Durham University

Inspiring the extraordinary

Machine Learning

COMP2261 Artificial Intelligence

Lecturer: **Dr. Yang Long**

Associate Professor, MRC Innovation Fellow, SMIEEE
Director of Hybrid Intelligence Lab, VIViD Group
Department of Computer Science,
Durham University

.

Durham University

Inspiring the extraordinary

Chapter 2: Learning Paradigms in Machine Learning (4 hours)

Overall Introduction
Machine learning encompasses a wide range of learning paradigms that allow models to learn and adapt from data. This chapter explores various paradigms, from foundational approaches like supervised and unsupervised learning to more advanced techniques involving multi-sourced data and hybrid models. The aim is to provide a comprehensive overview of the different ways machine learning models can be trained, tuned, and applied to real-world problems. We begin with foundational paradigms, which set the groundwork for understanding the basic principles of machine learning, followed by paradigms focused on learning from multiple data sources and, finally, advanced hybrid approaches that combine different agents and human participants to solve complex problems.

Section 1: Fundamental Learning Paradigms

Supervised Learning
- Linear Regression, Logistic Regression, Support Vector Machines (SVM)

Unsupervised Learning
- Clustering (e.g., K-means, DBSCAN)

Instance-Based Learning
- k-Nearest Neighbours (k-NN)

Contrastive Learning
- Metric Learning, Kernel Learning

Bayesian Learning
- Naive Bayes, Bayesian Networks, Gaussian Processes

Rule-Based Learning
- Decision Trees, Association Rule Mining

Generative Learning
- Gaussian Mixture Models (GMM), Hidden Markov Models (HMM)

Ensemble Learning
- Bagging, Boosting

Evolutionary Learning
- Genetic Algorithms, Evolution Strategies, Particle Swarm Optimisation

Representation Learning
- Autoencoders, Principal Component Analysis (PCA), Word2Vec

.

Manifold Learning
- Matrix / Tensor Factorisation

Constrained Learning
- Sparse Learning, Low-Rank Learning

Relationship Learning
- Graph-Based Learning, Probabilistic Graphical Learning

Other Learning Foundations

Deep Learning
- Attention-Based Learning, Self-Organising Learning, Hyperparameter Learning, Neural Architecture Search (NAS) Learning

Reinforcement Learning
- Inverse Reinforcement Learning

Recommender Systems
- Collaborative Filtering, Content-Based Filtering

Section 2: Multi-Sourced Learning

Semi-Supervised Learning
- Ladder Networks, Label Propagation

Self-Supervised Learning
- Contrastive Predictive Coding (CPC), SimCLR

Transfer Learning
- Fine-Tuning Pretrained Models (e.g., BERT, ResNet)

Multi-Instance Learning
- Diverse Density, mi-SVM

Multi-View Learning
- Co-Training, Multiple Kernel Learning (MKL)

Multi-Modal Learning
- Cross-Modal Retrieval, Audio-Visual Models

Multi-Task Learning
- Hard Parameter Sharing, Cross-Stitch Networks

Few-Shot Learning
- Prototypical Networks, Matching Networks

.

Meta-Learning
- Model-Agnostic Meta-Learning (MAML), Reptile

One-Shot Learning
- Siamese Networks

Zero-Shot Learning
- Attribute-Based Classification, Embedding-Based Methods

Uncertainty-Aware Learning
- Bayesian Neural Networks, Monte Carlo Dropout

Open World Learning
- OpenMax, Incremental Class Learning

Section 3: Advanced Hybrid Learning Paradigms

Active Learning
- Query-By-Committee, Uncertainty Sampling

Curriculum Learning
- Self-Paced Learning, Baby Steps Algorithm

Continual Learning
- Elastic Weight Consolidation (EWC), Progressive Neural Networks

Lifelong Learning
- Learning without Forgetting (LwF), Memory-Augmented Networks

Federated Learning
- Federated Averaging (FedAvg), Split Learning

Adversarial Learning
- Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD)

Multi-Agent Learning
- Deep Q-Learning, Multi-Agent Reinforcement Learning (MARL)

Knowledge Distillation Learning
- Teacher-Student Networks, Soft Label Distillation

Swarm Intelligence Learning
- Ant Colony Optimisation, Bee Colony Optimisation

Hybrid Learning Paradigms
- Reinforcement Learning with Imitation Learning, Neuro-Symbolic Integration

Symbiotic Learning

.

- Cooperative Multi-Agent Systems, Human-AI Teaming

World Model
- Dreamer, PlaNet

Main Contents

The main contents for each learning paradigm are developed using ten teaching methods, providing a comprehensive understanding of each topic for students. For each learning paradigm, we use four perspectives:

- **Concept:** Define the learning paradigm clearly. Explain its purpose and how it works.
    - Contrasting and Association: Compare this paradigm with similar or related paradigms. Highlight differences and associations.
- **Formulation:** Provide the relevant equations or mathematical foundations that explain how the paradigm works. Alternatively, create symbolic diagrams, charts, or other visuals to illustrate the paradigm.
- **References and Historical Cases:** Give a brief history of the paradigm, including key contributors and significant breakthroughs. Mention seminal papers or recent research that underpin this paradigm. Examples offer real-world examples where this paradigm is applied, and explain how it contributes to solving the problem.
- **Common Challenges and Limitations:** Discuss typical challenges faced when applying this paradigm, including limitations.
    - Interactive Discussion Questions: Pose questions that could stimulate discussion or critical thinking regarding this learning paradigm.

.

Durham University

Inspiring the extraordinary

Section 1: Fundamental Learning Paradigms

Supervised Learning Paradigm

Concept and Contrast: Supervised learning is a machine learning paradigm where a model is trained on a labelled dataset to map inputs to outputs. The goal is to learn from labelled data such that, after training, the model can generalise to make accurate predictions on new, unseen data. In supervised learning, each data point has an associated label, meaning the data comes with known outcomes. Training typically involves minimising an error function, also known as a loss function, which quantifies the difference between the model's predictions and actual labels. This iterative adjustment to minimise error is foundational to training a successful supervised learning model.
Contrast with Other Paradigms: Supervised learning differs from unsupervised learning, where data lacks labels and the goal is to identify patterns or structure within the data. It also contrasts with reinforcement learning, where the model learns through interactions with an environment and optimises for cumulative rewards rather than labelled outcomes. In semi-supervised or self-supervised learning, models rely on only partial labelling or use data patterns to generate labels, combining both labelled and unlabelled data sources.

General Formula for Supervised Learning

A general formula for supervised learning can be represented as follows:

$$\hat{y} = f(X;\theta)$$

where:

- $\hat{y}$ is the predicted output (e.g., class label or continuous value).
- X represents the input features.
- θ are the model parameters that are learned during training.

The goal of supervised learning is to find the optimal parameters θ by minimising a loss function $L(y, \hat{y})$, where y is the true label.

References and Historical Cases: Supervised learning has deep roots in statistics and pattern recognition, originating with early linear models and statistical approaches to classification. Seminal work includes **Ronald A. Fisher's** introduction of **Linear Discriminant Analysis (LDA)** in 1936, which provided a method to project high-dimensional data onto a lower-dimensional space for classification tasks (Fisher, 1936).

In the late 1950s, **Frank Rosenblatt** developed the **Perceptron**, one of the earliest artificial neural networks, designed to perform binary classification tasks by learning a linear boundary between classes (Rosenblatt, 1958). Although the perceptron initially gained significant attention, **Marvin**

.

Inspiring the extraordinary

Minsky and **Seymour Papert** highlighted its limitations in their 1969 book, "Perceptrons," noting that a single-layer perceptron could not solve non-linear problems (Minsky & Papert, 1969). This critique led to a temporary decline in neural network research, known as the "AI winter."

Support Vector Machines (SVM), developed by **Vladimir Vapnik** and **Alexey Chervonenkis** in the 1960s and popularised in the 1990s, provided a robust solution to classification by maximising the margin between classes (Vapnik, 1995). SVM's foundation in statistical learning theory set a standard for rigorous, theoretically grounded machine learning methods.

In recent decades, supervised learning has progressed significantly with the introduction of **deep neural networks**. The breakthrough came in 2012 with **AlexNet** (Krizhevsky, Sutskever, & Hinton, 2012), which demonstrated the power of deep learning in supervised tasks by achieving unprecedented performance on the ImageNet classification challenge. This model marked a new era of deep supervised learning and motivated the development of various architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs), widely applied today in fields from image recognition to natural language processing.

Recent research continues to refine supervised learning for real-world challenges. For example:

- **Robustness and interpretability: Ribeiro, Singh, and Guestrin's** work on **LIME (Local Interpretable Model-agnostic Explanations)** addressed the need for interpretable machine learning models, especially in high-stakes areas (Ribeiro et al., 2016).
- **Scalability and efficiency:** Advances in optimising deep learning architectures (Han, Mao, & Dally, 2015) have improved scalability, allowing supervised models to handle massive datasets.
- **Fairness and ethical considerations:** Recent studies, like **Zafar et al. (2017)**, have developed techniques to reduce bias in supervised learning models, an area of growing importance as AI is deployed in critical social applications.
-

These foundational studies and ongoing innovations drive supervised learning's adaptability and effectiveness across a wide range of domains.

Common Challenges and Limitations:

- Overfitting and Underfitting: A model may learn noise rather than general patterns, leading to poor performance on new data. Conversely, if the model is too simple, it may not capture essential patterns, resulting in underfitting.
- Data Dependency: Supervised learning requires substantial, high-quality labelled data, which can be costly or impractical to obtain in certain fields.

.

Durham University

Inspiring the extraordinary

- Bias-Variance Trade-Off: The challenge of balancing model complexity affects generalisation, requiring adjustments to avoid high bias (underfitting) or high variance (overfitting).
- Scalability: Training on large datasets with high-dimensional features demands significant computational power and memory resources.

Discussion Questions:

1. Why does supervised learning heavily rely on labelled data? What issues arise if labels are incorrect or inconsistent?
2. How can overfitting be identified and addressed in supervised models?
3. When selecting a loss function, what factors should be considered based on the data and task?
4. What are the potential ethical considerations when applying supervised learning to sensitive domains, such as healthcare?

References

1. Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*, 7(2), 179-188.
2. Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, 65(6), 386.
3. Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.
4. Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.
5. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). "ImageNet classification with deep convolutional neural networks." *Advances in Neural Information Processing Systems*, 25.
6. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135-1144.
7. Han, S., Mao, H., & Dally, W. J. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding." *arXiv preprint arXiv:1510.00149*.
8. Zafar, M. B., Valera, I., Gomez-Rodriguez, M., & Gummadi, K. P. (2017). "Fairness constraints: Mechanisms for fair classification." *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 962-970.

.

Durham University

Inspiring the extraordinary

Unsupervised Learning Paradigm

Concept and Contrast: Unsupervised learning is a machine learning paradigm where models are trained on unlabelled data, aiming to uncover patterns, groupings, or structure without pre-assigned outcomes. This is distinct from supervised learning, where models learn from labelled data with known outputs. Unsupervised learning tasks include clustering, dimensionality reduction, and anomaly detection. The primary objective is to find underlying structure or representations in the data that may inform further analysis or support downstream tasks.
Contrast with Other Paradigms: Unsupervised learning differs significantly from supervised learning, which relies on labelled datasets to make predictions. Unlike reinforcement learning, where a model interacts with an environment and optimises actions based on feedback or rewards, unsupervised learning explores the data without feedback. In contrast to semi-supervised learning, which uses partially labelled data, unsupervised learning operates purely on unlabelled data, focusing on identifying structures that can help interpret complex datasets or create useful representations for further learning.

General Formula for Unsupervised Learning

In unsupervised learning, there's no labelled output to compare against, so the objective function often seeks to optimise internal properties of the data, like compactness of clusters or lower-dimensional representation. A common objective in clustering tasks, for instance, is to minimise the within-cluster variance. This is formulated as:

$$\min \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

where:

- K is the number of clusters,
- C_k is the k-th cluster,
- x_i is a data point in cluster C_k,
- μ_k is the centroid (mean) of cluster C_k, and
- $\|x_i - \mu_k\|$ represents the Euclidean distance between x_i and the centroid.

In dimensionality reduction tasks, the objective might instead be to minimise the loss between original and projected data representations, such as:

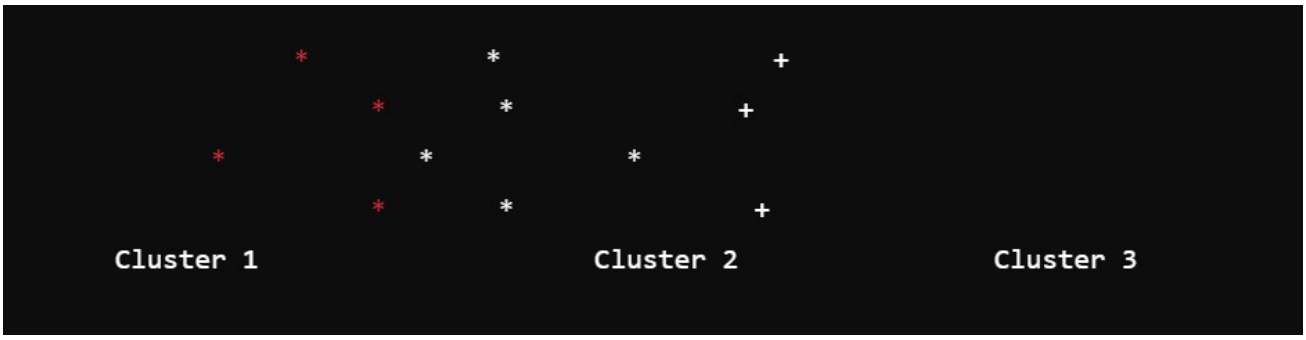
$$\min \|X - X'\|^2$$

where:

- X is the original high-dimensional data,
- X' is the low-dimensional representation.

.

Inspiring the extraordinary



References and Historical Cases: Unsupervised learning has origins in fields like exploratory data analysis, clustering, and multivariate statistics, with early techniques dating back to **Karl Pearson**'s work on **Principal Component Analysis (PCA)** in 1901. PCA has been fundamental in data reduction and visualization by projecting high-dimensional data into lower dimensions while retaining most of the variance (Pearson, 1901).

In the 1960s, **Hierarchical Clustering** gained prominence as a tool for data classification, grouping similar objects based on hierarchical relationships. Later, **K-means clustering** (MacQueen, 1967) became one of the most widely used algorithms for partitioning data into clusters by minimising the sum of squared distances between points and their assigned cluster centers. **J.B. MacQueen** formalised the K-means algorithm, which remains a cornerstone of clustering methods.
Unsupervised learning saw significant advancements with the development of **Self-Organising Maps (SOM)** by **Teuvo Kohonen** in the 1980s, which provided a foundation for dimensionality reduction and data visualization using neural networks. More recently, **t-SNE (t-distributed Stochastic Neighbor Embedding)**, introduced by **Laurens van der Maaten and Geoffrey Hinton** in 2008, has become a popular tool for visualising high-dimensional data (van der Maaten & Hinton, 2008). These approaches laid the groundwork for clustering, anomaly detection, and other unsupervised tasks used widely today.
Recent research has focused on deep unsupervised learning, particularly **autoencoders** and **Generative Adversarial Networks (GANs)**, which enable powerful representations and synthetic data generation without labelled data (Goodfellow et al., 2014).

Common Challenges and Limitations:

- **Interpretability:** Since unsupervised models often work with high-dimensional data and find patterns that may not correspond to human-interpretable features, the results can be difficult to interpret.
- **Defining Objectives:** Without labelled data, there's often no clear objective function to optimise. Selecting the number of clusters, dimensionality reduction criteria, or identifying anomalies requires intuition and trial-and-error.
- **Scalability and Efficiency:** Many unsupervised algorithms, particularly clustering methods, are computationally intensive, making them challenging to apply on large-scale datasets without optimisations.
- **Evaluation Difficulties:** Without a labelled dataset, evaluating the quality of clustering or dimensionality reduction becomes subjective, often relying on metrics like silhouette score or human validation.

Discussion Questions:

.

Durham University

Inspiring the extraordinary

1. How can we evaluate the quality of an unsupervised model's results without labelled data?
2. In what ways does dimensionality reduction benefit unsupervised learning? Are there trade-offs?
3. What are the potential ethical implications of using unsupervised learning in domains like surveillance or marketing?
4. How do we determine the optimal number of clusters in a dataset with K-means clustering?

References

1. Pearson, K. (1901). "On lines and planes of closest fit to systems of points in space." *Philosophical Magazine*, 2(11), 559-572.
2. MacQueen, J. B. (1967). "Some methods for classification and analysis of multivariate observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(14), 281-297.
3. Kohonen, T. (1982). "Self-organized formation of topologically correct feature maps." *Biological Cybernetics*, 43(1), 59-69.
4. van der Maaten, L., & Hinton, G. (2008). "Visualizing data using t-SNE." *Journal of Machine Learning Research*, 9(11).
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). "Generative adversarial nets." In *Advances in Neural Information Processing Systems* (pp. 2672-2680).

.

Inspiring the extraordinary

Instance-Based Learning Paradigm

Concept and Contrast: Instance-based learning is a machine learning paradigm where the model makes predictions based directly on specific instances of the training data rather than learning explicit parameters or patterns. Instead of generalising from the training data to create a model, instance-based algorithms, such as k-Nearest Neighbors (k-NN), store instances and use them at inference time to predict labels or outputs for new inputs by comparing them to stored instances. This paradigm is especially useful when the data distribution is highly irregular, as instance-based learning preserves local patterns. However, it can be computationally intensive because it requires the model to access and compare with potentially all stored data points each time it makes a prediction.

Contrast with Other Paradigms: Instance-based learning contrasts sharply with model-based paradigms like supervised learning, where the model is trained to learn a function or set of parameters that maps inputs to outputs. In model-based learning, predictions can be made directly based on learned parameters without referencing the original data, leading to faster inference. In contrast, instance-based learning relies on storing data and typically requires significant computational resources at inference time.

While unsupervised learning may also rely on instance relationships for clustering, instance-based learning specifically leverages these relationships to make predictions. In reinforcement learning, the model adapts based on feedback from an environment, which differs from instance-based learning's reliance on memorising and comparing specific instances.

General Formula for Instance-Based Learning:

Explanation:

- `@` is a new input to be classified.
- `*` and `o` represent stored instances of two classes (Class A and Class B).
- Based on the nearest instances, instance-based learning (e.g., k-NN) would classify `@` according to the majority class among its closest neighbours.

.

Durham University

Inspiring the extraordinary

General Formula for Instance-Based Learning: A general approach for instance-based learning involves finding the closest k instances to a new input x_{new} and making a prediction based on these neighbours. The distance function $d(x_{new}, x_i)$, often Euclidean, measures similarity. In k-Nearest Neighbors (k-NN), the prediction y_{new} is determined by the majority label (for classification) or average output (for regression) of the nearest k neighbours:

$$y_{new} = \text{majority or average}\{y_i : x_i \in \text{k-NN of } x_{new}\}$$

where:

- y_i represents the output for each instance x_i,
- x_i is within the set of k-nearest instances to x_{new},
- $d(x_{new}, x_i)$ is the distance metric used (e.g., Euclidean, Manhattan).

References and Historical Cases: The foundations of instance-based learning were established with the development of the k-Nearest Neighbors (k-NN) algorithm, formalised by **Cover and Hart** in 1967 (Cover & Hart, 1967). k-NN demonstrated a simple yet effective approach for classifying data based on similarity and has since been widely applied due to its interpretability and adaptability to non-linear data distributions.
In the 1980s, **Case-Based Reasoning (CBR)** emerged as an extension of instance-based learning, particularly for complex problem-solving tasks. CBR systems, like the ones developed by **Janet Kolodner** and **Roger Schank**, retrieve past cases similar to a current problem, adapt them, and apply them to the new situation, thereby improving system performance in various domains (Kolodner, 1983; Schank, 1982).
With the advent of more complex data, instance-based learning has further evolved. In recent years, it has been integrated with hybrid methods that combine instance-based learning with deep learning approaches, such as **Memory-Augmented Neural Networks (MANNs)**, to enable learning from and storing relevant instances in deep architectures (Santoro et al., 2016).

Common Challenges and Limitations:

- **Computational Complexity:** Instance-based learning requires calculating distances between the input and each stored instance, which can be computationally demanding, especially in high-dimensional spaces or large datasets.
- **Storage Requirements:** Since instance-based methods store all or most of the training data, they can be memory-intensive, particularly for large-scale datasets.
- **Sensitivity to Irrelevant Features:** If the data contains irrelevant features, instance-based learning can suffer from performance degradation. Feature selection or scaling techniques are often necessary.
- **Scalability Issues:** Due to high storage and computational demands, scaling instance-based learning methods to large datasets often requires optimisations, such as using approximate nearest neighbour search algorithms.

Discussion Questions:

.

Durham University

Inspiring the extraordinary

1. How does instance-based learning handle irregular or non-linear data distributions compared to model-based approaches?
2. In what scenarios might instance-based learning be more suitable than supervised or unsupervised learning?
3. How can we improve the efficiency of instance-based learning on large datasets?
4. What are potential limitations of using instance-based learning in real-time applications?

References

1. Cover, T., & Hart, P. (1967). "Nearest neighbor pattern classification." *IEEE Transactions on Information Theory*, 13(1), 21-27.
2. Kolodner, J. L. (1983). "Reconstructive memory: A computer model." *Cognitive Science*, 7(4), 281-328.
3. Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press.
4. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016). "Meta-learning with memory-augmented neural networks." *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

.

Durham University

Inspiring the extraordinary

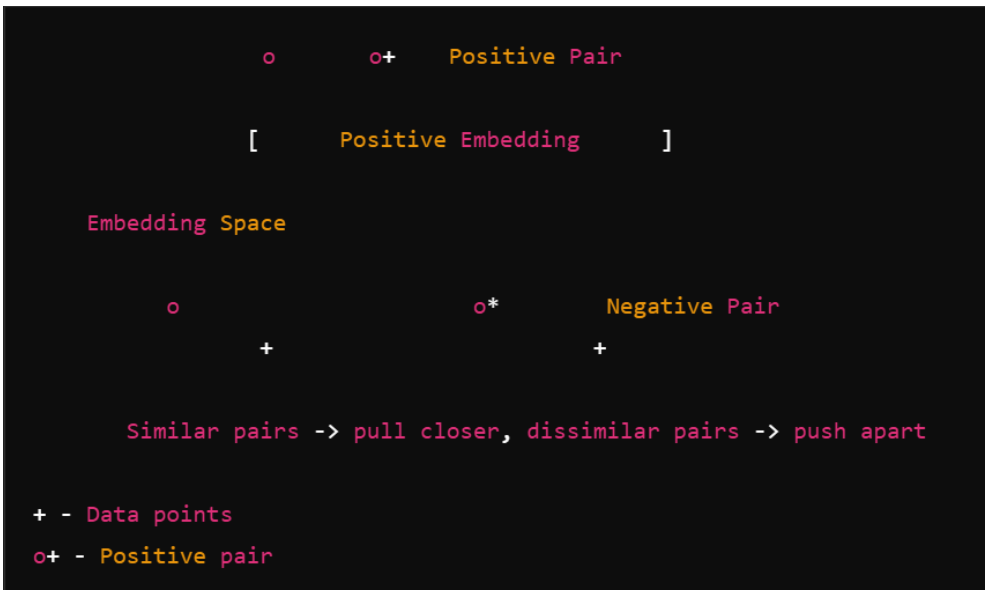
Contrastive Learning Paradigm

Concept and Contrast: Contrastive learning is a machine learning paradigm where models learn by comparing pairs of data points, focusing on distinguishing between similar and dissimilar samples. The core idea is to bring similar instances closer in a latent space while pushing dissimilar instances farther apart. This is particularly effective in representation learning, where the model aims to capture meaningful features that help in downstream tasks like classification, clustering, or retrieval.

Contrastive learning relies heavily on contrastive loss functions, such as **contrastive loss** or **triplet loss**, which quantify the similarity or dissimilarity between data points. In many cases, data augmentations are applied to create similar (positive) pairs by modifying the same data point, while different (negative) data points are sampled from the dataset to create dissimilar pairs.
Contrast with Other Paradigms: Contrastive learning is often associated with **metric learning** since both aim to learn distances between points in a feature space. In metric learning, however, the primary goal is to establish a distance metric that reflects similarity based on labelled pairs or triplets, whereas contrastive learning uses self-supervised approaches, often generating positive pairs through augmentations instead of labels.

Contrastive learning also shares similarities with **kernel learning**, as both aim to map data to a space where separation or clustering is easier. In contrastive learning, this space is typically high-dimensional and optimised to maintain relative distances rather than absolute classifications. In kernel learning, functions (kernels) are used to transform data points, enabling linear separability in higher-dimensional spaces. Kernel methods, however, are typically supervised and aim to optimise separability, whereas contrastive learning can be fully self-supervised, focusing on learned representations without explicit output labels.

Symbolic Diagram for Contrastive Learning



Durham University Palatine Centre Stockton Road Durham DH1 3LE UK
Telephone +44 (0)191 334 2000
www.durham.ac.uk

.

Durham University

Inspiring the extraordinary

General Formula for Contrastive Learning:

General Formula for Contrastive Learning: The objective function in contrastive learning, such as **contrastive loss** or **triplet loss**, encourages positive pairs to be close in the embedding space and negative pairs to be farther apart. For two embeddings z_i and z_j corresponding to a positive pair (i.e., augmentations of the same instance), and a temperature parameter τ, contrastive loss can be defined as:

$$\mathcal{L}_{contrastive} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{N} \exp(\text{sim}(z_i, z_k)/\tau)}$$

where:

- $\text{sim}(z_i, z_j)$ is a similarity function, typically cosine similarity.
- τ is a temperature parameter that controls the scale of the similarities.
- N is the number of negative samples used in the denominator.

For **triplet loss**, the objective function encourages the distance between positive pairs to be smaller than the distance between negative pairs by a margin α:

$$\mathcal{L}_{triplet} = \max(0, \|z_i - z_j\|^2 - \|z_i - z_k\|^2 + \alpha)$$

where:

- z_i and z_j are embeddings for a positive pair,
- z_k is the embedding for a negative sample,
- α is a margin parameter to enforce separation between positive and negative pairs.

References and Historical Cases: Contrastive learning has its roots in metric learning, where early methods such as **Large Margin Nearest Neighbour (LMNN)** (Weinberger & Saul, 2009) aimed to learn embeddings that respect pairwise constraints. However, contrastive learning as a paradigm gained prominence with the introduction of **Siamese Networks** by **Bromley et al. (1993)**, initially developed for signature verification, and later advanced with **triplet networks** (Schroff et al., 2015). In recent years, contrastive learning has flourished within self-supervised learning frameworks, such as **SimCLR (Simple Framework for Contrastive Learning of Visual Representations)** by **Chen et al. (2020)** and **MoCo (Momentum Contrast)** by **He et al. (2020)**. These methods demonstrated that contrastive learning could effectively pre-train representations in an unsupervised manner, achieving performance comparable to supervised methods on image recognition tasks.

.

Durham University

Inspiring the extraordinary

Contrastive learning has also expanded into text and multimodal domains, enabling applications in natural language processing and cross-modal learning, where aligning different data types, such as text and images, is essential for tasks like image-caption retrieval.

Common Challenges and Limitations:

- **Negative Sampling:** Obtaining a sufficient number of meaningful negative samples is critical for effective contrastive learning. If negative samples are too similar to positive pairs, the model may struggle to distinguish them.
- **Computational Complexity:** Computing contrastive loss requires processing pairs of samples, which can significantly increase computational requirements, especially in large datasets.
- **Mode Collapse:** Without diverse negative samples, the model may fail to learn distinct representations for different data points, leading to poor generalisation.
- **Sensitivity to Augmentations:** Contrastive learning relies on data augmentations to generate positive pairs. Inconsistent or inappropriate augmentations may lead to poor representations, especially in cases where augmentations alter class-relevant features.

Discussion Questions:

1. How does contrastive learning differ from metric learning, particularly in the context of supervised versus self-supervised learning?
2. What role do data augmentations play in the effectiveness of contrastive learning, and how could poor choices in augmentations impact model performance?
3. In contrastive learning, how can negative samples be selected to improve model generalisation?
4. What are potential applications of contrastive learning beyond image representation learning?

References

1. Weinberger, K. Q., & Saul, L. K. (2009). "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research*, 10, 207-244.
2. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). "Signature verification using a 'Siamese' time delay neural network." *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4), 669-688.
3. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). "FaceNet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815-823.
4. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). "A simple framework for contrastive learning of visual representations." *International Conference on Machine Learning (ICML)*.
5. He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). "Momentum contrast for unsupervised visual representation learning." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9729-9738.

.

Durham University

Inspiring the extraordinary

Bayesian Learning Paradigm

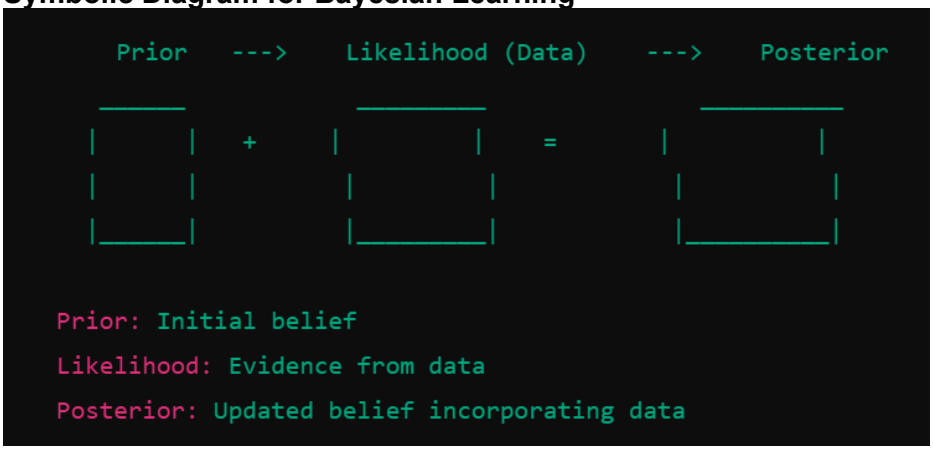
Concept and Contrast: Bayesian learning is a machine learning paradigm that incorporates principles from Bayesian probability to make predictions, quantify uncertainty, and update beliefs as new data is observed. Unlike many other paradigms that produce a single point estimate, Bayesian learning generates a probability distribution over possible predictions or model parameters, capturing the uncertainty inherent in the model. This is achieved through **Bayes' Theorem**, which combines prior beliefs (prior probability) with new evidence (likelihood) to yield an updated belief (posterior probability).

Bayesian models can be applied in both supervised and unsupervised settings. In classification, the **Naive Bayes** classifier is a well-known Bayesian method that assumes feature independence, simplifying computation while remaining effective for many tasks. Another example is **Bayesian k-Nearest Neighbors (k-NN)**, where Bayesian principles are incorporated into the k-NN algorithm to handle uncertainty in predictions.

Contrast with Other Paradigms: Bayesian learning differs fundamentally from frequentist approaches, which focus on estimating fixed model parameters based on observed data without incorporating prior beliefs. In contrast to paradigms like supervised learning, which make point predictions, Bayesian learning uses probability distributions to represent uncertainty and can refine these distributions with new data. Unlike instance-based learning, which stores and uses all data for predictions, Bayesian learning often relies on a subset of data and incorporates **prior distributions,** providing a more structured approach to uncertainty.

In contrast to metric learning or contrastive learning, which focus on similarity measures in a latent space, Bayesian learning emphasises the probabilistic nature of predictions and can produce uncertainty estimates that are crucial in decision-making under risk.

Symbolic Diagram for Bayesian Learning



Durham University Palatine Centre Stockton Road Durham DH1 3LE UK
Telephone +44 (0)191 334 2000
www.durham.ac.uk

.

Durham University

Inspiring the extraordinary

General Formula for Bayesian Learning

General Formula for Bayesian Learning: Bayesian learning revolves around **Bayes' Theorem**, which defines the posterior probability $P(\theta|X)$ of a parameter θ given observed data X:

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

where:

- $P(\theta|X)$ is the **posterior probability** of the parameter θ given the data X,
- $P(X|\theta)$ is the **likelihood** of observing the data X given θ,
- $P(\theta)$ is the **prior probability** of θ,
- $P(X)$ is the **evidence** or marginal likelihood of observing X, acting as a normalising constant.

In **Naive Bayes classification**, for instance, Bayes' Theorem is used to compute the probability of a class C given features $X = \{x_1, x_2, \ldots, x_n\}$, assuming feature independence:

$$P(C|X) \propto P(C) \prod_{i=1}^{n} P(x_i|C)$$

References and Historical Cases: Bayesian learning has roots in the work of **Thomas Bayes** in the 18th century, who developed the initial ideas behind Bayes' Theorem. Bayesian inference, however, saw limited application until computational advances made it feasible for machine learning. The **Naive Bayes classifier**, formalised by **Duda and Hart** in 1973, became a widely used probabilistic classifier, particularly due to its simplicity and efficiency in handling high-dimensional data (Duda & Hart, 1973).

The **Bayesian k-Nearest Neighbors (k-NN)**, an extension of the traditional k-NN algorithm, was later introduced to incorporate Bayesian inference principles into instance-based learning, allowing it to model prediction uncertainty better. **Bayesian Networks**, developed by **Judea Pearl** in the 1980s, further expanded Bayesian learning's reach, enabling structured probabilistic reasoning across domains and earning Pearl the Turing Award in 2011 (Pearl, 1985).

Modern Bayesian learning has benefited from **Markov Chain Monte Carlo (MCMC)** and **Variational Inference** techniques, which make it possible to perform inference in complex Bayesian models, allowing Bayesian learning to be applied in high-dimensional settings, such as deep learning and reinforcement learning.

.

Durham University

Inspiring the extraordinary

Common Challenges and Limitations:

- **Computational Complexity:** Bayesian inference can be computationally intensive, particularly in high-dimensional spaces, due to the need to calculate posterior distributions.
- **Choice of Priors:** Selecting an appropriate prior can significantly impact the model's performance and is often subjective, especially when limited prior knowledge is available.
- **Interpretability:** While Bayesian models provide uncertainty estimates, the underlying mathematics and probability distributions may be challenging to interpret for non-expert users.
- **Scalability:** Bayesian methods struggle with very large datasets due to the high computational demands of calculating posteriors, though techniques like approximate Bayesian computation can help.

Discussion Questions:

1. How does the use of prior distributions in Bayesian learning impact model predictions, and what are the challenges in selecting an appropriate prior?
2. How does Bayesian learning handle uncertainty, and why might this be valuable in applications with high-risk decisions?
3. What are the trade-offs between interpretability and computational complexity in Bayesian models?
4. In what scenarios might Bayesian k-Nearest Neighbors be more advantageous than traditional k-NN?

References

1. Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons.
2. Pearl, J. (1985). "Bayesian networks: A model of self-activated memory for evidential reasoning." *Proceedings of the 7th Conference of the Cognitive Science Society*, 329-334.
3. Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1995). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.
4. Neal, R. M. (1996). *Bayesian Learning for Neural Networks.* Springer.

.

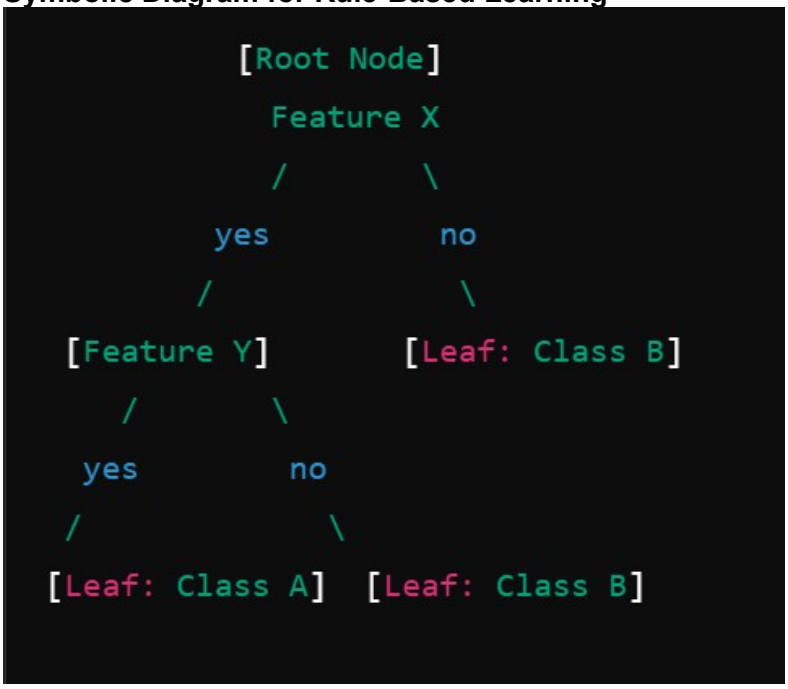
Durham University

Inspiring the extraordinary

Rule-Based Learning Paradigm

Concept and Contrast: Rule-based learning is a machine learning paradigm that uses logical rules derived from data to make predictions or decisions. This paradigm is particularly useful for interpretable models where the reasoning behind predictions can be explicitly traced through a series of conditions or rules. **Decision Trees** and **Association Rule Mining** are two common approaches in rule-based learning, each constructing rules differently to achieve transparent and interpretable models.
Decision Trees create rules by recursively splitting data based on feature values to form a tree structure where each path from root to leaf represents a rule for classification or regression.
Association Rule Mining, on the other hand, identifies relationships or correlations between variables within datasets, often used for tasks like market basket analysis. The generated association rules follow an “if-then” format, indicating how the presence of certain items or features predicts the presence of others.
Contrast with Other Paradigms: Unlike probabilistic paradigms such as Bayesian learning, which assign probability distributions to predictions, rule-based learning constructs deterministic rules based on data patterns. In contrast to instance-based learning (e.g., k-Nearest Neighbors), which relies on stored instances for prediction, rule-based learning generates explicit rules that can be applied independently of the training data.
While decision trees are often used in supervised settings, association rule mining operates on unlabeled data, identifying patterns and relationships without labels. Unlike neural networks, which act as "black boxes" by learning complex representations, rule-based learning offers high interpretability, making it suitable for applications where model transparency is critical.

Symbolic Diagram for Rule-Based Learning



.

Durham University

Inspiring the extraordinary

General Formula for Rule-Based Learning: In rule-based learning, each rule has a condition (antecedent) and a consequence (consequent) that dictates the prediction or association. For a decision tree, the rules can be summarised as:

1. Split the data based on feature X_i such that it maximises information gain or minimises impurity (e.g., Gini impurity, entropy).
2. Continue splitting recursively until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).
3. Each path from root to leaf represents a unique rule.

For Association Rule Mining, a common metric for rule quality is **confidence** and **support**:

$$\text{Support}(A \Rightarrow B) = \frac{\text{Number of transactions containing both } A \text{ and } B}{\text{Total transactions}}$$

$$\text{Confidence}(A \Rightarrow B) = \frac{\text{Number of transactions containing both } A \text{ and } B}{\text{Number of transactions containing } A}$$

These metrics help identify the strength of associations, with higher support and confidence indicating stronger relationships.

References and Historical Cases: Rule-based learning has roots in symbolic AI and early expert systems, with **Decision Trees** first formalised by **Ross Quinlan** in the form of **ID3 (Iterative Dichotomiser 3)** in 1986, which used information gain to decide splits. This method evolved into **C4.5** and **CART (Classification and Regression Trees)**, which incorporated additional techniques to handle both categorical and continuous data (Quinlan, 1993).
Association Rule Mining became popular with the development of the **Apriori algorithm** by **Agrawal et al.** in 1993, designed to discover frequent itemsets in large datasets and generate association rules. This work opened the door for applications in market analysis and retail, enabling insights into consumer behavior and product associations (Agrawal et al., 1993).
Rule-based learning has since evolved into advanced ensemble methods such as **Random Forests** and **Gradient Boosting Trees**, which combine multiple decision trees to improve accuracy and robustness. Recent research focuses on interpretable machine learning, where rule-based methods like decision trees are popular due to their transparency, especially in applications requiring explainability, such as healthcare and finance.

Common Challenges and Limitations:

- **Overfitting:** Decision trees are prone to overfitting, particularly when they grow too deep and capture noise in the data. Pruning and ensemble methods like Random Forests help mitigate this issue.

.

- **Scalability:** As data size and dimensionality increase, rule-based models, particularly association rule mining, may become computationally expensive due to the need to evaluate numerous item combinations.
- **Interpretability vs. Complexity:** While rule-based methods are generally interpretable, complex datasets may require an extensive number of rules, reducing model simplicity and interpretability.
- **Data Requirements:** Decision trees typically require large datasets to achieve stable splits, and association rule mining needs a high volume of transactions to identify meaningful patterns.

Discussion Questions:

1. What are the advantages of rule-based models like decision trees over black-box models, particularly in high-stakes applications?
2. How does association rule mining differ from supervised learning, and what types of problems is it best suited for?
3. How can pruning help reduce overfitting in decision trees, and what trade-offs does it introduce?
4. How does support and confidence in association rule mining impact the quality and interpretability of rules?

References

1. Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.
2. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Wadsworth.
3. Agrawal, R., Imieliński, T., & Swami, A. (1993). "Mining association rules between sets of items in large databases." *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207-216.

.

Durham University

Inspiring the extraordinary

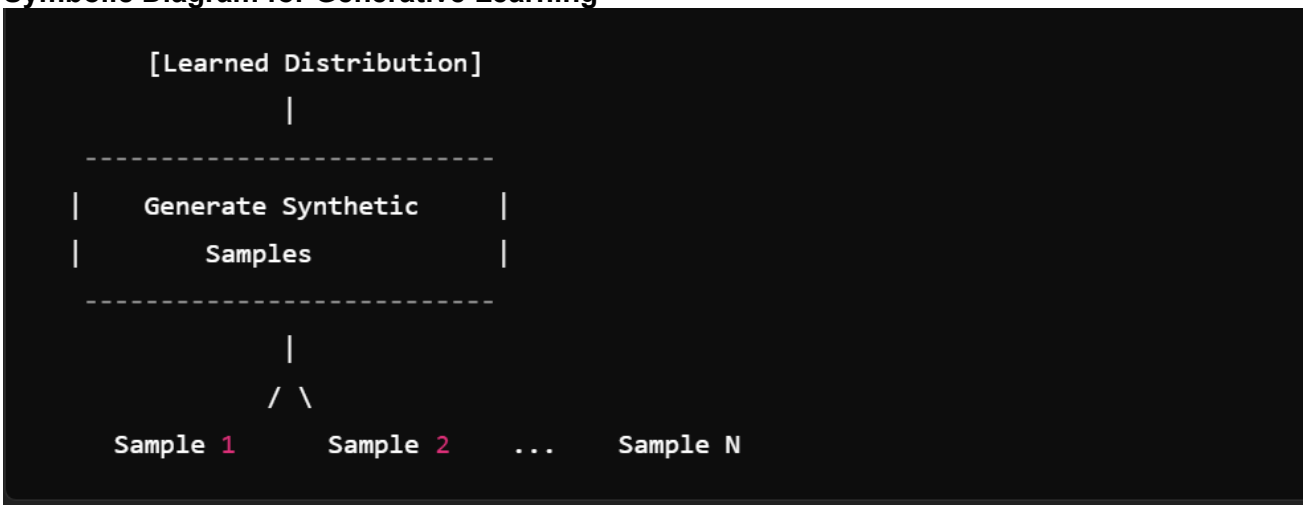
Generative Learning Paradigm

Concept and Contrast: Generative learning is a machine learning paradigm focused on learning the underlying distribution of data to generate new samples that resemble the original data. Unlike discriminative models, which focus solely on distinguishing between classes, generative models aim to model the joint probability distribution P(X,Y)P(X, Y)P(X,Y) (for labelled data) or P(X)P(X)P(X) (for unlabelled data) to learn how data is generated.
Generative learning is commonly employed in fields like image and text generation, anomaly detection, and semi-supervised learning. By capturing the distribution of data, generative models can simulate realistic data points, making them valuable in applications requiring synthetic data or creative content.

Contrast with Other Paradigms: Generative learning differs fundamentally from discriminative learning, such as supervised or instance-based learning, which model decision boundaries to classify data. Discriminative models focus on P(Y|X)P(Y|X)P(Y|X), or the probability of labels given features, rather than learning how the data was generated.

Generative models like **Gaussian Mixture Models (GMMs)** and **Hidden Markov Models (HMMs)** model the distribution of the data itself, often applied in unsupervised or semi-supervised settings. These models contrast with rule-based or instance-based learning, which rely on explicit rules or memorised instances. Generative models aim to generate realistic samples from learned distributions and are often more complex, making them computationally intensive.

Symbolic Diagram for Generative Learning



Explanation:

- The **Learned Distribution** represents the generative model's understanding of the data.
- **Synthetic Samples** are drawn from this distribution, resembling real data points.

.

General Formula for Generative Learning:

General Formula for Generative Learning: The objective in generative learning is to maximise the probability of observing the data given the model parameters, often using **Maximum Likelihood Estimation (MLE)** or **Maximum A Posteriori (MAP)** estimation.

For a generative model with parameters θ, the likelihood of data X is given by:

$$\mathcal{L}(\theta|X) = \prod_{i=1}^{n} P(x_i|\theta)$$

To find the optimal parameters, we maximise this likelihood:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} P(x_i|\theta)$$

For Gaussian Mixture Models, the distribution is modelled as a weighted sum of Gaussian distributions:

$$P(X) = \sum_{k=1}^{K} \pi_k \mathcal{N}(X|\mu_k, \Sigma_k)$$

where:

- K is the number of components,
- π_k is the mixture weight for component k,
- $\mathcal{N}(X|\mu_k, \Sigma_k)$ is the Gaussian distribution with mean μ_k and covariance Σ_k for component k.

References and Historical Cases: Generative learning has roots in statistical modelling, with early applications in **Gaussian Mixture Models (GMMs)** for clustering and density estimation. GMMs are widely used due to their flexibility in modelling data distributions as mixtures of Gaussians, making them applicable to complex, multi-modal data. Hidden Markov Models (HMMs), introduced by **Leonard E. Baum** in the 1960s, have been influential in sequential data modelling, particularly in speech recognition (Baum & Petrie, 1966).

Recent advancements include **Generative Adversarial Networks (GANs)** by **Ian Goodfellow et al. (2014)**, which revolutionised generative learning by introducing a game-theoretic approach to generation. GANs consist of two neural networks: a generator, which tries to create realistic samples, and a discriminator, which attempts to distinguish between real and synthetic data. This adversarial process pushes the generator to improve until it can produce highly realistic samples. Another impactful generative model is the **Variational Autoencoder (VAE)**, introduced by **Kingma**

.

Durham University

Inspiring the extraordinary

and Welling (2013), which uses a probabilistic approach to generate data through latent space encoding and decoding.
Applications of generative learning now span across text, audio, image synthesis, anomaly detection, and reinforcement learning, with applications like deep fakes, chatbots, and simulation environments.

Common Challenges and Limitations:

- **Mode Collapse:** Particularly in GANs, where the generator may produce limited patterns and fail to cover the full diversity of the data distribution.
- **High Computational Demand:** Generative models, especially deep neural architectures like GANs and VAEs, require substantial computational resources.
- **Evaluation Difficulty:** It's challenging to evaluate the quality of generated samples objectively, as common metrics may not always correlate with perceived quality.
- **Overfitting to Training Data:** Generative models may memorise training data, especially with limited datasets, leading to poor generalisation when generating new samples.
- **Interpretability:** While some simpler generative models like GMMs are interpretable, complex models like GANs lack transparency, which can be problematic in applications where interpretability is essential.

Discussion Questions:

1. What are the differences between generative and discriminative models, and when might each type be preferable?
2. How do Gaussian Mixture Models represent data distributions, and what are their strengths and limitations?
3. What are the challenges of training GANs, particularly with respect to mode collapse?
4. How do applications of generative learning differ across fields such as natural language processing and computer vision?

References

1. Baum, L. E., & Petrie, T. (1966). "Statistical inference for probabilistic functions of finite state Markov chains." *The Annals of Mathematical Statistics*, 37(6), 1554-1563.
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). "Generative adversarial nets." *Advances in Neural Information Processing Systems*, 2672-2680.
3. Kingma, D. P., & Welling, M. (2013). "Auto-encoding variational Bayes." *arXiv preprint arXiv:1312.6114*.

.

Durham University

Inspiring the extraordinary

Ensemble Learning Paradigm

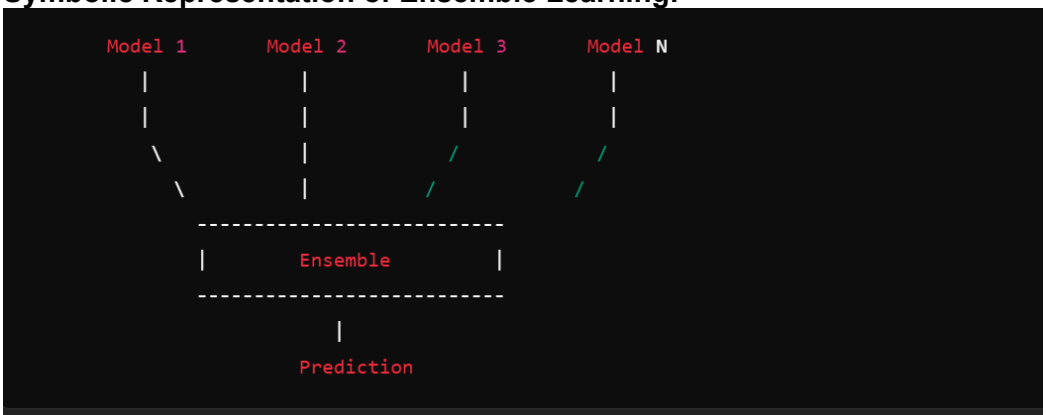
Concept: Ensemble learning is a machine learning paradigm that combines multiple models (often referred to as "weak learners") to improve overall predictive performance. By aggregating the predictions of multiple models, ensemble methods can achieve higher accuracy, reduce variance, and improve generalisation compared to individual models. Two widely used ensemble techniques are **Bagging** (Bootstrap Aggregating) and **Boosting**.

- **Bagging** aims to reduce model variance by training multiple models on different subsets of the data, created through random sampling with replacement. The final prediction is made by averaging (for regression) or taking a majority vote (for classification) from all models.
- **Boosting**, in contrast, is an iterative technique where each model attempts to correct the errors of the previous model. Boosting methods assign higher weights to misclassified samples in each iteration, allowing subsequent models to focus on hard-to-classify cases.

Ensemble learning provides robustness by averaging or adjusting multiple model outputs, making it especially valuable when individual models are prone to overfitting or have high variance.

Contrast with Other Paradigms: Ensemble learning contrasts with paradigms that rely on a single model to make predictions. Unlike **discriminative models**, which optimise a single model to classify data, ensemble learning achieves robustness by combining the strengths of multiple models. This aggregation helps reduce individual model weaknesses, resulting in higher accuracy and stability. Compared to **instance-based learning** (e.g., k-Nearest Neighbors), which uses stored data instances to make predictions, ensemble learning leverages multiple trained models and does not require storing the full training dataset. Additionally, ensemble methods differ from **Bayesian learning**, where probabilistic distributions quantify uncertainty. Ensemble learning does not provide probabilistic outcomes in the same way but still enhances robustness through model diversity.

Symbolic Representation of Ensemble Learning:



Explanation:

- Each individual model contributes to the **Ensemble**, which aggregates their outputs to make a final **Prediction**.
- Bagging and Boosting differ in how they train these models and aggregate predictions.

.

Durham University

Inspiring the extraordinary

General Formula for Ensemble Learning:

For Bagging, the ensemble prediction $\hat{y}_{\text{bag}}$ is given by averaging the predictions of M individual models $\hat{y}_i$ for regression tasks:

$$\hat{y}_{\text{bag}} = \frac{1}{M} \sum_{i=1}^{M} \hat{y}_i$$

For classification, Bagging takes a majority vote across models.

In Boosting, each model's prediction $\hat{y}_i$ is weighted based on its accuracy. The final prediction $\hat{y}_{\text{boost}}$ is a weighted sum of predictions:

$$\hat{y}_{\text{boost}} = \sum_{i=1}^{M} \alpha_i \hat{y}_i$$

where α_i is a weight assigned to model i, often based on its performance on previously misclassified samples.

References and Historical Cases: Ensemble learning has evolved through key techniques such as Bagging and Boosting:

1. **Bagging (Bootstrap Aggregating):** Introduced by **Leo Breiman** in 1996, Bagging trains multiple instances of the same model on different random subsets of data. **Random Forests**, an extension of Bagging, builds numerous decision trees on bootstrap samples and aggregates their predictions, making it one of the most popular ensemble methods (Breiman, 1996).
2. **Boosting:** Boosting emerged with **AdaBoost (Adaptive Boosting)**, developed by **Freund and Schapire** in 1997. AdaBoost iteratively trains weak classifiers, adjusting weights for misclassified examples to emphasise harder cases. Later, **Gradient Boosting** by **Friedman (2001)** extended the concept, using gradient descent to optimise each model's contribution to reducing residual errors (Freund & Schapire, 1997; Friedman, 2001).

These techniques have since inspired modern ensemble algorithms, such as **XGBoost** and **LightGBM**, which provide efficient, scalable implementations of gradient boosting for large datasets.

Discussion Questions:

1. How does ensemble learning reduce overfitting compared to individual models, particularly in Bagging and Boosting?
2. What are the computational trade-offs between Bagging (e.g., Random Forest) and Boosting (e.g., AdaBoost, Gradient Boosting)?
3. How does the iterative nature of Boosting make it effective in improving the performance of weak learners?
4. In what situations might Bagging outperform Boosting, and vice versa?

.

Durham University

Inspiring the extraordinary

References

1. Breiman, L. (1996). "Bagging predictors." *Machine Learning*, 24(2), 123-140.
2. Freund, Y., & Schapire, R. E. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences*, 55(1), 119-139.
3. Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*, 29(5), 1189-1232.

.

Durham University

Inspiring the extraordinary

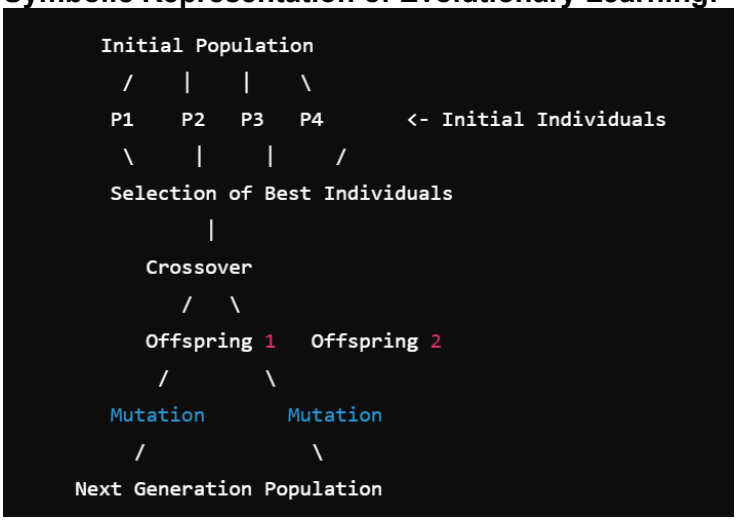
Evolutionary Learning Paradigm

Concept: Evolutionary learning is a machine learning paradigm inspired by the process of natural selection, where models evolve through generations to optimise a specific objective. Rather than relying on traditional optimisation techniques, evolutionary learning uses mechanisms like selection, mutation, crossover (recombination), and fitness evaluation to iteratively improve model performance. Popular techniques within this paradigm include **Genetic Algorithms (GA)**, **Evolution Strategies (ES)**, and **Particle Swarm Optimisation (PSO)**.

- **Genetic Algorithms (GA)** create a population of solutions (individuals) represented by encoded "genes." Through selection, crossover, and mutation, better solutions are generated over successive generations, simulating the process of natural selection.
- **Evolution Strategies (ES)** are a variant focusing on the evolution of parameters through mutation and selection, typically with real-valued parameters.
- **Particle Swarm Optimisation (PSO)** models a population of particles that "swarm" towards optimal solutions based on individual and collective experiences.

Contrast with Other Paradigms: Evolutionary learning contrasts sharply with gradient-based optimisation, such as those used in supervised learning and deep learning, where gradients guide the search for optimal parameters. In evolutionary learning, gradients are not required; instead, random variations, guided by fitness evaluations, drive the search process. Unlike ensemble learning, which combines multiple models, evolutionary algorithms optimise a single population of solutions, with weak or low-performing individuals removed from the population.
Compared to **reinforcement learning**, which also incorporates feedback from the environment, evolutionary algorithms do not require continuous or sequential feedback but instead evaluate fitness after each generation. This makes evolutionary learning effective for complex optimisation problems where gradients are hard to compute or where the objective function is not smooth.

Symbolic Representation of Evolutionary Learning:



Explanation:

- **P1, P2, P3, P4** represent initial individuals in the population.
- Selected individuals undergo **crossover** to produce offspring, followed by **mutation** to introduce variation.
- The process continues through generations, iterating toward optimal solutions.

.

Durham University

Inspiring the extraordinary

General Formula for Evolutionary Learning:

In evolutionary algorithms, the fitness function $f(x)$ is maximised or minimised based on the objective. Let P be the population of solutions, with each individual represented by x_i:

1. **Selection:** Choose individuals based on their fitness:

$$\text{Select}(x_i) \propto f(x_i)$$

2. **Crossover:** Combine pairs of individuals (parents) to produce offspring:

$$\text{Offspring} = \alpha x_i + (1 - \alpha) x_j$$

where $\alpha \in [0, 1]$ is a mixing ratio.

3. **Mutation:** Introduce random variations:

$$x_i = x_i + \delta$$

where δ is a small random perturbation, allowing exploration of the solution space.

4. **Fitness Evaluation:** Evaluate the new population and repeat until convergence or the maximum number of generations is reached.

References and Historical Cases: Evolutionary learning has its foundations in biological evolution, formalised in computational form in the 1960s:

- **Genetic Algorithms (GA):** Developed by **John Holland** in the 1970s, genetic algorithms are designed to simulate evolution in finding solutions to optimisation problems. Holland's work laid the groundwork for encoding solutions as chromosomes and evolving them through selection, crossover, and mutation (Holland, 1975).
- **Evolution Strategies (ES):** Originating in the 1960s with **Ingo Rechenberg** and **Hans-Paul Schwefel**, evolution strategies focus on optimising continuous parameters and have been widely applied in engineering fields for optimisation tasks (Rechenberg, 1973).
- **Particle Swarm Optimisation (PSO):** Proposed by **Kennedy and Eberhart** in 1995, PSO models optimisation as a swarm of particles moving through a solution space, adjusting based on individual and swarm experiences. This method quickly gained popularity for its simplicity and effectiveness across various optimisation problems (Kennedy & Eberhart, 1995).

Modern evolutionary algorithms have found applications in neural architecture search, where the evolutionary process searches for optimal neural network architectures, and in optimisation problems that are non-differentiable, making gradient-based methods infeasible.

5. Discussion Questions:

1. How does evolutionary learning differ from gradient-based optimisation, and when is it preferable?
2. What are some challenges in balancing exploration and exploitation in evolutionary algorithms?

.

Durham University

Inspiring the extraordinary

3. How does mutation contribute to evolutionary algorithms' ability to explore diverse solutions?
4. In what types of real-world problems might particle swarm optimisation or genetic algorithms be particularly useful?

References

1. Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
2. Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog.
3. Kennedy, J., & Eberhart, R. (1995). "Particle swarm optimization." *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942-1948.

.

Durham University

Inspiring the extraordinary

Representation Learning Paradigm

Concept: Representation learning is a machine learning paradigm aimed at discovering useful representations or features from raw data, often without relying on human-designed features. The goal is to learn compact, informative, and generalisable embeddings that capture the essential characteristics of the data, making it suitable for downstream tasks like classification, clustering, and regression. Techniques like **Autoencoders**, **Principal Component Analysis (PCA)**, and **Word2Vec** are commonly used in representation learning to uncover low-dimensional structures in high-dimensional data.

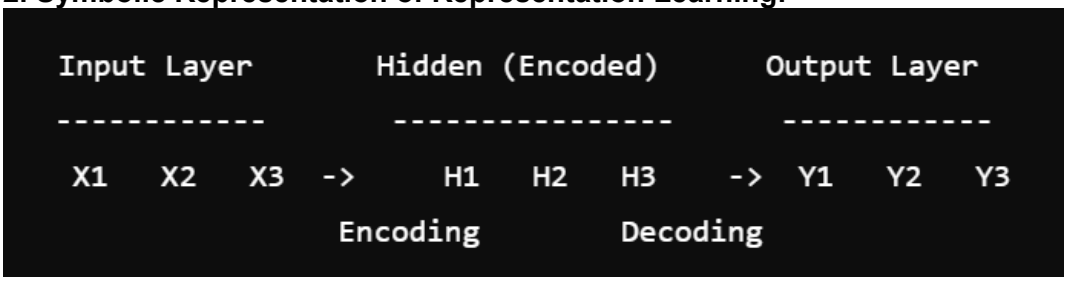
- **Autoencoders** are neural networks that learn compressed representations by encoding data into a lower-dimensional space and then reconstructing it.
- **Principal Component Analysis (PCA)** is a linear method that projects data onto a lower-dimensional space, retaining the maximum variance.
- **Word2Vec** and other word embedding models transform high-dimensional words into dense vector representations, capturing semantic relationships in text data.

Contrast with Other Paradigms: Representation learning contrasts with paradigms like supervised learning, where the focus is on direct predictions using labelled data. Instead of merely outputting predictions, representation learning focuses on finding informative data encodings. This contrasts with **instance-based learning**, which relies on storing raw data points rather than transforming them into new representations. Unlike **rule-based learning**, representation learning does not produce explicit if-then rules but instead learns continuous, abstract representations that can generalise across data types.

Compared to unsupervised learning, representation learning can operate in both supervised and unsupervised contexts, though it is often self-supervised. In contrast to generative learning, which models the data distribution for generating new samples, representation learning focuses on capturing the structure of data for representation, not generation.

---

2. Symbolic Representation of Representation Learning:



Explanation:

- **Input Layer:** The raw data features (e.g., X1, X2, X3).
- **Hidden (Encoded) Layer:** The compressed representation or encoding (e.g., H1, H2, H3), capturing essential data patterns.
- **Output Layer:** Reconstructed data, ideally similar to the input, allowing the model to learn representations.

.

Durham University

Inspiring the extraordinary

3. General Formula for Representation Learning:

In autoencoders, the objective is to minimise the reconstruction error between the input X and the reconstructed output Y. Let f represent the encoding function, and g the decoding function; the loss function L can be defined as:

$$L(X, Y) = \|X - g(f(X))\|^2$$

where:

- X is the original input data,
- $g(f(X))$ is the reconstructed output after encoding and decoding,
- $\|\cdot\|^2$ denotes the squared error, commonly used to measure reconstruction accuracy.

For **Principal Component Analysis (PCA)**, the objective is to maximise the variance captured in the lower-dimensional representation, which is achieved by projecting the data onto the principal components (eigenvectors) of the covariance matrix.

For **Principal Component Analysis (PCA)**, the objective is to maximise the variance captured in the lower-dimensional representation, which is achieved by projecting the data onto the principal components (eigenvectors) of the covariance matrix.

References and Historical Cases: Representation learning has deep roots in data transformation and dimensionality reduction techniques:

1. **Principal Component Analysis (PCA):** Introduced by **Karl Pearson** in 1901, PCA has been widely applied in reducing data dimensionality by transforming it onto principal components that capture the most variance (Pearson, 1901).
2. **Autoencoders:** Developed in the 1980s, autoencoders were initially designed for unsupervised pre-training of neural networks. They became popular in deep learning, enabling non-linear dimensionality reduction and feature learning (Hinton & Salakhutdinov, 2006).
3. **Word2Vec:** Proposed by **Tomas Mikolov et al. (2013)**, Word2Vec and other word embedding methods transformed NLP by creating dense, continuous vector representations of words, preserving semantic relationships (Mikolov et al., 2013).

Modern advances include **Variational Autoencoders (VAEs)**, which add probabilistic interpretations to autoencoders, and **Transformers**, which enable complex representation learning in NLP by capturing long-range dependencies in text.

Discussion Questions:

1. How does representation learning benefit machine learning models in comparison to using raw features?
2. What are the advantages and disadvantages of linear methods like PCA versus non-linear methods like autoencoders?

.

Durham University

Inspiring the extraordinary

3. How does representation learning differ in applications such as image processing versus natural language processing?
4. In what scenarios might an autoencoder's learned representation be preferable over traditional dimensionality reduction methods?

References

1. Pearson, K. (1901). "On lines and planes of closest fit to systems of points in space." *Philosophical Magazine*, 2(11), 559-572.
2. Hinton, G. E., & Salakhutdinov, R. R. (2006). "Reducing the dimensionality of data with neural networks." *Science*, 313(5786), 504-507.
3. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781*.

.

Durham University

Inspiring the extraordinary

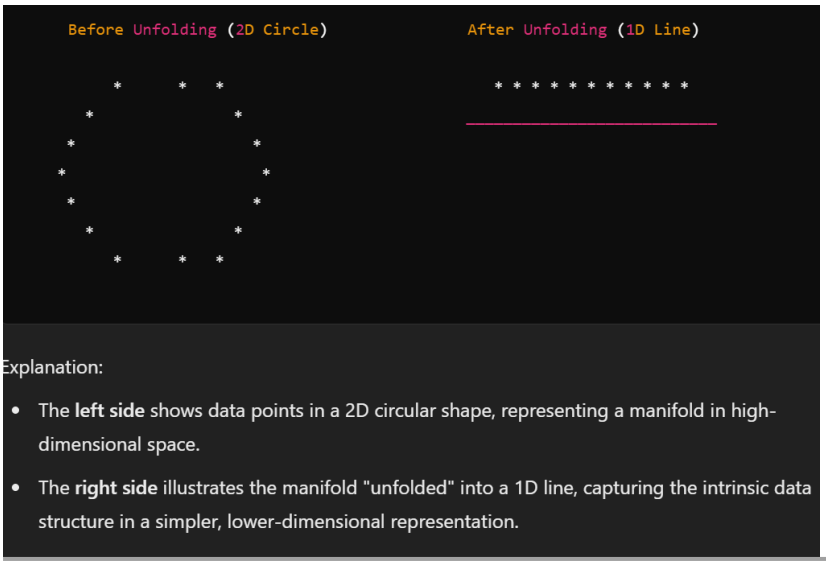
Manifold Learning Paradigm

Concept:
Manifold learning is a machine learning paradigm focused on understanding the low-dimensional structure of high-dimensional data by mapping it to a manifold—a lower-dimensional space that captures essential relationships and patterns. This paradigm assumes that data in high-dimensional space actually resides on a lower-dimensional manifold, allowing complex relationships to be represented in a simplified, compact form. **Matrix Factorisation**, **Tensor Factorisation**, **Locally Linear Embedding (LLE)**, **t-SNE (t-distributed Stochastic Neighbour Embedding)**, and **Tucker Decomposition** are key techniques in manifold learning, each with specific strengths in extracting and representing manifold structures.

- **Matrix Factorisation** and **Tensor Factorisation** decompose data matrices or tensors into lower-dimensional representations, useful for tasks like recommendation systems.
- **Locally Linear Embedding (LLE)** preserves local relationships, mapping points in high-dimensional space to lower-dimensional space while maintaining neighbourhood information.
- **t-SNE** is a non-linear dimensionality reduction technique that preserves local similarities in data and is often used for visualisation of high-dimensional data in 2D or 3D space.
- **Tucker Decomposition** is a type of tensor factorisation that generalises matrix factorisation to multi-dimensional arrays, effectively capturing multi-modal data patterns.

Contrast with Other Paradigms:
Manifold learning differs from traditional dimensionality reduction methods like PCA, which assumes linear relationships in data and projects it onto a linear subspace. In contrast, manifold learning can capture non-linear structures, making it suitable for complex, high-dimensional datasets that follow non-linear patterns. Unlike representation learning, which learns informative embeddings for downstream tasks, manifold learning specifically focuses on finding a low-dimensional, interpretable structure within data. Unlike instance-based learning, which stores data and uses it directly for prediction, manifold learning abstracts a compressed representation, focusing on the geometry of the data itself rather than individual instances.

Symbolic Representation of Manifold Learning:



.

Durham University

Inspiring the extraordinary

General Formula for Manifold Learning:

General Formula for Manifold Learning:

In manifold learning, an objective is often to minimise reconstruction error between original and mapped points while preserving local or global structure.

For **Locally Linear Embedding (LLE)**, we aim to find weights W that best reconstruct each point x_i using its neighbours:

$$\min_W \sum_i \|x_i - \sum_j W_{ij} x_j\|^2$$

where W_{ij} represents the weight of neighbour x_j in the reconstruction of x_i. The goal is to find a lower-dimensional embedding Y that preserves these weights.

For **t-SNE**, we minimise the Kullback-Leibler (KL) divergence between the similarity distributions of points in high-dimensional space P and lower-dimensional space Q:

$$\min_Y \sum_{i \neq j} P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

where P_{ij} is the probability of similarity between points i and j in high-dimensional space, and Q_{ij} is the similarity in the lower-dimensional embedding.

References and Historical Cases

Manifold learning has grown from techniques in data decomposition and visualisation:

1. **Principal Component Analysis (PCA):** While not a manifold technique, PCA set the stage for linear dimensionality reduction. **Hotelling (1933)** and **Pearson (1901)** first explored it as a statistical tool for variance maximisation (Hotelling, 1933; Pearson, 1901).
2. **Locally Linear Embedding (LLE):** Developed by **Saul and Roweis** in 2000, LLE aimed to preserve local neighbourhood relationships while mapping high-dimensional data onto a lower-dimensional manifold, making it useful for non-linear structures (Roweis & Saul, 2000).
3. **t-SNE (t-distributed Stochastic Neighbor Embedding):** Proposed by **van der Maaten and Hinton** in 2008, t-SNE became widely used for visualising complex high-dimensional data in 2D or 3D while preserving local structure, making it a popular choice for exploring data like word embeddings or gene expression profiles (van der Maaten & Hinton, 2008).
4. **Tensor Factorisation (Tucker Decomposition): Tucker (1966)** introduced a form of multi-way data decomposition that extends matrix factorisation into multi-dimensional arrays (Tucker, 1966). This method is commonly applied in image processing and recommendation systems.

.

Durham University

Inspiring the extraordinary

Recent advancements focus on scalable manifold learning for large datasets, deep learning variants of manifold techniques, and adaptive manifold learning methods to better capture complex, dynamic structures.

Discussion Questions:

- What types of data are best suited for manifold learning, and how does it compare to linear methods like PCA in these contexts?
- How does LLE's focus on local neighbourhoods differ in approach and outcome from t-SNE's probabilistic mapping?
- What are the challenges associated with using tensor factorisation in high-dimensional multi-modal data?
- How might manifold learning be adapted to improve scalability for large datasets?

References

1. Hotelling, H. (1933). "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, 24(6), 417-441.
2. Pearson, K. (1901). "On lines and planes of closest fit to systems of points in space." *Philosophical Magazine*, 2(11), 559-572.
3. Roweis, S. T., & Saul, L. K. (2000). "Nonlinear dimensionality reduction by locally linear embedding." *Science*, 290(5500), 2323-2326.
4. van der Maaten, L., & Hinton, G. (2008). "Visualizing data using t-SNE." *Journal of Machine Learning Research*, 9(11).
5. Tucker, L. R. (1966). "Some mathematical notes on three-mode factor analysis." *Psychometrika*, 31(3), 279-311.

.

Durham University

Inspiring the extraordinary

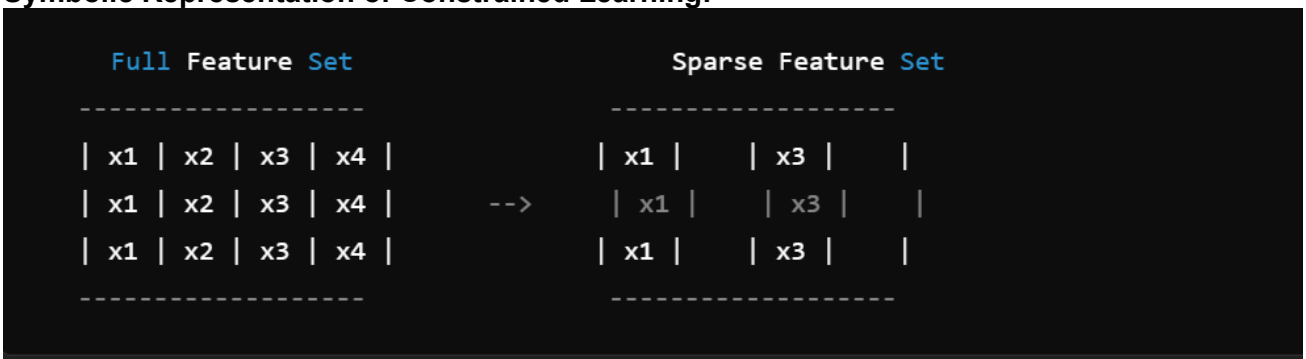
Constrained Learning Paradigm

Concept:
Constrained learning is a machine learning paradigm that incorporates constraints into the learning process to enforce specific properties on the model or its outputs. By applying constraints, constrained learning can achieve more interpretable, efficient, or generalisable models, particularly in scenarios where data or computational resources are limited. **Sparse Learning** and **Low-Rank Learning** are two common forms of constrained learning:

- **Sparse Learning** enforces sparsity by encouraging models to select only a subset of features or parameters, reducing complexity and focusing on the most informative parts of the data.
- **Low-Rank Learning** imposes a low-rank constraint on matrices or tensors, enabling the model to identify low-dimensional structures, which is especially useful in tasks like matrix completion and collaborative filtering.

Contrast with Other Paradigms:
Constrained learning differs from standard machine learning paradigms that do not enforce specific structural restrictions on model parameters or outputs. Unlike **representation learning**, which aims to discover meaningful data encodings, constrained learning applies explicit constraints, such as sparsity or low rank, to encourage efficient, interpretable models. In contrast to **generative learning**, which models the full data distribution, constrained learning may ignore certain aspects of the data to achieve a simpler, more focused model.
While **manifold learning** aims to reduce data to a lower-dimensional manifold, constrained learning focuses on sparsity or low-rank structures to simplify models directly, often resulting in easier-to-interpret and faster-to-compute solutions.

Symbolic Representation of Constrained Learning:



Explanation:

- **Left:** A matrix with all features included.
- **Right:** A sparse matrix where certain features (e.g., $x2$ and $x4$) are omitted to encourage sparsity.

.

Durham University

Inspiring the extraordinary

General Formula for Constrained Learning:

In constrained learning, constraints are incorporated into the objective function to encourage sparse or low-rank solutions.

For **Sparse Learning**, an L_1 regularisation term is added to the objective to promote sparsity. The objective function is:

$$\min_W \|Y - XW\|^2 + \lambda \|W\|_1$$

where:

- Y is the target,
- X is the input data,
- W is the weight matrix,
- $\|W\|_1$ is the L_1-norm, promoting sparsity,
- λ is a regularisation parameter controlling the degree of sparsity.

For **Low-Rank Learning**, a low-rank constraint is applied, often using the nuclear norm $\|\cdot\|_*$:

$$\min_W \|Y - XW\|^2 + \lambda \|W\|_*$$

where $\|W\|_*$ is the nuclear norm, encouraging a low-rank solution in W.

References and Historical Cases

Constrained learning, particularly sparse and low-rank approaches, has been essential in various fields:

1. **Sparse Learning: Tibshirani (1996)** introduced **Lasso (Least Absolute Shrinkage and Selection Operator)**, a widely used sparse learning method that incorporates an L1L_1L1 regularisation penalty to enforce sparsity in the model, especially for high-dimensional data (Tibshirani, 1996).
2. **Low-Rank Learning:** Low-rank methods gained prominence with **SVD (Singular Value Decomposition)** and applications in collaborative filtering, where low-rank matrix factorisation helps in recommendation systems. **Candes and Recht (2009)** applied low-rank learning to matrix completion, showing that it could recover missing entries from partially observed data (Candes & Recht, 2009).

Modern applications of constrained learning include high-dimensional genomics, compressed sensing in imaging, and efficient feature selection in large-scale machine learning, where

.

Durham University

Inspiring the extraordinary

interpretability and computational efficiency are critical. Here lists some extended examples of constraints:

- **Sparsity Constraint**: Limits the number of non-zero elements in a vector or matrix, promoting models that use fewer features or parameters (e.g., L1L_1L1 regularisation in Lasso).
- **Low-Rank Constraint**: Enforces a low rank on matrices or tensors, often applied in matrix completion and collaborative filtering tasks to capture simplified representations.
- **Non-Negativity Constraint**: Requires all elements in a solution to be non-negative, commonly used in fields like image processing (e.g., Non-negative Matrix Factorisation).
- **Smoothness Constraint**: Encourages smooth changes across neighbouring elements, often applied in signal processing and image reconstruction (e.g., Total Variation regularisation).
- **Orthogonality Constraint**: Enforces orthogonal (perpendicular) components, often used in dimensionality reduction and feature learning to encourage independence between features.
- **Trace Constraint**: Limits the trace (sum of diagonal elements) of a matrix, useful in controlling the complexity of covariance matrices in optimisation problems.
- **Box Constraint**: Specifies upper and lower bounds on each variable in a model, commonly used in optimisation tasks to prevent values from exceeding practical ranges.
- **Equality Constraint**: Forces certain variables or functions to have fixed relationships, such as enforcing symmetry or fixed values, common in linear programming.
- **Budget Constraint**: Imposes a limit on resource usage, such as the number of selected features, model size, or computation time, ensuring efficient use of resources.
- **Group Sparsity Constraint**: Extends sparsity constraints by enforcing entire groups of variables to be zero, used for structured feature selection (e.g., Group Lasso).

Discussion Questions:

- How does adding sparsity constraints in model training impact model interpretability and performance?
- In what types of applications would low-rank constraints be more beneficial than sparse constraints?
- What are the computational challenges associated with enforcing low-rank constraints in large-scale data?
- How does constrained learning contribute to model efficiency, particularly in resource-limited environments?

References

1. Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267-288.
2. Candes, E. J., & Recht, B. (2009). "Exact matrix completion via convex optimization." *Foundations of Computational Mathematics*, 9(6), 717-772.

.

Durham University

Inspiring the extraordinary

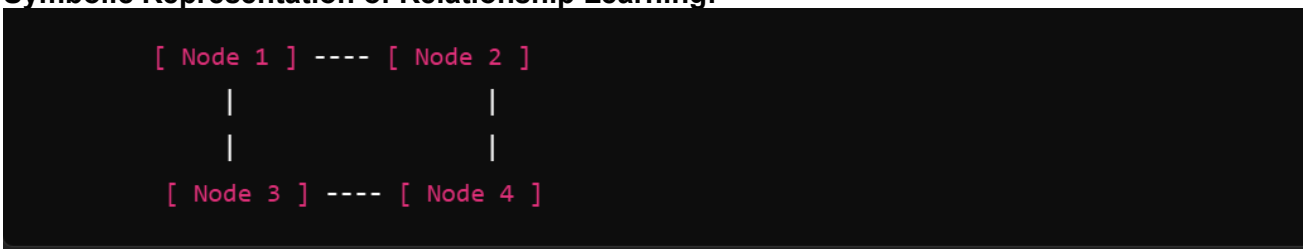
Relationship Learning Paradigm

Concept:
Relationship learning is a machine learning paradigm focused on identifying and modelling the relationships or dependencies among entities within a dataset. It's particularly valuable when data points are interconnected, as in social networks, recommendation systems, or molecular structures. Two popular techniques in relationship learning are **Graph-Based Learning** and **Probabilistic Graphical Models (PGMs)**.

- **Graph-Based Learning** models data as nodes (entities) and edges (relationships) in a graph, where graph structures represent dependencies. Techniques like **Graph Neural Networks (GNNs)** learn embeddings for nodes or edges by aggregating information from neighbouring nodes, capturing relational patterns in data.
- **Probabilistic Graphical Models (PGMs)**, such as **Bayesian Networks** and **Markov Random Fields**, use a probabilistic framework to model the dependencies among random variables. PGMs allow for uncertainty in relationships, making them suitable for structured probabilistic reasoning in areas like genetics, robotics, and recommendation systems.

Contrast with Other Paradigms:
Relationship learning differs from paradigms like supervised learning, which generally assumes data points are independent and identically distributed. In relationship learning, data points are dependent, and the structure of these dependencies is as important as the data values themselves. Compared to **representation learning**, which focuses on finding informative data encodings, relationship learning explicitly models connections among data points.
Relationship learning also differs from **instance-based learning** in that it doesn't treat each instance independently; instead, it uses interdependencies to make predictions. Probabilistic Graphical Models offer a probabilistic approach, unlike deterministic models in rule-based learning, by representing uncertainties in relationships and allowing for probabilistic inference.

Symbolic Representation of Relationship Learning:



Explanation:

- **Nodes** represent entities (e.g., people in a social network, or genes in a gene network).
- **Edges** represent relationships or dependencies between nodes, capturing relational patterns within the data.

.

Durham University

Inspiring the extraordinary

General Formula for Relationship Learning:

In **Graph-Based Learning**, many tasks involve calculating properties or metrics of the graph, such as node degrees, paths, or adjacency structures. For example, the **adjacency matrix** A of a graph G captures direct relationships between nodes:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

This matrix A serves as a foundational tool to measure and utilise relationships among nodes. Metrics like **node degree** $d(i)$, which represents the number of connections for node i, can be derived as:

$$d(i) = \sum_{j \in V} A_{ij}$$

In **Probabilistic Graphical Models (PGMs)**, the structure of a graph (directed or undirected) is used to model dependencies among random variables represented as nodes. For example, in a **Bayesian Network**, the joint probability distribution of nodes $X = \{X_1, X_2, \ldots, X_n\}$ given their dependency structure is:

$$P(X) = \prod_{i=1}^{n} P(X_i | \text{Parents}(X_i))$$

where **Parents**(X_i) refers to the set of nodes with directed edges pointing to X_i. This structure simplifies calculations by taking advantage of conditional independence, which reduces the number of terms in the joint distribution.

References and Historical Cases

Relationship learning emerged from network theory and probabilistic reasoning:

1. **Graph Theory:** Early work in graph theory by **Euler** in 1736 laid the foundation for representing data relationships as nodes and edges (Euler, 1736). Modern Graph Neural Networks (GNNs) gained traction with the **Graph Convolutional Network (GCN)** by **Kipf and Welling (2017)**, which generalised convolution operations for graphs and became popular in social network analysis and recommendation systems (Kipf & Welling, 2017).
2. **Probabilistic Graphical Models:** Developed by **Judea Pearl** in the 1980s, Bayesian Networks (a type of PGM) allowed for structured probabilistic reasoning with conditional dependencies. Pearl's work in causal inference and probabilistic reasoning earned him the Turing Award in 2011 (Pearl, 1988). Markov Random Fields, another type of PGM, provide undirected models suited for image processing and spatial data.

.

Durham University

Inspiring the extraordinary

Today, relationship learning is widely applied in recommendation systems, molecular property prediction, social network analysis, and knowledge graph construction, where understanding the interconnections among entities is essential. Here lists extended graph approaches:

- **Spectral Graph Theory**: Uses the **spectrum of the graph Laplacian** (eigenvalues and eigenvectors) to study graph properties. Spectral methods are widely used in tasks like clustering, community detection, and dimensionality reduction by mapping nodes to a lower-dimensional space where relationships are preserved.
- **Graph Cuts**: Divides a graph into disjoint subsets by removing edges (cuts) to minimise the connection between nodes in different partitions. **Min-cut** and **Normalized-cut** algorithms are popular for image segmentation and network partitioning, where the goal is to achieve balanced and meaningful splits based on graph connectivity.
- **Hypergraphs**: Extends traditional graphs by allowing edges (hyperedges) to connect more than two nodes. This approach is useful for modelling complex relationships among groups of entities, such as multi-party interactions or collaborative filtering in recommendation systems.
- **Random Walks on Graphs**: Utilises random traversal of nodes to explore graph structures. A **PageRank** algorithm, for instance, models a random surfer on a web graph to rank the importance of pages. Random walks are also employed in node ranking, feature extraction, and link prediction.
- **Graph Matching**: Finds correspondences between nodes in two graphs, identifying similar substructures. It's used in applications like image matching, where similar objects in different images are matched based on graph representations. **Subgraph Isomorphism** is a common algorithm in graph matching.
- **Graph Kernels**: Measures similarity between graphs by comparing node and edge patterns, allowing kernel-based learning methods like SVMs to operate on graph-structured data. Examples include the **Weisfeiler-Lehman graph kernel**, which maps nodes to labels iteratively, capturing structural similarity.
- **Graph Clustering and Community Detection**: Groups nodes based on their connectivity patterns. **Modularity maximisation** and **Louvain algorithm** are commonly used methods to identify communities within social networks or biological graphs, detecting clusters of closely connected nodes.
- **Graph Embedding**: Maps nodes or edges to a continuous vector space, preserving structural properties. Techniques like **DeepWalk** and **node2vec** use random walks to generate embeddings, capturing complex node relationships for tasks like node classification and link prediction.
- **Graph Convolutional Networks (GCNs)**: Extends traditional convolutional networks to graph data, using a message-passing framework to aggregate information from neighbouring nodes. GCNs are powerful for semi-supervised learning on graph-structured data, such as citation networks and social networks.
- **Multi-Relational Graphs and Knowledge Graphs**: Represents entities and relationships in a graph with multiple types of nodes and edges, commonly seen in knowledge graphs (e.g., entities and their connections). Multi-relational models like **TransE** and **RotatE** embed nodes and edges into a space that captures diverse relationships.

Discussion Questions:

- How does relationship learning leverage dependencies between data points to improve predictions?

.

Durham University

Inspiring the extraordinary

- What are the primary challenges associated with learning from graph-structured data?
- In what contexts would probabilistic graphical models be more advantageous than deterministic graph-based models?
- How does relationship learning apply to real-world networks, such as social networks or biological networks?

References

1. Euler, L. (1736). "Solutio problematis ad geometriam situs pertinentis." *Commentarii academiae scientiarum Petropolitanae*, 8, 128-140.
2. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
3. Kipf, T. N., & Welling, M. (2017). "Semi-supervised classification with graph convolutional networks." *International Conference on Learning Representations (ICLR)*.

.

Durham University

Inspiring the extraordinary

Deep Learning

Deep learning is a machine learning foundation focused on using multi-layered neural networks to automatically learn representations from raw data. It is particularly powerful in handling unstructured data like images, text, and audio. Within deep learning, several specialised approaches have emerged:

- **Attention-Based Learning**: Utilises attention mechanisms that allow models to focus selectively on specific parts of the input, greatly improving performance in sequence-based tasks such as language translation (e.g., Transformer models).
- **Self-Organising Learning**: Uses unsupervised techniques, like self-organising maps (SOMs), where the model autonomously discovers patterns by grouping similar data points, often applied in data clustering and visualisation.
- **Hyperparameter Learning**: Optimises the selection of hyperparameters (parameters set before model training) using methods like grid search, random search, or Bayesian optimisation, critical for tuning deep learning models.
- **Neural Architecture Search (NAS) Learning**: Automates the process of designing neural network architectures by searching for optimal architectures using reinforcement learning or evolutionary algorithms.

Reinforcement Learning

Reinforcement learning (RL) is a paradigm where agents learn to make decisions by interacting with an environment and maximising cumulative rewards. It's especially effective for tasks with sequential decision-making, like robotics or game playing.

- **Inverse Reinforcement Learning (IRL)**: Seeks to infer the underlying reward function based on observed behaviours, commonly used when it's difficult to define the reward explicitly but examples of expert behaviour are available, such as in imitation learning.

Recommender Systems

Recommender systems are designed to suggest items of interest to users, commonly seen in e-commerce, streaming services, and social media.

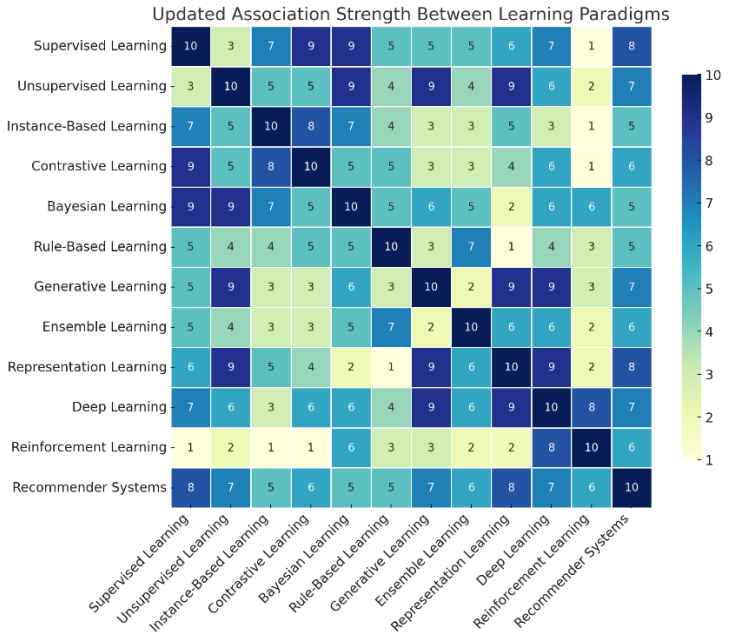
- **Collaborative Filtering**: Recommends items by finding patterns in user-item interactions, such as users with similar preferences or items often rated similarly, typically using matrix factorisation or neighbourhood-based methods.
- **Content-Based Filtering**: Suggests items based on their attributes, comparing item features with user preferences, often used in conjunction with collaborative filtering for hybrid recommendation systems.

These learning foundations support a wide variety of applications, enabling systems to capture complex patterns, interact with dynamic environments, and personalise user experiences. At the end of this section, we have a fun table to think about as follows. You can zoom in to find out and think about associations between the learning paradigms.

Durham University

Inspiring the extraordinary

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Paradigm**	**Supervised**	**Unsupervised**	**Instance-Based**	**Contrastive**	**Bayesian**	**Rule-Based**	**Generative**	**Ensemble**	**Representation**	**Deep Learning**	**Reinforcement**	**Recommender**
2	Supervised	-	3: Limited interaction, as unsupervised lacks labelled data, unlike supervised	7: Many instance-based models (e.g., k-NN) work within supervised contexts	9: Strong association; contrastive methods like metric learning are often supervised	9: Bayesian methods (e.g., Naive Bayes) are common supervised classifiers	5: Moderate association through decision trees for rule learning	5: Generative models like GMMs can assist supervised classification	5: Ensemble techniques like bagging & boosting improve supervised model performance	6: Supervised models often use embeddings from representation learning	7: Supervised deep models excel in labelled data tasks	1: Weak association; RL focuses on action-reward loops, not on labelled data	8: Recommender systems frequently use supervised learning to classify preferences
3	Unsupervised	3: Different approaches due to lack of labelled data	-	5: Some instance-based methods like clustering align with unsupervised methods	5: Moderate; unsupervised data pairs are used in contrastive learning (e.g., clustering)	9: Bayesian methods model data distributions, often in unsupervised settings	4: Limited rule discovery in unsupervised contexts	9: Generative models are commonly unsupervised, learning data distributions	4: Limited overlap as ensemble methods need labelled data for evaluation	9: Strong; many representation techniques (e.g., PCA) are unsupervised	6: Deep networks often pre-train with unsupervised learning for representation	2: Minimal overlap; unsupervised has less focus on reward-based learning	7: Recommenders use unsupervised clustering for user-item groups
4	Instance-Based	7: Works well in supervised contexts as in k-NN	5: Uses data structure for clustering	-	8: Distance metrics in instance-based learning connect well with contrastive learning	7: Bayesian k-NN and probabilistic neighbours share similarity in learning	4: Occasional use in rule-based, but limited structure here	3: Weak association; instance-based lacks generative focus	3: Rarely combines with ensemble techniques	5: Moderate association; embeddings can improve distance calculations in instance models	3: Limited overlap, deep learning rarely relies on memorisation	1: Very weak association, as RL relies on decision-making not instance similarity	5: Recommenders use k-NN for user-item similarity
5	Contrastive	9: Contrastive methods in supervised learning (e.g., triplet loss)	5: Semi-use in unsupervised contrastive learning with augmentations	8: Relies on instance-based distance metrics	-	5: Probabilistic distances relate indirectly to contrastive comparisons	5: Limited overlap; rule-based doesn't rely on comparisons	3: Weak link; generative focuses on new data, not comparison	3: Minor role; ensembles do not focus on contrastive learning	4: Uses representations from contrastive learning for embeddings	6: Deep models use contrastive learning for self-supervised and unsupervised tasks	1: Weak link as RL does not rely on sample comparisons	6: Contrastive embeddings improve content-based recommendations
6	Bayesian	9: Probabilistic methods are common in supervised contexts (e.g., Naive Bayes)	9: Bayesian models like GMM work unsupervised	7: Shared use with Bayesian k-NN in probabilistic neighbour learning	5: Some similarity in probabilistic distance metrics	-	5: Overlap with rule-based Bayesian networks for decision trees	6: Probabilistic models assist in generative models, e.g., Gaussian processes	5: Ensemble methods can be Bayesian (Bayesian Boosting)	2: Less overlap, representation learning doesn't use probabilistic approaches heavily	6: Bayesian networks support probabilistic aspects in deep models	6: Bayesian RL explores probabilistic decision-making	5: Bayesian networks improve uncertainty in recommendation systems
7	Rule-Based	5: Rule extraction from decision trees in supervised settings	4: Limited unsupervised usage, minimal overlap	4: Rule discovery does not typically rely on instance similarity	5: Limited interaction; rule-based focuses on decision paths, not comparisons	5: Bayesian networks used for probabilistic rules	-	3: Generative methods are less structured around rules	7: Ensembles of rules (e.g., random forests) in rule-based learning	1: Less connection; rule-based does not seek embeddings	4: Minimal overlap, deep models are not rule-driven	3: Weak link, RL focuses on actions rather than rule sets	5: Rule-based methods assist in knowledge-based recommenders
8	Generative	5: Uses supervised classification with GMMs	9: Common in unsupervised learning to learn distributions	3: Minimal overlap; generative does not rely on memorised instances	3: Limited link; generative focuses on data creation	6: Bayesian models, e.g., Gaussian processes, assist generative models	3: Minor overlap; rule-based models are deterministic, not generative	-	2: Weak link; ensemble techniques rarely focus on generation	9: Strong link as embeddings improve generative models	9: GANs in deep learning are central to generative models	3: Limited connection; RL focuses on optimisation, not generation	7: Generative models create diverse item representations for recommendations
9	Ensemble	5: Common in boosting, bagging in supervised	4: Limited interaction in unsupervised models	3: Rarely uses instance-based learning	3: Limited link; contrastive focuses on embeddings, not ensembling	5: Bayesian ensembles in boosting	7: Strong use in rule-based random forests	2: Weak connection; ensembles rarely focus on generating data	-	6: Some representation methods, e.g., stacking, use ensembles	6: Used in stacked architectures within deep learning	2: Minimal overlap, as RL does not rely on ensembling	6: Ensemble methods combine various recommendations
10	Representation	6: Common in extracting embeddings for supervised tasks	9: Strong foundation in unsupervised learning (e.g., PCA)	5: Moderate, embeddings assist instance-based models	4: Moderate use in contrastive embeddings	2: Minimal overlap, probabilistic models less representation-focused	1: Weak link; rules don't utilise representations	9: Embeddings used extensively in generative models	6: Some ensemble methods incorporate representation	-	9: Embeddings are core in deep learning	2: Minimal overlap with RL	8: Content-based recommenders rely on embeddings
11	Deep Learning	7: Common in supervised deep models (CNNs, RNNs)	6: Unsupervised pre-training for deep models	3: Deep models rarely rely on instance similarity	6: Contrastive learning enhances self-supervised deep tasks	6: Probabilistic deep models (e.g., Bayesian neural nets)	4: Minimal overlap with rule-based, deep models not rule-driven	9: GANs and VAEs are core to deep generative models	6: Stacked ensembles in deep architectures	9: Strong foundation as embeddings in representation learning	-	8: Deep RL integrates deep networks	7: Deep embeddings improve recommendation
12	Reinforcement	1: Rarely intersects with supervised paradigms	2: Minimal overlap with unsupervised learning	1: Minimal association; instance similarity not used in RL	1: Weak association with contrastive methods	6: Bayesian RL adds probabilistic decision-making	3: Weak link as RL is action-focused, not rule-focused	3: Minimal link; RL does not generate data but optimises rewards	2: Minor role, ensembling uncommon in RL	2: Minimal overlap with representation models	8: Deep RL heavily integrates deep learning	-	6: Some recommenders leverage RL for user interactions
13	Recommender	8: Recommender systems often rely on supervised models	7: Use of unsupervised clustering for user-item groups	5: k-NN models item-user similarity in recommend	6: Contrastive embeddings improve content-based recommendations	5: Bayesian networks improve uncertainty in recommendation systems	5: Rule-based methods assist in knowledge-based recommenders	7: Generative models create diverse item representations for recommendations	6: Ensemble methods combine various recommendations	8: Content-based recommenders rely on embeddings	7: Deep embeddings improve recommendation	6: Some recommenders leverage RL for user interactions	



Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

.

Durham University

Inspiring the extraordinary

Section 2 Multi-sourced Learning

Semi-Supervised Learning Paradigm

Concept:
Semi-supervised learning is a machine learning paradigm that combines a small amount of labelled data with a large amount of unlabelled data during training. This approach leverages the vast availability of unlabelled data, which is often easier to collect than labelled data, to improve model performance. Semi-supervised methods aim to exploit the structure in unlabelled data to guide the learning process, making them effective in scenarios where labelling is expensive or time-consuming.
In semi-supervised learning, models typically learn from both labelled examples and the patterns found in unlabelled data, creating a balance between supervised and unsupervised learning. Two common techniques are **Ladder Networks** and **Label Propagation**:

- **Ladder Networks** use a neural network architecture where the model combines supervised learning with unsupervised learning of latent representations, encouraging the network to learn useful features from both labelled and unlabelled data.
- **Label Propagation** is a graph-based technique that spreads label information from labelled nodes (data points) to unlabelled nodes based on the structure of the data, such as its similarity graph, helping unlabelled data points to inherit labels from their closest labelled neighbours.

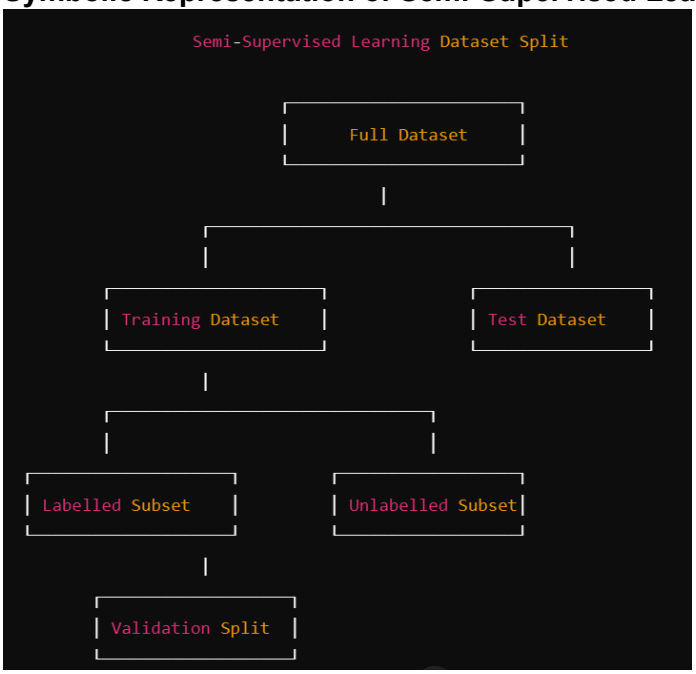
Contrast with Other Paradigms:
Semi-supervised learning contrasts with fully **supervised learning**, where only labelled data is used for training. In supervised learning, the reliance on labelled data makes it difficult to scale in cases where labelling is resource-intensive. Semi-supervised learning bridges this gap by incorporating unlabelled data to improve generalisation.
Compared to **unsupervised learning**, where no labels are available and the model's goal is to uncover hidden structures or patterns, semi-supervised learning can use labelled data as an anchor for better structure discovery.
Unlike **self-supervised learning**, which generates its own labels from the data using transformations, semi-supervised learning relies on existing human-labelled data as a guide while leveraging unlabelled data for refinement.

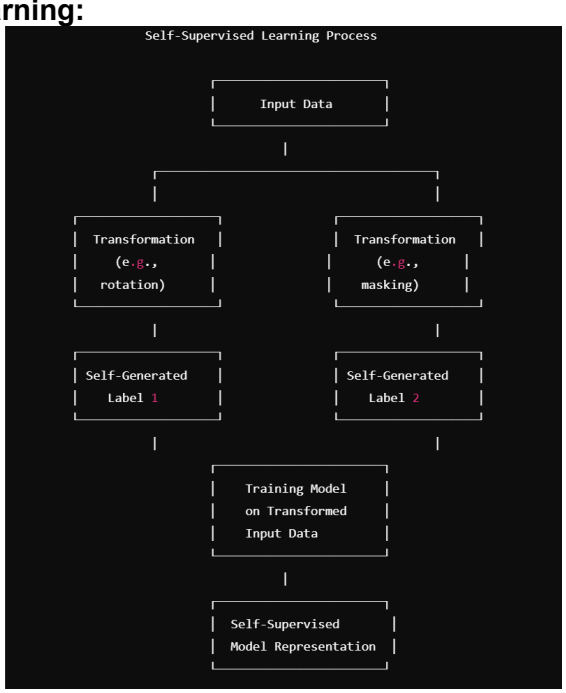
.

Durham University

Inspiring the extraordinary

Symbolic Representation of Semi-Supervised Learning:





Explanation:

- **Labelled Subset**: A small portion of the training data that has human-provided labels.
- **Unlabelled Subset**: The larger portion of the training data that is unlabelled, used to learn patterns.
- **Validation Split**: A labelled validation set used to tune hyperparameters and assess model performance during training.
- **Test Dataset**: A fully labelled dataset used for final evaluation, unseen during training.

.

Durham University

Inspiring the extraordinary

General Formula for Semi-Supervised Learning:

General Formula for Semi-Supervised Learning:

The loss function in semi-supervised learning typically combines both supervised loss L_s and unsupervised loss L_u components:

$$L = L_s(\text{labelled data}) + \lambda L_u(\text{unlabelled data})$$

where:

- L_s is the supervised loss, calculated using the labelled data (e.g., cross-entropy for classification tasks),
- L_u is the unsupervised loss, often based on consistency between the model's predictions for unlabelled data,
- λ is a hyperparameter controlling the balance between the supervised and unsupervised components.

For **Ladder Networks**, the total cost function is composed of supervised and unsupervised reconstruction losses, where the model tries to minimise the reconstruction error at each layer while learning from labelled data:

$$L = L_s + \sum_{i=1}^{n} \alpha_i L_u^{(i)}$$

where $L_u^{(i)}$ is the reconstruction loss at layer i, and α_i are the weights for each layer's unsupervised loss.

References and Historical Cases

Semi-supervised learning has its roots in both supervised and unsupervised techniques, with many methods developed to maximise the use of unlabelled data:

1. **Label Propagation**: Introduced by **Zhu and Ghahramani (2002)**, label propagation is a graph-based approach where the labels of a few nodes (labelled data) are propagated to nearby nodes in the graph based on similarity (Zhu & Ghahramani, 2002). This method has been effective in a range of applications, including natural language processing and image classification.
2. **Ladder Networks**: Developed by **Valpola and Rasmus et al. (2015)**, ladder networks combine supervised learning with unsupervised learning using denoising autoencoders to perform feature extraction from unlabelled data. The ladder architecture allows each layer of the network to learn unsupervised representations while simultaneously training on labelled data (Rasmus et al., 2015).

.

Inspiring the extraordinary

Semi-supervised learning has become critical in domains where labelling is expensive or impractical, such as medical imaging, speech recognition, and web content classification.

Discussion Questions:

1. How does semi-supervised learning balance between labelled and unlabelled data? What factors affect this balance?
2. What are the advantages of using Ladder Networks in semi-supervised learning compared to more traditional models like logistic regression?
3. How does label propagation leverage graph structure to improve the learning process, and what are the limitations in sparsely connected graphs?
4. In what types of real-world applications would semi-supervised learning provide the greatest benefit?

References

1. Zhu, X., & Ghahramani, Z. (2002). "Learning from Labeled and Unlabeled Data with Label Propagation." *CMU Technical Report CMU-CALD-02-107*.
2. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., & Raiko, T. (2015). "Semi-supervised learning with Ladder Networks." *Advances in Neural Information Processing Systems*, 27.

.

Durham University

Inspiring the extraordinary

Self-Supervised Learning Paradigm

Concept:

Self-supervised learning is a machine learning paradigm where models learn to predict parts of the data using other parts as labels. The key idea is to create supervised tasks from unlabelled data by generating pseudo-labels through transformations or data manipulations. These tasks guide the model in learning meaningful representations of the data without requiring human-labelled annotations.

Self-supervised learning is widely used in domains such as computer vision and natural language processing, where acquiring large amounts of labelled data is challenging. Two notable techniques in this paradigm are **Contrastive Predictive Coding (CPC)** and **SimCLR**.

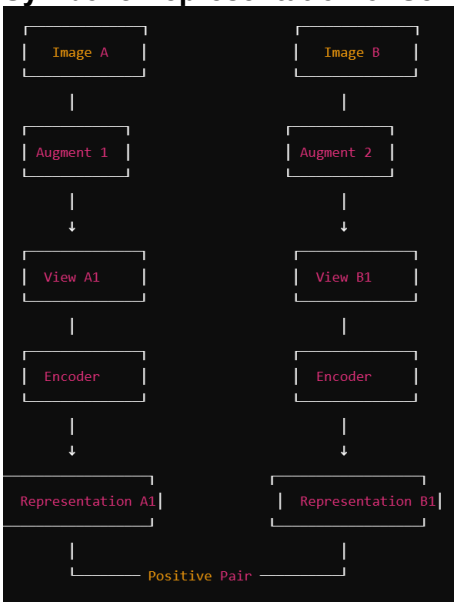
- **Contrastive Predictive Coding (CPC)**: This method leverages contrastive learning to predict future parts of a signal, such as predicting future audio or video frames, using learned representations from past data. The model contrasts correct future predictions against incorrect ones to learn useful features.
- **SimCLR** (Simple Framework for Contrastive Learning of Visual Representations): In SimCLR, the model learns by contrasting augmented views of the same image (positive pairs) against augmented views of different images (negative pairs). The contrastive loss encourages the model to bring positive pairs closer in representation space while pushing negative pairs farther apart.

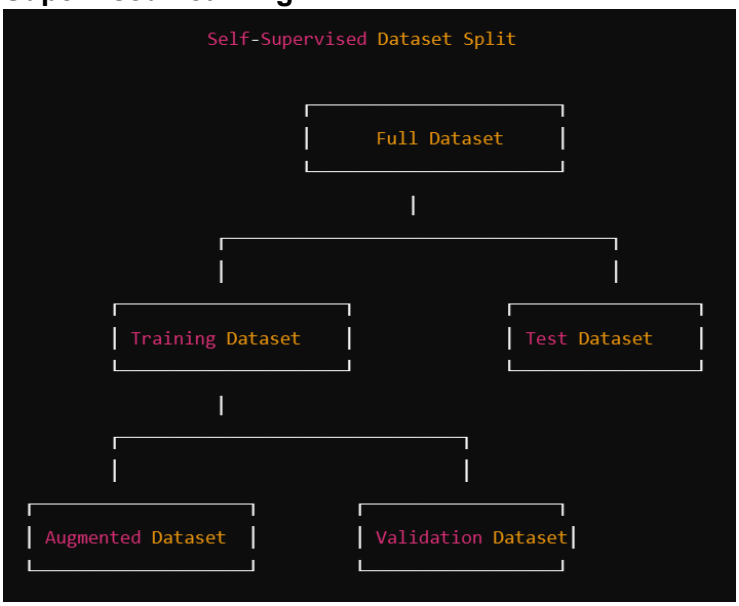
Contrast with Other Paradigms:

Self-supervised learning contrasts with both **supervised learning** and **unsupervised learning** by generating pseudo-labels from the data itself, rather than relying on human-provided labels or uncovering patterns without labels. In contrast to **semi-supervised learning**, which uses a combination of labelled and unlabelled data, self-supervised learning relies entirely on unlabelled data and creates tasks to guide the model.

Compared to **contrastive learning**, which is used as a method in self-supervised learning (e.g., CPC, SimCLR), self-supervised learning refers to the broader paradigm, with contrastive techniques being one way to implement it.

Symbolic Representation of Self-Supervised Learning:





.

Durham University

Inspiring the extraordinary

Explanation:

- **Image A** is augmented in two different ways to produce two views: **View A1** and **View B1**. These form a **positive pair** since they are two different views of the same image.
- The goal is to learn representations that pull positive pairs closer together in the embedding space, while pushing **negative pairs** (different images, such as A and B) farther apart.

General Formula for Self-Supervised Learning:

General Formula for Self-Supervised Learning:

Self-supervised learning with contrastive loss functions aims to maximise the similarity between positive pairs while minimising the similarity between negative pairs. The **contrastive loss** is often represented as:

$$L = -\log \frac{\exp(\text{sim}(h_i, h_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(h_i, h_k)/\tau)}$$

where:

- h_i and h_j are the embeddings of two augmented views of the same image (positive pair),
- $\text{sim}(h_i, h_j)$ represents the cosine similarity between the embeddings,
- τ is a temperature hyperparameter,
- The denominator sums over all other views (negative pairs), penalising the similarity between embeddings of different images.

References and Historical Cases

Self-supervised learning has become a central paradigm in learning representations from unlabelled data:

1. **Contrastive Predictive Coding (CPC)**: Introduced by **van den Oord et al. (2018)**, CPC learns to predict future representations of sequential data (e.g., audio or video frames) using contrastive learning. This approach has been successful in learning meaningful representations for downstream tasks (van den Oord et al., 2018).
2. **SimCLR**: Developed by **Chen et al. (2020)**, SimCLR demonstrated the power of contrastive learning in image representation by using data augmentations to create pairs for self-supervised learning. It achieved competitive performance on image classification tasks compared to supervised methods (Chen et al., 2020).

These methods have significantly advanced self-supervised learning, particularly in domains such as image recognition, speech processing, and video analysis.

Discussion Questions:

1. How do contrastive methods like SimCLR help models learn effective representations without explicit labels?

.

Durham University

Inspiring the extraordinary

2. What role do data augmentations play in self-supervised learning, and how might poor augmentations affect the performance?
3. How does Contrastive Predictive Coding (CPC) differ from SimCLR in its application and objectives?
4. In which domains might self-supervised learning be most effective compared to supervised or semi-supervised approaches?

References

1. van den Oord, A., Li, Y., & Vinyals, O. (2018). "Representation Learning with Contrastive Predictive Coding." *arXiv preprint arXiv:1807.03748*.
2. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). "A Simple Framework for Contrastive Learning of Visual Representations." *International Conference on Machine Learning (ICML)*.

.

Durham University

Inspiring the extraordinary

Transfer Learning Paradigm

Concept:
Transfer learning is a machine learning paradigm where a model trained on one task is adapted to perform well on a different, but related, task. Rather than starting the learning process from scratch, transfer learning leverages knowledge gained from one domain (source task) to improve learning in another domain (target task). This is particularly useful when labelled data in the target domain is scarce but abundant in the source domain. Transfer learning can be applied in various settings, including feature extraction, domain adaptation, and even model fine-tuning.
While **Fine-Tuning Pretrained Models** like BERT and ResNet has become a dominant approach in deep learning, transfer learning also has a strong presence in traditional machine learning. One such traditional approach is **dictionary learning**, which provides a robust mechanism for transferring learned representations across tasks.

- **Dictionary Learning**: In dictionary learning, the model learns a set of basis functions (or "dictionary atoms") that can sparsely represent data in one domain. These learned representations can be transferred to a different, but related, task or dataset by adapting the dictionary to fit new data, while retaining the original dictionary structure.

Other traditional approaches to transfer learning include **instance-based transfer** (where data from the source task is re-weighted to apply to the target task) and **feature-based transfer** (where features learned in one task are reused in another task).

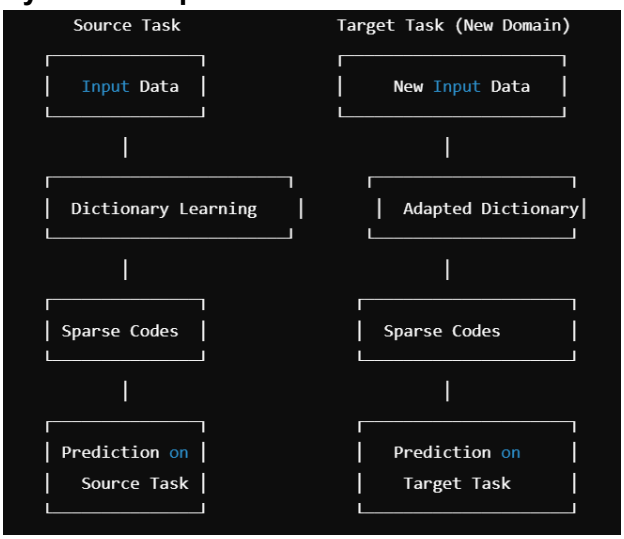
Contrast with Other Paradigms:
Transfer learning differs from paradigms like **supervised learning**, where the model is trained from scratch on a new task, without any prior knowledge. Instead, transfer learning attempts to transfer knowledge from an existing model, reducing the need for large amounts of labelled data in the target domain.
Compared to **unsupervised learning**, transfer learning typically involves labelled data from a source domain, though the target task might have limited or no labels. **Semi-supervised learning** uses unlabelled data within the same domain, whereas transfer learning works across domains, transferring knowledge between different but related tasks.
Transfer learning also contrasts with **representation learning**, where the focus is on learning compact, informative representations from the data itself. In transfer learning, these representations are already pre-learned in one domain and are adapted for use in another.

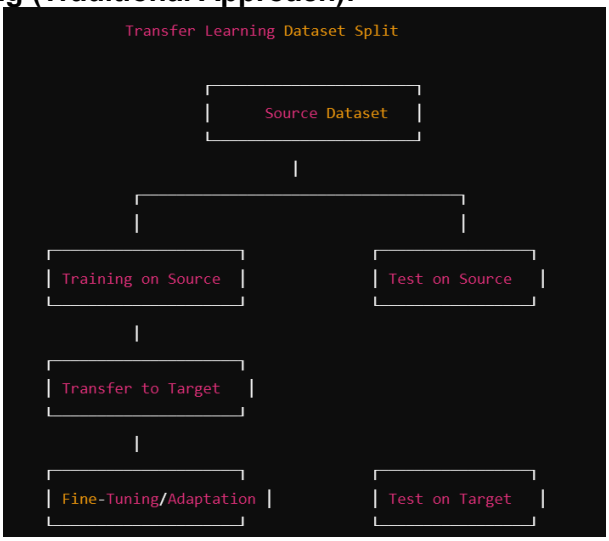
.

Durham University

Inspiring the extraordinary

Symbolic Representation of Transfer Learning (Traditional Approach):





Explanation:

- **Source Task**: The model learns a dictionary from the input data.
- **Dictionary Learning**: The learned dictionary represents the input data in a sparse form.
- **Target Task**: The dictionary is adapted to fit the new domain, and sparse representations are reused for predictions in the target domain.

General Formula for Transfer Learning (Dictionary Learning):

In dictionary learning, the objective is to learn a dictionary D and sparse codes Z such that the input data X can be approximated as $X \approx DZ$, where D is the dictionary, and Z represents the sparse codes. The learning problem can be formulated as:

$$\min_{D,Z} \|X - DZ\|_2^2 + \lambda \|Z\|_1$$

where:

- X is the input data from the source domain,
- D is the learned dictionary,
- Z is the sparse representation of X,
- λ is a regularisation parameter that controls the sparsity of Z.

When transferring the dictionary to a new task, D is adapted to fit the new domain, while retaining much of the structure learned from the source domain.

References and Historical Cases

Transfer learning has been explored in traditional machine learning for various tasks, long before its recent popularity in deep learning:

.

Durham University

Inspiring the extraordinary

1. **Dictionary Learning**: Dictionary learning has been applied to tasks like image denoising, inpainting, and classification. **Mairal et al. (2009)** developed algorithms for sparse dictionary learning that are widely used in image processing and machine learning tasks (Mairal et al., 2009).
2. **Instance-Based Transfer Learning**: One early method for transferring instances from a source task to a target task involves re-weighting source data points to make them more applicable to the target task. This method is commonly used in domain adaptation.
3. **Domain Adaptation:** Focuses on adapting a model trained in one domain (source) to perform well in a different but related domain (target), where the data distributions between the domains are different.
4. **Robustness Learning**: Aims to make models more resilient to changes in data distribution or adversarial examples, ensuring the model performs well even when exposed to new or noisy data.

Traditional transfer learning methods have found applications in areas such as computer vision, natural language processing, and even finance, where tasks often benefit from leveraging pre-existing models or representations.

Discussion Questions:

1. How does dictionary learning support the transfer of sparse representations between tasks?
2. What are the key challenges in applying transfer learning in traditional machine learning, compared to deep learning?
3. How does instance-based transfer learning differ from feature-based or model-based transfer learning, and when would each be useful?
4. In what scenarios would transfer learning provide the most benefit, especially in domains with limited labelled data?

References

1. Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009). "Online Learning for Matrix Factorization and Sparse Coding." *Journal of Machine Learning Research*, 11, 19-60.

.

Durham University

Inspiring the extraordinary

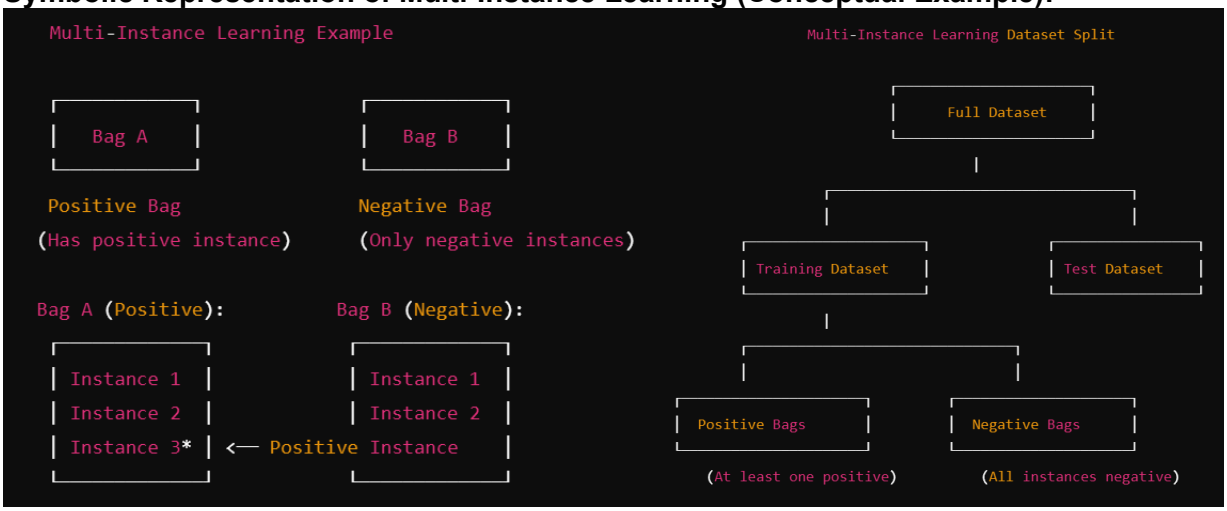
Multi-Instance Learning Paradigm

Concept:
Multi-instance learning (MIL) is a machine learning paradigm where the data is represented in **bags** of instances rather than individual instances. Each bag is labelled as positive or negative, but the labels of the individual instances within the bag are unknown. The main challenge is to classify bags correctly, knowing that a positive bag contains at least one positive instance, while a negative bag contains only negative instances.
Two widely used approaches in multi-instance learning are **Diverse Density** and **mi-SVM**:

- **Diverse Density (DD)**: A probabilistic approach that finds the concept shared by positive bags while being absent in negative bags. It attempts to identify a point in feature space where many positive instances from different positive bags are clustered together, but negative instances are far away.
- **mi-SVM (Multiple Instance Support Vector Machines)**: A generalisation of SVMs where the model seeks a hyperplane that classifies bags by optimising the margin between bags, considering the unknown labels of individual instances in the bag. The classifier learns by treating the most positive instance in each positive bag as crucial.

Contrast with Other Paradigms:
Multi-instance learning differs from **supervised learning**, where each instance has a label. In MIL, labels are assigned to bags, not instances, so the learner must figure out how the bag label is determined by the instances.
Compared to **instance-based learning**, where each data point is independent, in MIL, instances within the same bag are not independent, as their labels collectively determine the bag's label. MIL is also distinct from **semi-supervised learning**, as it doesn't rely on partially labelled data but rather on aggregate information from bags of instances.

Symbolic Representation of Multi-Instance Learning (Conceptual Example):



.

- **Bag A** is a positive bag because it contains at least one positive instance (**Instance 3**).
- **Bag B** is a negative bag because all instances in the bag are negative.
- The goal of multi-instance learning is to classify the entire bag based on the presence (or absence) of positive instances, without knowing which instances are positive in advance.

General Formula for Multi-Instance Learning:

In **Diverse Density** (DD), the objective is to find a point in feature space t that maximises the probability that instances from positive bags are close to t and instances from negative bags are far from t. The density is calculated as:

$$DD(t) = \prod_{i \in \text{Positive Bags}} \max_{x_{ij} \in B_i} P(t|x_{ij}) \prod_{i \in \text{Negative Bags}} \prod_{x_{ij} \in B_i} (1 - P(t|x_{ij}))$$

where:

- t is the point in feature space,
- x_{ij} is an instance in bag B_i,
- $P(t|x_{ij})$ is the probability that x_{ij} belongs to the concept defined by t.

For **mi-SVM**, the objective is to learn a hyperplane that maximises the margin between positive and negative bags. The key constraint is that at least one instance in each positive bag must be classified as positive, which is formalised as:

$$\min_{w} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i \max_{x_{ij} \in B_i} (w^T x_{ij}))$$

where:

- w is the weight vector,
- y_i is the label of bag B_i,
- x_{ij} is an instance in bag B_i.

References and Historical Cases

Multi-instance learning has evolved through both probabilistic and SVM-based approaches, with applications in tasks like drug activity prediction and image classification:

1. **Diverse Density (DD)**: Proposed by **Maron and Lozano-Perez (1998)**, DD was one of the earliest methods to formalise multi-instance learning probabilistically. It identified the point in feature space shared by positive bags (Maron & Lozano-Perez, 1998).

.

Durham University

Inspiring the extraordinary

2. **mi-SVM**: Developed by **Andrews, Tsochantaridis, and Hofmann (2003)**, mi-SVM extended the traditional SVM framework to multi-instance learning by enforcing that at least one instance in each positive bag must be classified as positive (Andrews et al., 2003).

These methods have been successfully applied in various domains, including medical diagnosis, object detection in images, and drug discovery.

Discussion Questions:

1. How does Diverse Density locate the concept common to positive bags, and how might this approach struggle with noisy or ambiguous instances?
2. How does mi-SVM balance between identifying key instances within bags and generalising across all instances?
3. In what domains would multi-instance learning provide significant advantages over traditional supervised learning, where labels for individual instances are unavailable?
4. How does the assumption that positive bags contain at least one positive instance affect the performance of multi-instance learning models?

References

1. Maron, O., & Lozano-Perez, T. (1998). "A framework for multiple-instance learning." *Advances in Neural Information Processing Systems*, 10.
2. Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). "Support vector machines for multiple-instance learning." *Advances in Neural Information Processing Systems*, 15.

.

Durham University

Inspiring the extraordinary

Multi-View Learning Paradigm

Concept:

Multi-view learning is a machine learning paradigm where data is represented from multiple views or perspectives, with each view providing different but complementary information about the data. The goal is to leverage all views to improve learning performance. This paradigm is particularly useful when the same set of instances can be described by different feature sets, such as text and image descriptions of the same objects or different sensor data capturing the same event.

Two popular techniques in multi-view learning are **Co-Training** and **Multiple Kernel Learning (MKL)**:

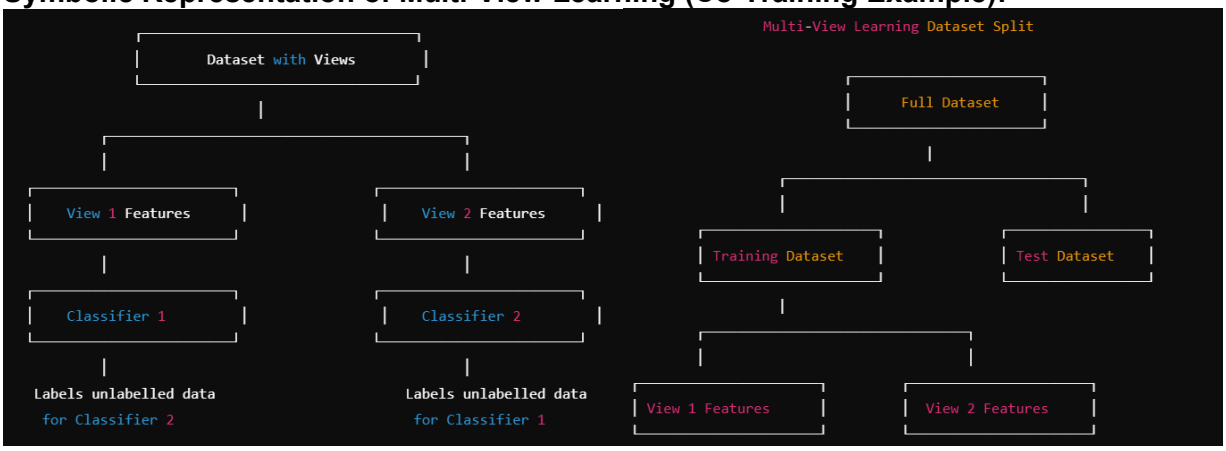
- **Co-Training**: A semi-supervised method where two or more classifiers are trained on different views of the data. These classifiers iteratively teach each other by labelling unlabelled data that one classifier is confident about, which the other classifier then uses to improve its learning.
- **Multiple Kernel Learning (MKL)**: This method combines kernels from different views of the data into a single, optimised kernel. The idea is to find an optimal combination of kernels (one per view) to improve classification or regression tasks by learning from multiple representations of the data.

Contrast with Other Paradigms:

Multi-view learning differs from **single-view supervised learning**, where all features are combined into a single view. In contrast, multi-view learning maintains the separation of views and leverages the complementary information from each.

Unlike **multi-instance learning**, which deals with multiple instances grouped into bags, multi-view learning focuses on different representations of the same instances. It also contrasts with **ensemble learning**, which aggregates multiple models on the same data, while multi-view learning uses different views of the data for training.

Symbolic Representation of Multi-View Learning (Co-Training Example):



- The dataset is split into two different views, each with its own set of features.
- Two classifiers are trained on these different views, and they iteratively label unlabelled data to help improve each other's performance.

General Formula for Multi-View Learning:

In multi-view learning, the goal is to leverage multiple views of the data, $V_1, V_2, \ldots, V_n$, to improve the learning process. Each view V_i corresponds to a different set of features or representations of the same instances. The fundamental principle is to combine these views in such a way that the model benefits from the complementary information provided by each view.

Here's a general framework that can be applied across different multi-view learning approaches:

Objective Function:

The general objective in multi-view learning is to minimize the total loss across all views, where each view provides a separate loss function L_i. These losses are combined to form the overall objective:

$$\min_{W_1, W_2, \ldots, W_n} \sum_{i=1}^{n} \alpha_i L_i(W_i; V_i)$$

where:

- W_i are the parameters (e.g., weights, kernels) learned from the i-th view,
- L_i is the loss function for the i-th view, such as cross-entropy or hinge loss,
- V_i represents the data from the i-th view,
- α_i are weights or coefficients controlling the contribution of each view to the final model.

The goal is to jointly learn from all views while minimising the combined loss.

Regularization Term:

In many multi-view learning methods, it's also crucial to enforce consistency between views, ensuring that the learned representations from different views agree with each other. This is often done through a regularization term that encourages similarity between the models learned from different views:

$$R(W_1, W_2, \ldots, W_n) = \sum_{i,j} \lambda_{ij} \| h(W_i; V_i) - h(W_j; V_j) \|^2$$

where:

- $h(W_i; V_i)$ represents the feature representation or output learned from view V_i,
- λ_{ij} controls the regularization strength between view i and view j,
- The regularization term encourages the representations from different views to be consistent or similar.

.

Durham University

Inspiring the extraordinary

References and Historical Cases

Multi-view learning has been widely applied in domains where data comes from multiple sources, such as bioinformatics, multimedia, and natural language processing:

1. **Co-Training**: First introduced by **Blum and Mitchell (1998)**, Co-Training is a semi-supervised learning approach that leverages multiple views of the data to improve classifier performance. It has been widely used in natural language processing and web page classification (Blum & Mitchell, 1998).
2. **Multiple Kernel Learning (MKL)**: **Lanckriet et al. (2004)** introduced the idea of combining multiple kernels for classification tasks, using optimization techniques to find the best combination of kernels from different views (Lanckriet et al., 2004). MKL has been successfully applied in areas such as genomics, image classification, and object recognition.

Discussion Questions:

1. How does Co-Training exploit the independence of views to improve the performance of semi-supervised learning?
2. What are the advantages of using Multiple Kernel Learning (MKL) compared to using a single kernel in traditional kernel methods?
3. In what situations would multi-view learning provide significant benefits, particularly when data from multiple modalities or perspectives is available?
4. What challenges arise in combining views when they provide contradictory or weakly correlated information?

References

1. Blum, A., & Mitchell, T. (1998). "Combining labeled and unlabeled data with Co-Training." *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 92-100.
2. Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). "Learning the kernel matrix with semidefinite programming." *Journal of Machine Learning Research*, 5, 27-72.

.

Durham University

Inspiring the extraordinary

Multi-Modal Learning Paradigm

Concept:

Multi-modal learning is a machine learning paradigm that integrates information from multiple modalities (e.g., text, images, audio, video) to create richer, more robust models. Each modality provides different perspectives on the same underlying data, allowing the model to capture more comprehensive patterns that wouldn't be possible with a single modality. For example, in a video analysis task, combining visual information (images) with audio (speech) improves the model's understanding of the scene.

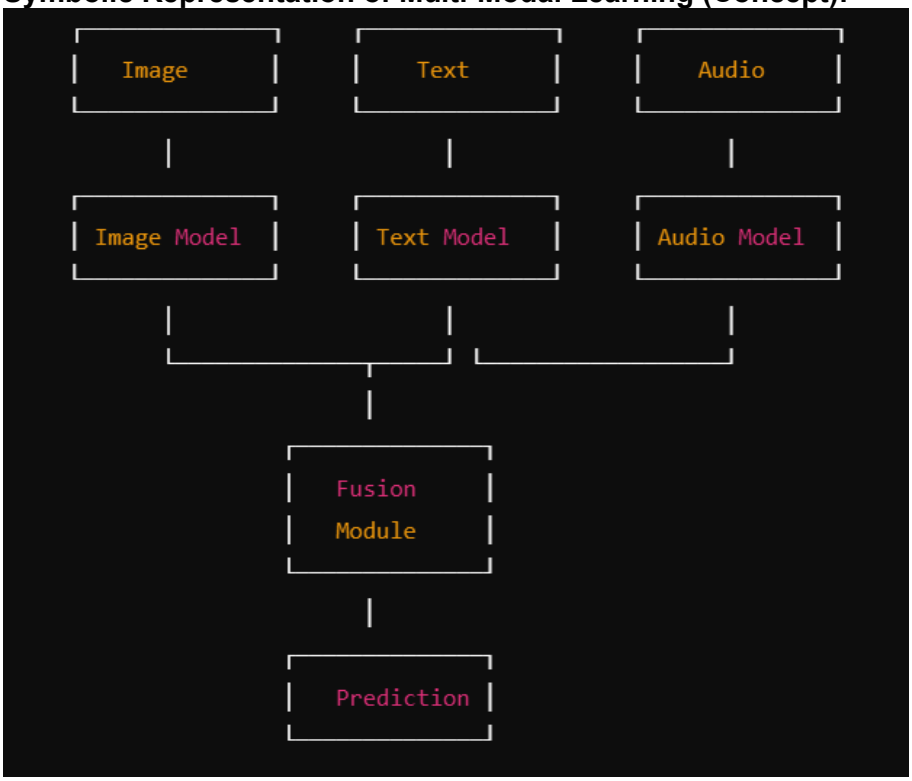
Multi-modal learning leverages the complementary nature of modalities, such as combining natural language and vision in tasks like image captioning or video understanding. The challenge is to effectively fuse the modalities and learn meaningful representations across them.

Contrast with Other Paradigms:

Multi-modal learning differs from **multi-view learning**, where multiple views represent different feature sets of the same data but often from the same modality (e.g., different sensors capturing the same physical event). In multi-modal learning, the modalities can be entirely different types of data (e.g., image and text).

Unlike **multi-task learning**, where a model learns several related tasks at the same time, multi-modal learning focuses on combining multiple types of input data for a single task. It also differs from **representation learning**, where the goal is often to discover compact feature representations of the data within a single modality.

Symbolic Representation of Multi-Modal Learning (Concept):



.

- **Image, Text, and Audio**: The raw inputs from different modalities.
- **Image Model, Text Model, Audio Model**: Each modality has its own dedicated model to extract features.
- **Fusion Module**: The outputs of each modality-specific model are combined (e.g., concatenation, attention-based fusion).
- **Prediction**: The fused representation is used to make the final prediction.

General Formula for Multi-Modal Learning:

In multi-modal learning, the objective is to learn a function $f(x_1, x_2, \ldots, x_n)$, where x_i are inputs from different modalities. These inputs are passed through modality-specific models $f_1, f_2, \ldots, f_n$, which are then fused into a single representation z:

$$z = \text{Fusion}(f_1(x_1), f_2(x_2), \ldots, f_n(x_n))$$

The final prediction $\hat{y}$ is made from the fused representation:

$$\hat{y} = g(z)$$

where:

- $f_i(x_i)$ is the feature representation learned from the i-th modality,
- Fusion is a function that combines the outputs of the modality-specific models (e.g., concatenation, attention),
- $g(z)$ is the prediction function (e.g., a softmax classifier for classification tasks).

For example, if the task is video understanding with visual and audio data, the two modalities (image and audio) are processed separately, and their representations are fused before making the prediction.

References and Historical Cases

1. **State-of-the-Art**:
   A state-of-the-art multi-modal model is **CLIP (Contrastive Language-Image Pretraining)**, introduced by **Radford et al. (2021)**. CLIP learns from natural language and images in a contrastive manner to perform various vision tasks without task-specific fine-tuning. By leveraging large-scale text-image pairs, CLIP demonstrates significant advances in zero-shot learning and multi-modal understanding, making it a powerful and flexible model for multi-modal tasks (Radford et al., 2021).
2. **Fundamental Work**:
   A fundamental work in multi-modal learning is **Canonical Correlation Analysis (CCA)**, a statistical method introduced by **Hotelling (1936)**. CCA finds linear relationships between two sets of variables (e.g., two modalities) by maximizing the correlation between projections

.

of the variables. It laid the groundwork for multi-modal learning by enabling the integration of multiple types of data in tasks such as speech and vision (Hotelling, 1936).

Discussion Questions:

1. How does multi-modal learning improve performance compared to single-modal learning, and what are the challenges of effectively fusing different modalities?
2. What are the key differences between early fusion (combining modalities before learning) and late fusion (combining after learning)?
3. How do state-of-the-art models like CLIP leverage contrastive learning to bridge the gap between different modalities, and how does this impact multi-modal tasks?
4. In what applications would multi-modal learning be most beneficial, and how could it improve tasks like medical diagnosis, autonomous driving, or multimedia retrieval?

References

1. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., & Krueger, G. (2021). "Learning Transferable Visual Models From Natural Language Supervision." *Proceedings of the 38th International Conference on Machine Learning (ICML)*.
2. Hotelling, H. (1936). "Relations Between Two Sets of Variates." *Biometrika*, 28(3/4), 321-377.

.

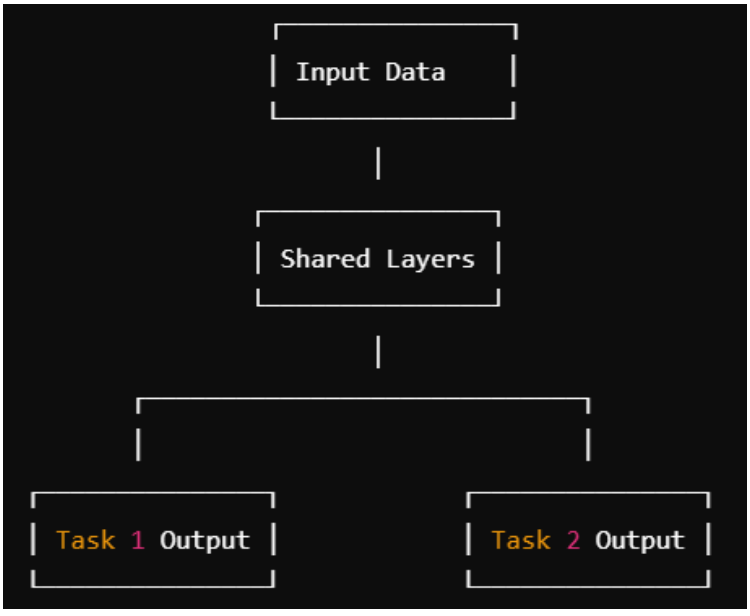
Durham University

Inspiring the extraordinary

Multi-Task Learning Paradigm
Concept:
Multi-task learning (MTL) is a machine learning paradigm where a model is trained to perform multiple related tasks simultaneously. Instead of training separate models for each task, MTL leverages shared representations and commonalities across tasks to improve generalisation and learning efficiency. The key idea is that learning one task can provide beneficial inductive bias for learning another, as the tasks share some underlying structure.
MTL is particularly effective when tasks are related or have overlapping features. For example, in computer vision, learning to detect objects in images might help a model also learn to segment those objects. In natural language processing, tasks such as part-of-speech tagging and named entity recognition often benefit from shared learning.

Contrast with Other Paradigms:
Multi-task learning differs from **single-task learning**, where a model is trained independently for each task. In MTL, tasks are learned jointly, and the model shares representations across them, which can improve learning efficiency and reduce overfitting, especially in low-data scenarios.
In contrast to **multi-modal learning**, which focuses on integrating different types of input data (e.g., images, text), multi-task learning deals with multiple objectives or outputs for the same data. MTL also differs from **transfer learning**, where knowledge from a source task is transferred to a target task sequentially, while MTL tackles all tasks concurrently.

Symbolic Representation of Multi-Task Learning (Concept):



Explanation:
- **Input Data**: The same data is used for multiple tasks.
- **Shared Layers**: The model uses shared layers to learn representations that are useful across all tasks.
- **Task-Specific Layers**: After the shared layers, each task has its own output head, which is specialised for its specific objective (e.g., classification for Task 1 and regression for Task 2).

.

Durham University

Inspiring the extraordinary

General Formula for Multi-Task Learning:

The objective function for multi-task learning combines the losses of each individual task. The total loss is typically a weighted sum of the losses for each task:

$$L_{\text{total}} = \sum_{i=1}^{n} \lambda_i L_i$$

where:

- L_i is the loss for the i-th task,
- λ_i is a weight that controls the importance of each task's loss,
- n is the total number of tasks.

The goal is to minimise the total loss across all tasks while optimising shared representations that benefit each task. The weights λ_i can be learned or set based on task importance.

For example, in a computer vision scenario where the model is jointly learning object detection and segmentation, the total loss might be a combination of the classification loss for detection and the pixel-wise loss for segmentation.

References and Historical Cases

1. **State-of-the-Art**:
   A state-of-the-art method in multi-task learning is **MT-DNN (Multi-Task Deep Neural Networks)**, introduced by **Liu et al. (2019)**. MT-DNN leverages shared representations across multiple natural language understanding tasks. By sharing parameters across tasks such as text classification, natural language inference, and sentiment analysis, MT-DNN achieved state-of-the-art results in various NLP benchmarks (Liu et al., 2019).
2. **Fundamental Work**:
   A fundamental work in multi-task learning is **Caruana's (1997)** paper, which introduced multi-task learning as a way to improve generalisation by using shared representations across tasks. Caruana's work demonstrated that multi-task learning reduces overfitting by allowing the model to learn commonalities between tasks, especially when training data for individual tasks is limited (Caruana, 1997).

Discussion Questions:

1. How does multi-task learning leverage shared representations across tasks to improve generalisation? What challenges arise in balancing the tasks?
2. How would task similarity affect the performance of a multi-task model? Would dissimilar tasks harm performance due to conflicting representations?
3. What are the benefits and drawbacks of jointly training tasks with different difficulty levels or data availability?

.

Durham University

Inspiring the extraordinary

4. How do state-of-the-art multi-task models like MT-DNN manage task-specific layers versus shared layers to ensure that task-specific information is preserved?

References

1. Liu, X., He, P., Chen, W., & Gao, J. (2019). "Multi-Task Deep Neural Networks for Natural Language Understanding." *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.
2. Caruana, R. (1997). "Multitask Learning." *Machine Learning*, 28(1), 41-75.

.

Durham University

Inspiring the extraordinary

Few-Shot Learning Paradigm

Concept:

Few-shot learning is a machine learning paradigm where models are trained to perform a task with only a few labelled examples. This is in stark contrast to traditional machine learning approaches, which typically require large amounts of labelled data for effective performance. Few-shot learning aims to generalise from a limited number of training instances by leveraging prior knowledge or learning strategies that allow the model to adapt quickly to new tasks with minimal data.

Two popular techniques in few-shot learning are **Prototypical Networks** and **Matching Networks**:

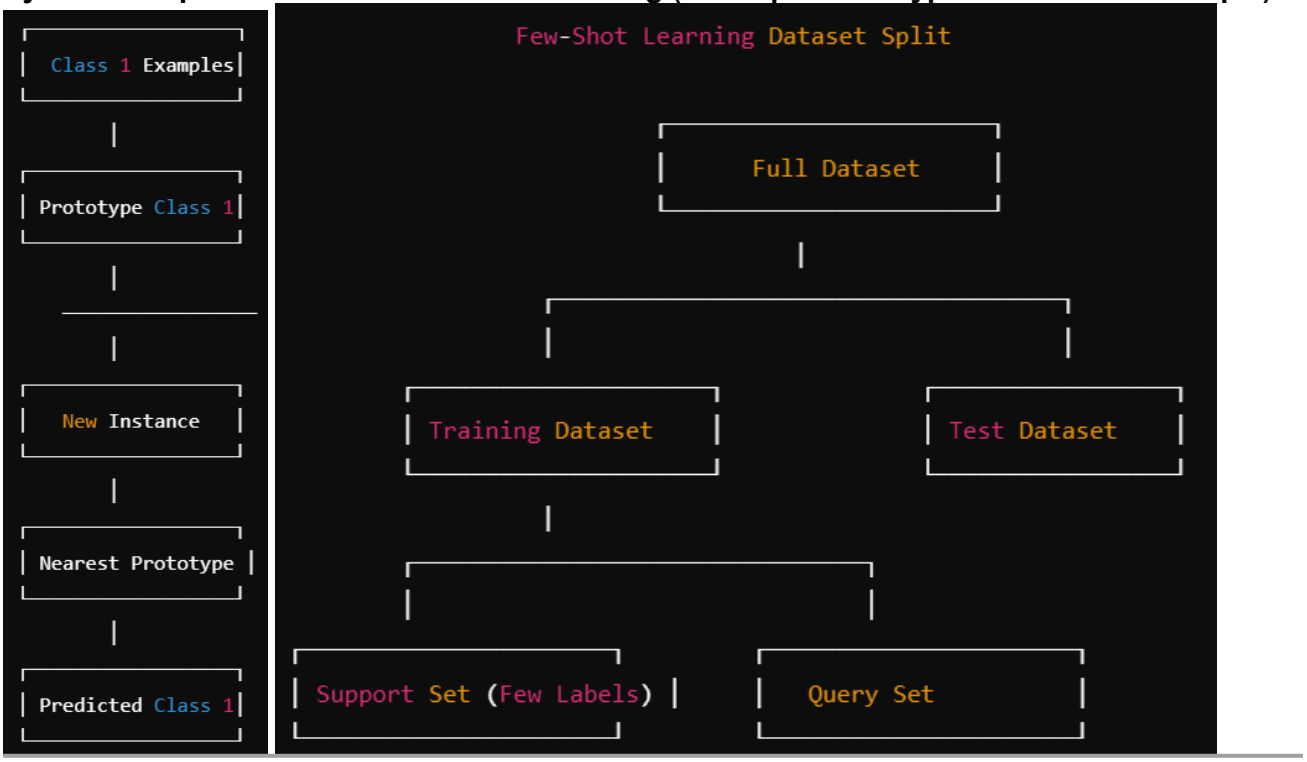
- **Prototypical Networks**: This method represents each class by a "prototype" – a central point in the embedding space – computed as the mean of the few-shot examples. Classification of new instances is done based on their proximity to these prototypes.
- **Matching Networks**: Matching networks use a memory-augmented neural network that computes the similarity between new examples and a small support set of labelled examples. The model then makes predictions based on these learned similarities.

Contrast with Other Paradigms:

Few-shot learning differs from traditional **supervised learning**, where large labelled datasets are required to achieve high accuracy. In few-shot learning, the focus is on developing models that can perform well with very few examples, typically by exploiting previously learned knowledge.

In comparison to **transfer learning**, where a model is first pre-trained on a large dataset and then fine-tuned on a new task, few-shot learning requires the model to adapt quickly without the need for large-scale pre-training. It is also distinct from **meta-learning**, though few-shot learning is often considered a subfield of meta-learning, as it focuses on learning how to adapt to new tasks with minimal data.

Symbolic Representation of Few-Shot Learning (Concept - Prototypical Networks Example):



.

Durham University

Inspiring the extraordinary

General Formula for Few-Shot Learning:

In **Prototypical Networks**, the goal is to compute a prototype p_c for each class c by averaging the embedded representations $f_\theta(x_i)$ of the few-shot examples x_i from class c:

$$p_c = \frac{1}{|S_c|} \sum_{x_i \in S_c} f_\theta(x_i)$$

where:

- S_c is the support set for class c (i.e., the set of few-shot examples for that class),
- $f_\theta(x_i)$ is the embedding of example x_i from class c,
- p_c is the prototype for class c.

For classification, the new instance x is assigned to the class whose prototype is closest in the embedding space, using a distance metric (e.g., Euclidean distance):

$$\hat{y} = \arg\min_c \|f_\theta(x) - p_c\|$$

In **Matching Networks**, the model computes the similarity between a query example and each example in the support set using a similarity function $g(x, x_i)$, and the prediction is made by a weighted combination of the support set labels:

$$\hat{y} = \sum_{x_i \in S} g(x, x_i) y_i$$

where:

- $g(x, x_i)$ is the similarity between the query x and support example x_i,
- y_i is the label of x_i,
- $\hat{y}$ is the predicted label for the query example.

References and Historical Cases

1. **State-of-the-Art**:
   Prototypical Networks, introduced by **Snell, Swersky, and Zemel (2017)**, are widely used in few-shot classification tasks. Prototypical Networks leverage simple and effective prototypes for each class and have been successful in both image classification and natural language processing tasks (Snell et al., 2017).

.

Durham University

Inspiring the extraordinary

2. **Fundamental Work**:
   Matching Networks, developed by **Vinyals et al. (2016)**, is a foundational approach to few-shot learning. Matching Networks introduced the idea of memory-augmented neural networks for few-shot classification, where the model directly compares new examples to a support set, making it highly effective for rapid adaptation to new tasks (Vinyals et al., 2016).

Discussion Questions:

1. How do Prototypical Networks and Matching Networks use the idea of learning from similarities in different ways?
2. What are the main challenges of few-shot learning, and how does the reliance on prior knowledge help models overcome the lack of data?
3. In what real-world applications might few-shot learning provide significant advantages, especially in situations where labelled data is scarce?
4. How does the embedding space representation in Prototypical Networks impact the model's ability to generalise to new tasks?

References

1. Snell, J., Swersky, K., & Zemel, R. S. (2017). "Prototypical Networks for Few-Shot Learning." *Advances in Neural Information Processing Systems (NeurIPS)*.
2. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). "Matching Networks for One Shot Learning." *Advances in Neural Information Processing Systems (NeurIPS)*.

.

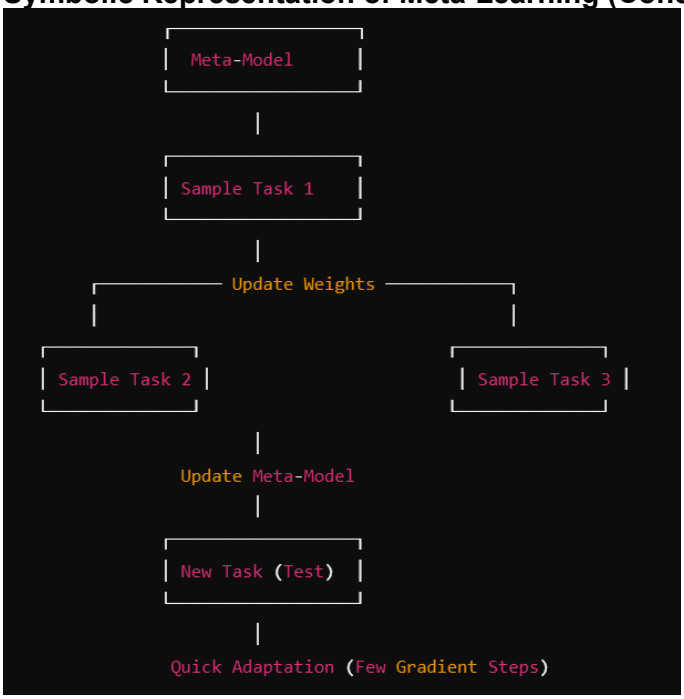
Inspiring the extraordinary

Meta-Learning Paradigm
Concept:
Meta-learning, often referred to as "learning to learn," is a machine learning paradigm where models are trained to quickly adapt to new tasks with minimal data. Instead of focusing solely on performing a specific task, meta-learning teaches models how to learn efficiently from new data, allowing them to generalise across various tasks. The meta-learner typically operates on a distribution of tasks rather than a single task, enabling rapid adaptation when faced with novel scenarios.
Two prominent methods in meta-learning are **Model-Agnostic Meta-Learning (MAML)** and **Reptile**:

- **MAML**: This approach aims to find an initial set of model parameters that can be rapidly fine-tuned to new tasks using just a few gradient steps and a small amount of task-specific data. It is model-agnostic, meaning it can be applied to any learning algorithm that uses gradient descent.
- **Reptile**: Similar to MAML, Reptile also optimises for fast adaptation but with a simpler approach. Instead of requiring second-order gradients like MAML, Reptile averages the model parameters after performing multiple task-specific updates, leading to a computationally more efficient meta-learning process.

Contrast with Other Paradigms:
Meta-learning differs from **supervised learning**, where a model is trained to solve a single task using a large amount of labelled data. In meta-learning, the goal is to generalise across multiple tasks, even when data is limited for each individual task. It is also distinct from **few-shot learning**, although related, because few-shot learning focuses on performing a task with limited examples, while meta-learning equips the model with the ability to rapidly learn new tasks in general.
Compared to **transfer learning**, which adapts a pre-trained model to a new task, meta-learning focuses on training the model itself to be adaptable across tasks, learning how to learn efficiently rather than simply transferring pre-trained knowledge.

Symbolic Representation of Meta-Learning (Concept - MAML Example):



.

- **Meta-Model**: The initial model that learns from multiple tasks.
- **Sample Tasks**: During training, the meta-model is updated based on different tasks, with the goal of learning how to adapt quickly to new ones.
- **New Task**: When a new, unseen task is presented, the meta-model adapts to it using only a few gradient steps, leveraging the knowledge it has gained from previous tasks.

General Formula for Meta-Learning:

In **MAML**, the objective is to find model parameters θ that can be quickly adapted to new tasks using a few gradient updates. Given a distribution of tasks $p(\mathcal{T})$, MAML optimises for the initial parameters by minimising the loss across tasks after adaptation.

1. **Inner Update**: For each task $\mathcal{T}_i$, the model adapts the parameters θ by performing one or more gradient updates on the task-specific loss $L_{\mathcal{T}_i}$:

$$\theta_i' = \theta - \alpha \nabla_\theta L_{\mathcal{T}_i}(\theta)$$

where α is the inner learning rate.

2. **Meta-Update**: The meta-objective is to minimise the task loss after the inner update. The overall objective is to find the initial parameters θ that lead to low task-specific loss after adaptation:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} L_{\mathcal{T}_i}(\theta_i')$$

This requires second-order gradients, as the objective depends on the adapted parameters θ_i'.

In **Reptile**, the meta-update is simplified by avoiding the second-order derivatives. Reptile updates the meta-parameters by taking the difference between the initial and final task-specific parameters:

$$\theta \leftarrow \theta + \epsilon(\theta_i' - \theta)$$

where ϵ is the meta-learning rate. This approach simplifies the computation while still facilitating fast adaptation to new tasks.

References and Historical Cases

1. **State-of-the-Art**:
   Model-Agnostic Meta-Learning (MAML), introduced by **Finn, Abbeel, and Levine (2017)**, is one of the most widely used meta-learning methods. It has been applied to a variety of tasks, including few-shot classification, reinforcement learning, and robotics. MAML's ability to quickly adapt to new tasks makes it a powerful tool in environments where new tasks are continually encountered (Finn et al., 2017).
2. **Fundamental Work**:
   Reptile, developed by **Nichol et al. (2018)**, is a simpler alternative to MAML. It avoids the

.

complexity of second-order gradients while achieving similar results. Reptile has been applied to reinforcement learning and classification tasks, making it a fundamental contribution to meta-learning methods (Nichol et al., 2018).

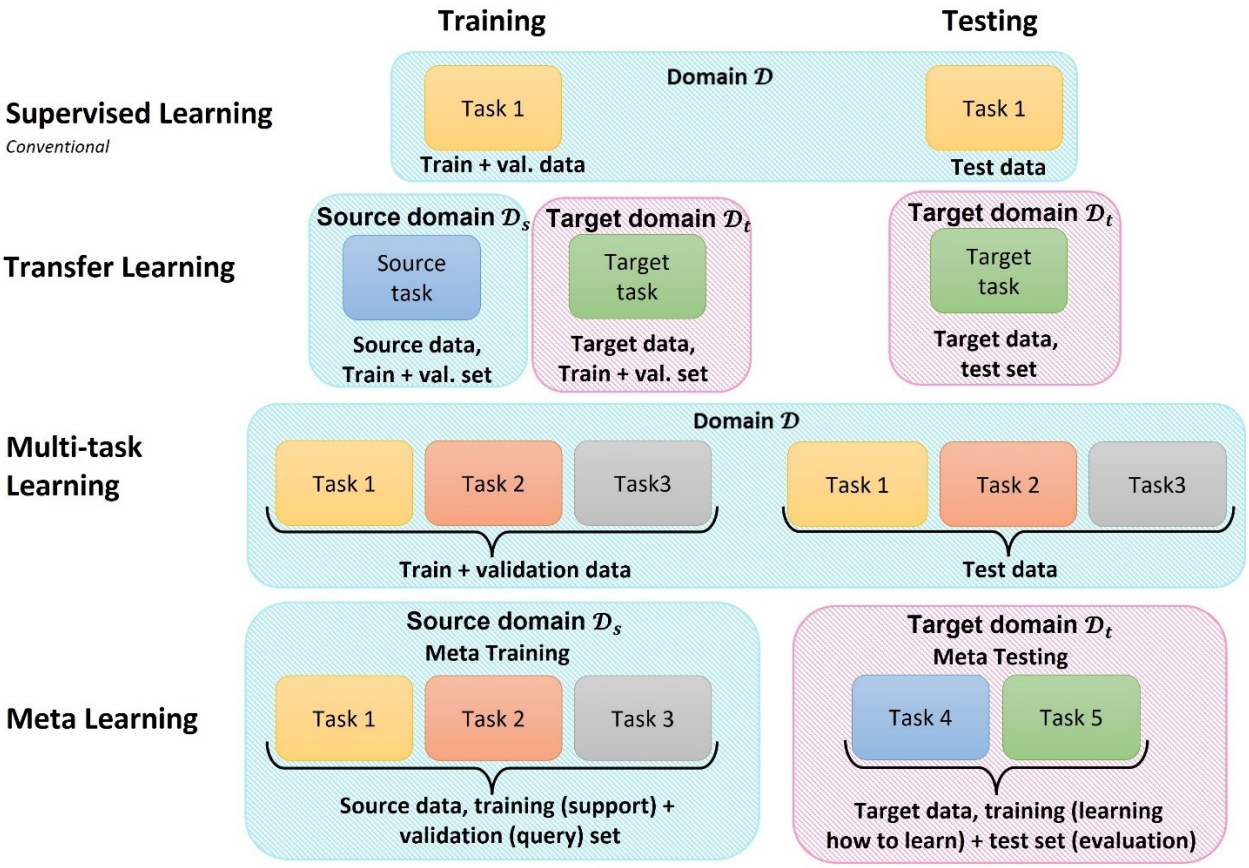
Discussion Questions:

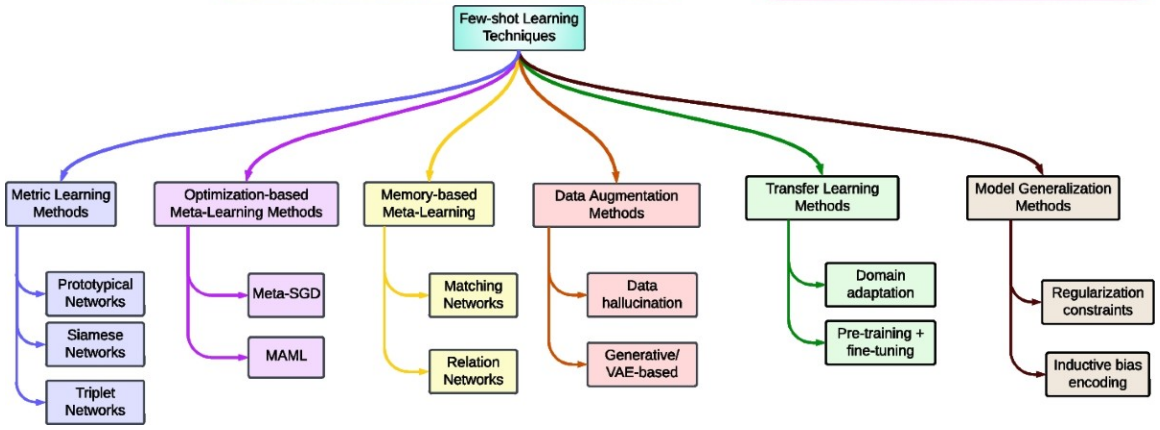
1. How does MAML facilitate fast adaptation to new tasks, and what are the trade-offs of using second-order gradients in the meta-update?
2. What are the key differences between MAML and Reptile in terms of computational efficiency and learning performance?
3. How does the concept of meta-learning extend to real-world applications, such as robotics, where new tasks frequently arise?
4. In what ways can meta-learning be combined with reinforcement learning to improve the generalisation of agents in dynamic environments?

References

1. Finn, C., Abbeel, P., & Levine, S. (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." *Proceedings of the 34th International Conference on Machine Learning (ICML)*.
2. Nichol, A., Achiam, J., & Schulman, J. (2018). "On First-Order Meta-Learning Algorithms." *arXiv preprint arXiv:1803.02999*.

1.

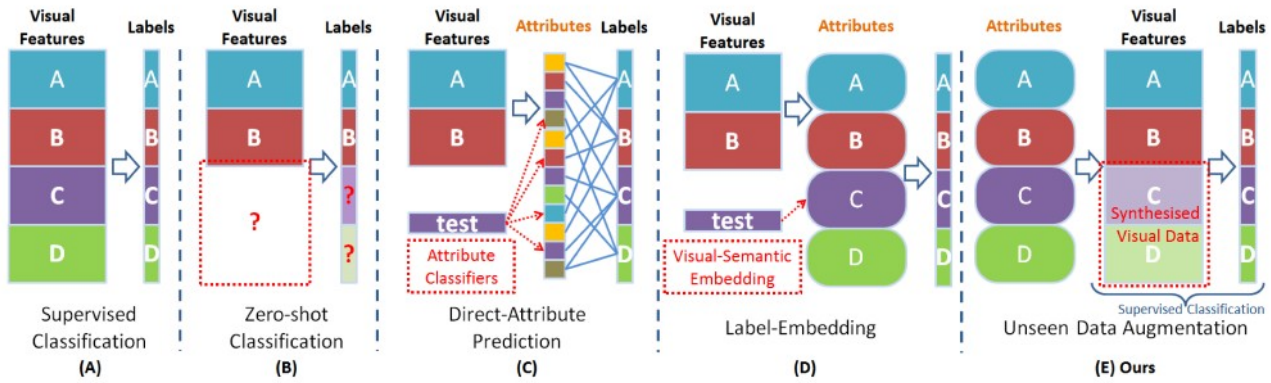

Training
Testing
Supervised Learning
Conventional
Domain $\mathcal{D}$
Task 1
Train + val. data
Task 1
Test data
Transfer Learning
Source domain $\mathcal{D}_s$
Source task
Source data, Train + val. set
Target domain $\mathcal{D}_t$
Target task
Target data, Train + val. set
Target domain $\mathcal{D}_t$
Target task
Target data, test set
Multi-task Learning
Domain $\mathcal{D}$
Task 1
Task 2
Task3
Train + validation data
Task 1
Task 2
Task3
Test data
Meta Learning
Source domain $\mathcal{D}_s$
Meta Training
Task 1
Task 2
Task 3
Source data, training (support) + validation (query) set
Target domain $\mathcal{D}_t$
Meta Testing
Task 4
Task 5
Target data, training (learning how to learn) + test set (evaluation)

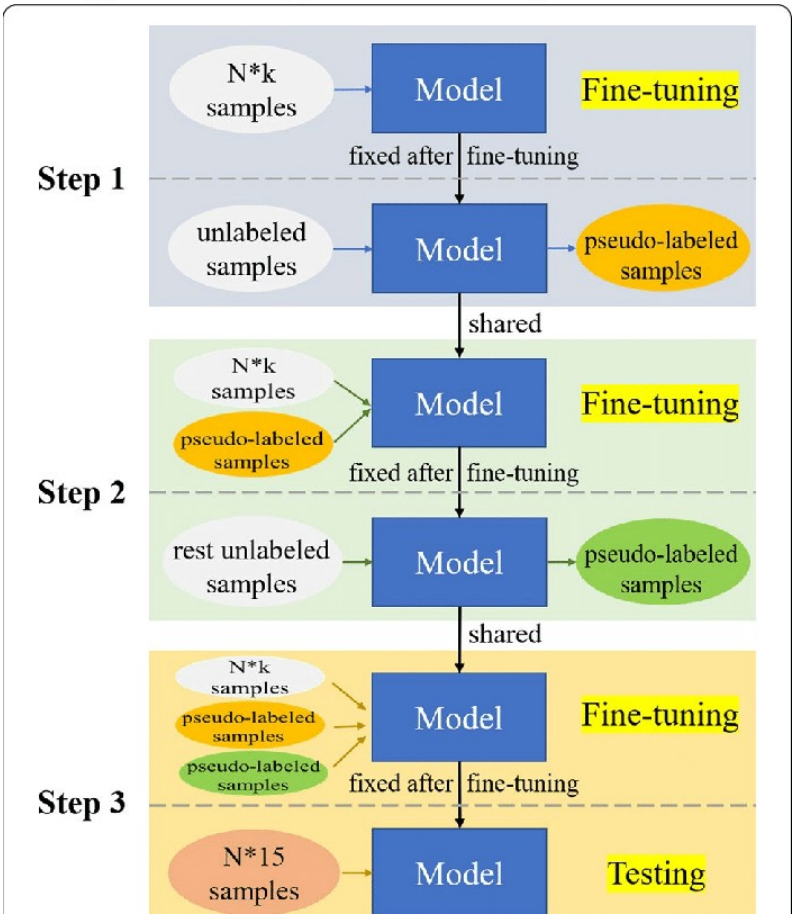

Few-shot Learning Techniques
Metric Learning Methods
Prototypical Networks
Siamese Networks
Triplet Networks
Optimization-based Meta-Learning Methods
Meta-SGD
MAML
Memory-based Meta-Learning
Matching Networks
Relation Networks
Data Augmentation Methods
Data hallucination
Generative/ VAE-based
Transfer Learning Methods
Domain adaptation
Pre-training + fine-tuning
Model Generalization Methods
Regularization constraints
Inductive bias encoding

.

Durham University

Inspiring the extraordinary





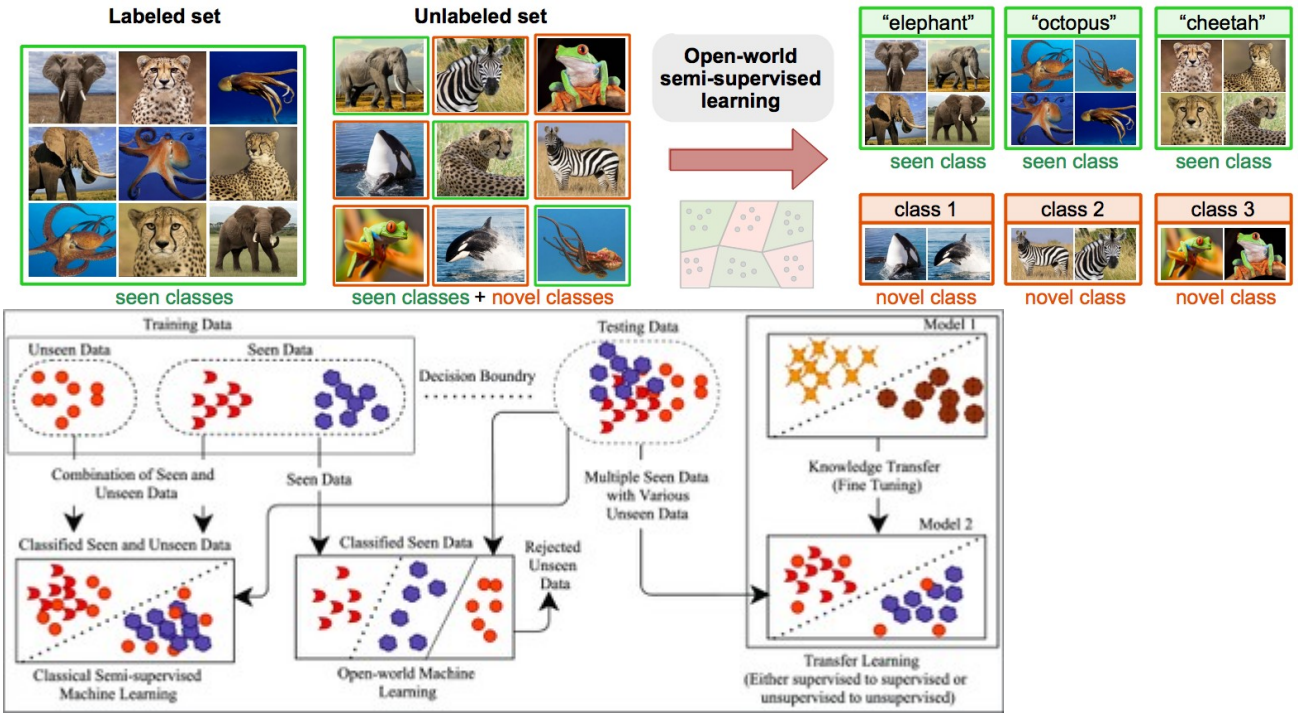
Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

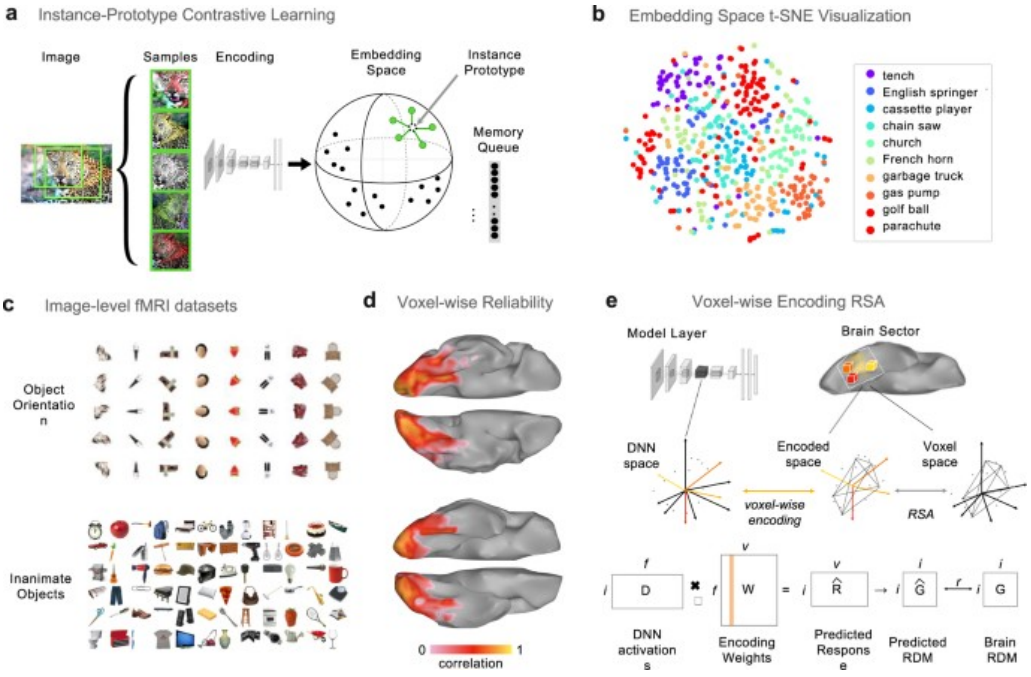
.

Durham University

Inspiring the extraordinary

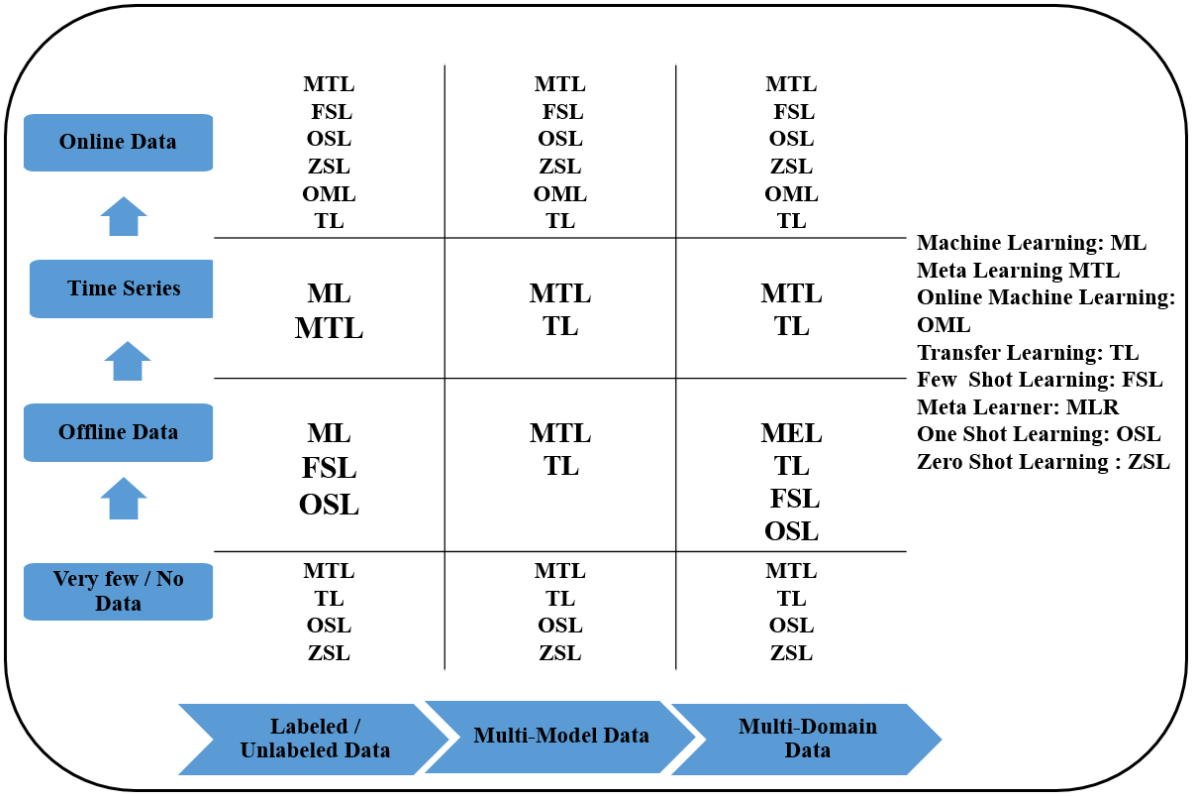


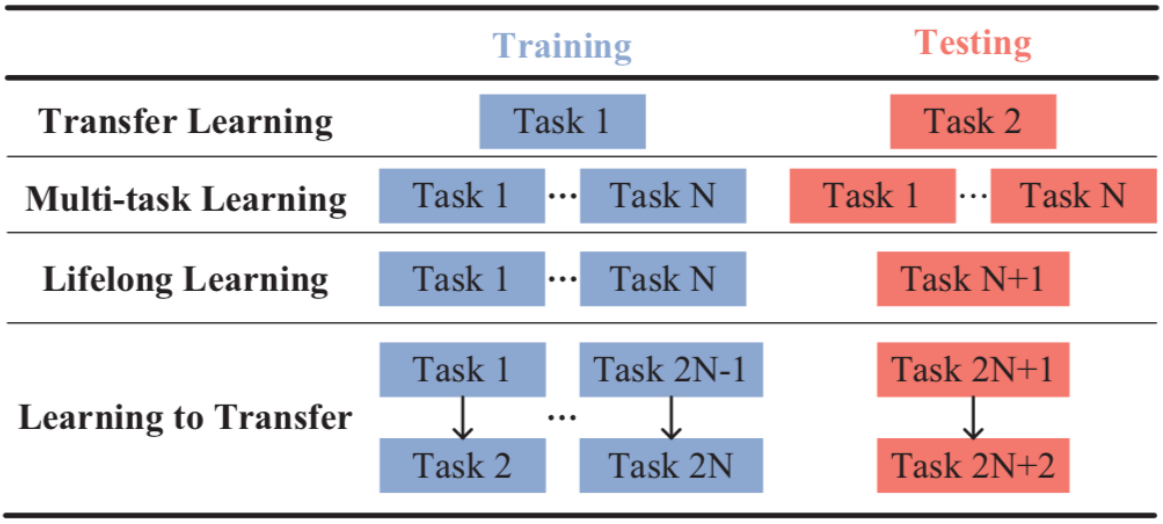
[Open-world Machine Learning: Applications, Challenges, and Opportunities]



Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

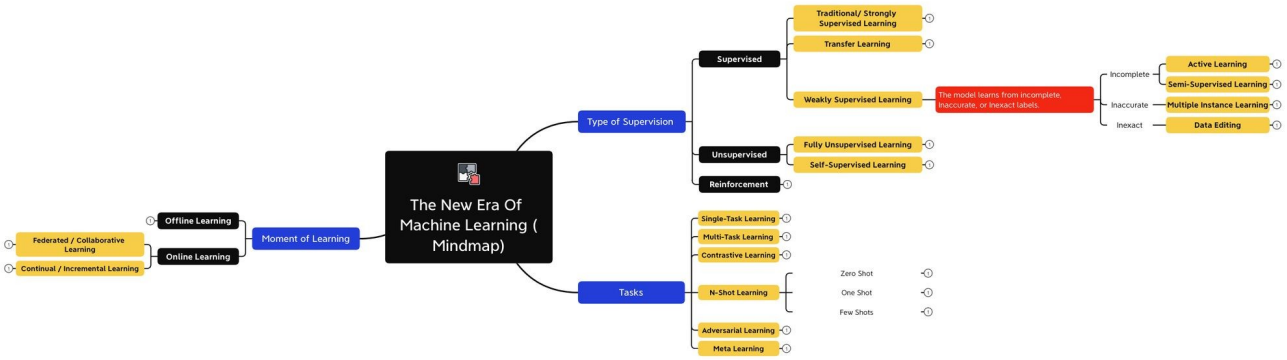
Telephone +44 (0)191 334 2000
www.durham.ac.uk

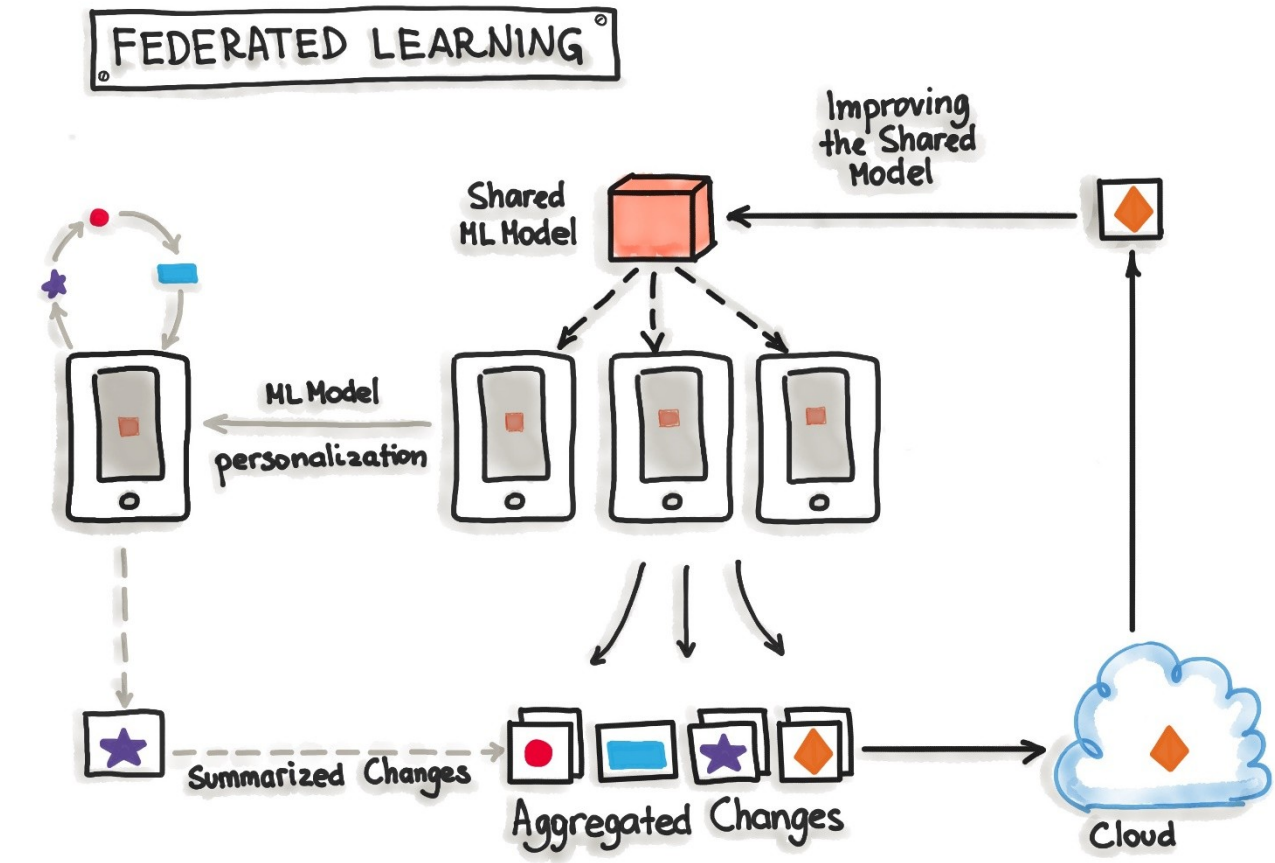

Online Data
Time Series
Offline Data
Very few / No Data
MTL FSL OSL ZSL OML TL
MTL FSL OSL ZSL OML TL
MTL FSL OSL ZSL OML TL
ML MTL
MTL TL
MTL TL
ML FSL OSL
MTL TL
MEL TL FSL OSL
MTL TL OSL ZSL
MTL TL OSL ZSL
MTL TL OSL ZSL
Machine Learning: ML
Meta Learning MTL
Online Machine Learning: OML
Transfer Learning: TL
Few Shot Learning: FSL
Meta Learner: MLR
One Shot Learning: OSL
Zero Shot Learning : ZSL
Labeled / Unlabeled Data
Multi-Model Data
Multi-Domain Data


Training
Testing
Transfer Learning
Task 1
Task 2
Multi-task Learning
Task 1 ··· Task N
Task 1 ··· Task N
Lifelong Learning
Task 1 ··· Task N
Task N+1
Learning to Transfer
Task 1
Task 2N-1
Task 2N+1
Task 2
Task 2N
Task 2N+2

Durham University

Inspiring the extraordinary





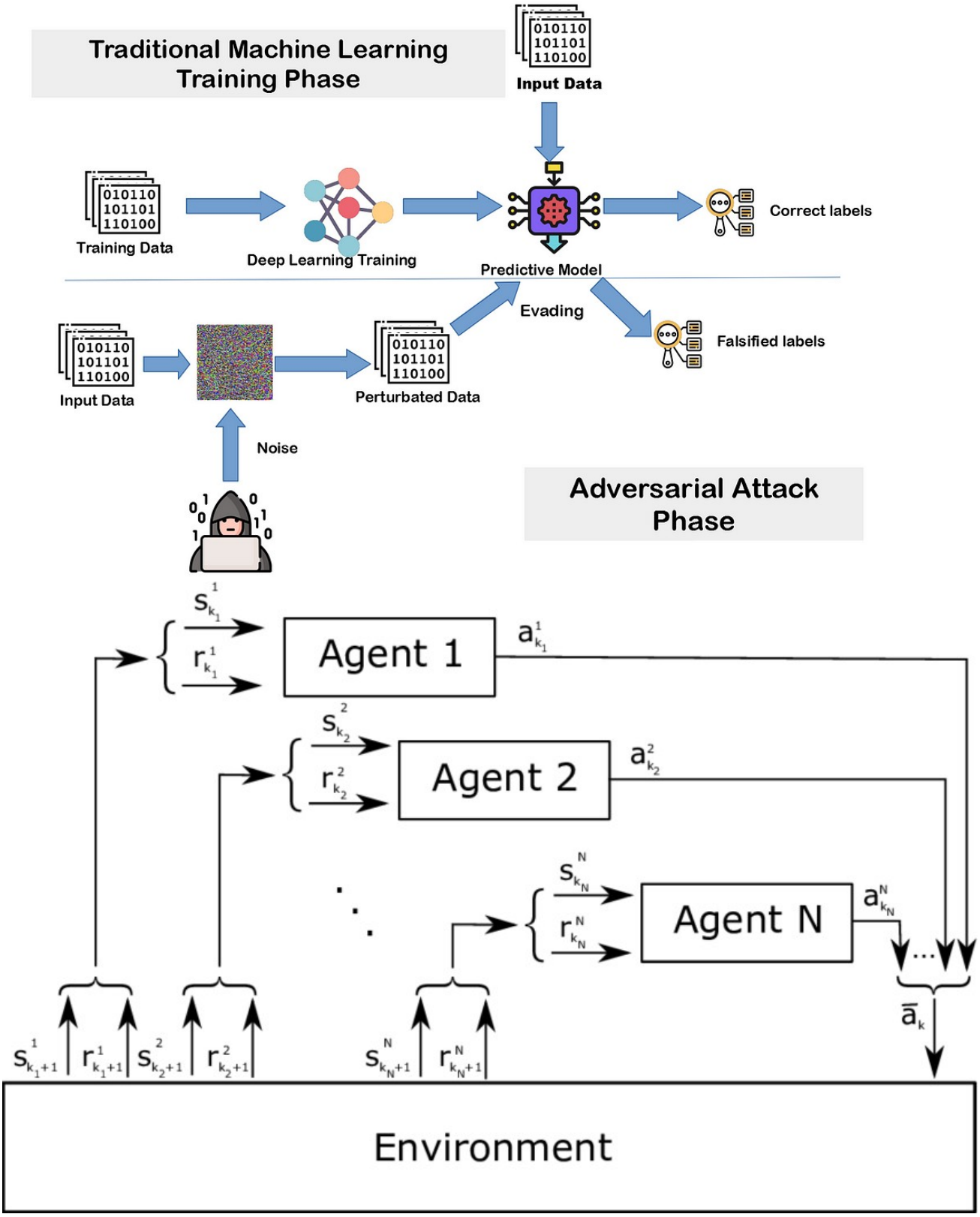
Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

.

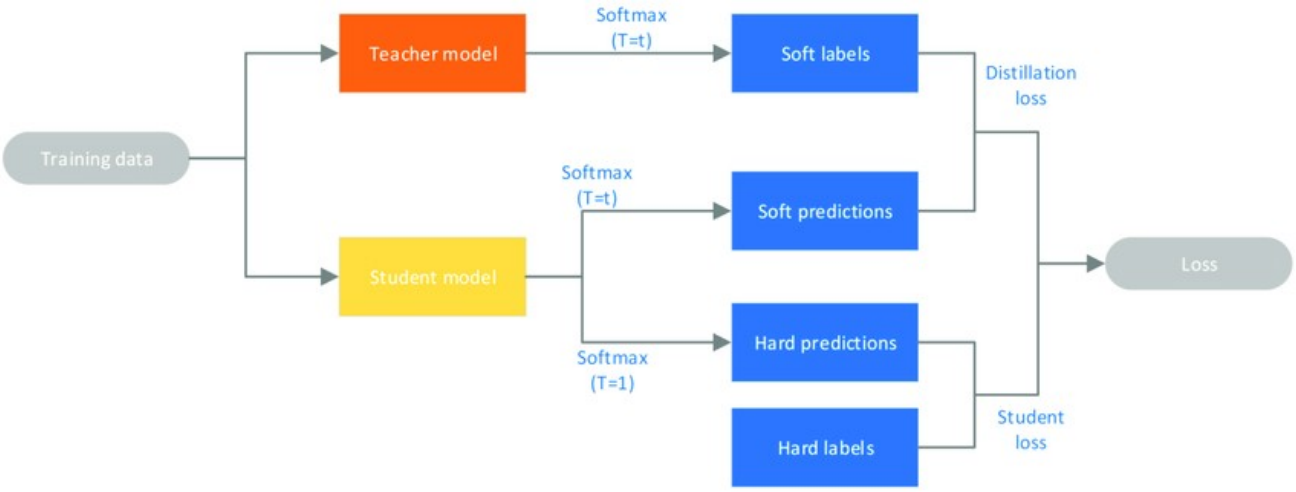
Durham University

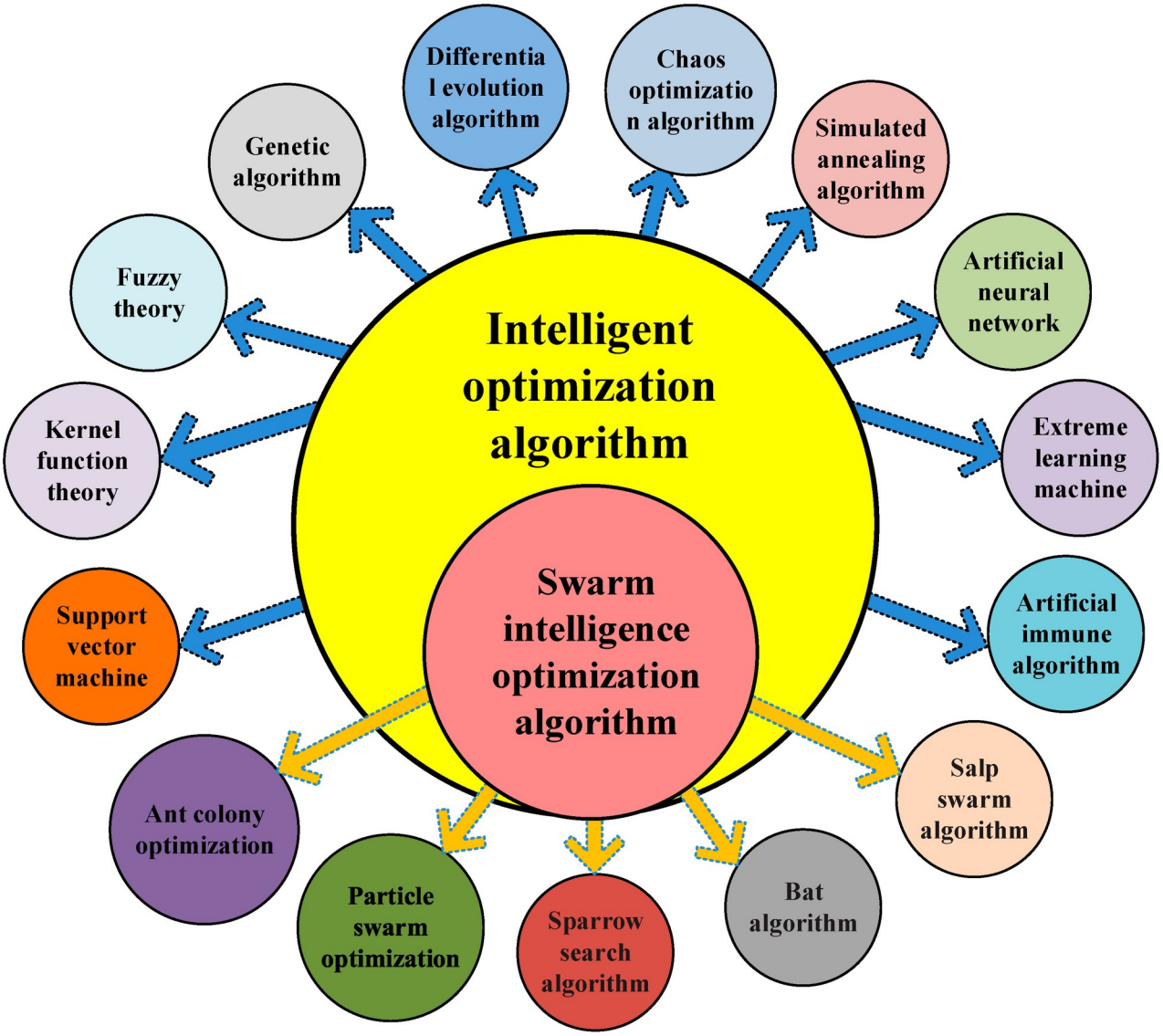
Inspiring the extraordinary



Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk



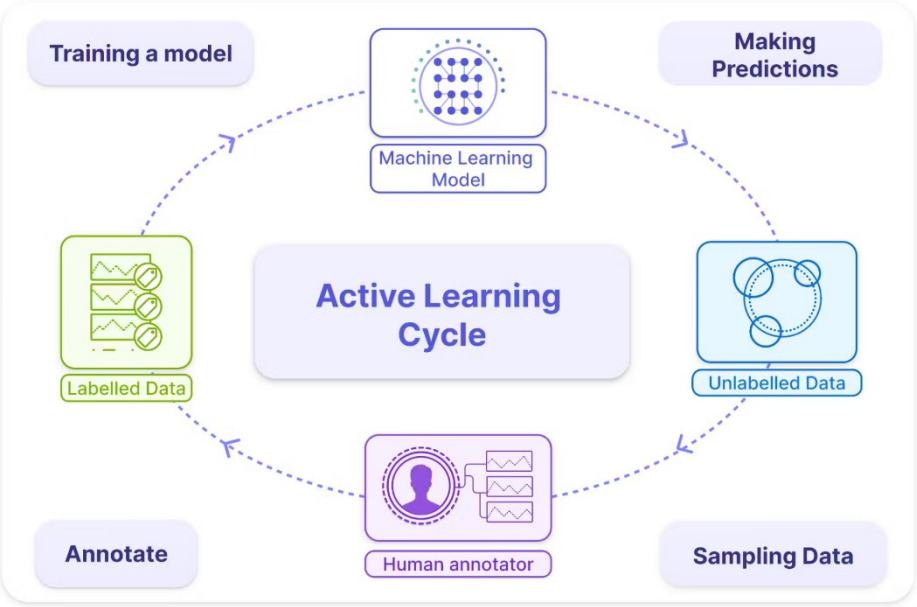


Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

Durham University

Inspiring the extraordinary



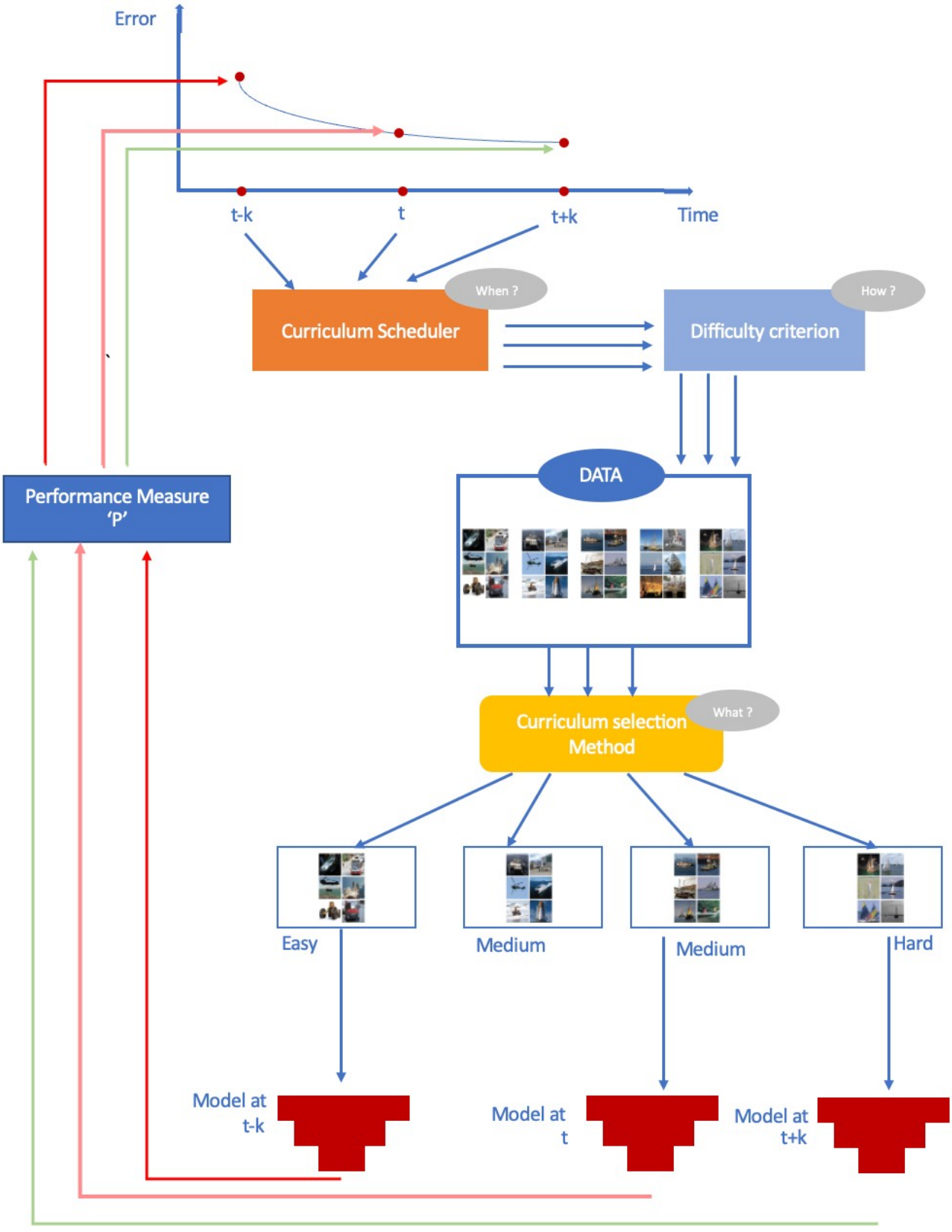
ENCORD

Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

Durham University

Inspiring the extraordinary



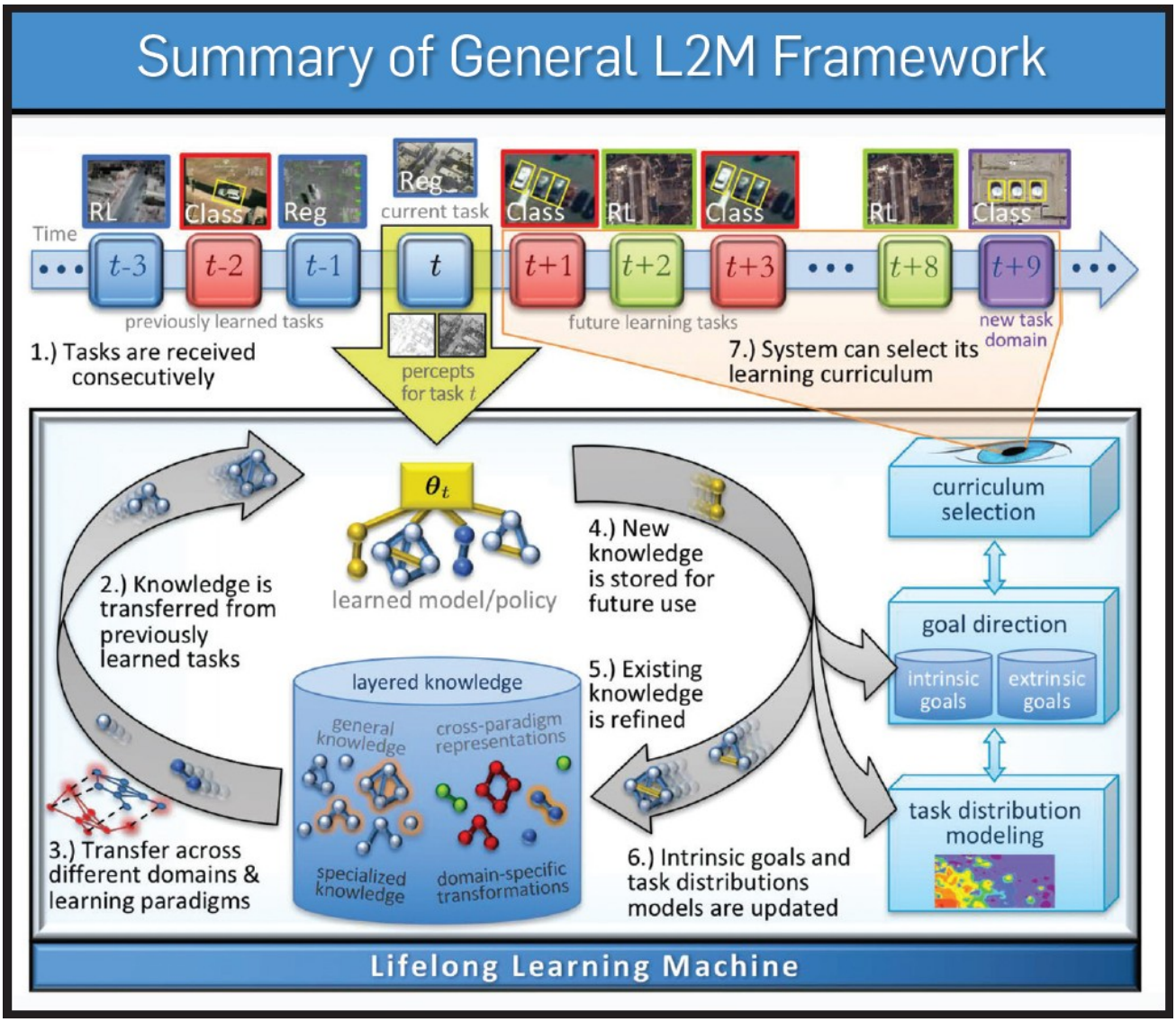
Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

Telephone +44 (0)191 334 2000
www.durham.ac.uk

.

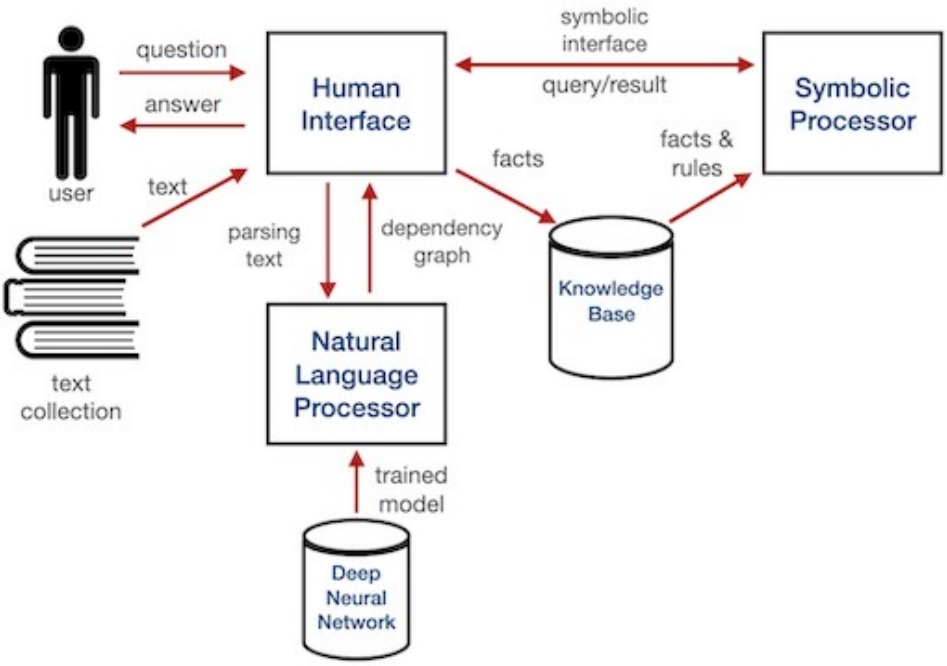
Durham University

Inspiring the extraordinary



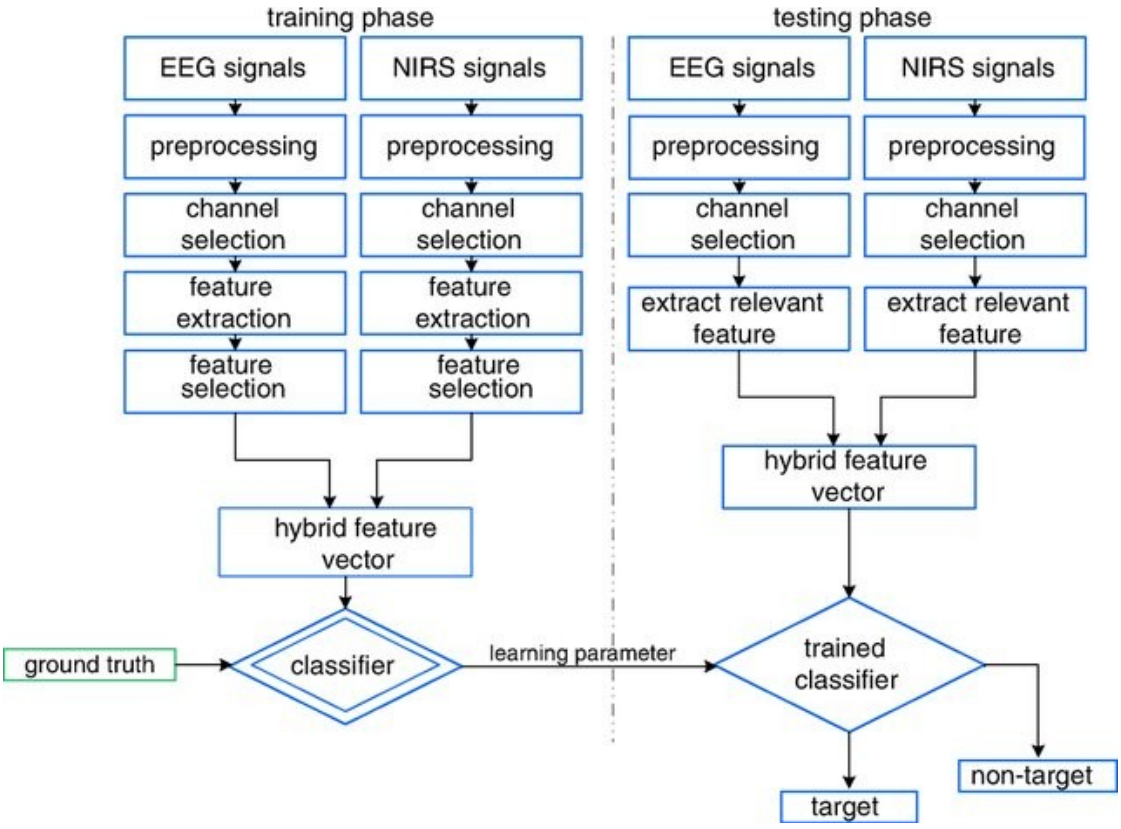
Durham University Palatine Centre Stockton Road Durham DH1 3LE UK

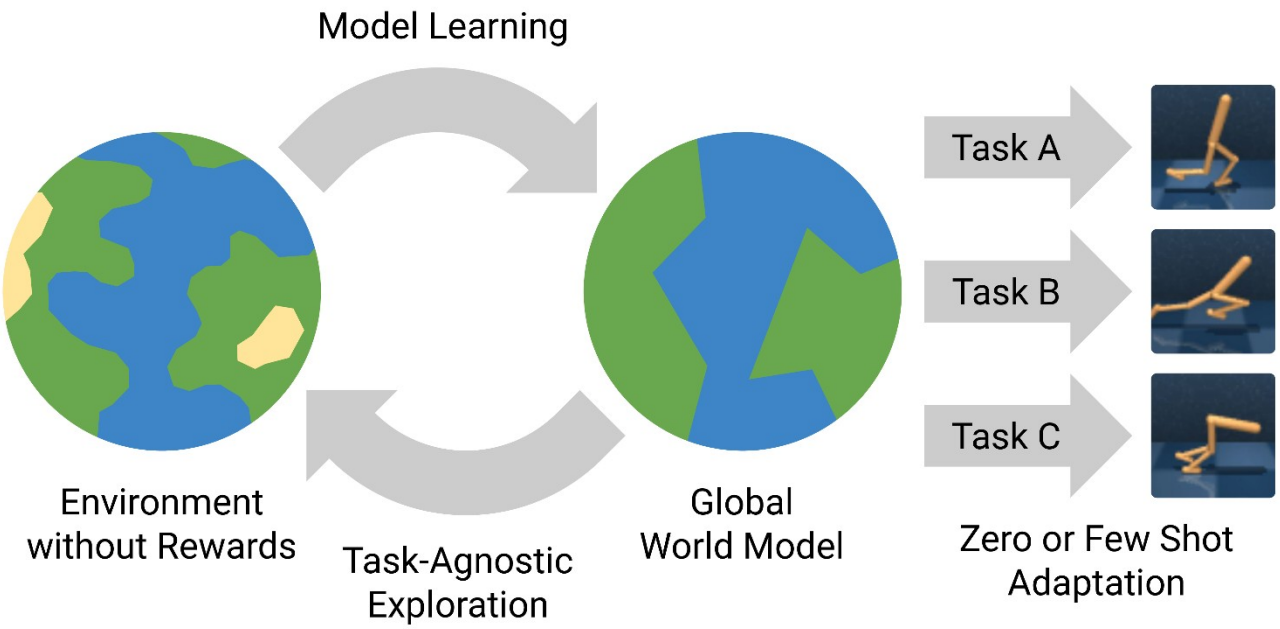
Telephone +44 (0)191 334 2000
www.durham.ac.uk


question
answer
user
text
text collection
Human Interface
symbolic interface
query/result
Symbolic Processor
facts
facts & rules
parsing text
dependency graph
Knowledge Base
Natural Language Processor
trained model
Deep Neural Network

Durham University

Inspiring the extraordinary





Durham University Palatine Centre Stockton Road Durham DH1 3LE UK
Telephone +44 (0)191 334 2000
www.durham.ac.uk

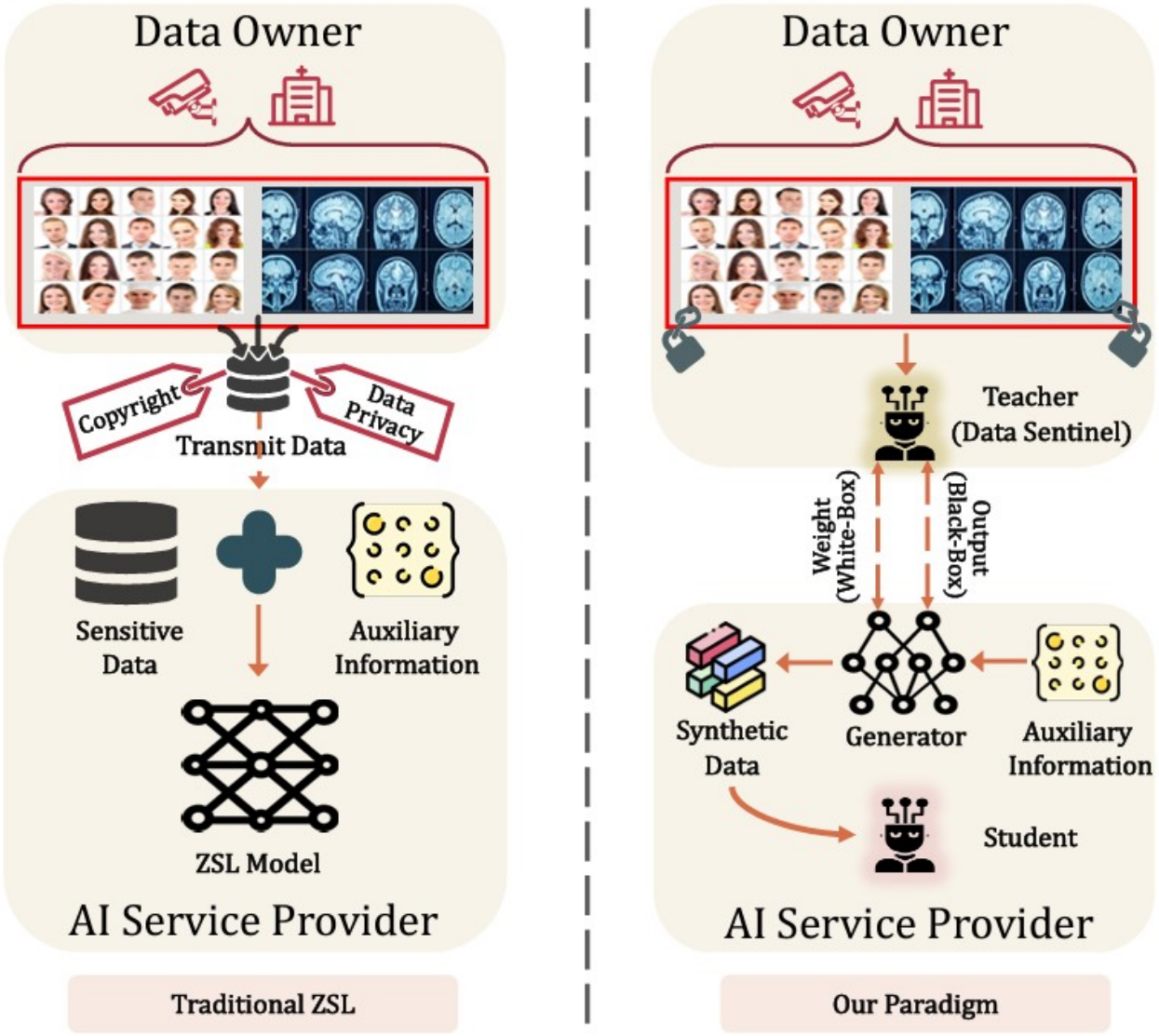


Fig. 1: In traditional ZSL approaches, real data is necessitated to establish the visual-semantic association. Conversely, SG-ZSL introduces a teacher model, which acts as a data sentinel, enabling the execution of ZSL tasks without the need for direct access to real data.