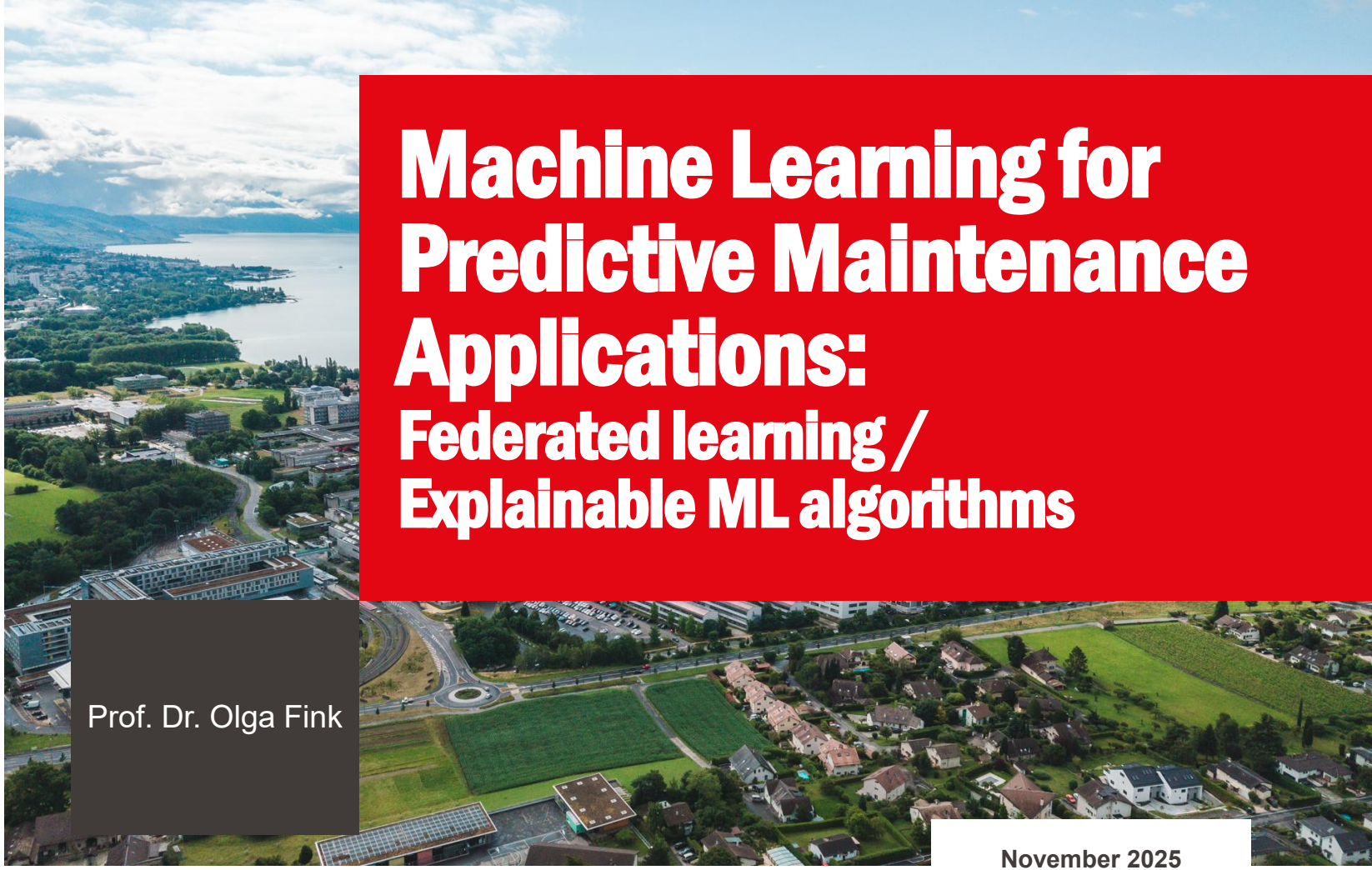


Machine Learning for Predictive Maintenance Applications:

Federated learning / Explainable ML algorithms

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

November 2025

Domain generalization

- Domain generalization focuses on training models that generalize to unseen target domains.
- Unlike domain adaptation, it does not require access to target domain data during training.
- It aims to learn robust and invariant features from source domains that perform well under different conditions.
- Useful when target environments are unknown or change dynamically.
- Helps create models that are more adaptable and resilient to distribution shifts or novel scenarios.

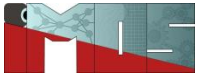
Continuous domain adaptation

- Continuous domain adaptation focuses on adapting a model continuously as the target domain evolves over time.
- Unlike traditional domain adaptation, it handles scenarios where data distributions change dynamically and require ongoing adjustments.
- It seeks to maintain model performance by adapting incrementally to new conditions without needing to retrain from scratch.
- Useful for real-world applications where environments and data distributions are constantly shifting.
- Often involves strategies for updating learned representations, retaining past knowledge, and mitigating catastrophic forgetting.

Test-time adaptation

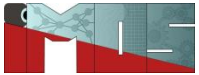
- Test-time adaptation refers to adapting a model to distribution shifts in the target domain during inference, without further training on the source domain.
- It focuses on updating model parameters or representations using only test data, typically through self-supervised or unsupervised strategies.
- Allows models to respond to domain shifts or changes at the point of inference, improving performance in unseen or evolving conditions.
- Reduces the reliance on having a large amount of labeled data in the target domain.
- Useful for scenarios where models encounter dynamic environments or previously unseen data distributions at test time.

Which questions to follow to select an ML model



1. **What is the nature of the problem?**
   - Is it a classification, regression, clustering, or recommendation problem?
   - Are you dealing with supervised, unsupervised, or reinforcement learning?
2. **What is the size and quality of the dataset?**
   - How many samples and features are available?
   - Is the data clean, or does it contain a lot of noise and missing values?
   - Is the data balanced, or are there significant class imbalances?
3. **Is the training dataset representative of the expected operating / application conditions**
   - Is there a high diversity of operating conditions expected?
4. **What are the characteristics of the data?**
   - Is the data structured or unstructured (e.g., text, images, audio)?
   - Are the features numerical, categorical, or a mix of both?
   - Do the features have a temporal or spatial component?

Which questions to follow to select an ML model



5. What are the performance requirements?

- Do you prioritize accuracy, interpretability, or computational efficiency?
- Is the problem domain one where explainability is crucial (e.g., healthcare, finance)?
- What are the acceptable trade-offs between bias and variance?

6. What is the computational complexity and resources available?

- How much time and computational power do you have for training?
- Do you have access to specialized hardware (e.g., GPUs)?

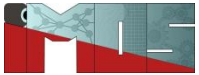
7. What are the specific goals and constraints?

- What is the end goal of applying the ML model (e.g., prediction, anomaly detection, optimization)?
- Are there specific business or application constraints to consider (e.g., real-time processing, deployment environment)?

8. What is the level of domain knowledge available?

- How well do you understand the domain and the problem?
- Is there domain expertise to help with feature engineering and model interpretation?

Which questions to follow to select an ML model



9. What is the expected model lifecycle?

- How often will the model need to be updated or retrained?
- Is the model expected to handle concept drift (changes in the underlying data distribution over time)?

10. What is the potential for model interpretability?

- Do stakeholders need to understand how the model makes decisions?
- Is there a regulatory requirement for transparency?

11. What are the available benchmarks or baselines?

- Are there existing solutions or benchmarks that can provide a performance reference?
- How do different algorithms perform on similar problems in literature or industry standards?

Data sharing/ Federated learning

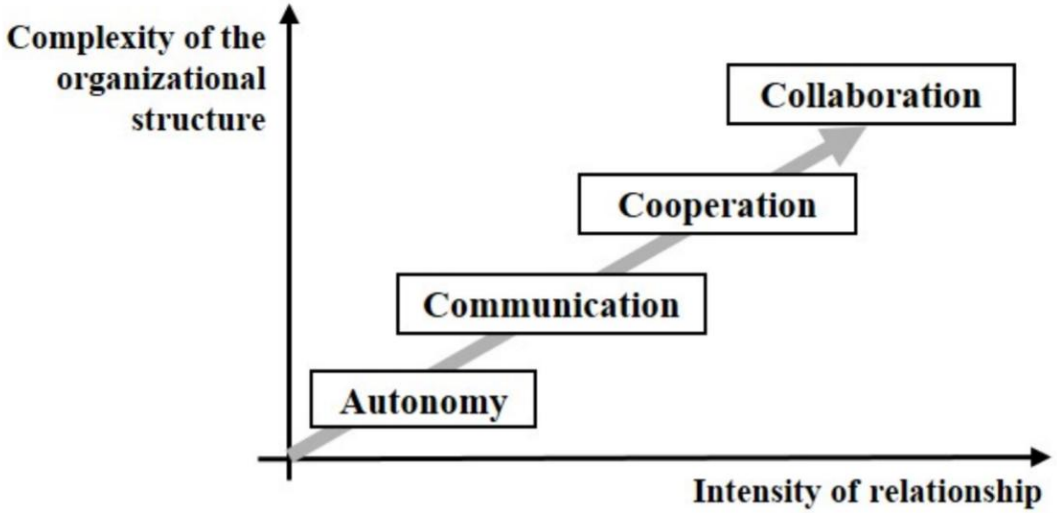
Different stakeholders in PHM

The importance of data for ML

> “The biggest obstacle to using advanced data analysis isn’t skill base or technology; it’s plain old access to the data.”
>
> -Edd Wilder-James, Harvard Business Review

Source: Min Du, 2019

New forms of sharing and collaboration required

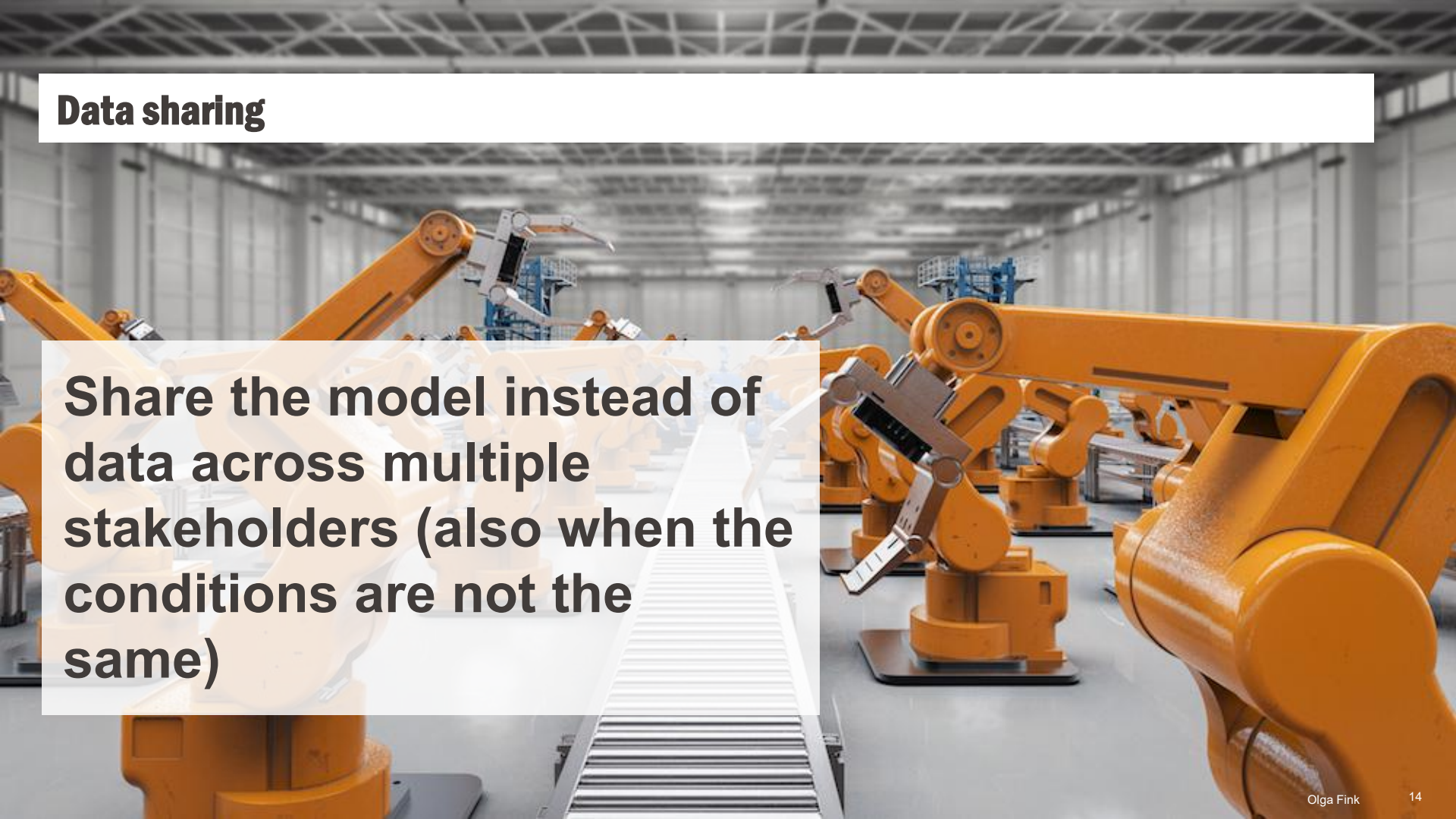


Source: Chebel-Morello, Nicod, Varnier, 2017

Data sharing

How could data be shared across stakeholders?

Data sharing



Share the model instead of data across multiple stakeholders (also when the conditions are not the same)

A new paradigm – Federated Learning

a synchronous update scheme that proceeds in rounds of communication

McMahan, H. Brendan, Eider Moore, Daniel Ramage, and Seth Hampson. "Communication-efficient learning of deep networks from decentralized data." *AISTATS, 2017*.

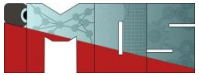
Source: Min Du, 2019

Federated learning

- **Federated learning** is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider.
- Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

Source: *Advances and Open Problems in Federated Learning*
https://arxiv.org/pdf/1912.04977.pdf

Federated learning characteristics (1/2)



1. Data Privacy and Security

- **Data Locality**: Raw data remains on local devices (e.g., smartphones, edge devices, or distributed servers) instead of being sent to a central server, enhancing privacy and security.
- **Secure Aggregation**: Only model updates (e.g., gradients) are sent to a central server, and these updates can be further encrypted or anonymized, reducing the risk of exposing sensitive information.
- **Privacy-Preserving Techniques**: Techniques like differential privacy or homomorphic encryption are often integrated to ensure that individual data points cannot be inferred from the shared updates.

2. Decentralized Training

- **Local Training**: Each device trains a model locally on its own data, which means training occurs independently across a distributed network.
- **Global Model Aggregation**: After local training, model parameters are sent to a central server that aggregates these updates to form a new global model, which is then sent back to each local device to continue training.

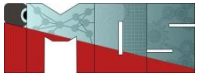
3. Handling Data Heterogeneity

- **Non-IID Data**: In federated learning, data on different devices is often non-independent and identically distributed (non-IID), meaning data distributions may vary significantly between devices.
- **Variable Data Quality and Quantity**: Some devices may have abundant data, while others have minimal or noisy data, requiring federated learning algorithms to account for varying data quality and volume.

4. Communication Efficiency

- **Minimizing Data Transfer**: To limit network usage, federated learning focuses on minimizing the frequency and size of data transferred between the server and devices.
- **Compression Techniques**: Methods like quantization, sparsification, and pruning are used to reduce the size of model updates, which is essential for devices with limited bandwidth.

Federated learning characteristics (2/2)



5. Device Heterogeneity and Scalability

- **Diverse Device Capabilities**: Federated learning systems are designed to work across a wide variety of devices, from smartphones to IoT devices, each with different computational capabilities, network stability, and battery life.
- **Scalability**: Federated learning is scalable, as it can handle thousands or even millions of devices simultaneously, with participants joining or leaving the training process dynamically.

6. Model Personalization

- **Personalized Models**: Federated learning allows each device to maintain its own personalized version of the model, especially if local data significantly deviates from the global model's training data. This feature can improve the model's performance on individual devices.

7. Fault Tolerance

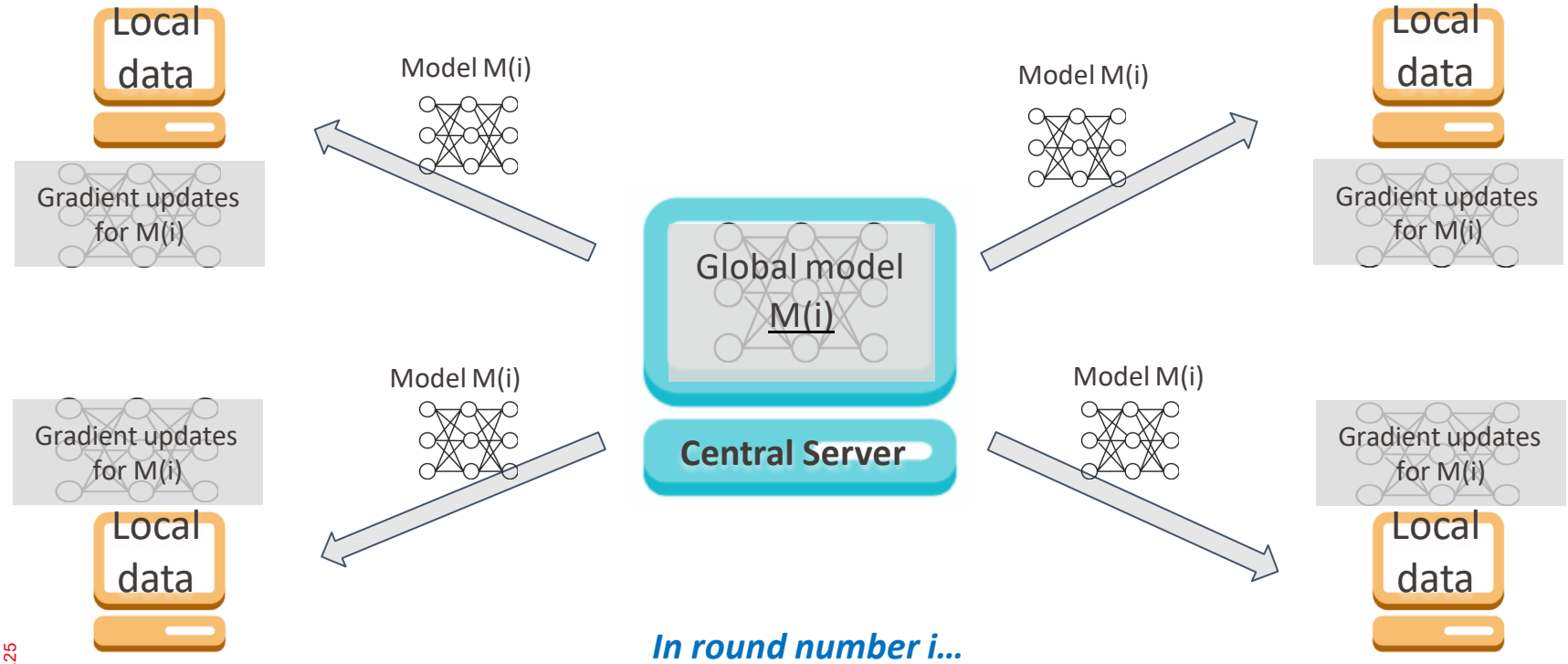
- **Robust to Device Failures**: Federated learning systems can continue training even if some devices drop out or are temporarily unavailable, making them resilient to intermittent connectivity and device churn.

Federated Learning vs. Peer-to-peer learning

	Federated learning	**Fully decentralized (peer-to-peer) learning**
Orchestration	A central orchestration server or service organizes the training, but never sees raw data.	No centralized orchestration.
Wide-area communication	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	Peer-to-peer topology, with a possibly dynamic connectivity graph.

Source: *Advances and Open Problems in Federated Learning*
https://arxiv.org/pdf/1912.04977.pdf

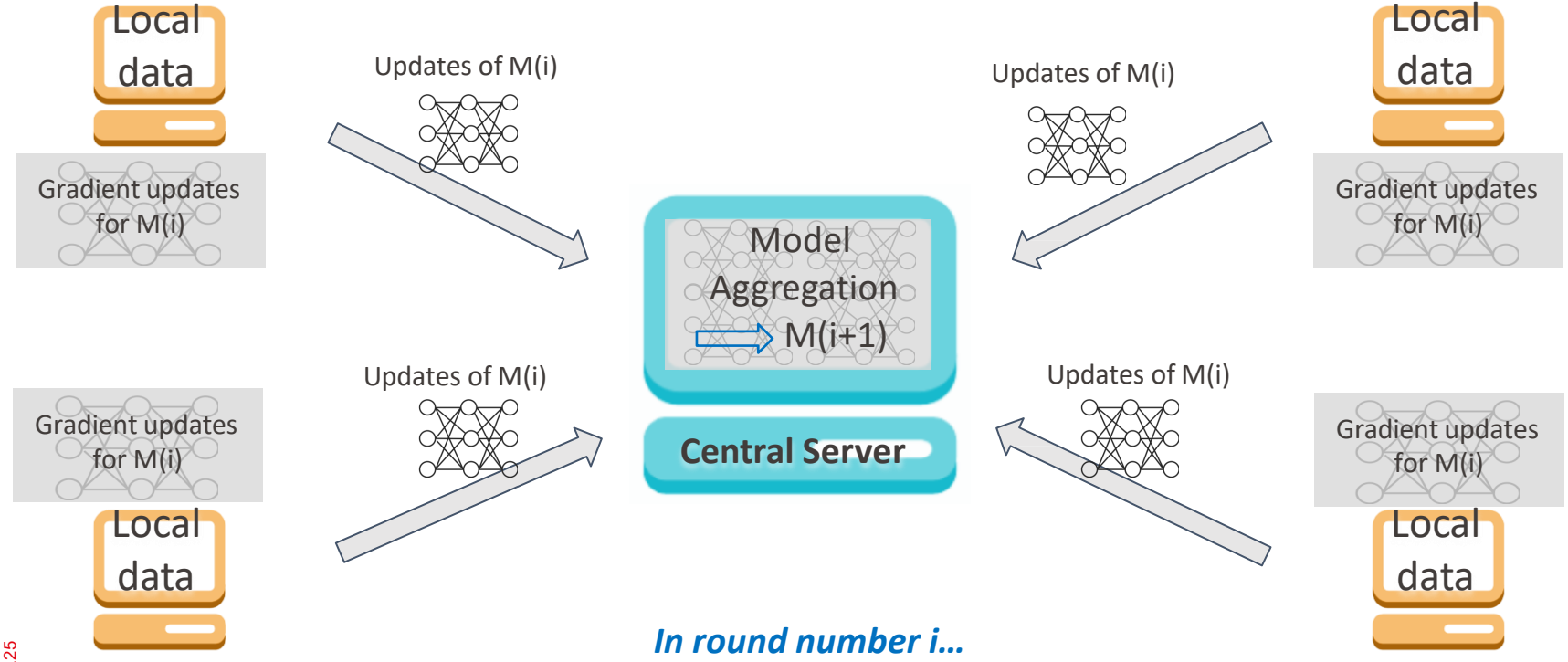
Federated learning – overview



In round number i…

Source: Min Du, 2019

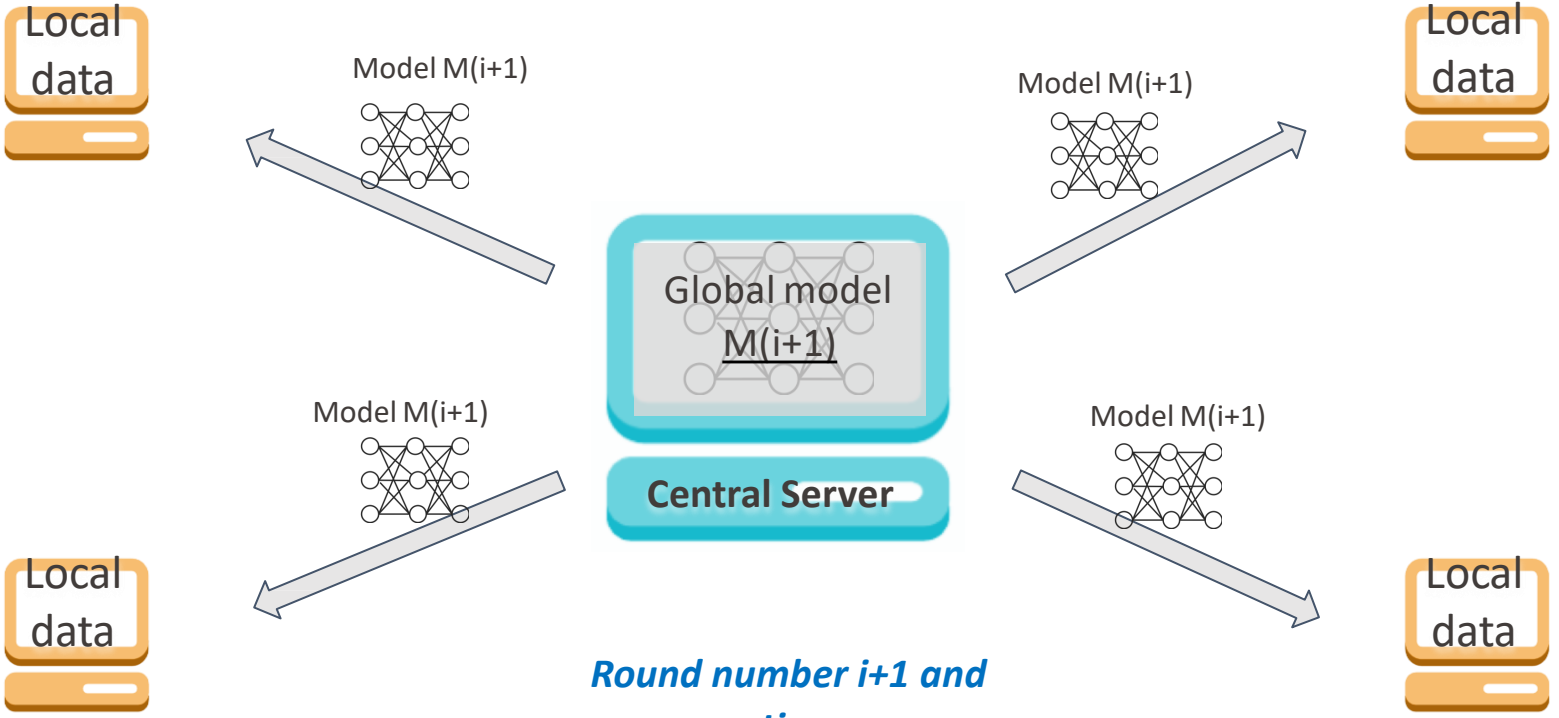
Federated learning – overview



In round number i…

Source: Min Du, 2019

Federated learning – overview



Round number i+1 and continue…

Source: Min Du, 2019

Federated learning – detail

- Recall in traditional deep learning model training
  - For a training dataset containing n samples (x_i, y_i), $1 \leq i \leq n$, the training objective is:

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

$f(w) = l(x_i, y_i, w)$ is the loss of the prediction on example (x_i, y_i)

  - Deep learning optimization relies on SGD and its variants, through ***mini-batches***

$$w_{t+1} \leftarrow w_t - \eta \nabla f(w_t; x_k, y_k)$$

Source: Min Du, 2019

Federated learning – detail

- In federated learning
  - Suppose n training samples are distributed to K clients, where P_k is the set of indices of data points on client k, and $n_k = |P_k|$
  - For training objective: $\min_{w \in \mathbb{R}^d} f(w)$

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \qquad \text{where} \qquad F_k(w) \stackrel{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$$

Source: Min Du, 2019

A baseline – FederatedSGD (FedSGD)

- A randomly selected client that has n_k training data samples in federated learning ≈ *A randomly selected sample in traditional deep learning*
- Federated SGD (FedSGD): a single step of gradient descent is done per round
- Recall in federated learning, a *C*-fraction of clients are selected at each round.
  - *C*=1: full-batch (non-stochastic) gradient descent
  - *C*<1: stochastic gradient descent (SGD)

Source: Min Du, 2019

FederatedSGD (FedSGD) / FederatedAveraging (FedAvg)

Learning rate: η; total #samples: n; total #clients: K; #samples on a client k: n_k; clients fraction $C = 1$

- In a round t:
  - The central server broadcasts current model w_t to each client; each client k computes gradient: $g_k = \nabla F_k(w_t)$, on its local data.
    - Approach 1: Each client k submits g_k; the central server aggregates the gradients to generate a new model:
      - $w_{t+1} \leftarrow w_t - \eta \nabla f(w_t) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$.

      Recall $f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$
    - Approach 2: Each client k computes: $w_{t+1}^k \leftarrow w_t - \eta g_k$; the central server performs aggregation:
      - $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

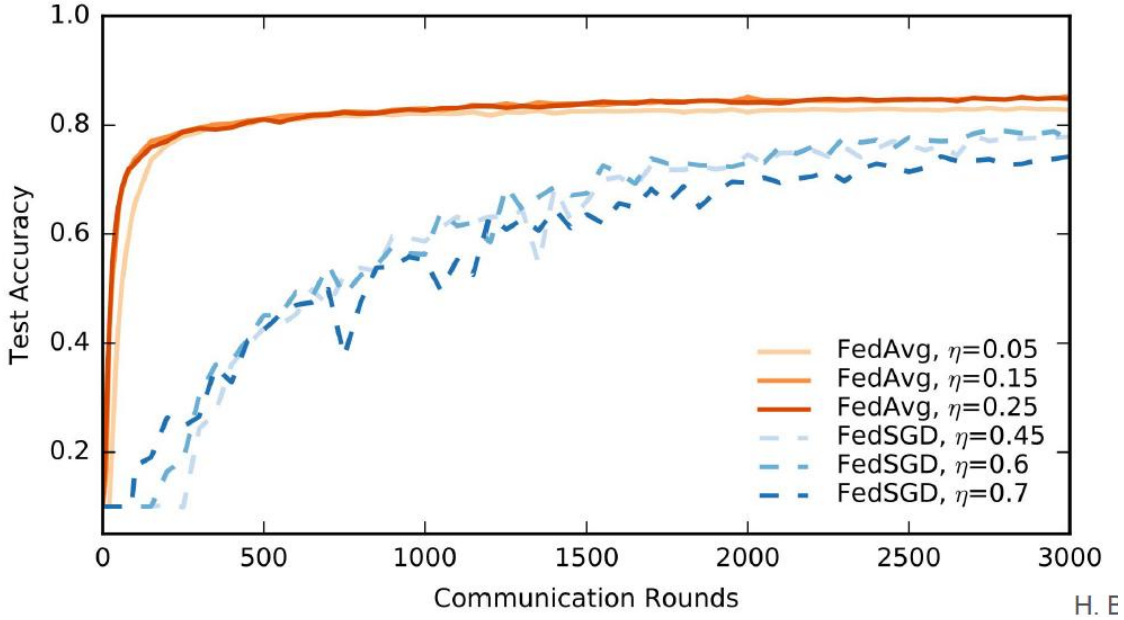
      For multiple times ⟹ FederatedAveraging (FedAvg)

Source: Min Du, 2019

FedSGD vs FedAvg

- **FederatedSGD (FedSGD)**
  - **Process**: In FedSGD, each device performs a single step of stochastic gradient descent (SGD) on its local data and then sends the computed gradients (i.e., updates) to the central server. The server then aggregates these gradients from all devices and applies them to the global model.
  - **Communication**: FedSGD involves frequent communication between the devices and the server, as each gradient update requires synchronization with the server.
  - **Advantages**: FedSGD ensures that each update to the global model reflects the most recent local gradient information from each device. This can sometimes lead to faster convergence if the communication overhead is manageable.
  - **Drawbacks**: The high frequency of communication in FedSGD is a major drawback, especially in federated learning scenarios where devices might have limited bandwidth. Frequent communication also consumes more energy on devices, which is a concern for battery-powered devices like mobile phones.
- **FederatedAveraging (FedAvg)**
  - **Process**: In FedAvg, each device performs multiple local updates (i.e., several steps of SGD) on its local data before sending the updated model parameters to the central server. The server then averages the model parameters from all participating devices to update the global model.
  - **Communication**: By performing multiple local updates before communicating, FedAvg significantly reduces the frequency of communication between devices and the server.
  - **Advantages**: FedAvg is much more communication-efficient than FedSGD since each device communicates with the server only after several local updates. This makes it more suitable for federated learning, where communication costs are high. Additionally, local updates often lead to faster convergence in practical applications, as they allow each device to adapt the model more effectively to its own data before syncing.
  - **Drawbacks**: FedAvg might converge more slowly or be less stable in scenarios where data on each device is highly heterogeneous (non-IID). In such cases, performing multiple local updates can lead to models that diverge from each other, making aggregation less effective.

CIFAR-10 Convolutional model



Updates to reach 82%

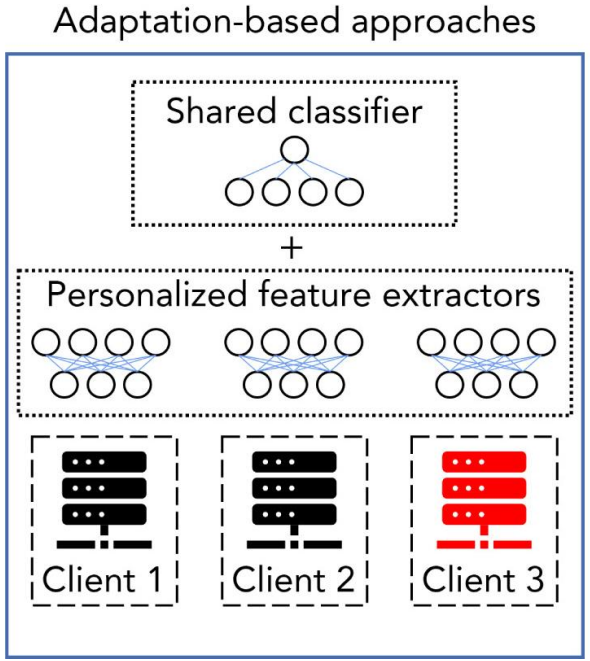
SGD	31,000
FedSGD	6,600
FedAvg	630

49x decrease in communication (updates) vs SGD

(IID and balanced data)

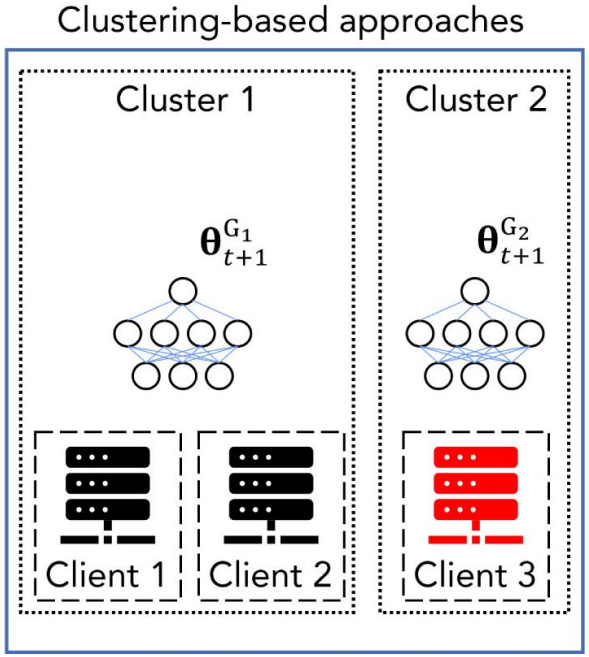
McMahan, H. Brendan, Eider Moore, Daniel Ramage, and Seth Hampson. "Communication-efficient learning of deep networks from decentralized data." *AISTATS, 2017*.

Adaptation-based approach to FL



Overview of an adaptation-based approach to FL that uses a global classifier to accommodate client-specific feature extractors.

Clustering-based approach to FL



Overview of a clustering-based approach to FL that groups clients for training based on non-private comparison criteria

Example

Data heterogeneity in industrial settings

1. *Domain shift:* Clients do not share a similar data distribution.
2. *Label heterogeneity:* Clients' datasets have a different number and types of faults.

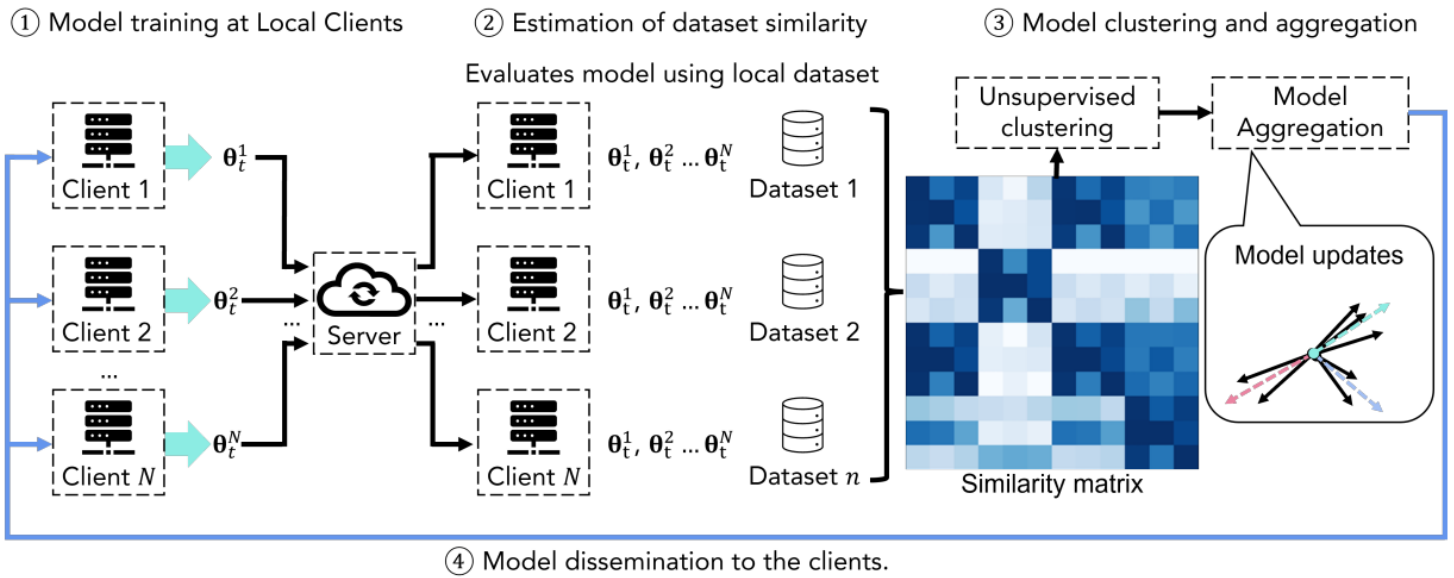
Federated learning adapted to the specificities of the unit

- Distance awareness: the model's ability to quantify the distance of a testing example from the training data.

- For personalized federated learning, during model aggregation → a local client would assign higher weights to the model that was trained using similar training data.

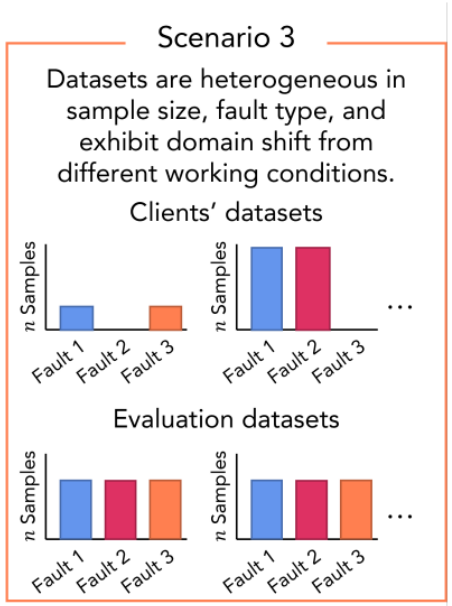
- Spectral-normalized Neural Gaussian Process (SNGP) used to quantify the prediction uncertainty

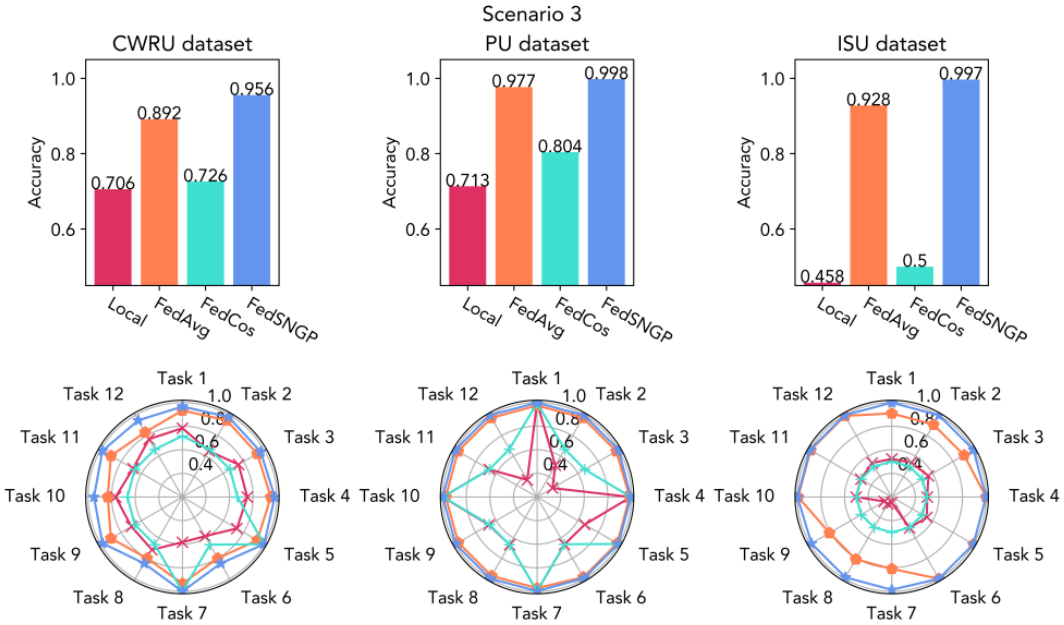
Federated Learning with Uncertainty-Based Client Clustering for Fleet-Wide Fault Diagnosis



Lu, H., Thelen, A., Fink, O., Hu, C., & Laflamme, S. (2024). Federated Learning with Uncertainty-Based Client Clustering for Fleet-Wide Fault Diagnosis, Mechanical Systems and Signal Processing, 210, p.111068.

Results on FedSNGP





Lu, H., Thelen, A., Fink, O., Hu, C., & Laflamme, S. (2024). Federated Learning with Uncertainty-Based Client Clustering for Fleet-Wide Fault Diagnosis, Mechanical Systems and Signal Processing, 210, p.111068.

Explainability / Interpretability

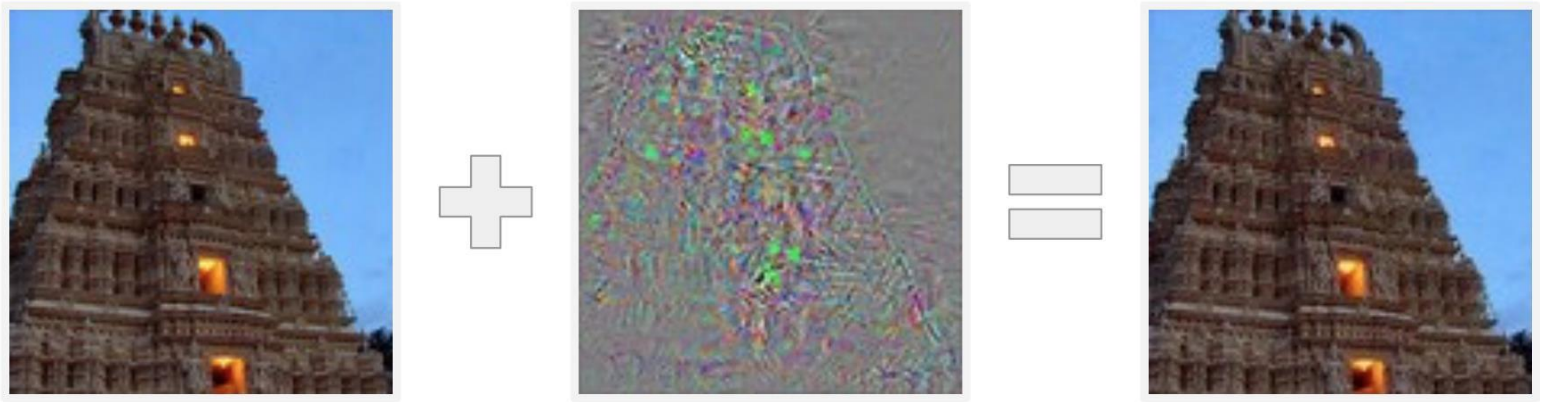
Most of the time, the algorihtms work really well but sometimes…

A refrigerator filled with lots of food and drinks

Tesla Autopilot Misidentified On-Road Horse-Drawn Carriage

Source: https://incidentdatabase.ai/cite/398/

Adversarial Examples



Original image

Temple (97%)

Perturbations

Adversarial example

Ostrich (98%)

Source: Byron Wallace

Explainability in XAI

- Refers to how well humans can understand and interpret a model's decision-making process
- Involves making AI predictions and reasoning transparent and accessible
- Helps users comprehend **why** and **how** a model arrives at specific outcomes
- Aims to build trust, accountability, and the ability to debug or improve AI systems
- Especially important for complex "black-box" models (e.g., deep neural networks)

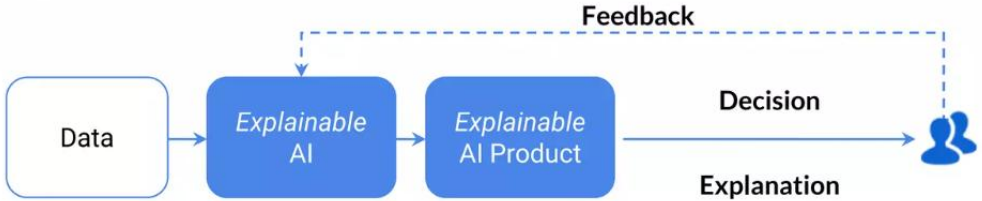
Why explainable AI?

Black Box AI

Confusion with Today's AI Black Box

- Why did you do that?
- Why did you not do that?
- When do you succeed or fail?
- How do I correct an error?

Explainable AI



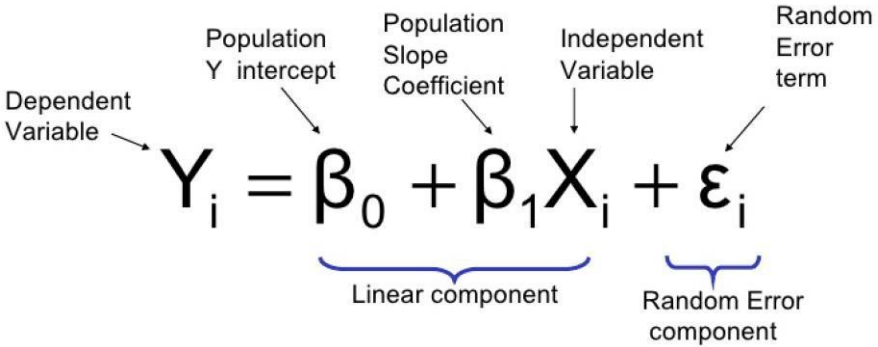
Clear & Transparent Predictions

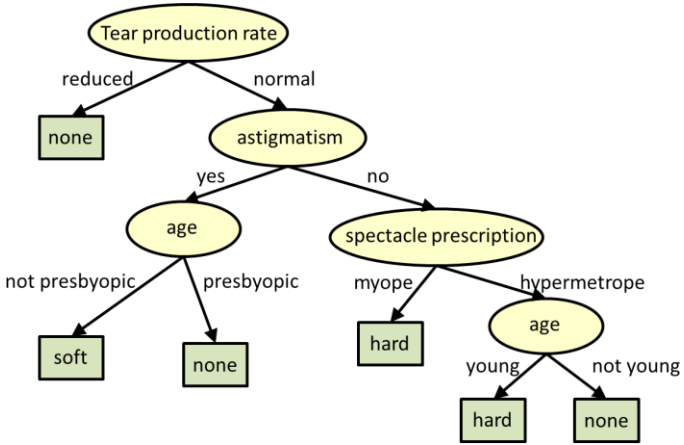
- I understand why
- I understand why not
- I know why you succeed or fail
- I understand, so I trust you

Source: WWW 2020 Tutorial

Simple explainability

- In pre-deep learning models, some models are considered “interpretable”



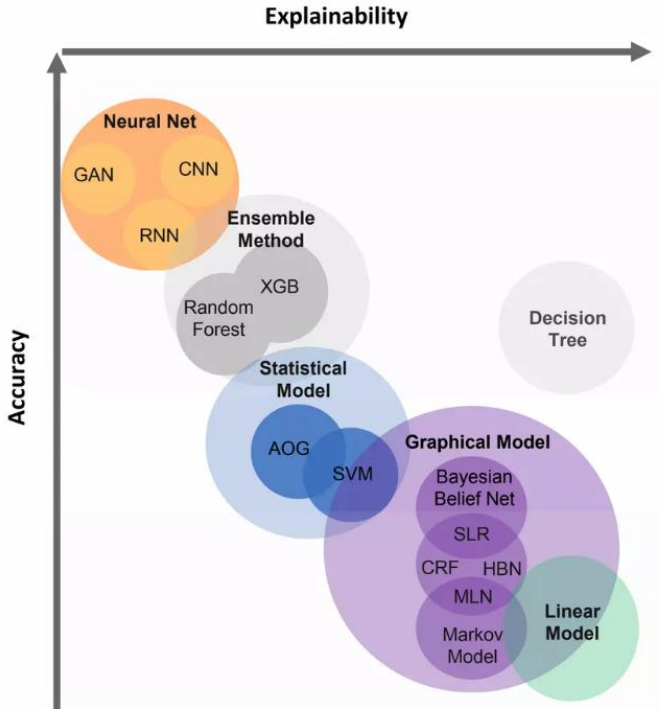


Source: Byron Wallace

Accuracy vs. Explainability

Learning

- Challenges:
  - Supervised
  - Unsupervised learning
- Approach:
  - Representation Learning
  - Stochastic selection
- Output:
  - **Correlation**
  - **No causation**



Interpretability

- Non-Linear functions
- Polynomial functions
- Quasi-Linear functions

Source: WWW 2020 Tutorial

Different ways to achieve interpretability

Build interpretability into the model

Post-hoc approach to interpretability → trying to explain given models and their output

Some properties of Interpretations

- Faithfulness - how to provide explanations that accurately represent the true reasoning behind the model's final decision.
- Plausibility – Is the explanation correct or something we can believe is true, given our current knowledge of the problem ?
- Understandable – Can I put it in terms that end user without in-depth knowledge of the system can understand ?
- Stability – Do similar instances have similar interpretations ?

Source: Byron Wallace

Global vs Local

- Do we explain individual prediction ?
- Example :
- Heatmaps
- Rationales

- Do we explain entire model?
- Example :
- Linear Regression
- Decision Trees

Source: Byron Wallace

Inherent vs Post-hoc

- Is the explainability built into the model ?

- Examples:

- Linear Regression
- Decision Trees

- Is the model black-box and we use external method to try to understand it ?

- Examples:

- Heatmaps (Some forms)

Source: Byron Wallace

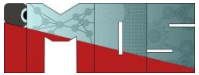
Model based vs Model Agnostic

- Can it explain only few classes of models?
- Examples:
- Decision Trees
- Attention
- Gradients (Differentiable Models only)

- Can it explain any model ?
- Examples:
- LIME – Locally Interpretable Model Agnostic Explanations
- SHAP – Shapley Values

Source: Byron Wallace

Different approaches to explain the behavior of ML approaches post-hoc



- Explaining with Surrogates
- Explaining with local perturbations
- Propagation-Based Approaches (Leveraging Structure)
- Meta-explanations
- Attribution-based methods
- Counterfactual explanations
- Interaction explanations
- Attention Mechanisms
- …

Explainable AI (1/2)

Feature Importance and Attribution Methods

- These methods identify the most influential features or components in making a prediction.
- **SHAP (SHapley Additive exPlanations)**: Based on cooperative game theory, SHAP assigns each feature a "Shapley value" that indicates its contribution to a particular prediction. SHAP values provide consistent and fair attribution by treating each feature as a "player" in a game.
- **LIME (Local Interpretable Model-agnostic Explanations)**: LIME approximates complex models by locally fitting an interpretable model (like linear regression) around a specific instance. It perturbs input features to see how predictions change, helping identify which features influence the prediction.
- **Integrated Gradients**: A technique for neural networks that calculates feature importance by integrating the gradients of a model's output with respect to its inputs, starting from a baseline (e.g., zero). Integrated gradients quantify each input's contribution to a model's prediction.
- **Permutation Feature Importance**: Measures feature importance by shuffling each feature in turn and observing the impact on model accuracy. If a feature's importance is high, shuffling its values will significantly degrade model performance.

Visualization Techniques

- Visualization is key to making model insights understandable for humans, especially in image and text processing.
- **Saliency Maps**: Often used in image processing, saliency maps highlight the parts of an image that have the most influence on a model's prediction. This can help in understanding what parts of an image a convolutional neural network (CNN) is focusing on.
- **Partial Dependence Plots (PDP)**: Show the relationship between one or two features and the predicted outcome, averaging out the effect of other features. PDPs help understand how the model's predictions change as the value of a feature changes.
- **Layer-wise Relevance Propagation (LRP)**: Assigns relevance scores to each neuron in the network, which can be aggregated to show which pixels or words are most influential in a decision. This technique is especially useful for deep neural networks.
- **t-SNE and UMAP**: Dimensionality reduction techniques that help visualize high-dimensional data in two or three dimensions, making it easier to observe clusters or patterns in the data that the model may have learned.

Explainable AI (2/2)

Surrogate Models

- **Surrogate Models**: These are simpler, interpretable models (like linear or decision tree models) trained to approximate a complex model's predictions. Surrogate models can provide a global understanding of a complex model's behavior by approximating its decision boundaries.

Counterfactual Explanations

- Counterfactual explanations help answer "what if" questions by indicating how a model's prediction would change if the input features were altered. For example, in loan applications, a counterfactual explanation might state, "If the applicant's income were $5,000 higher, the loan would be approved." This type of explanation is valuable for understanding decision boundaries and fairness.

Causal Inference and Causal Models

- Causal inference goes beyond correlation by trying to understand causal relationships between variables. Causal models can help identify which factors genuinely influence predictions rather than merely correlate with the outcome. These methods are particularly important in applications where knowing causality is crucial, such as medicine or social sciences.

Rule-Based Explanations and Decision Rules

- Some models, such as decision trees and rule-based models, are naturally interpretable. These models use a series of if-then rules that are easy for humans to follow. For more complex models, rule extraction techniques can derive decision rules that approximate the model's decision process, making it more interpretable.

Prototype and Example-Based Explanations

- **Prototypes**: These are representative examples from the training set that encapsulate common patterns or features. Showing a prototype can help users understand typical cases within a class.
- **Influence Functions**: These methods determine which training examples are most responsible for a specific prediction. They help identify cases in the training data that the model relied on heavily, which can be especially useful for diagnosing model biases.

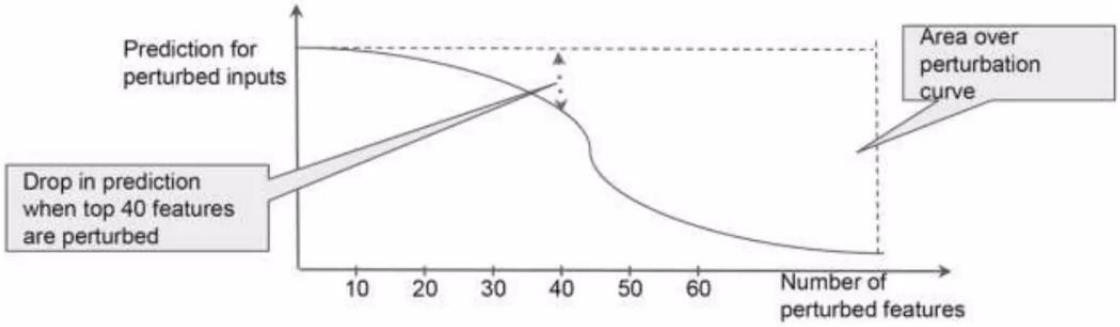
Bayesian and Probabilistic Approaches

- Probabilistic models, such as Bayesian networks, can inherently quantify uncertainty in their predictions. By providing a probabilistic interpretation of model decisions, these methods allow users to understand the confidence level associated with each prediction, helping to address questions of model reliability.

Perturbation-based approaches

Perturb top-k features by attribution and observe change in prediction

- Higher the change, better the method
- Perturbation may amount to replacing the feature with a random value
- Samek et al. formalize this using a metric: **Area over perturbation curve**
  - Plot the prediction for input with top-k features perturbed as a function of k
  - Take the area over this curve



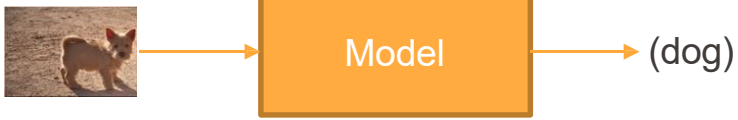
Source: KDD 2019 Tutorial

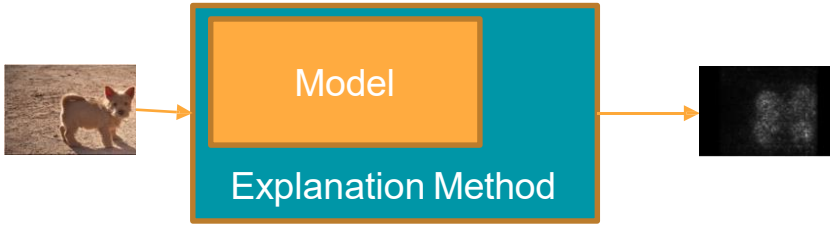
Attribution-based methods

- Attribute a model's prediction on an input to features of the input
- Attribiution methods
  - Ablations (drop each feature and attribute the change in prediction to that feature)
  - Gradient-based methods (attribution to a feautre is feature value times gradient)
  - Score backpropation based methods
    - **Guided BackProp**: Only consider ReLUs that are on (linear regime), and which contribute positively
    - **LRP**: Use first-order Taylor decomposition to linearize activation function
    - **DeepLift**: Distribute activation difference relative a reference point in proportion to edge weights

Saliency Based Methods

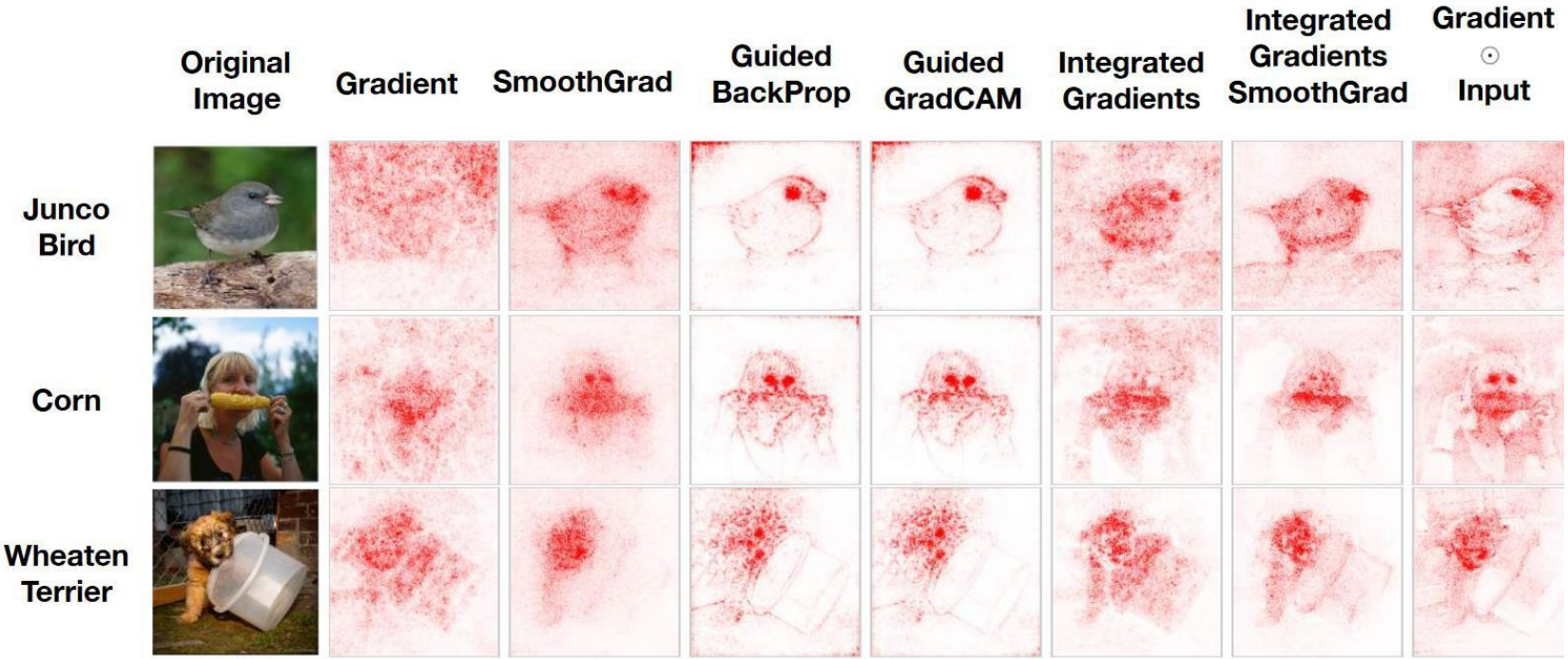
- Heatmap based visualization
- Need differentiable model in most cases
- Normally involve gradient





Source: Byron Wallace

Saliency Based Methods



[Adebayo et al 2018]

Saliency Example - Gradients

$$f(x): R^d \rightarrow R$$

$$E(f)(x) = \frac{df(x)}{dx}$$

Source: Byron Wallace

Saliency Example – Leave-one-out

$$f(x): R^d \rightarrow R$$

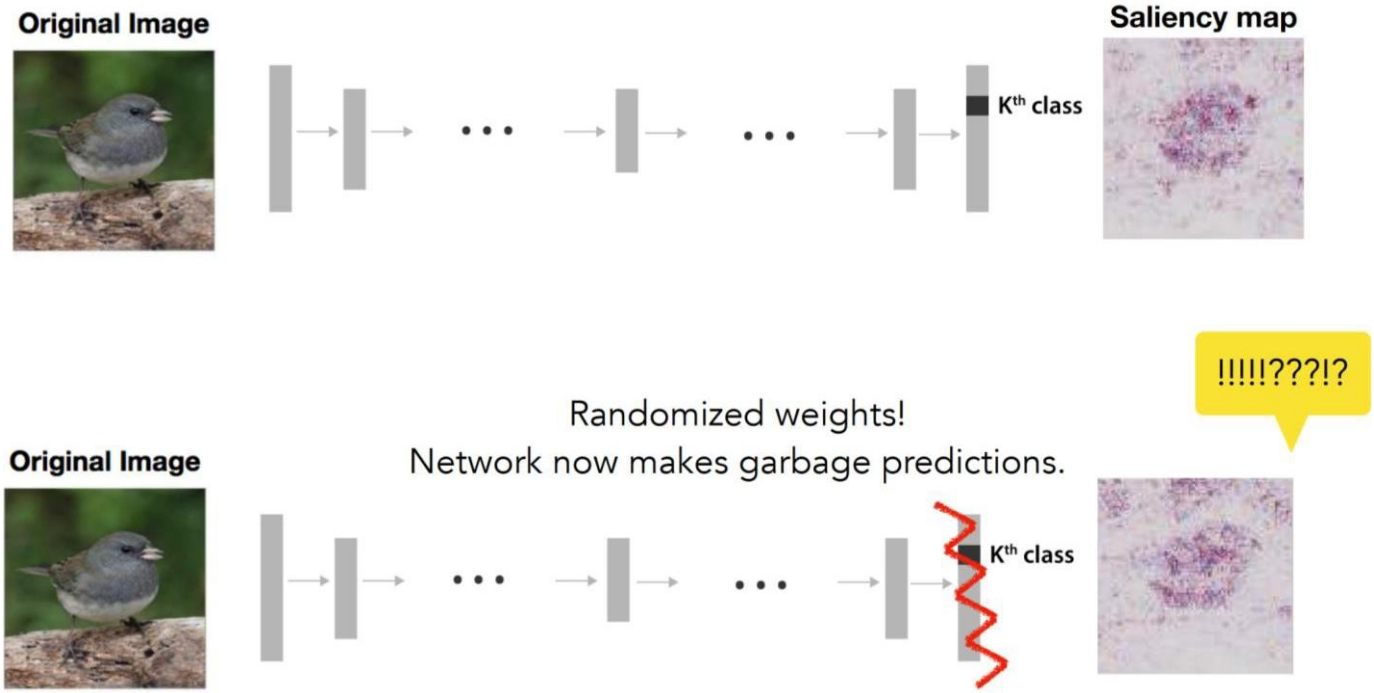
$$E(f)(x)_i = f(x) - f(x\backslash \mathrm{i})$$

How to remove ?

1. Zero out pixels in image
2. Remove word from the text
3. Replace the value with population mean in tabular data

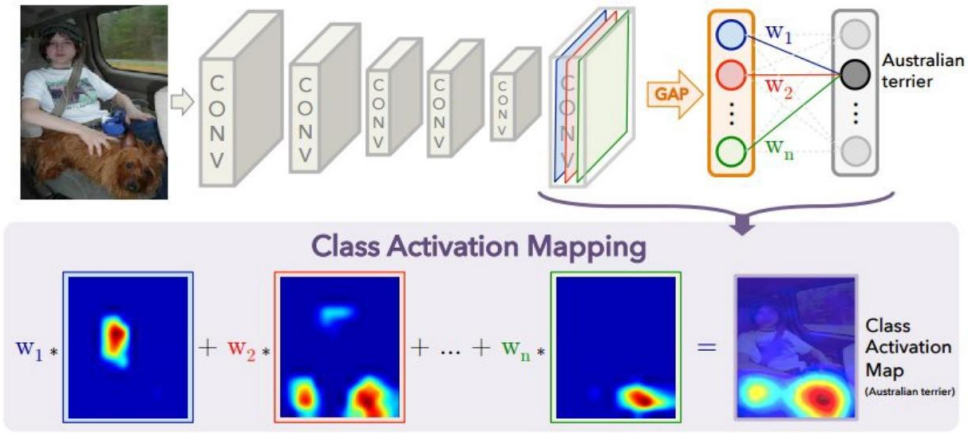
Source: Byron Wallace

Sanity check: When predictions change, do the explanations change?



Source: Julius Adebayo

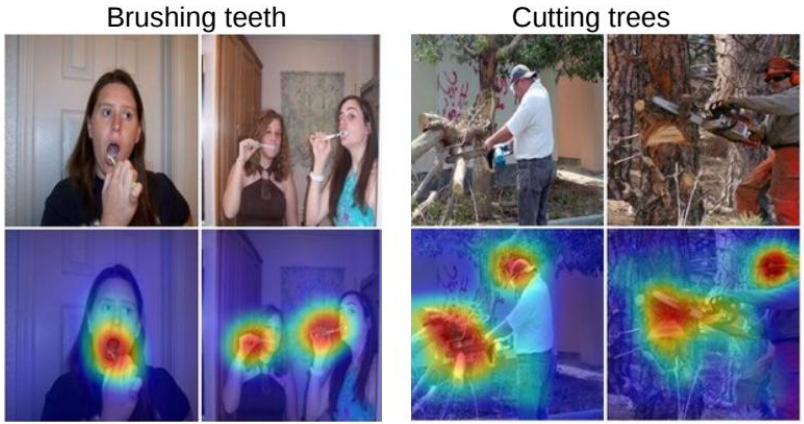
Class Activation Mapping

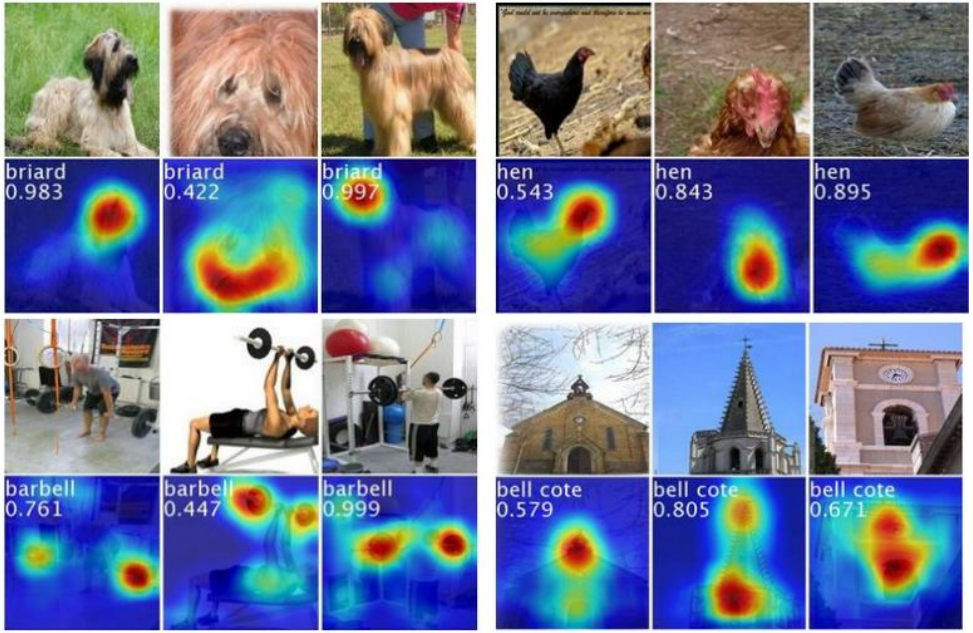


- GAP: Global average pooling
- We can identify the importance of the image regions by projecting back the weights of the output layer on the convolutional feature maps obtained from the last Convolution Layer.

Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

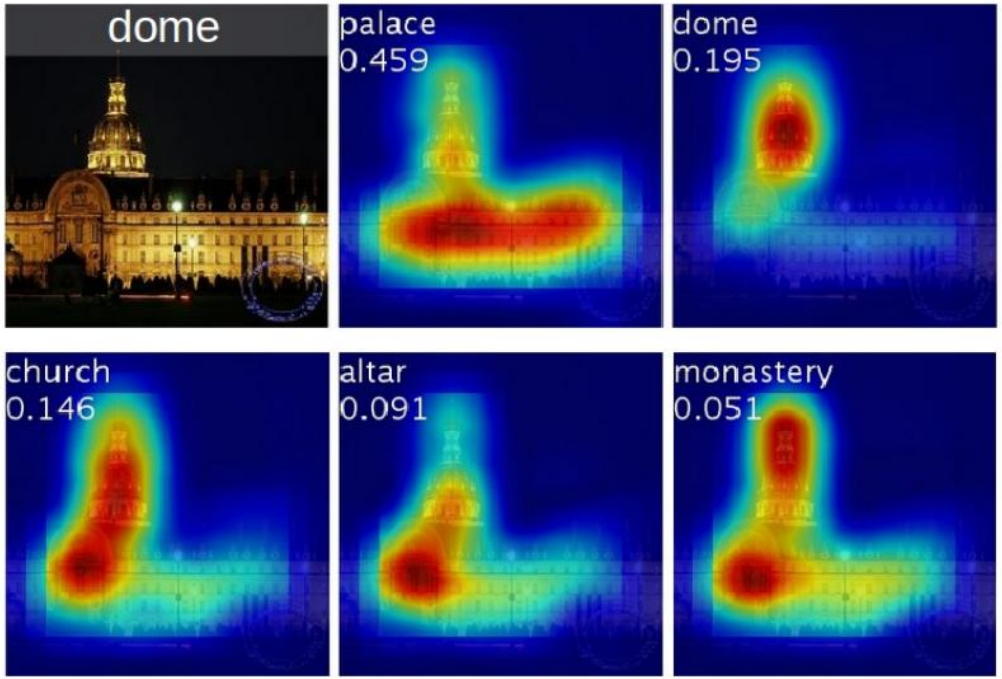
Class Activation Mapping: examples





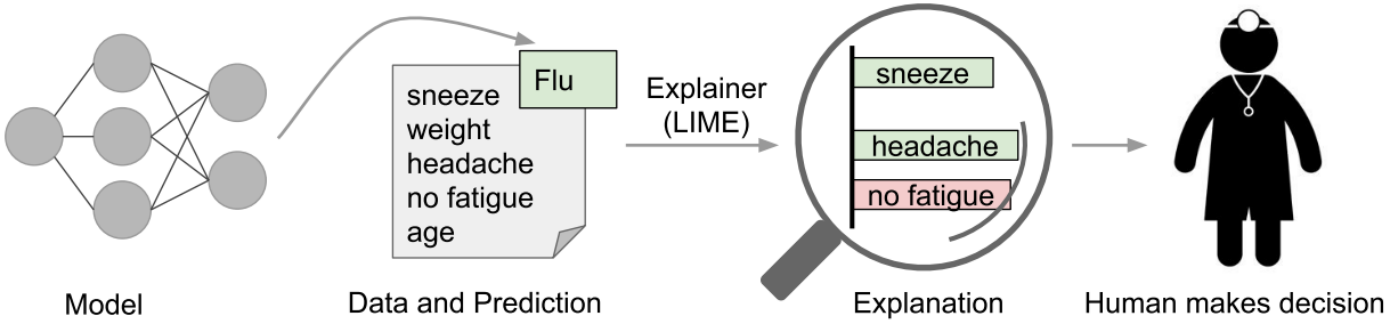
Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Class Activation Mapping



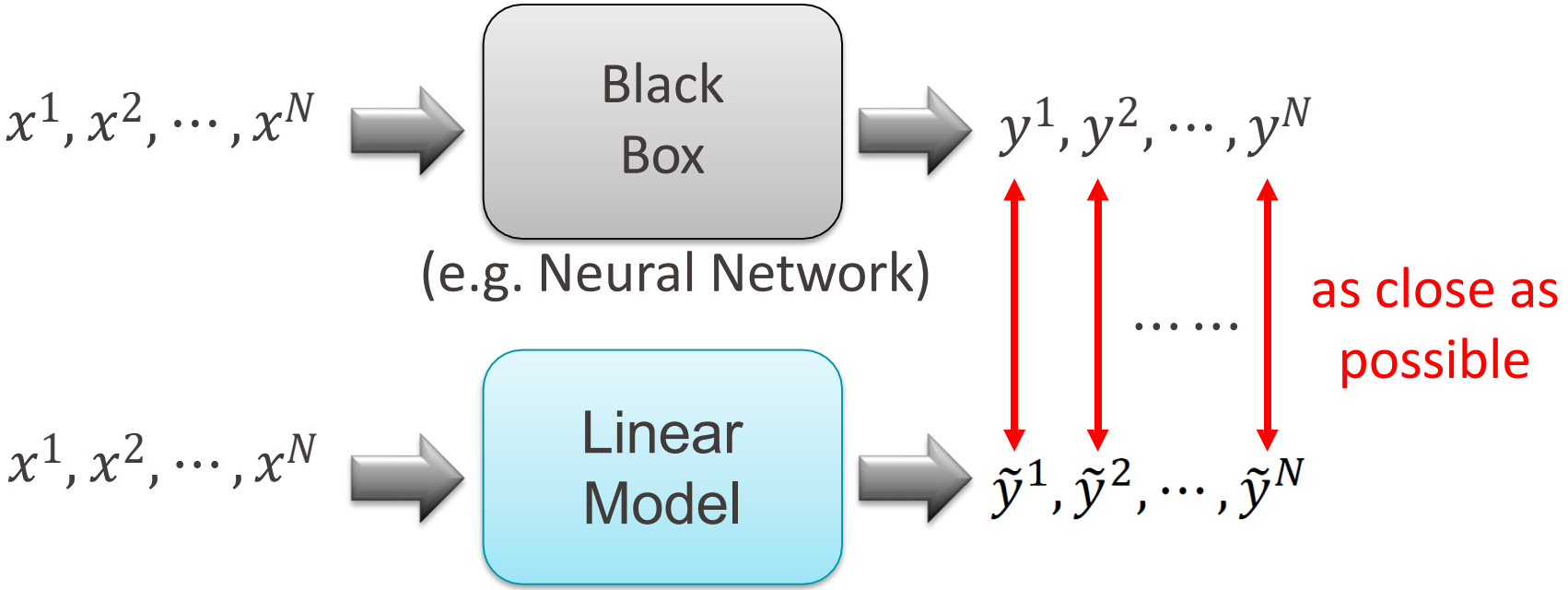
Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Local Interpretable Model-Agnostic Explanations (LIME): basic idea



M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

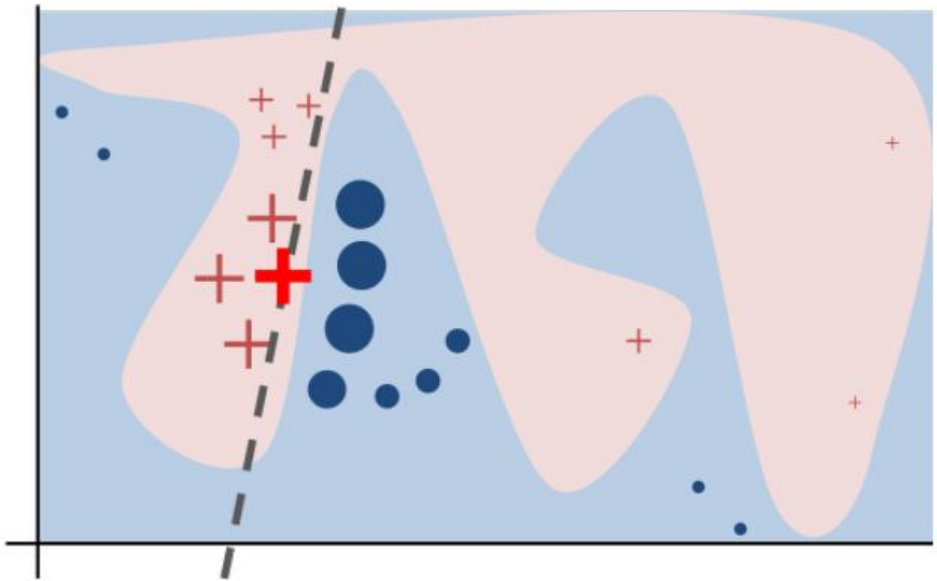
LIME – locally interpretable model agnostic



Can't do it globally of course, but locally ? Main Idea behind LIME

Source: Hung-Yi Lee

LIME: Toy example of the basic concept



M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

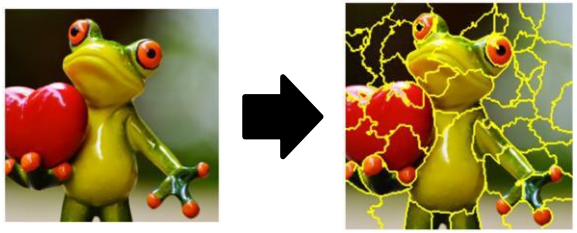
LIME: divide images into interpretable components (contiguous superpixels)

Original Image

Interpretable Components

M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

LIME – Image



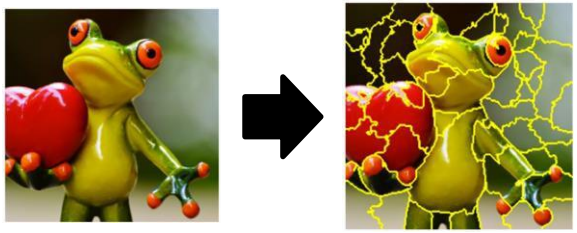
1. Given a data point you want to explain
2. Sample at the nearby - Each image is represented as a set of superpixels (segments).

- Randomly delete some segments.

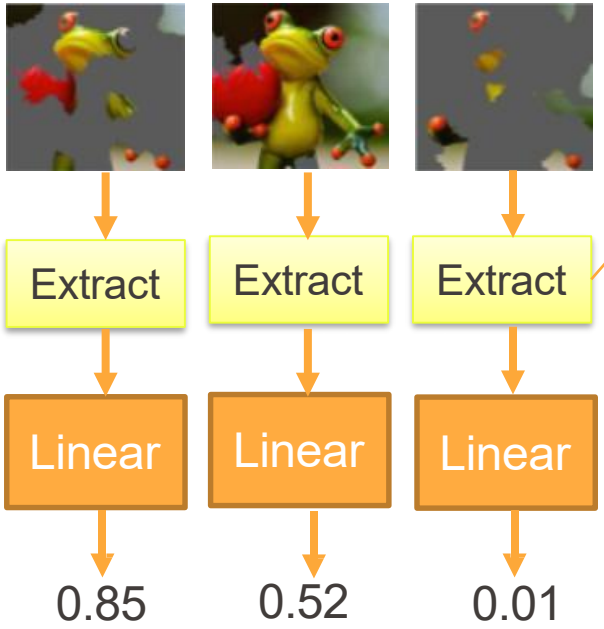
Compute the probability of “frog” by black box

Source: Hung-Yi Lee

LIME – Image



3. Fit with linear (or interpretable) model



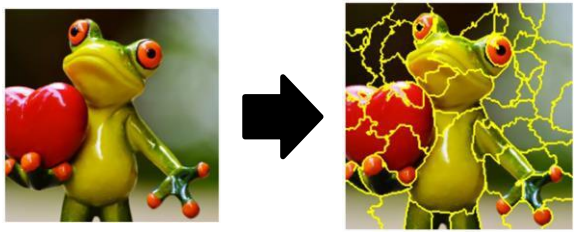
$x_1 \quad \cdots\cdots \quad x_m \quad \cdots\cdots \quad x_M$

$$x_m = \begin{cases} 0 & \text{Segment m is deleted.} \\ 1 & \text{Segment m exists.} \end{cases}$$

M is the number of segments.

Source: Hung-Yi Lee

LIME – Image



4. Interpret the model you learned

$$y = w_1 x_1 + \cdots + w_m x_m + \cdots + w_M x_M$$

$$x_m = \begin{cases} 0 & \text{Segment m is deleted.} \\ 1 & \text{Segment m exists.} \end{cases}$$

M is the number of segments.

If $w_m \approx 0$ ➡ segment m is not related to “frog”

If w_m is positive ➡ segment m indicates the image is “frog”

If w_m is negative ➡ segment m indicates the image is not “frog”

Source: Hung-Yi Lee

LIME: underlying algorithm

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K

$\mathcal{Z} \leftarrow \{\}$
for $i \in \{1, 2, 3, ..., N\}$ **do**
  $z'_i \leftarrow sample\_around(x')$
  $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$
end for
$w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ ▷ with z'_i as features, $f(z)$ as target
return w

Match interpretable model to black box

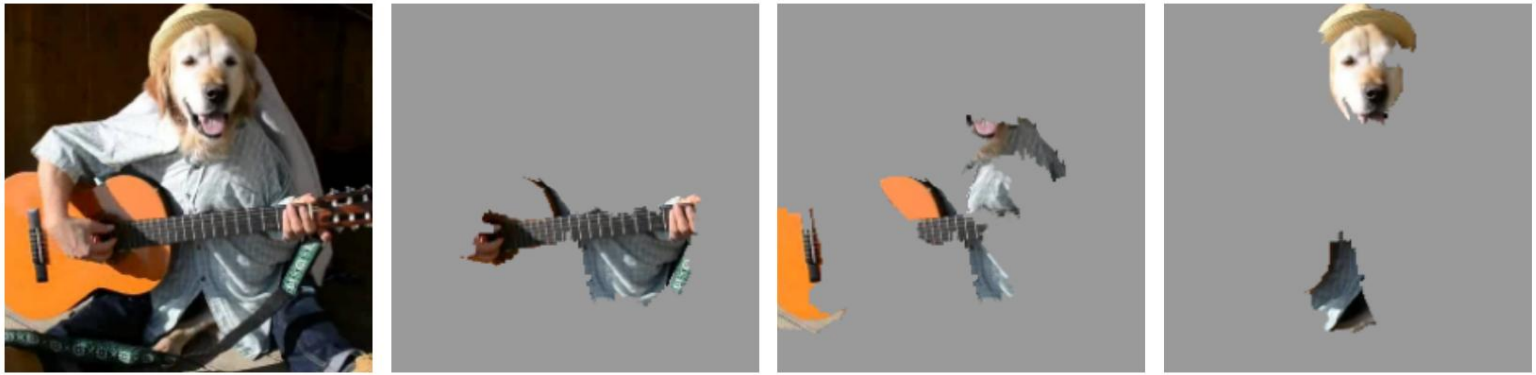
Control complexity of the model

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

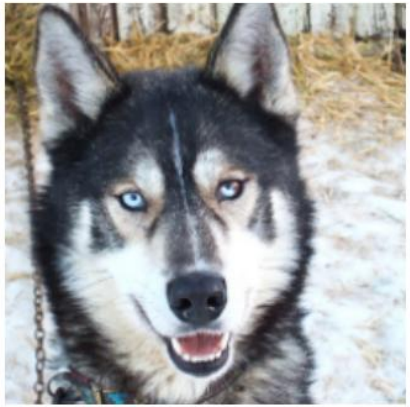
LIME: Example



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

LIME: bringing trust («Husky vs Wolf)



(a) Husky classified as wolf

(b) Explanation

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 1135–1144.

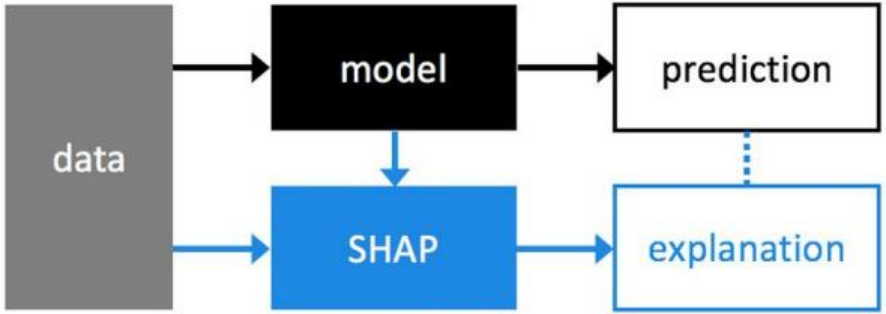
SHAP (Shapley values Additive exPlanations) – additive feature attribution method to explain the output of any ML model

Classic result in game theory on distributing gain in a **coalition game**

- **Coalition Games**
  - Players collaborating to generate some **gain** (think: revenue)
  - Set function v(S) determining the gain for any subset S of players
- **Shapley Values** are a fair way to attribute the total gain to the players based on their contributions
  - Concept: **Marginal contribution** of a player to a subset of other players (v(S U {i}) - v(S))
  - Shapley value for a player is a **specific weighted aggregation of its marginal** over all possible subsets of other players

Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

SHAP (Shapley values Additive exPlanations) – additive feature attribution method to explain the output of any ML model



The Shapley value is a mathematical concept in game theory that was introduced by Lloyd Shapley in 1951.

Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

Shapley values

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right]$$

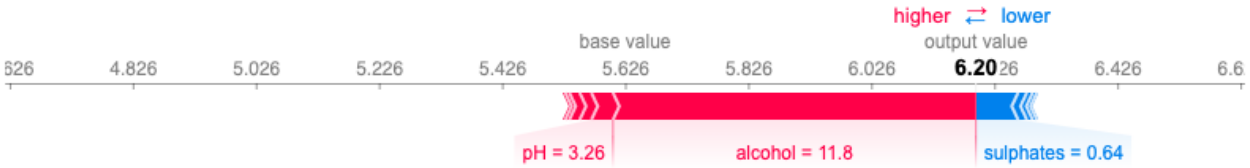
where S feature subset, F – set of all features, $S \subseteq F \setminus \{i\}$ all possible subsets

$f_{S \cup \{i\}}$ is trained with that feature present and f_S with that feature withheld

predictions from the two models are then compared $f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)$

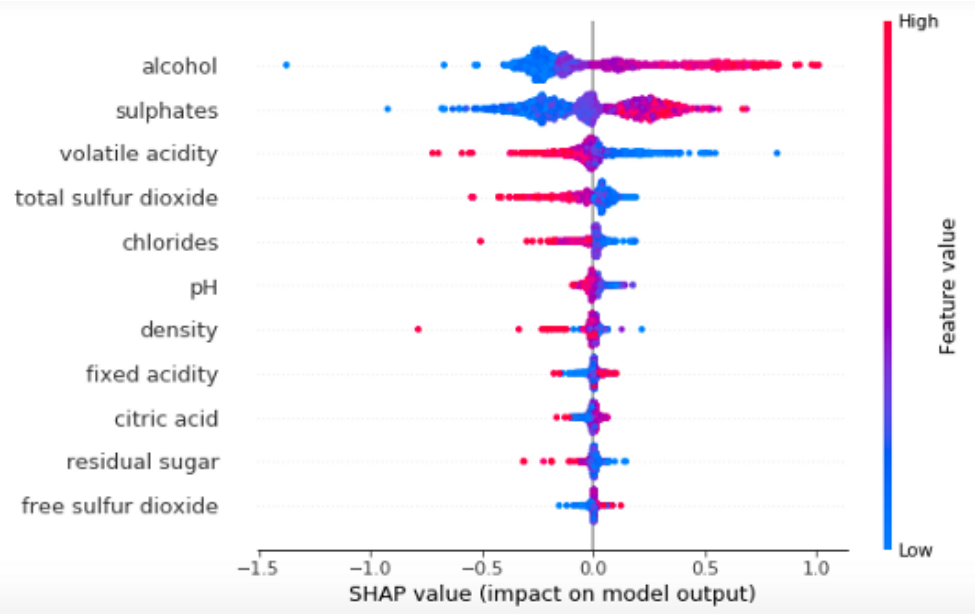
Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

Individual SHAP value plot



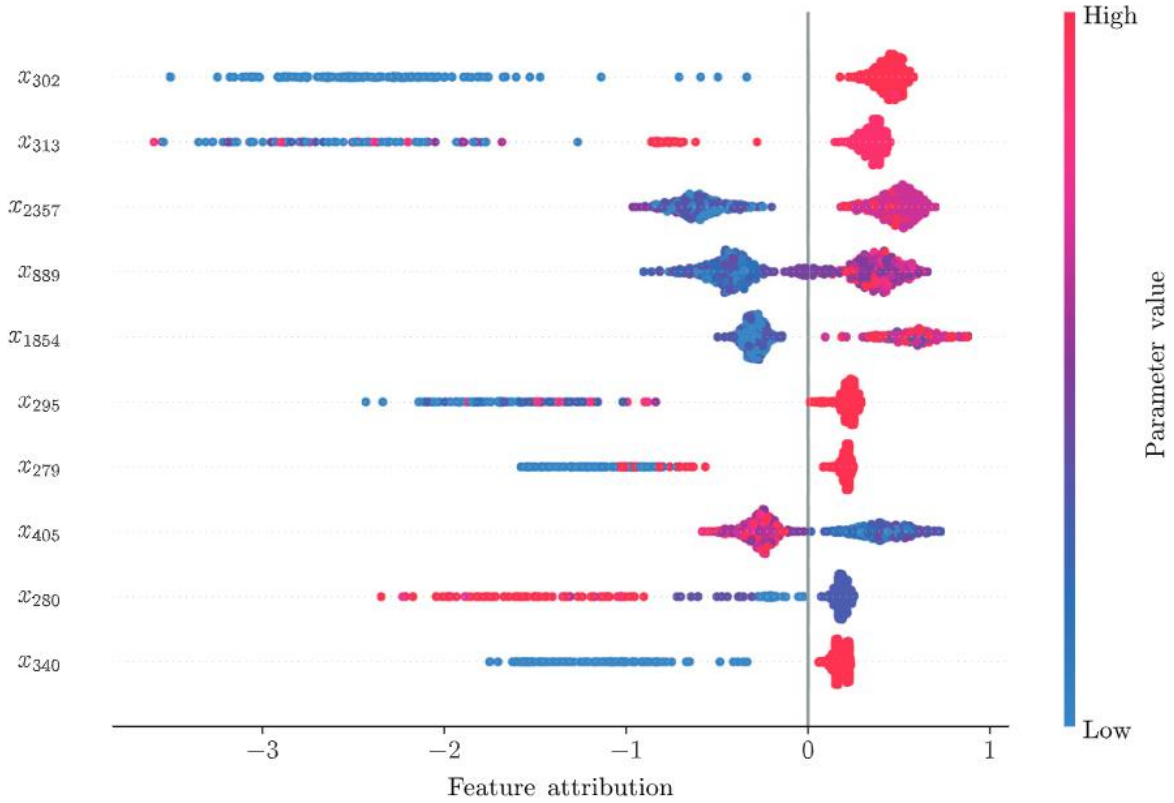
Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

SHAP variables importance plot



Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

Drivers of the semiconductor production quality



Senoner, J., Netland, T., & Feuerriegel, S. (2022). Using explainable artificial intelligence to improve process quality: Evidence from semiconductor manufacturing. *Management Science*, *68*(8), 5704-5723.

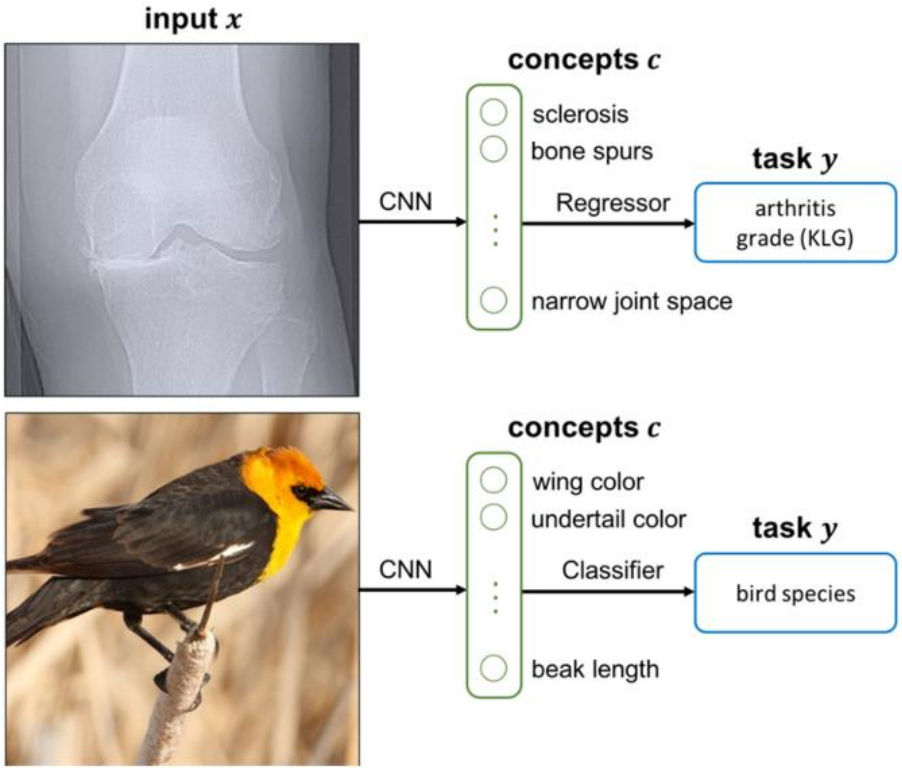
What Are CBMs?

- Two-stage architecture:

1. Predict interpretable concepts from input
2. Predict output from those concepts

- Makes the decision process transparent and modular

Concept bottleneck models



Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., & Liang, P. (2020, November). Concept bottleneck models. In *International conference on machine learning* (pp. 5338-5348). PMLR.

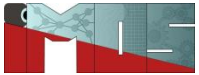
Benefits of CBMs

- ✓Interpretability: Understand model reasoning
- ✓Debuggability: Trace and fix errors
- ✓Human Intervention: Override concept predictions
- ✓Generalization: Encourages disentangled features

Challenges

- Requires labeled concepts during training
- Concept leakage (information shortcut)
- May limit model performance if concepts are incomplete or noisy

What Are Concept Embedding Models (CEMs)?

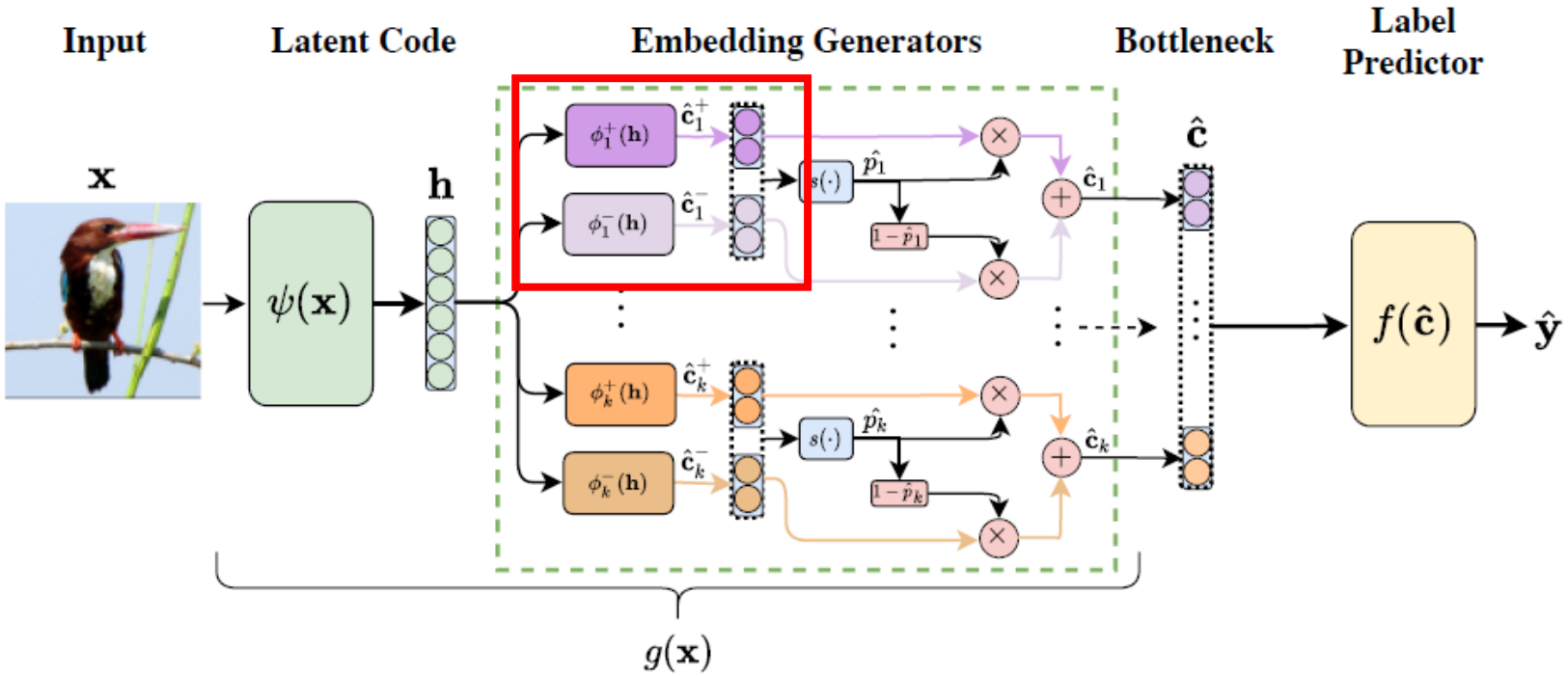


- Models that learn a latent concept space where each dimension aligns with semantically meaningful features
- Concepts are supervised but embedded, allowing compact, flexible representation
- Enables downstream tasks to use interpretable embeddings

CEM Architecture

- Input → Encoder → Concept Embedding (latent space) → Decoder
- Concept embedding trained to satisfy:
  - Disentanglement
  - Compactness
  - Predictive utility
  - Partial concept supervision

Concept Embedding Model



Espinosa Zarlenga, Mateo, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." Advances in Neural Information Processing Systems 35 (2022): 21400-21413.