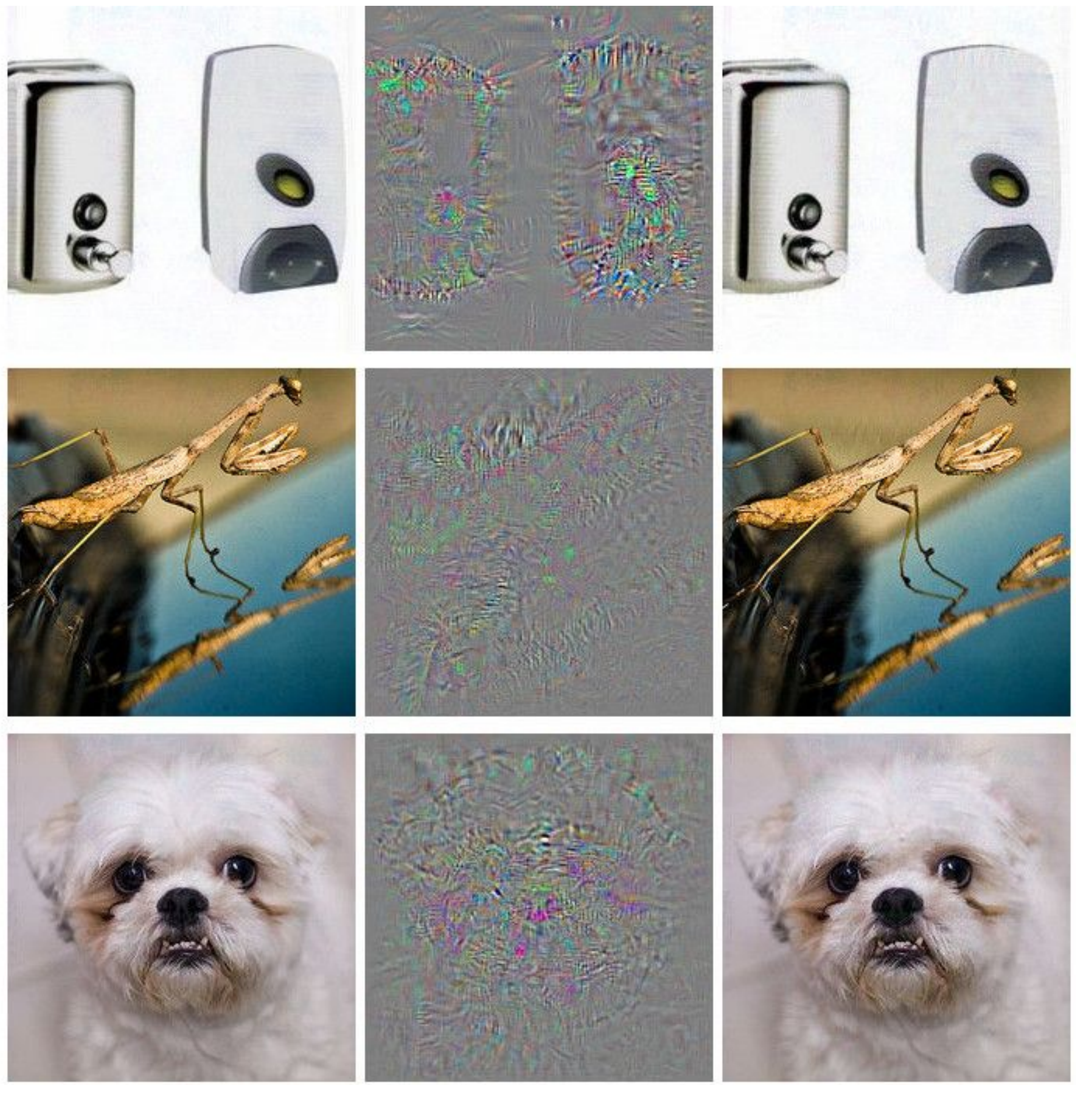
- Perform gradient descent in order to maximize the loss (the goal of adversarial attack).
- Consider the input image $x$ to be a trainable parameter and compute the gradient with respect to the input image to create a perturbation.

$$\eta = \epsilon \cdot \text{sign}\big(\nabla_x \mathcal{L}(\theta, x, y)\big)$$

- An adversarial example can be created as:

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

# Explainability: Adversarial Samples



# Explainability: Influential Instances

An influential instance is a data instance whose removal has a strong effect on the trained model. The more the model parameters or predictions change when the model is retrained with a particular instance removed from the training data, the more influential that instance is.

There are two ways to identify these instances:

- **Deletion diagnostics** evaluates different versions of the training dataset w/out the individual samples to determine the relative predictive importance of these samples (in some ways similar to SHAP).

- **Influence functions** uses a double-differentiable function to approximate the influence of an instance. Instead of deleting the instance, the method upweights the instance in the loss by a very small step.

# So can we analyse the predictions made by deep learning models?

Yes, but usually indirectly:

- The models are often large (millions/billions of parameters) and it is infeasible to identify the contribution of each parameter
- Often encountered in a black-box setting, prohibiting any view of the internal state of the model
- Non-linear and non-convex, meaning that standard mathematical analysis tools cannot be used

# References


*The Interpretable ML Book*: https://christophm.github.io/interpretable-ml-book/

GradCAM: https://arxiv.org/pdf/1610.02391.pdf

PyTorch library for CAM methods: https://github.com/jacobgil/pytorch-grad-cam

Saliency for Fine-Grained Object Recognition: www.sciencedirect.com/science/article/pii/S0031320319301773.

FGSM (Introduction to adversarial samples): https://arxiv.org/abs/1412.6572

DeconvNet for feature visualisation: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Guided BackProp: https://arxiv.org/abs/1412.6806

Feature Visualisation: https://distill.pub/2017/feature-visualization/

Short break now!

We will then proceed to uncertainty…

# Uncertainty

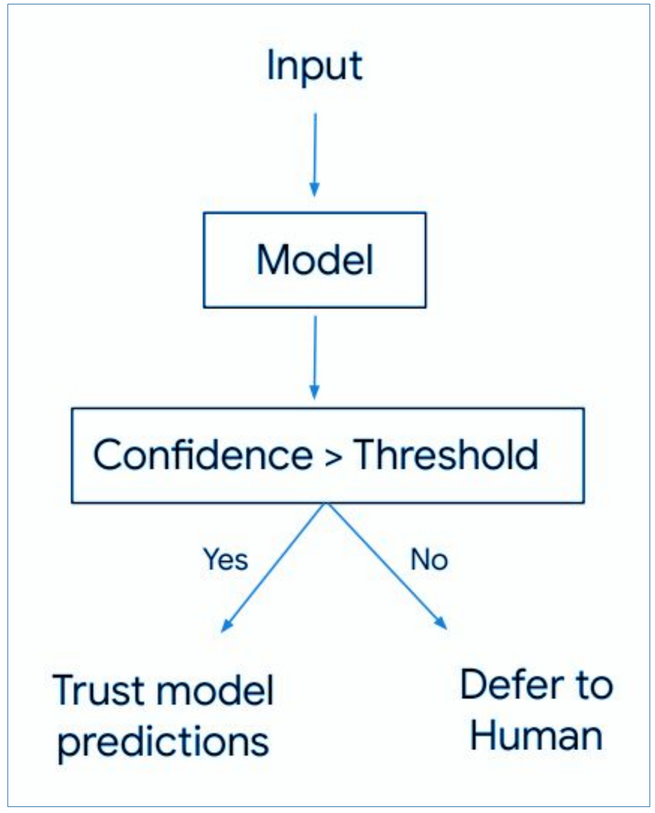
So what is uncertainty?

# Uncertainty

And why do we need it?

Answer:

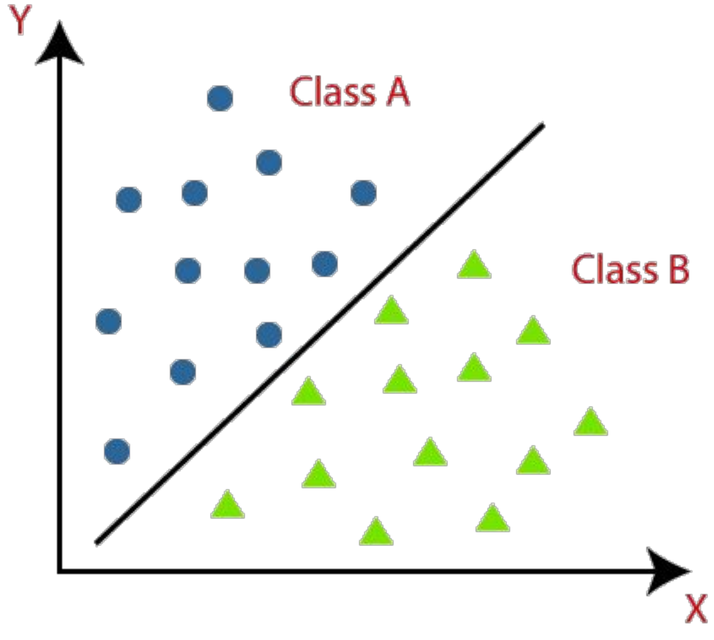
To provide confidence bounds for data analysis and decision making



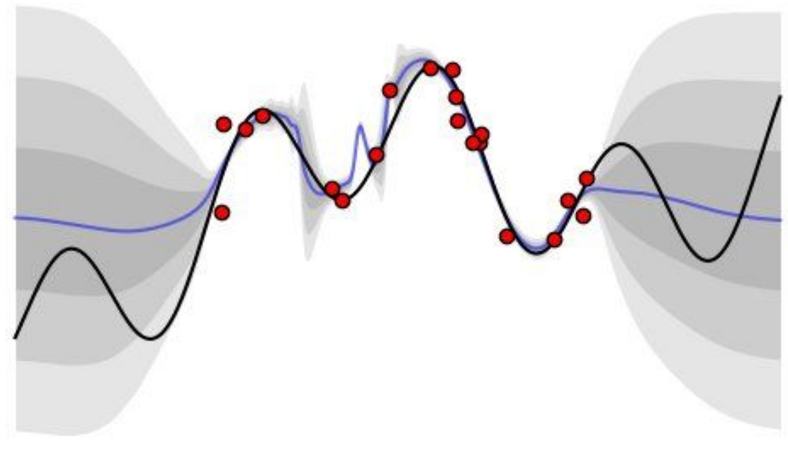

From https://www.gatsby.ucl.ac.uk/~balaji/balaji-uncertainty-talk-cifar-dlrl.pdf

# Uncertainty

Classification
(predict class label and its confidence)




Regression
(predict mean and its variance)



*In both cases we could obtain a prediction distribution instead of just a single value prediction*

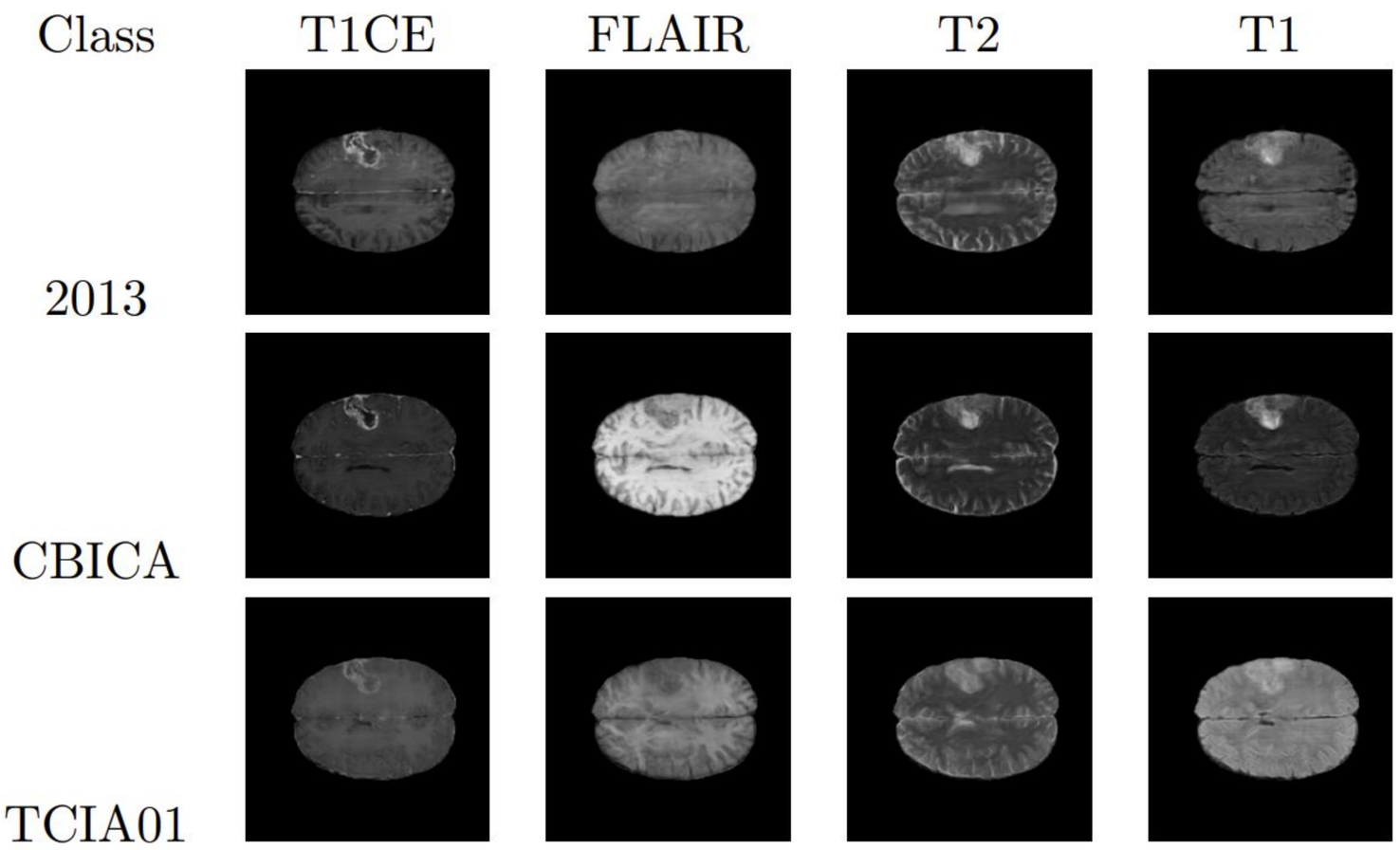
# Uncertainty

But is it really needed?

*“If your model predicts with >95% accuracy on this data, I don’t really care how confident it is…”*
From a conversion with some doctor.

# Uncertainty

It seems to be actually needed. Because of the distribution shift (when you apply your model on new data coming from a different distribution, you might not be able to evaluate the accuracy. There is a high chance it deteriorates, so it would be useful at least to know how confident your model is).




From *“Red-GAN: Attacking class imbalance via conditioned generation. Yet another medical imaging perspective”*

# Uncertainty

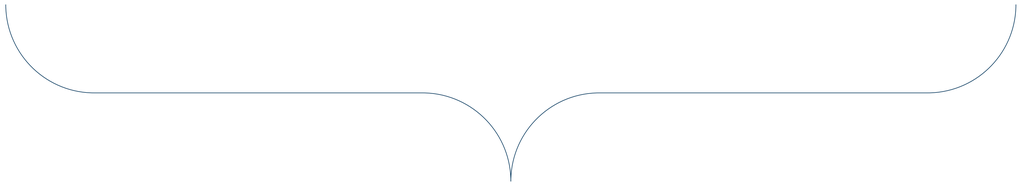
What are possible causes for data distribution shift?

- Population shift (ages, gender, ethnicity, etc)
- Acquisition shift (scanner, modality, protocol, etc)
- Annotation shift (annotation policy, annotator experience)

# Uncertainty

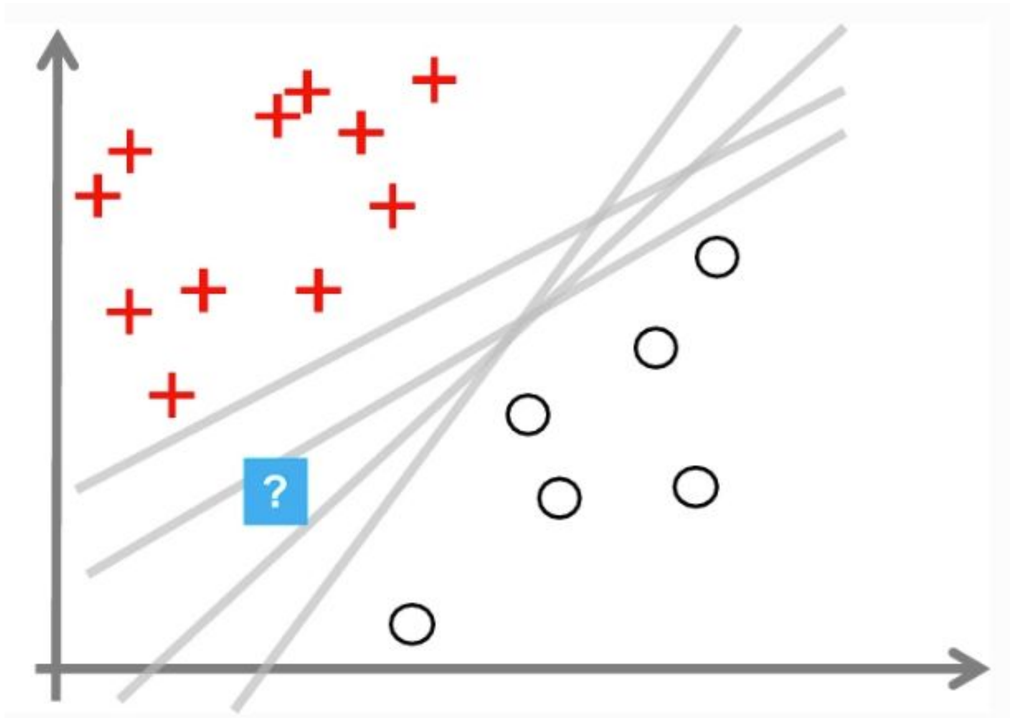
What are possible causes for data distribution shift?

- Population shift (ages, gender, ethnicity, etc)
- Acquisition shift (scanner, modality, protocol, etc)
- Annotation shift (annotation policy, annotator experience)



*Data (aleatoric) uncertainty*

# Uncertainty




Model (epistemic) uncertainty

★ Due to a lack of knowledge about the right hypothesis which is caused by a lack of data

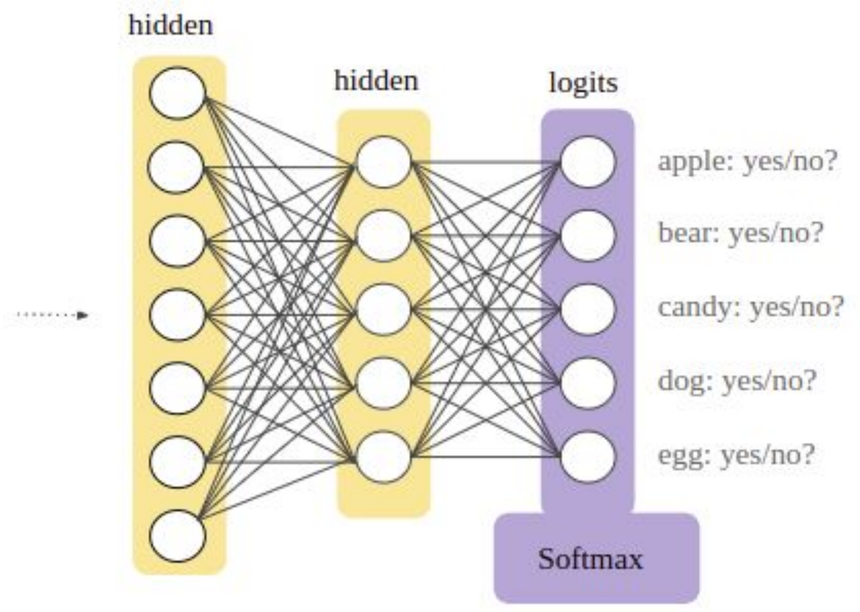
★ Epistemic uncertainty is reducible (in contrast to aleatoric uncertainty)

From *“Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods” by Hüllermeier E.*

# Methods to estimate uncertainty

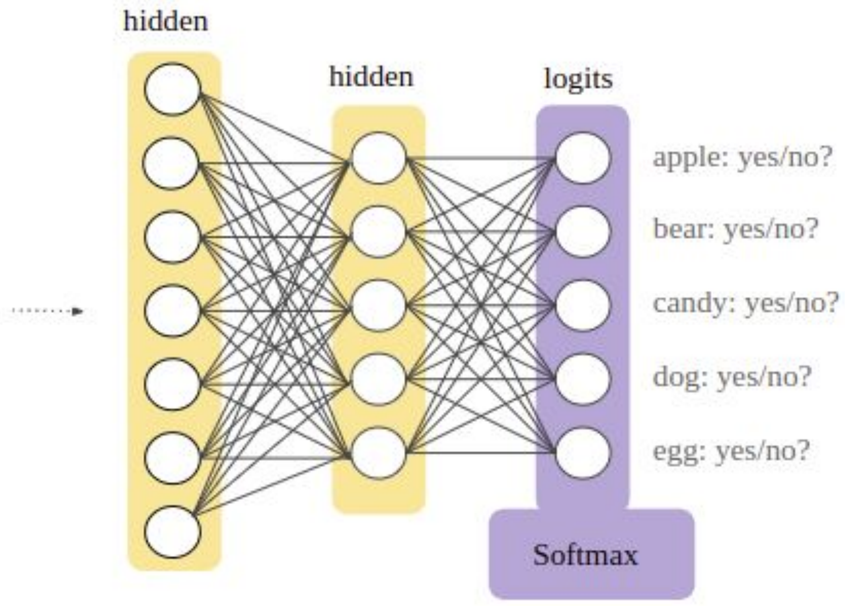
- Looking at softmax
- Ensembling
- Test-time augmentation
- Monte-Carlo Drop-out
- Bayesian neural networks
- Probabilistic U-Net

# Looking at softmax

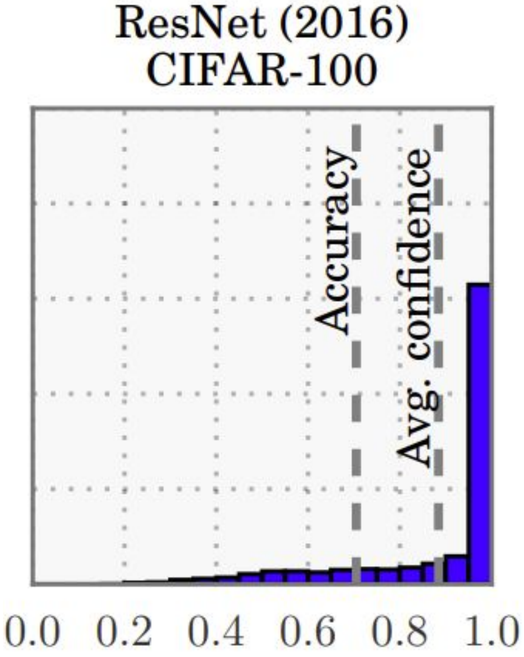



From https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

# Looking at softmax

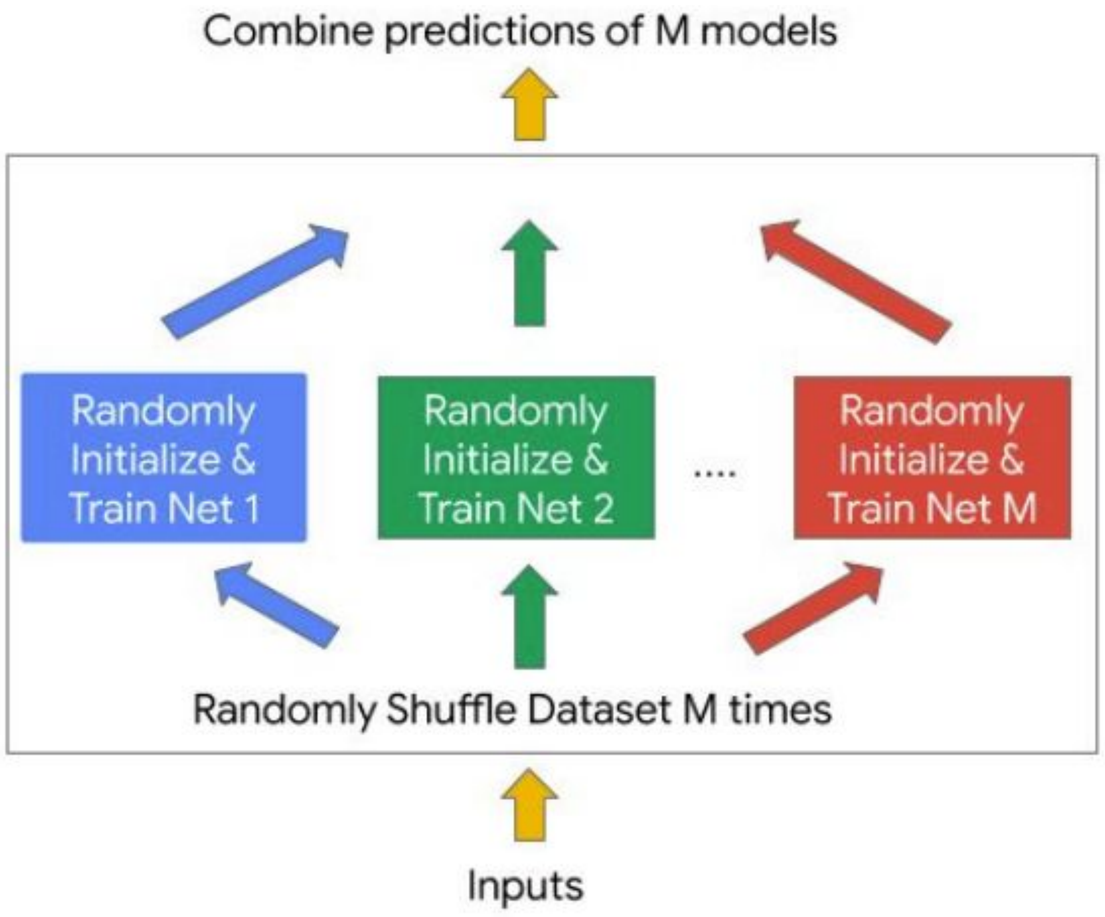



Problem!
Softmax in modern nets is prone to overconfidence




From *“On Calibration of Modern Neural Networks” by Guo C. et al.*

# Ensembling




From *“Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”,* Lakshminarayanan et al.

# Test-Time Augmentation

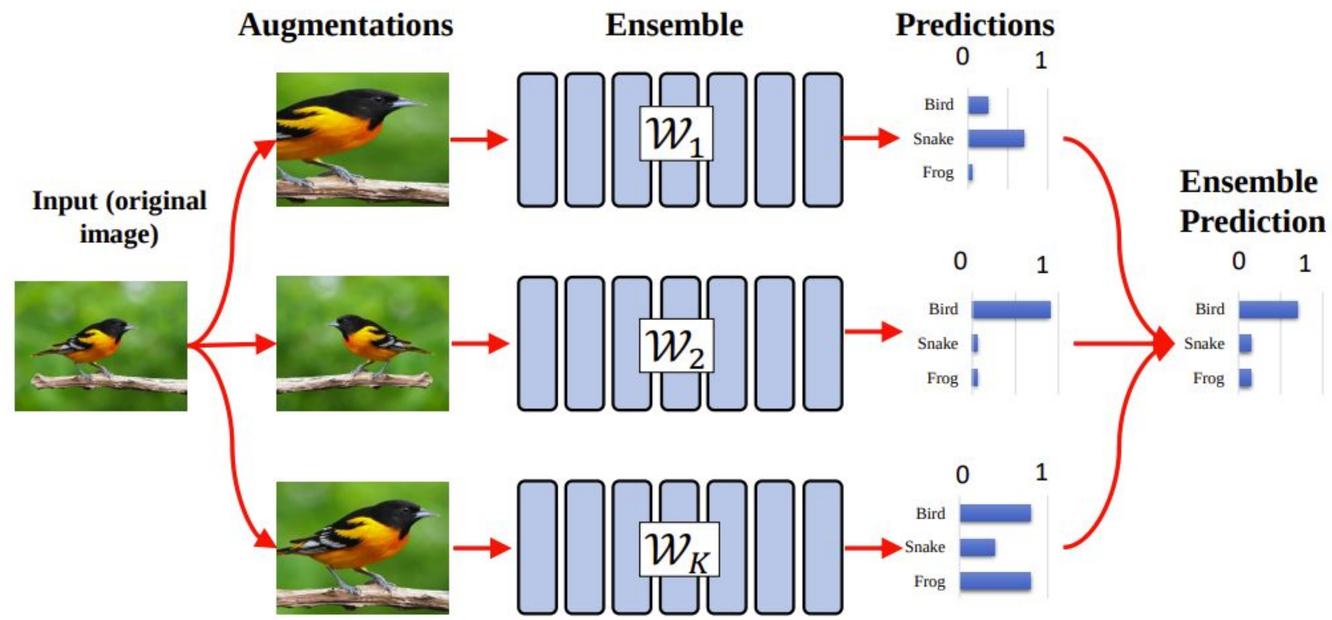
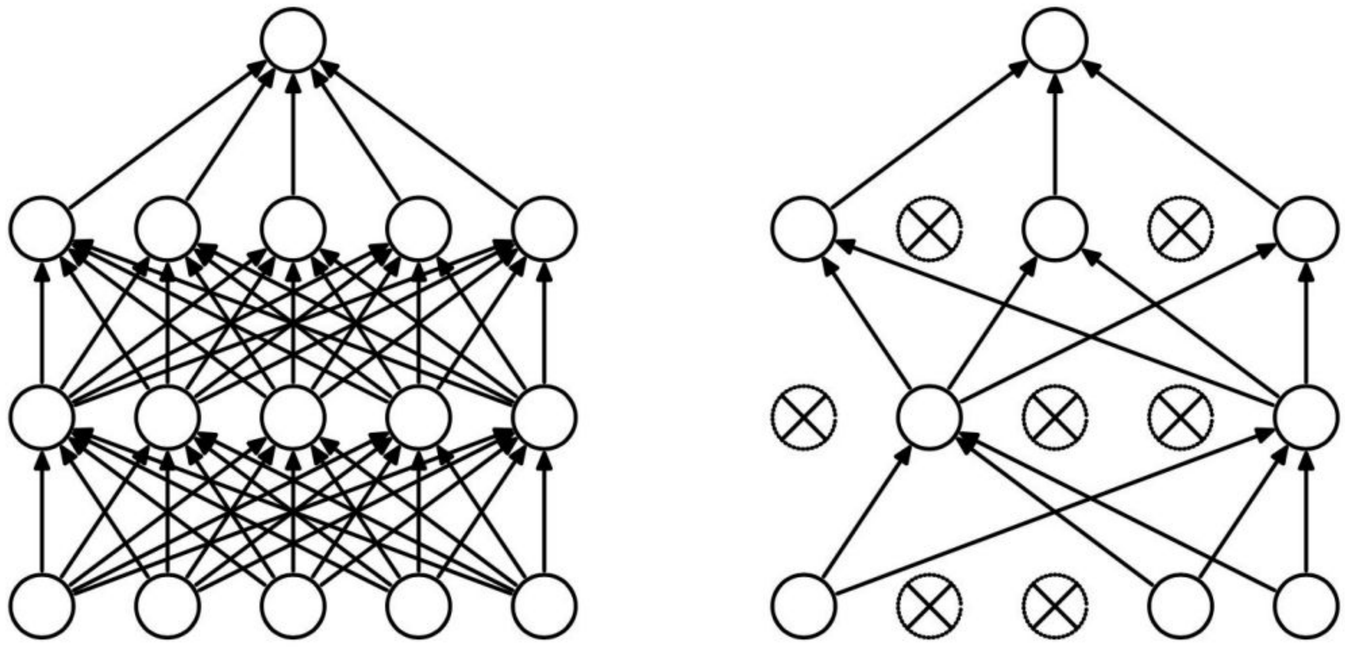



Fig. 17: A schematic view of TTA for ensembling techniques which is reproduced based on [212].

From “A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges” M. Abdar et al.

# Monte-Carlo Drop Out

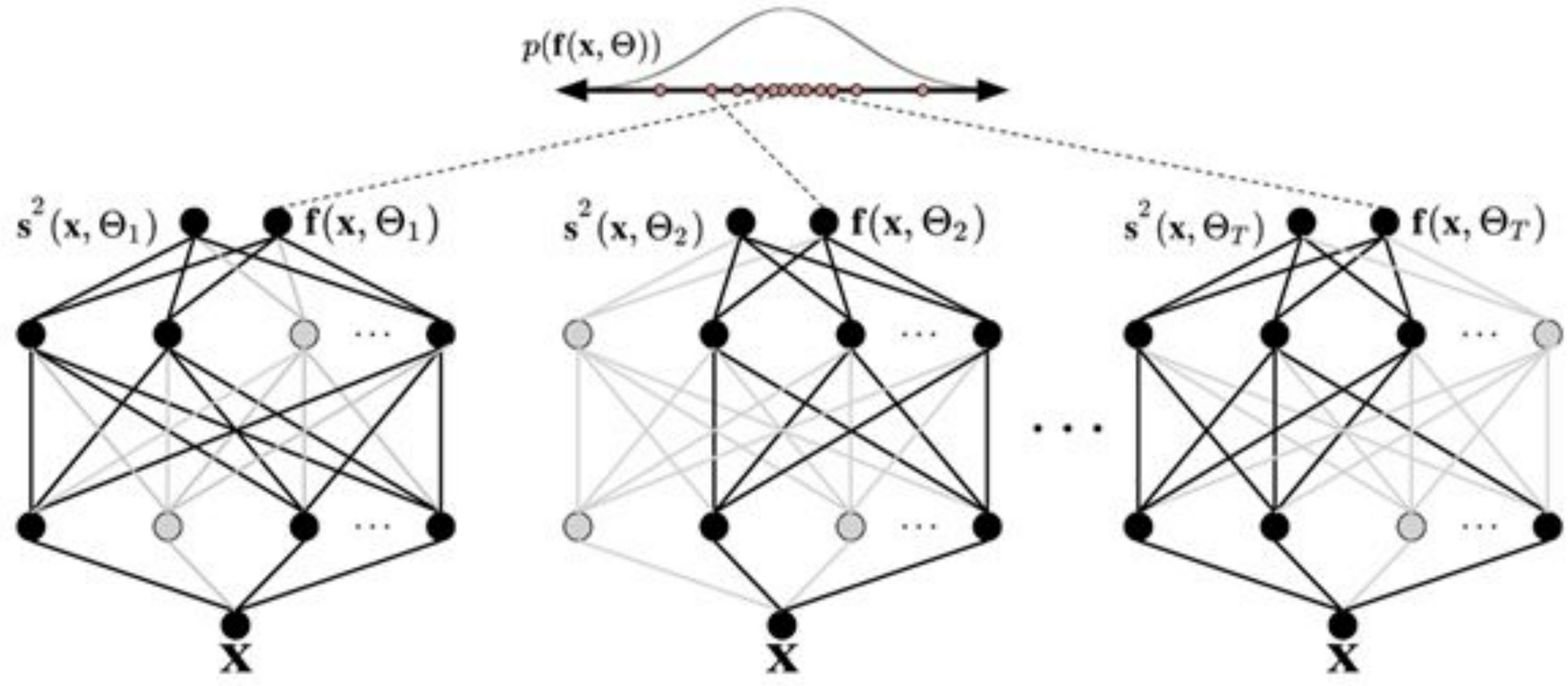


(a) Standard Neural Net

(b) After applying dropout.

From https://www.gatsby.ucl.ac.uk/~balaji/balaji-uncertainty-talk-cifar-dlrl.pdf

# Monte-Carlo Drop Out




From https://docs.aws.amazon.com/prescriptive-guidance/latest/ml-quantifying-uncertainty/mc-dropout.html

# Bayesian neural networks

When we train deterministic networks, we optimize the loss function to obtain parameters ($w$).

| $w$ | $\rightarrow$ | $Y_w(X)$ |
|---|---|---|
| parameters | | predictions |

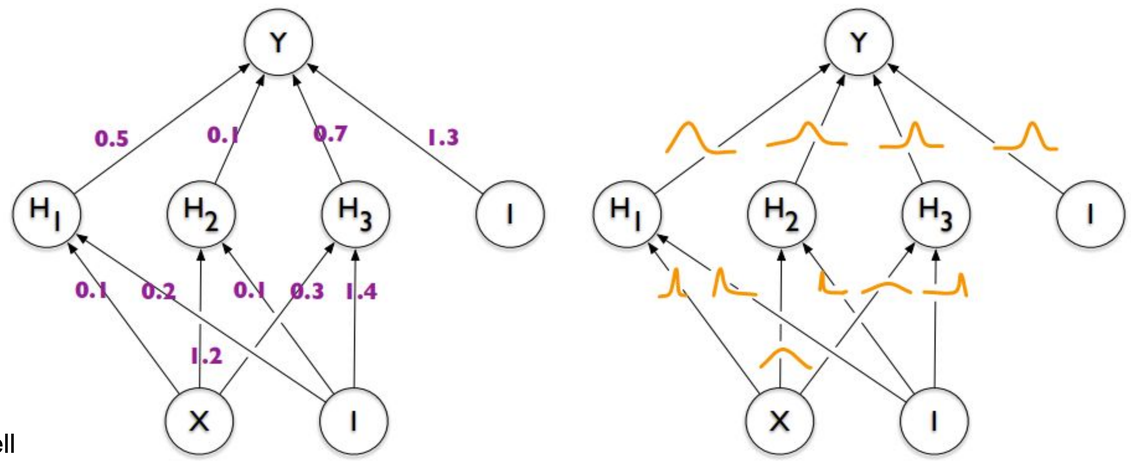
# Bayesian neural networks

When we train deterministic networks, we optimize the loss function to obtain parameters $w$.

$$\underset{\text{parameters}}{w} \quad \rightarrow \quad \underset{\text{predictions}}{Y_w(X)}$$

In Bayesian inference, instead of learning the parameters ($w$), we seek to compute p($w$ | D_train)

$$\underset{\text{distribution over parameters}}{p(w \mid D\_train)} \quad \rightarrow \quad \underset{\text{distribution over predictions}}{p(Y_w(X) \mid D\_train)}$$




From "*Weight Uncertainty in Neural Networks*" by C. Blundell

# Bayesian neural networks

When we train deterministic networks, we optimize the loss function to obtain parameters $w$.

| $w$ | $\rightarrow$ | $Y_w(X)$ |
|---|---|---|
| parameters | | predictions |

In Bayesian inference, instead of learning the parameters ($w$), we seek to compute p($w$ | D_train)

| p($w$ \| D_train) | $\rightarrow$ | p($Y_w(X)$ \| D_train) |
|---|---|---|
| distribution over parameters | | distribution over predictions |

$$\int_w p(\hat{y}(x)|w)p(w|D)dw = p(\hat{y}(x)|D)$$

Simple!

# Bayesian neural networks

When we train deterministic networks, we optimize the loss function to obtain parameters $w$.

| $w$ | $\rightarrow$ | $Y_w(X)$ |
|---|---|---|
| parameters | | predictions |

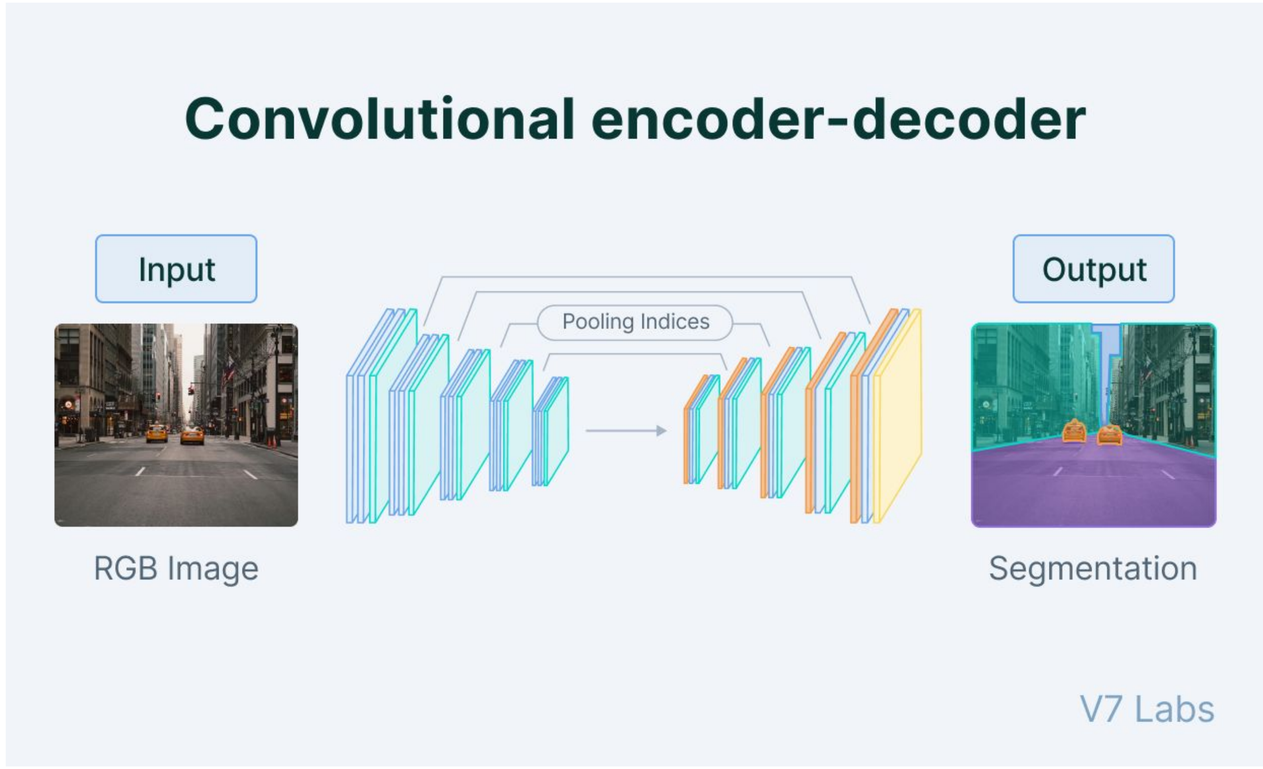
In Bayesian inference, instead of learning the parameters ($w$), we seek to compute p($w$ | D_train)

| p($w$ \| D_train) | $\rightarrow$ | p($Y_w(X)$ \| D_train) |
|---|---|---|
| distribution over parameters | | distribution over predictions |

$$\int_w p(\hat{y}(x)|w)p(w|D)dw = p(\hat{y}(x)|D)$$

Not Simple at all!

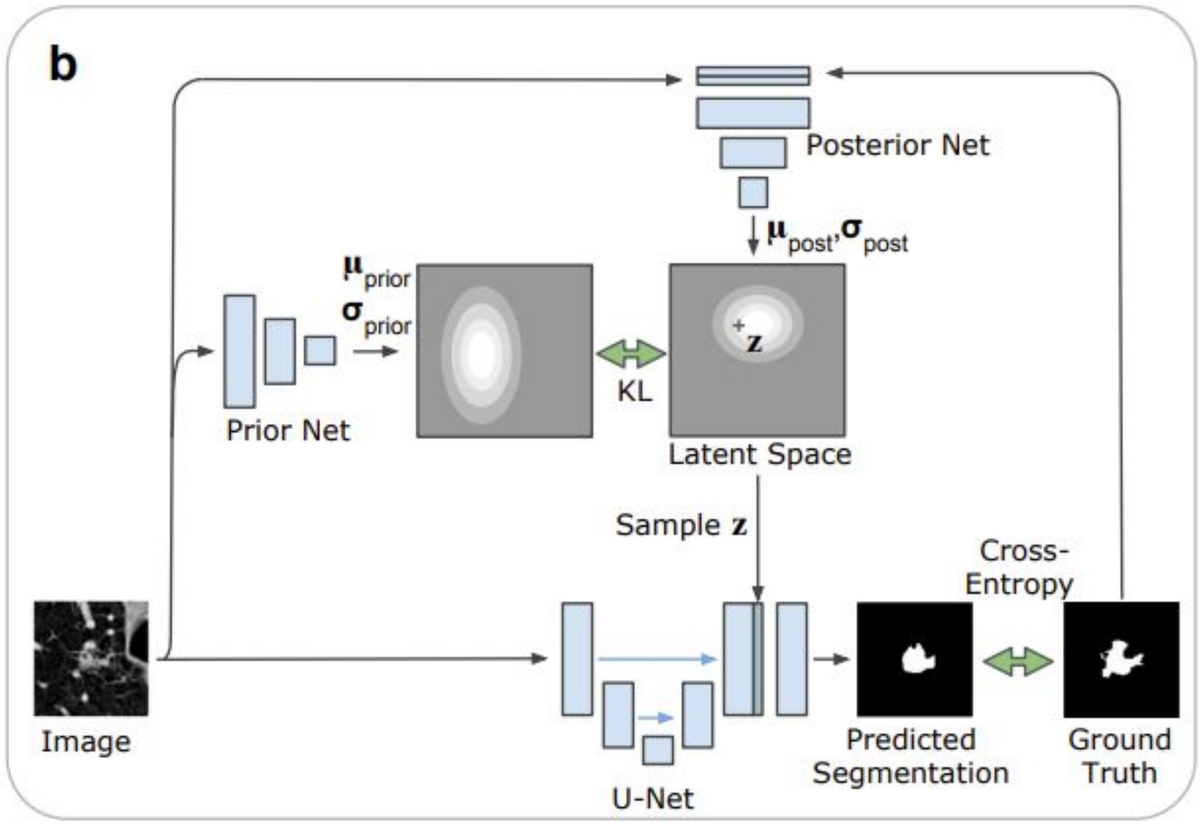
# Probabilistic U-Net




A U-Net architecture (deterministic one)

From https://www.v7labs.com/blog/semantic-segmentation-guide

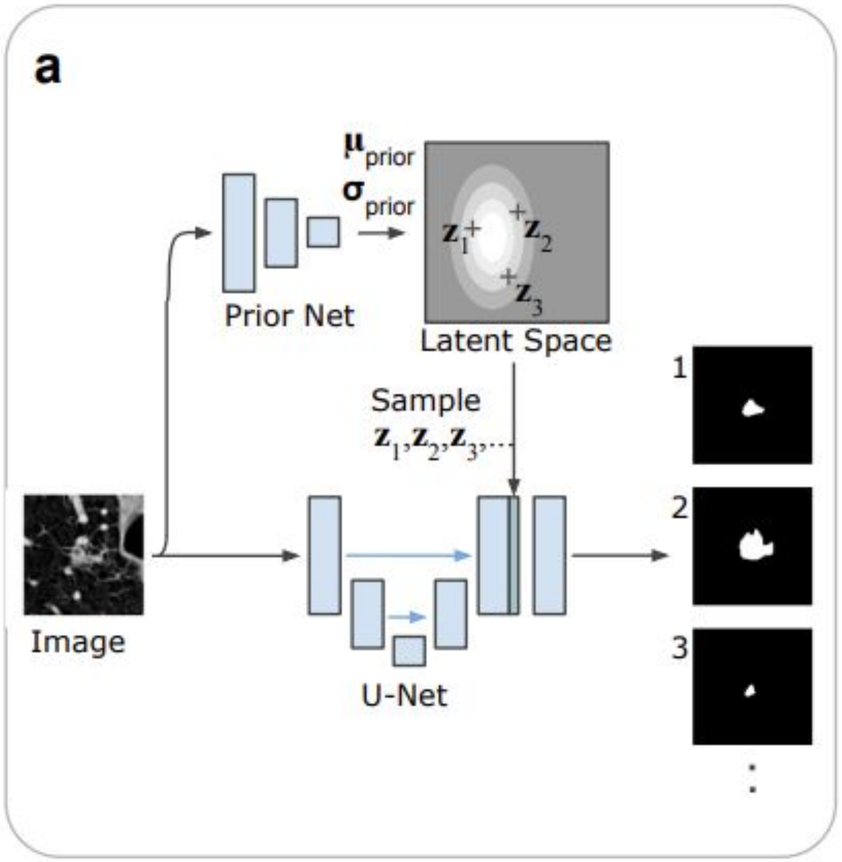
# Probabilistic U-Net




Training phase of the probabilistic U-Net

From *"A Probabilistic U-Net for Segmentation of Ambiguous Images" by Simon A. A. Kohl*

# Probabilistic U-Net




Test phase of the probabilistic U-Net

From *"A Probabilistic U-Net for Segmentation of Ambiguous Images" by Simon A. A. Kohl*

# This is the end of the chapter

But not of the course…

See you soon (during the tutorial hours on Thursday: 4pm-6pm)!