OOD Generalization

Classical machine learning algorithms are constructed on the *i.i.d.* assumption. As you know, in the real world, this assumption is hardly satisfied (high-stake applications such as healthcare, military, and autonomous driving) since the test distribution may be shifted and different from the train distribution, so performance may decrease. Hence OOD Generalization is raised.

The distribution shift can occur for many reasons, such as the temporal/spatial evolution of data or the sample selection bias in the data collection process.

The test distribution is usually unknown due to the nature of applications like stream-based online scenarios, where test data are generated in the future.

OOD generalization problem can be defined as instantiating a supervised learning problem.

supervised learning: $$f_\theta^* = \arg\min_{f_\theta} \mathbb{E}_{X,Y\sim P_{te}}[\ell(f_\theta(X), Y)]$$

i.i.d. Learning: $$\mathcal{L}_{ERM} = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) \quad , P_{tr}(X,Y) = P_{te}(X,Y)$$

Generally, the OOD problem is infeasible unless we make some assumptions on how the test distribution may change. Among the different distribution shifts, the covariate shift is the most common one and has been studied in depth (other forms of distribution shifts exist, like label shift and concept shift). Specifically, the covariate shift is defined as $P_{tr}(Y|X) = P_{te}(Y|X)$ and $P_{tr}(X) \neq P_{te}(X)$, which means the marginal distribution of X shifts from training phase to test phase while the label generation mechanism keeps unchanged.

for evaluating the OOD performance we require specific datasets and evaluation metrics, as the classic ways for *i.i.d.* setting are inapplicable under distributional shifts. And this also motivates the generation of different datasets and evaluations.

Methods for designing an algorithm with good OOD Generalization performance:

1. **unsupervised representation learning methods:** Includes Disentangled Representation Learning and Causal Representation Learning, which exploits the unsupervised representation learning techniques (e.g., variational Bayes) to embed prior knowledge into the learning process.

2. **supervised learning models:** Includes Causal Learning (formulate training and testing distributions with causal structures and the distributional shifts mainly originate from interventions or confounders), Stable Learning (introduce distributional shifts via selection bias), and Domain Generalization (mainly focus on real scenarios and utilize data collected from different domains), which designs various model architectures and learning strategies to achieve OOD generalization.
3. **optimization methods**: Includes Distributionally Robust Optimization and Invariance-Based Optimization, which directly formulates the objective of OOD generalization and conduct optimization with theoretical guarantees for OOD optimality.

Theoretical Connections:

For branches of methods for OOD generalization, there are some inherent connections among them. For example, the connections among causal learning methods, distributionally robust optimization (DRO) methods, and stable learning methods.

Fairness:

Fairness has recently been linked to OOD issues. Generally speaking, subgroups split by sensitive attributes in fairness literature correspond to environments in OOD literature. In addition, both areas need to specify learning objectives with respect to the subgroups/environments.

Explainability:

Causality has recently been introduced to model explanation, especially in deep learning methods. Actually, causality is the crux for both OOD and explainability. The models will have good OOD performance and explainability simultaneously if they utilize the causal relationship between the features and the outcomes.

Anomaly Detection

Anomaly detection, a.k.a. outlier detection or novelty detection, is referred to as the process of detecting data instances that significantly deviate from the majority of data instances.

Due to the increasing demand and applications in broad domains, such as risk management, compliance, security, financial surveillance, health and medical risk, and AI safety, anomaly detection plays increasingly important roles, highlighted in various communities, including data mining, machine learning, computer vision, and statistics.

In recent years, deep learning has shown tremendous capabilities in learning expressive representations of complex data such as high-dimensional data, temporal data, spatial data, and graph data, pushing the boundaries of different learning tasks. Deep learning for anomaly detection aim at learning feature representations or anomaly scores via neural networks for the sake of anomaly detection.

PROBLEM COMPLEXITIES AND CHALLENGES:

- **Unknownness:** Anomalies are associated with many unknowns, e.g., instances with unknown abrupt behaviors, data structures, and distributions. They remain unknown until they actually occur, such as novel terrorist attacks, frauds, and network intrusions.
- **Heterogeneous anomaly classes:** Anomalies are irregular, and thus, one class of anomalies may demonstrate completely different abnormal characteristics from another class of anomalies. For example, in video surveillance, abnormal events, robbery, traffic accidents, and burglary are visually highly different.
- **Rarity and class imbalance:** Anomalies are typically rare data instances, contrasting to normal instances that often account for an overwhelming proportion of the data. Therefore, it is difficult, if not impossible, to collect a large amount of labeled abnormal instances.
- **Diverse types of anomaly:** Three completely different types of anomaly have been explored. 1.Point anomalies. 2.Conditional anomalies, a.k.a. contextual anomalies. 3.Group anomalies, a.k.a. collective anomalies.
- **Low anomaly detection recall rate:** Since anomalies are highly rare and heterogeneous, it is difficult to identify all of the anomalies. Actually, how to reduce false positives and enhance detection recall rates is one of the most important and yet difficult challenges.
- **Anomaly detection in high-dimensional and/or not-independent data:** Anomalies often exhibit evident abnormal characteristics in a low-dimensional space yet become hidden and unnoticeable in a high-dimensional space.
- **Data-efficient learning of normality/abnormality:** Due to the difficulty and cost of collecting large-scale labeled anomaly data, fully supervised anomaly detection is often impractical as it assumes the availability of labeled training data with both normal and anomaly classes. In the last decade, major research efforts have been focused on unsupervised anomaly detection that does not require any labeled training data. Semi-supervised anomaly detection, which assumes a set of labeled normal training data, is a research direction dedicated to this problem. Another research line is weakly-supervised anomaly detection.
- **Noise-resilient anomaly detection:** Many weakly/semi-supervised anomaly detection methods assume the labeled training data is clean, which can be vulnerable to noisy instances that are mistakenly labeled as an opposite class label. In such cases, we may use unsupervised methods instead, but this fails to utilize the genuine labeled data. Additionally, there often exists large-scale anomaly-contaminated unlabeled data. Noise-resilient models can leverage those unlabeled

data for more accurate detection. Thus, the noise here can be either mislabeled data or unlabeled anomalies. The main challenge is that the amount of noise can differ significantly from datasets, and noisy instances may be irregularly distributed in the data space.

- **Detection of complex anomalies:** Most of the existing methods are for point anomalies, which cannot be used for conditional anomaly and group anomalies since they exhibit completely different behaviors from point anomalies. Also, current methods mainly focus on detecting anomalies from single data sources, while many applications require the detection of anomalies with multiple heterogeneous data sources.
- **Anomaly explanation:** In many safety-critical domains, there may be some major risks if anomaly detection models are directly used as black-box models. To derive anomaly explanation from specific detection methods is still a largely unsolved problem, especially for complex models.

Algorithms (popular methods):

Supervised methods: Bayesian networks, k-nearest neighbors, decision trees, supervised neural networks, and SVMs.

Unsupervised methods: Autoencoders, K-means, GMMs, hypothesis tests-based analysis, and PCAs.

Categorization of Deep Anomaly Detection:

- Deep anomaly detection
    - Deep learning for feature extraction
    - Learning feature representations of normality
        - Generic normality feature learning
            - Autoencoders
            - Generative adversarial networks
            - Predictability modeling
            - Self-supervised classification
        - Anomaly measure-dependent feature learning
            - Distance-based measures
            - One-class classification measures
            - Clustering-based measures
    - End-to-end anomaly score learning
        - Ranking models
        - Prior-driven models
        - Softmax likelihoold models
        - End-to-end one-class classification

Explainable AI

Explainable AI is a research field on ML interpretability techniques whose aims are to understand machine learning model predictions and explain them in human and understandable terms to build trust with stakeholders. Explainable AI is a key part of broader, human-centric responsible AI practices. Interpretability focuses on model understanding techniques, while explainability more broadly focuses on model explanations and the interface for translating these explanations in human, understandable terms for different stakeholders.

At a high level, there are three **stakeholder** groups for machine learning systems that drive the dual aims of model understanding and providing human interpretable explanations to different stakeholders in order to build trust. For **engineers** building models, the focus is more on interpretable ML techniques for model understanding and improving model performance. More complex models can be challenging to debug, understand, and control, thereby impeding their adoption. Conversely, trust is critical for model **consumers** looking to understand the impact of model predictions more than the model internals. Interpretable explanations build trust with these end-users that model decisions are reliable, equitable, and can be influenced to achieve better outcomes. The role of **regulators** is to ensure model decisions comply with laws and do not amplify undesirable bias from underlying data sets. Interpretable explanations provide audible metadata to regulators to trace unexpected predictions back to their inputs to inform corrective actions.

Model understanding is critical to many tasks as part of building and operating machine learning systems:

1. explain predictions to inform and support human decision-making processes.
2. debugging model performance to inform corrective actions.
3. refining modeling and data collection processes.
4. verifying model behavior is acceptable.
5. presenting the model's predictions to stakeholders.

There's no agreed-upon mathematical definition of interpretability. Interpretability is about how a cause (this can be a change in a future value, group of features, or model parameters) can affect your model's predictions. Or, to put it another way, it is the extent to which you, the modeler, can predict what will happen given a change in input or algorithm parameters.

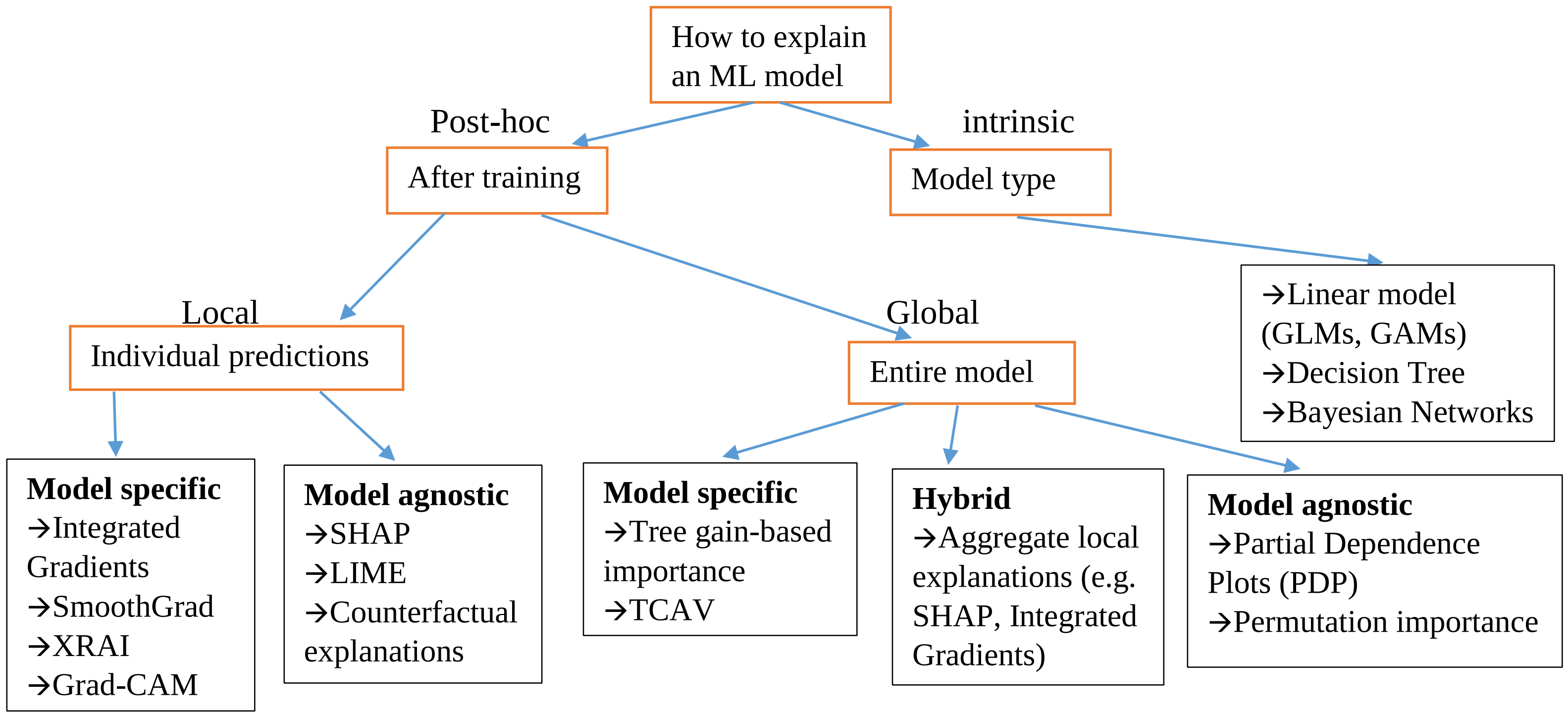
The success of deep learning and achieving high performance has come with a trade-off in explainability. This has resulted in some high-profile ML deployment failures from these black-box models. Many were too opaque, uncontestable, exhibited unreliable behavior, and worse, reinforced undesirable biases that resulted in poor outcomes for many stakeholders. In response, the field of explainable AI has shifted considerable attention to interpretable ML methods to unpack these more complex deep learning models.

Some criteria of what comprises a good explanation:

Each ML interpretability method comes with different guarantees and limitations, computational requirements, and explanation output. When picking a method, it's important to always consider your models decisions within the context of the requirements of the broader system that it operates in. There are a few general properties of explanations that can be used to guide a selection of ML interpretability methods. explanations must be:

1. Complete 2. Accurate 3. Meaningful 4. Consistent

Taxonomy: Interpretable Machine Learning methods



Intrinsic:

- Restricting the complexity of the machine learning model
- Simple structure

Post-hoc:

- Applying methods that analyze trained models
    - e.g. Permutation feature importance
- Can be applied to intrinsically interpretable models

Local:

- Interpretability of individual predictions or a small part of the model's prediction space.
- Higher precision but lower recall understanding of model behavior.

Global:

- Aggregated, ranked contributions of input variables for the entire model's prediction space.
- Higher recall view of the entire model prediction space, but lower precision due to aggregations e.g. averages.

Model Specific:

- Only work for specific models due to definition
    - e.g. neural network gradients, tree-based feature importances

Model Agnostic:

- Portable across model definitions
    - e.g. Permutation feature importances, explainable surrogates

Explanation type:

Many methods return feature statistics, measuring a feature's proportional contribution to the prediction, while others extract concepts, decision rules, feature summary visualizations, counterfactual data points, or even simpler approximate models.

Note:

Integrated gradients can highlight the pixels critical to that decision, giving you insight into what the model sees.

Uncertainty

These factors are mainly based on an uncertainty already included in the data (data uncertainty) or a lack of knowledge of the neural network (model uncertainty):

- the lack of expressiveness and transparency of a deep neural network's inference model, which makes it difficult to trust their outcomes.
- the inability to distinguish between in-domain and out-of-domain samples and the sensitivity to domain shifts.
- the inability to provide reliable uncertainty estimates for a deep neural network's decision and frequently occurring overconfident predictions.
- the sensitivity to adversarial attacks that make deep neural networks vulnerable for sabotage.

Sources and types of uncertainty:

consider four different steps from the raw information in the environment to a prediction by a neural network with quantified uncertainties:

1) the data acquisition process: The occurrence of some information in the environment and a measured observation of this information.

2) the DNN building process: The design and training of a neural network.

3) the applied inference model: The model which is applied for inference (e.g. a Bayesian neural network or an ensemble of neural networks).

4) the prediction's uncertainty model: The modelling of the uncertainties caused by the neural network and by the data (an additional explicit modeling of *distributional uncertainty* is used to model the uncertainty, caused by examples from a region not covered by the training data).

these four steps contain several potential sources of uncertainty and errors, which again affect the final prediction of a neural network. The five factors that are the most vital for the cause of uncertainty in a DNN's predictions are:

• the variability in real world situations

• the errors inherent to the measurement systems

• the errors in the architecture specification of the DNN

• the errors in the training procedure of the DNN

• the errors caused by unknown data

On the basis of the input data domain, the predictive uncertainty can also be classified into three main classes:

1. In-domain uncertainty 2. Domain-shift uncertainty 3. Out-of-domain uncertainty

the methods for estimating the uncertainty can be split in four different types based on the number (single or multiple) and the nature (deterministic or stochastic) of the used DNNs:

- **Single deterministic methods:** give the prediction based on one single forward pass within a deterministic network. The uncertainty quantification is either derived by using additional (external) methods or is directly predicted by the network.
    - External Methods
        1. Gradient Metrics 2. Additional Network for Uncertainty 3. Distance to Training Data
    - Internal Methods
        1. Prior Networks 2. Evidential Neural Networks 3. Gradient penalties
- **Bayesian methods:** cover all kinds of stochastic DNNs, i.e. DNNs where two forward passes of the same sample generally lead to different results.
    - Variational Inference
        1. Application of Variational Inference 2. Stochastic Variational Inference 3. Normalizing flows 4. Monte-Carlo Dropout
    - Sampling Methods
        1. Original works 2. Stochastic MCMC 3. Theoretic Advances
    - Laplace Approximation
        1. Diagonal Information Matrix 2. KroneckerFactorization 3. Sparse Information Matrix
- **Ensemble methods:** combine the predictions of several different deterministic networks at inference.
    - Weight Sharing
        1. Sub-Ensembles 2. Batch-Ensembles
    - Reduce Members
        1. Model Pruning 2. Distillation
    - Training Strategies
        1. Random Initialization/ Data Shuffling 2. Bagging/ Boosting 3. Single Training Run

- **Test-time augmentation methods:** give the prediction based on one single deterministic network but augment the input data at test-time in order to generate several predictions that are used to evaluate the certainty of the prediction.
    - Augmentation Policies

Note:

In order to use the presented methods on real-life tasks, several different considerations have to be taken into account. The memory and computational power are often restricted, while many real-world tasks may be time-critical.

there are several reasons why evaluating the quality of the uncertainty estimates is a challenging task:

- First, the quality of the uncertainty estimation depends on the underlying method for estimating uncertainty.
- Second, there is a lack of ground truth uncertainty estimates and defining ground truth uncertainty estimates is challenging.
- Third, there is a lack of a unified quantitative evaluation metric. To be more specific, the uncertainty is defined differently in different machine learning tasks such as classification, segmentation, and regression. For instance, prediction intervals or standard deviations are used to represent uncertainty in regression tasks, while entropy (and other related measures) are used to capture uncertainty in classification and segmentation tasks.

Uncertainty measures and quality:

- Evaluating uncertainty in classification tasks:

  For classification tasks, the network's softmax output already represents a measure of confidence. But since the raw softmax output is neither very reliable nor can it represent all sources of uncertainty, further approaches and corresponding measures were developed.
    - Measuring data, model, and distributional uncertainty in classification tasks
    - Performance measure on complete data set
- Evaluating uncertainty in regression tasks
    - Measuring data, and model uncertainty in regression predictions
- Evaluating uncertainty in segmentation tasks

Calibration

A predictor is called well-calibrated if the derived predictive confidence represents a good approximation of the actual probability of correctness. Therefore, in order to make use of uncertainty quantification methods, one has to be sure that the network is well calibrated.

The different types of uncertainty calibration methods:

- Post-Processing Methods
  1.Ensemble of Post Processing Models 2. Gaussian Processes 3. Histogram Binning 4. Temperature Scaling
- Regularization Methods
  1.Objective Function Modification 2. Data Augmentation 3. Label Smoothing 4. Exposure to OOD examples
- Uncertainty Estimation Approaches
  1. Ensemble of Neural Networks 2. Bayesian Neural Networks

Dataset and baselines

We should consider commonly used tasks and data sets for evaluating uncertainty estimation among existing works. Besides, a variety of baseline approaches are commonly used as a comparison against the methods proposed by the researchers.

Applications of uncertainty estimates

- Active Learning
- Reinforcement Learning
- Uncertainty in Real-World Applications
  - Medical Analysis
  - Robotics
  - Earth Observation(EO)

Open Set Recognition

Open Set Recognition (OSR) is where incomplete knowledge of the world exists at training time, and unknown classes can be submitted to an algorithm during testing, requiring the classifiers to not only accurately classify the seen classes but also effectively deal with unseen ones.

In a closed set (or static environment) assumption, the training and testing data are drawn from the same label and feature spaces. However, a more realistic scenario is usually open and non-stationary such as driverless, fault/medical diagnosis, etc., where unseen situations can emerge unexpectedly, which drastically weakens the robustness of these existing methods.

To meet this challenge, several related research topics have been explored, including lifelong learning transfer learning, domain adaptation, zero-shot, one-shot (few-shot) recognition/learning, open set recognition/classification, etc.

Recognition should consider four basic categories of classes as follows:

1. **known known classes (KKCs):** the classes with distinctly labeled positive training samples (also serving as negative samples for other KKCs), and even have the corresponding side-information like semantic/attribute information, etc.
2. **Known unknown classes (KUCs):** labeled negative samples, are not necessarily grouped into meaningful classes, such as the background classes, the Universum classes, etc.
3. **Unknown known classes (UKCs):** classes with no available samples in training but available side-information (e.g., semantic/attribute information) of them during training.
4. **Unknown unknown classes (UUCs):** classes without any information regarding them during training: not only unseen but also having no side-information (e.g., semantic/attribute information, etc.) during training.

The space far from known data (including KKCs and KUCs) is usually considered as *open space O*. So labeling any sample in this space as an arbitrary KKC inevitably incurs risk, which is called *open space risk* R_O: (where f denotes the measurable recognition function-the overall measure space S_O)

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f(x)dx}{\int_{S_o} f(x)dx}$$

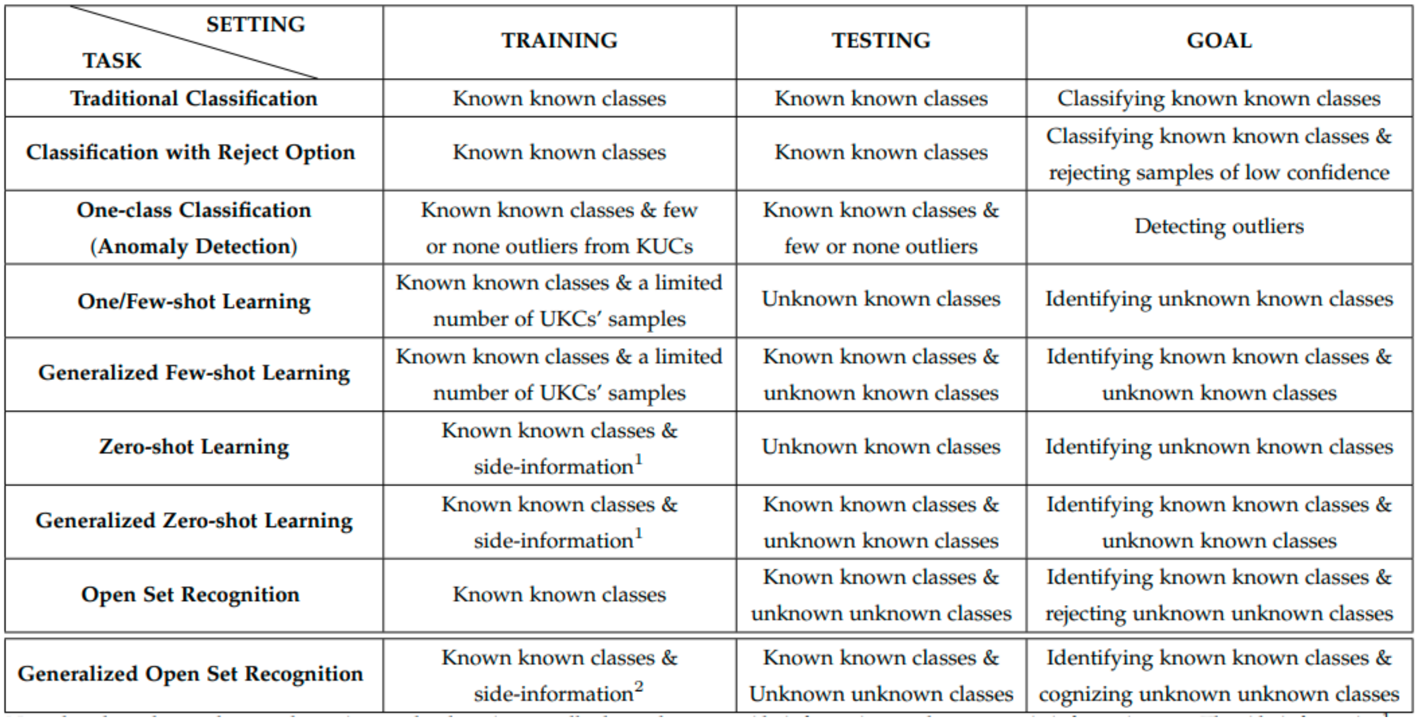
the openness of the corresponding recognition task O is: (Let CTA, CTR, and CTE respectively represent the set of classes to be recognized, the set of classes used in training and the set of classes used during testing.)

$$O = 1 - \sqrt{\frac{2 \times |C_{\mathrm{TR}|}}{|C_{\mathrm{TA}}| + |C_{\mathrm{TE}}|}}$$

The open set risk denoted as the following formula balances the empirical risk and the open space risk over the space of allowable recognition functions:(where $\lambda_r$is a regularization constant)

$$\arg\min_{f \in \mathcal{H}} \{R_{\mathcal{O}}(f) + \lambda_r R_{\varepsilon}(f(V))\}$$

Differences between Open Set Recognition and its related tasks:



SETTING TASK	TRAINING	TESTING	GOAL
Traditional Classification	Known known classes	Known known classes	Classifying known known classes
Classification with Reject Option	Known known classes	Known known classes	Classifying known known classes & rejecting samples of low confidence
One-class Classification (Anomaly Detection)	Known known classes & few or none outliers from KUCs	Known known classes & few or none outliers	Detecting outliers
One/Few-shot Learning	Known known classes & a limited number of UKCs' samples	Unknown known classes	Identifying unknown known classes
Generalized Few-shot Learning	Known known classes & a limited number of UKCs' samples	Known known classes & unknown known classes	Identifying known known classes & unknown known classes
Zero-shot Learning	Known known classes & side-information[1]	Unknown known classes	Identifying unknown known classes
Generalized Zero-shot Learning	Known known classes & side-information[1]	Known known classes & unknown known classes	Identifying known known classes & unknown known classes
Open Set Recognition	Known known classes	Known known classes & unknown unknown classes	Identifying known known classes & rejecting unknown unknown classes
Generalized Open Set Recognition	Known known classes & side-information[2]	Known known classes & Unknown unknown classes	Identifying known known classes & cognizing unknown unknown classes

Different Categories of Models for Open Set Recognition:

- Discriminative Model (DM)
    - Traditional ML-based
        - SVM-based
            - corresponding method
            - corresponding method additionally adopts the statistical Extreme Value Theory
        - Sparse Representation-based
            - corresponding method additionally adopts the statistical Extreme Value Theory
        - Distance-based
            - corresponding method
        - Margin Distribution-based
            - corresponding method additionally adopts the statistical Extreme Value Theory
        - Others
    - Deep Neural Network-based
        - corresponding method
        - corresponding method additionally adopts the statistical Extreme Value Theory

- Generative Model (GM)
  - Instance Generation-based
    - corresponding method
    - corresponding method additionally adopts the statistical Extreme Value Theory
  - Non-Instance Generation-based
    - corresponding method
- Hybrid GM DM (research direction in the future)

Note:

the existing open set recognition is indeed in an open scenario but not incremental and does not scale gracefully with the number of classes.

Datasets:

In open set recognition, most existing experiments are usually carried out on a variety of recast multi-class benchmark datasets at present, where some distinct labels in the corresponding dataset are randomly chosen as KKCs while the remaining ones as UUCs.

Evaluation Criteria:

- Accuracy
- F-measure
- Youden’s index

Currently, F-measure and AUROC is the most commonly used evaluation metrics.

Counterfactual Learning

A **counterfactual explanation** of a prediction describes the smallest change to the feature values that changes the prediction to a predefined output.

Human-understandable explanations for machine-produced decisions are advantageous in several ways. For example, focusing on a use case of applicants applying for loans, the benefits would include:

- An explanation can be beneficial to the applicant whose life is impacted by the decision. For example, it helps an applicant understand which of their attributes were strong drivers in determining a decision.

- Further, it can help an applicant challenge a decision if they feel an unfair treatment has been meted, e.g., if one's race was crucial in determining the outcome. This can also be useful for organizations to check for bias in their algorithms.
- In some instances, an explanation provides the applicant with feedback that they can act upon to receive the desired outcome at a future time.
- Explanations can help the machine learning model developers identify, detect, and fix bugs and other performance issues.
- Explanations help in adhering to laws surrounding machine-produced decisions, e.g., GDPR.

Major research themes have sought to incorporate increasingly complex constraints on counterfactuals, all to ensure the resulting explanation is truly actionable and useful. Development in this field has focused on addressing these desiderata in a generalizable way across algorithms and is computationally efficient:

1. **Validity:** counterfactual explanation as an optimization problem.

$$\arg\min_{x'} d(x,x') \text{ subject to } f(x') = y'$$

$$\arg\min_{x'} \max_{\lambda} \lambda(f(x') - y')^2 + d(x,x')$$

The first term encourages the output of the classifier on the counterfactual to be close to the desired class and the second term forces the counterfactual to be close the original datapoint.

2. **Actionability:** An important consideration while making recommendation is which features are mutable and which aren't. A recommended counterfactual should never change the immutable features. In fact, if change to a legally sensitive feature produces a change in prediction, it shows inherent bias in the model. Also an applicant might have a preference order amongst the mutable features. The optimization problem is modified to take this into account. We might call the set of actionable features $\mathcal{A}$, and update our loss function to be:

$$\arg\min_{x' \in \mathcal{A}} \max_{\lambda} \lambda(f(x') - y')^2 + d(x,x')$$

3. **Sparsity:** There can be a trade-off between the number of features changed and the total amount of change made to obtain the counterfactual. We update our loss function to include a penalty function which encourages sparsity in the difference between the modified and the original datapoint, $g(x' - x)$, e.g. L0/L1 norm:

$$\arg\min_{x' \in \mathcal{A}} \max_{\lambda} \lambda(f(x') - y')^2 + d(x,x') + g(x' - x)$$

4. **Data Manifold closeness:** It would be hard to trust a counterfactual if it resulted in a combination of features which were utterly unlike any observations the classifier has seen before. In this sense, the counterfactual would be "unrealistic" and not easy to realize. Therefore, it is desirable that a generated counterfactual is realistic in the sense that it is near the training data and adheres to observed correlations among the features. We might update our loss function to include a penalty for adhering to the data manifold defined by the training set X, denoted by $l(x' ; X)$:

$$\arg\min_{x' \in \mathcal{A}} \max_{\lambda} \lambda(f(x') - y')^2 + d(x, x') + g(x' - x) + l(x'; \mathcal{X})$$

5. **Causality:** Features in a dataset are rarely independent, therefore changing one feature in the real world affects other features. In order to be realistic and actionable, a counterfactual should maintain any known causal relations between features.
6. **Amortized inference:** Thus learning to predict the counterfactual allows the algorithm to quickly compute a counterfactual (or several) for any new input x, without requiring to solve an optimization problem.
7. **Alternative methods:** Finally, several papers solve the counterfactual generation problem using linear programming, mixed-integer programming, or SMT solvers. These approaches give guarantees and optimize fast, but are limited to classifiers with linear (or piece-wise linear) structure.

Different terminology often captures the basic idea of counterfactual explanations, although subtle differences exist between the terms. Several terms worth noting include:

1. Recourse 2. Inverse classification 3. Contrastive explanation 4. Adversarial learning

Properties of counterfactual algorithms:

1. **Model access:** We identify three distinct access levels - access to complete model internals, access to gradients, and access to only the prediction function (black-box)
2. **Model domain:** Agnostic- Differentiable- Tree ensemble- Linear- Linear and causal graph- Linear and tree ensemble- Random Forest
3. **Optimization amortization:** a) Amortized Inference b) Multiple counterfactual
4. **Counterfactual attributes:** a) Sparsity b) Data manifold c) Causal relation
5. **Counterfactual optimization problem attributes:** a) Feature preference b) Categorical distance function

Commonly used datasets for evaluation:

- Image: MNIST
- Tabular: Adult income, German credit, Compas recidivism, LSAT, Pima diabetes

Metrics for evaluation of counterfactual generation algorithms:

1.Validity 2.Proximity 3.Sparsity 4. Diversity
5.Counterfactual generation time 6.Closeness to the training data
7.Causal constraint satisfaction (feasibility) 8.IM1 and IM2

$$IM1 = \frac{\|x_{cf} - AE_t(x_{cf})\|_2^2}{\|x_{cf} - AE_o(x_{cf})\|_2^2 + \epsilon}$$

$$IM2 = \frac{\|AE_t(x_{cf}) - AE(x_{cf})\|_2^2}{\|x_{cf}\|_1 + \epsilon}$$

Open Questions:

1. Unify counterfactual explanations with traditional "explainable AI."
2. Counterfactual explanations should account for bias in the classifier.
3. Lack of a structural causal model (SCM).
4. Lack of interventional data.
5. ML models are not static.
6. CFEs should be an interactive service, while guarding against privacy attacks.
7. CFEs should tell what should not change.
8. Consider bias in the ML model.
9. Capturing personal preferences.
10. Generate common-sensical CFEs.
11. Lack of visualization of CFEs.
12. Lack of understanding of how to apply CFEs in the regression (as opposed to classification) case.

Self-supervised Learning

Self-supervised learning is a subset of unsupervised learning methods. Self-supervised learning refers to learning methods in which ConvNets are explicitly trained with automatically generated labels.

To avoid time-consuming and expensive data annotations, many self-supervised methods are proposed to learn visual features from large-scale unlabeled images or videos without using any human annotations. To learn visual features from unlabeled data, a popular solution is to propose various pretext tasks for networks to solve, while the networks can be trained by learning objective functions of the pretext tasks, and the features are learned through this process. Various pretext tasks have been proposed for self-supervised learning, including colorizing grayscale images, image inpainting, image jigsaw puzzles, etc. The pretext tasks share two common properties: (1) visual features of images or videos need to be captured by ConvNets to solve the pretext tasks, and (2) pseudo labels for the pretext task can be automatically generated based on the attributes of images or videos.

After self-supervised pretext task training finished, the learned parameters serve as a pre-trained model and are transferred to other downstream computer vision tasks by fine-tuning. The performance on these downstream tasks is used to evaluate the quality of the learned features. During the knowledge transfer for downstream tasks, the general features from only the first several layers are unusually transferred to downstream tasks.

Since no human annotations are needed to generate pseudo labels during self-supervised training, very largescale datasets can be used for self-supervised training. Trained with these pseudo labels, self-supervised methods achieved promising results and the gap with supervised methods in performance on downstream tasks becomes smaller.

Formulation: Self-supervised learning trained with data X_i along with its pseudo label P_i; Given a set of N training data $D = \{P_i\}_{i=0}^{N}$, the training loss function is defined as:

$$loss(D) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(X_i, P_i)$$

Common Deep Network Architectures:

- Architectures for Learning Image Features: AlexNet, VGG, ResNet, GoogLeNet, DenseNet
- Architectures for Learning Video Features: 1. Two-Stream Network 2. Spatiotemporal Convolutional Neural Network 3. Recurrent Neural Network

Commonly Used Pretext Tasks: According to the data attributes used to design pretext tasks we summarize the pretext tasks into four categories:

- **Generation-based Methods:** This type of methods learn visual features by solving pretext tasks that involve image or video generation.
  - **Image Generation:** Visual features are learned through the process of image generation tasks.
    - Image Generation with GAN
    - Super-Resolution
    - Image Inpainting
    - Image Colorization

    - **Video Generation:** Visual features are learned through the process of video generation tasks.
        - Video Generation with GAN
        - Video Colorization
        - Video Future Prediction
- **Context-based pretext tasks:** The design of context-based pretext tasks mainly employ the context features of images or videos.
    - **Context Similarity:** Pretext tasks are designed based on the context similarity between image patches.
        - Clustering-based methods
        - graph constraint-based methods
    - **Spatial Context Structure:** Pretext tasks are used to train ConvNets are based on the spatial relations among image patches.
        - Image Jigsaw Puzzle
        - Geometric Transformation recognition
        - context prediction
    - **Temporal Context Structure:** The temporal order from videos is used as supervision signal.
        - Frame Order Verification
        - Frame Order Recognition
- **Free Semantic Label-based Methods:** This type of pretext tasks train networks with automatically generated semantic labels. The labels are generated by traditional hardcode algorithms, or by game engines.
    - Moving Object Segmentation
    - Relative Depth Prediction
    - Surface Normal Prediction
    - Contour Detection
    - Depth Estimation
    - Semantic Segmentation
- **Cross Modal-based Methods:** This type of pretext tasks train ConvNets to verify whether two different channels of input data are corresponding to each other.
    - Flow-RGB Correspondence
        - Optical Flow Estimation
        - Flow-RGB Correspondence Verification
    - Visual-Audio Correspondence
        - Audio Visual Correspondence
    - Ego-motion
        - Ego-motion

Commonly Used Downstream Tasks for Evaluation:

- Semantic Segmentation
- Object Detection
- Image Classification
- Human Action Recognition
- Qualitative Evaluation
  - Kernel Visualization
  - Feature Map Visualization
  - Nearest Neighbor Retrieval

Datasets

- **Image Datasets:** ImageNet, Places, Places365, SUNCG, MNIST, SVHN, CIFAR10, STL-10, PASCAL Visual Object Classes (VOC)
- **Video Datasets:** YFCC100M, SceneNet RGB-D, Moment in Time, Kinetics, AudioSet, KITTI, UCF101, HMDB51

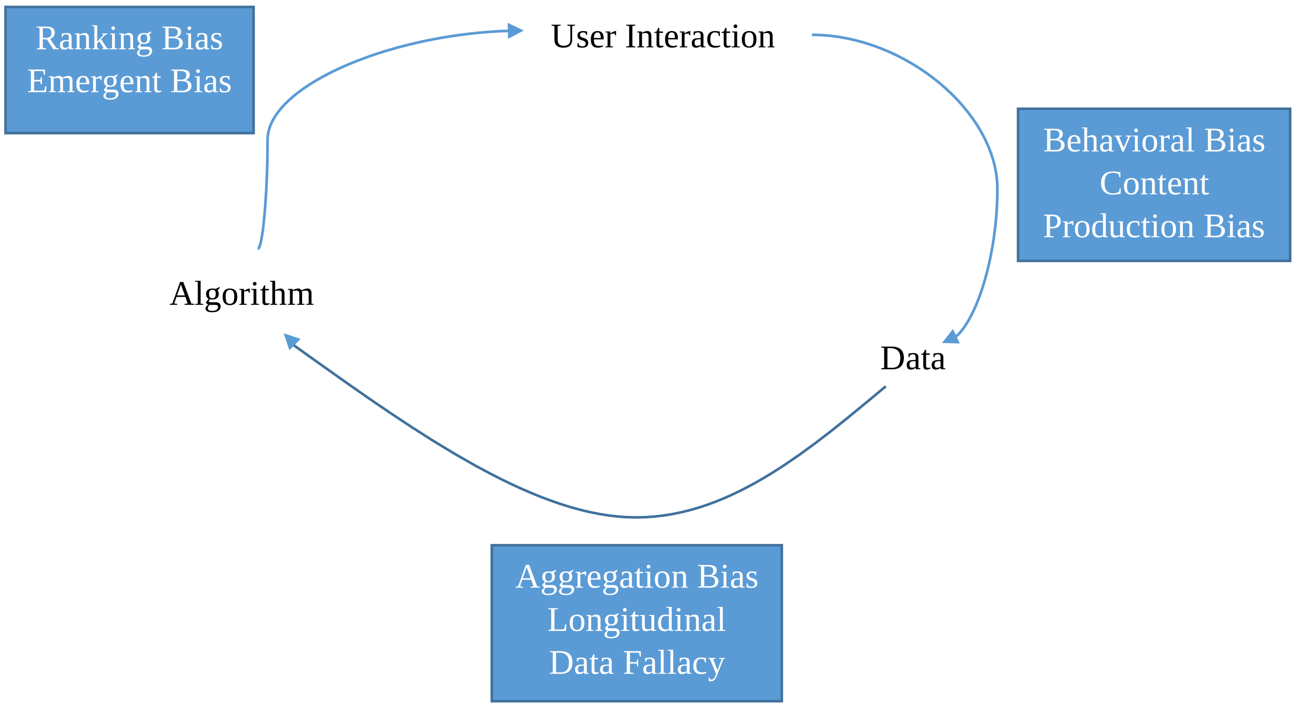
Future Directions:

1. Learning Features from Synthetic Data
2. Learning from Web Data
3. Learning Spatiotemporal Features from Videos
4. Learning with Data from Different Sensors
5. Learning with Multiple Pretext Tasks

Fairness

Algorithms are vulnerable to biases that render their decisions "unfair". In the context of decision-making, fairness is the absence of any prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics. Thus, an unfair algorithm is one whose decisions are skewed toward a particular group of people.

Examples of bias definitions placed in the data, algorithm, and user interaction feedback loop:



Types of Bias:

1. **Data to Algorithm**: biases in data, which, when used by ML training algorithms, might result in biased algorithmic outcomes.
    - **a.** Measurement (or reporting) Bias
    - **b.** Omitted Variable Bias
    - **c.** Representation Bias
    - **d.** Aggregation Bias (or ecological fallacy)
        - **i.** Simpson's Paradox
        - **ii.** Modifiable Areal Unit Problem
    - **e.** Sampling Bias
    - **f.** Longitudinal Data Fallacy
    - **g.** Linking Bias
2. **Algorithm to User**: the biases that are as a result of algorithmic outcomes and affect user behavior as a consequence.
    - **a.** Algorithmic Bias
    - **b.** User Interaction Bias
        - **i.** Presentation Bias
        - **ii.** Ranking Bias

   - **c.** Popularity Bias
   - **d.** Emergent Bias
   - **e.** Evaluation Bias
3. **User to Data**: Many data sources used for training ML models are user-generated. Any inherent biases in users might be reflected in the data they generate. Furthermore, when user behavior is affected/modulated by an algorithm, any biases present in those algorithm might introduce bias in the data generation process.
   - **a.** Historical Bias
   - **b.** Population Bias
   - **c.** Self-Selection Bias
   - **d.** Social Bias
   - **e.** Behavioral Bias
   - **f.** Temporal Bias
   - **g.** Content Production Bias

Discrimination:

Similar to bias, discrimination is also a source of unfairness. Discrimination can be considered as a source for unfairness that is due to human prejudice and stereotyping based on the sensitive attributes, which may happen intentionally or unintentionally, while bias can be considered as a source for unfairness that is due to the data collection, sampling, and measurement.

1. Explainable Discrimination
2. Unexplainable Discrimination
   - a. Direct Discrimination
   - b. Indirect Discrimination
3. Sources of Discrimination
   - a. Systemic Discrimination
   - b. Statistical Discrimination

Definitions of Fairness:

Definition 1. Equalized Odds (Group Fairness)

Definition 2. Equal Opportunity (Group Fairness)

Definition 3. Demographic Parity (Group Fairness)

Definition 4. Fairness Through Awareness (Individual Fairness)

Definition 5. Fairness Through Unawareness (Individual Fairness)

Definition 6. Treatment Equality (Group Fairness)

Definition 7. Test Fairness (Group Fairness)

Definition 8. Counterfactual Fairness (Individual Fairness)

Definition 9. Fairness in Relational Domains

Definition 10. Conditional Statistical Parity (Group Fairness)

The target biases in the algorithms fall under three categories:

1. Pre-processing 2. In-processing 3. Post-processing

Bias and fairness in different areas:

1. Variational auto encoders 2. Community detection 3. Word embedding
4. Adversarial learning 5. Causal inference 6. Graph embedding
7. Classification 8. Regression 9. PCA
10. Clustering

Challenges:

1. Synthesizing a definition of fairness 2. From Equality to Equity
3. Searching for Unfairness

Datasets for Fairness Research:

1. UCI Adult Dataset 2. German Credit Dataset 3. WinoBias 4. COMPAS Dataset
5. Communities and Crime Dataset 6. Recidivism in Juvenile Justice Dataset
7. Pilot Parliaments Benchmark Dataset 8. Diversity in Faces Dataset