**Introduction to Inverse Reinforcement Learning (IRL)**

Inverse Reinforcement Learning, or IRL, aims to infer the **reward function** that explains an expert’s behavior. Unlike imitation learning, which directly mimics actions, IRL tries to understand **why** the expert makes certain decisions. By learning this reward function, an agent can generalize beyond demonstrations and adapt to new situations.

IRL relies on **state-action pairs**, representing decisions made by an expert. Instead of directly learning a policy, IRL estimates the reward function that best explains these choices. Once inferred, this reward function is used in Reinforcement Learning to train an agent that optimizes its behavior.

**Challenges in IRL**

Despite its advantages, IRL is computationally expensive since it involves solving multiple RL problems. Another key challenge is **reward ambiguity**—a single expert policy can correspond to multiple reward functions, making it difficult to determine the true underlying motivation. Additionally, most IRL methods assume knowledge of the **environment’s transition model**, which is often unknown in real-world applications.

**Types of IRL**

To address these challenges, different IRL methods have been developed:

1. **Maximum-Margin IRL**: Finds a reward function that maximizes the margin between the expert’s behavior and all other possible behaviors, ensuring the expert’s actions are significantly better than alternatives.
2. **Maximum Entropy IRL (MaxEnt IRL)**: Assumes that the expert is acting optimally but accounts for uncertainty by modeling the expert’s behavior probabilistically, assigning higher probability to trajectories with higher rewards while maintaining maximum entropy to avoid overfitting.
3. **Bayesian IRL**: Models the reward function as a probability distribution and updates beliefs about it as more expert demonstrations are observed, incorporating prior knowledge and uncertainty into the learning process.

**Graph-Based Representations for IRL**

In video-based IRL, one way to represent human behavior is through **graph-based representations**, where actions, objects, and interactions are structured in a way that captures temporal relationships. Unlike raw image-based methods, graph representations focus on meaningful relationships, filtering out unnecessary visual details.

**Egocentric Action Scene Graphs (EASGs) – Long-Term Behavior Understanding** *(Action Scene Graphs for Long-Form Understanding of Egocentric Videos)*

One effective graph representation is **Egocentric Action Scene Graphs (EASGs)**, which model human actions in **three key stages**: **Precondition, Point of No Return, and Postcondition**. A **structured representation** of actions over time, used for **long-term understanding** of egocentric videos. **Nodes** represent the camera wearer, action verb, and objects involved, while **edges** capture relationships between them.

In IRL, this structured representation is valuable because it provides **richer learning signals**, making it easier for agents to infer reward functions. Additionally, **EASGs can predict future scene graphs**, meaning an agent can anticipate upcoming actions and adjust its behavior accordingly—something traditional image-based representations struggle with.

**Action-Conditioned Scene Graphs (ACSGs) – Interactive Learning in Robotics** *(RoboEXP: Action-Conditioned Scene Graph via Interactive Exploration for Robotic Manipulation)*

A similar approach in robotics is the **Action-Conditioned Scene Graph (ACSG)**, which is actively constructed as a robot interacts with its environment. This graph encodes both **spatial relationships** and **action effects**, allowing the robot to understand how its interactions influence the world.

Unlike passive observation, this enables an IRL agent to **learn through exploration**, updating its graph dynamically as new objects and actions are encountered.

This is particularly useful in IRL because:

* **It captures causal relationships** between actions and objects, aiding in more accurate reward inference.
* **It supports incremental learning**, meaning agents refine their behavior over time instead of relying on fixed datasets.
* **It enables zero-shot generalization**, allowing an agent trained in one environment to transfer knowledge to new scenarios without additional training.

By structuring interactions in this way, IRL agents can learn **not just what actions to take, but why**, improving their ability to generalize behavior from human demonstrations.

**Graph-Based Video IRL – Learning from Diverse Demonstrations** *(Graph Inverse Reinforcement Learning from Diverse Videos)*

A key challenge in video-based IRL is dealing with diverse, unstructured videos. This paper proposes using **graph abstraction** to replace raw pixel data, filtering out irrelevant information like textures or lighting variations. Objects and interactions are mapped as nodes and edges, making it easier to extract meaningful behavioral patterns.

To improve reward learning, this method also integrates **Interaction Networks (INs)** to explicitly model object-object interactions over time. Additionally, it uses **Temporal Cycle Consistency (TCC)**, which ensures that sequences align properly across different videos—this helps IRL models better track task progression and infer reward signals even in varied environments.

Another key innovation is the **embedding space representation**, where videos are mapped based on task progression. This allows the agent’s reward function to be continuously adjusted based on how closely its actions align with expert demonstrations, rather than just evaluating final outcomes.

**Image-Based Representations**

Now, let’s shift our focus to **image-based representations**, a powerful approach for learning human behaviors from video. These methods rely on visual data to extract meaningful information for reward learning and decision-making.

**Value-Implicit Pretraining (VIP): Learning Rewards from Unlabeled Human Videos (***Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training***)**

This method, called **VIP**, introduces a new way of learning generalizable visual reward functions from large, unlabeled human video datasets.

Unlike traditional reinforcement learning models that require task-specific fine-tuning, **VIP learns from human behavior passively**, making it more scalable. This is crucial for **Inverse Reinforcement Learning (IRL)** because it allows agents to derive **reward functions that generalize across multiple tasks**, just as humans can adapt their behaviors to different contexts.

But how does VIP do this?

1. **Learning a Visual Representation**

* VIP first **pretrains an encoder** on large, unlabeled human video datasets.
* This encoder maps video frames into an **embedding space** where similar states (frames) are close together, and different states are farther apart.

1. **Defining the Reward Function**

* Instead of relying on explicit robot action labels, **VIP uses goal-conditioned reinforcement learning** to extract reward signals from human goal-directed behavior.
* It measures the **distance** between the robot’s current state and a goal state **in an embedding space**—the closer the robot is to the goal representation, the higher the reward.
* This makes **VIP a self-supervised learning approach**, meaning it can train on **massive** datasets without the need for hand-annotated labels.

We obtain a **generalizable reward model** that can be applied to **unseen tasks**. And this is where VIP becomes particularly valuable for IRL—since IRL aims to **extract reusable reward functions**, VIP provides a structured way to transfer these functions across different tasks and environments.

So how does VIP actually obtain its reward function?

* **First**, it learns from **unlabeled human videos**, observing **changes in visual data** rather than explicit action labels.
* **Second**, it uses **goal-conditioned rewards**, measuring how close the current visual embedding is to the goal state.
* **Finally**, VIP operates in a **self-supervised manner**, where the reward signal emerges naturally from **visual progress toward the goal**.

VIP doesn’t require explicit demonstrations or manually designed reward labels—it learns by simply watching humans in action.

**Unsupervised Perceptual Rewards for Imitation Learning (***Unsupervised Perceptual Rewards for Imitation Learning***)**

This method **automatically discovers task sub-goals** from human demonstration videos.

Instead of requiring predefined rewards, the model learns which steps matter by **training classifiers on high-level visual features** extracted from a deep learning model.

It provides **structured feedback** at different stages of a task, helping an agent learn **not just the final goal but also the intermediate steps**—something that’s often missing in traditional RL and IRL methods.

The learned reward function is based on **step-wise classifiers**, which means the agent receives **continuous feedback** throughout the task rather than just at the end.

This aligns well with **IRL methodologies** that focus on decomposing complex tasks into smaller, more manageable steps.

This approach is similar to **Sub-goal Weighted IRL (SWIRL)** and other segmentation-based IRL methods, which also aim to **break down tasks into meaningful sub-goals for improved learning efficiency**.

**Model-Based IRL from Visual Demonstrations (***Model-Based Inverse Reinforcement Learning from Visual Demonstrations***)**

A **model-based approach** allows for **planning over predicted future states**, reducing the need for trial-and-error learning.

The system consists of three main components:

* **A Keypoint Detector** – extracts meaningful low-dimensional visual features from images.
* **A Dynamics Model** – predicts state transitions based on these keypoints.
* **A Gradient-Based Model Predictive Planner (MPC)** – optimizes actions using the learned cost function and transition model.

This enables **visual model-predictive control (MPC)**, where the agent predicts and optimizes its actions based on learned **visual representations**.

A major advantage of this approach is that it **doesn’t rely on full raw images**. Instead, it **extracts keypoints**, which serve as a structured representation of the environment.

**Traditional IRL struggles with high-dimensional visual inputs**, making it computationally expensive.

Keypoints provide a **compact, structured representation** that simplifies learning and planning.

The system learns a **time-dependent cost function**, meaning different keypoints are weighted **dynamically** depending on the stage of the task—this ensures that the agent focuses on the most **task-relevant** features at each moment.

**Time-Contrastive Networks (TCNs): Learning Structured Representations from Video (***Time-Contrastive Networks: Self-Supervised Learning from Video****)***

One of the biggest challenges in learning from video is **viewpoint variation**. **Time-Contrastive Networks (TCNs)** tackle this by learning viewpoint-invariant representations that focus on motion-relevant features rather than raw appearance.

**Viewpoint-Invariant Embeddings for Imitation Learning**

* TCNs use **self-supervised learning** to align similar actions across different viewpoints.
* This is crucial for learning human behaviors from videos without requiring the same camera angle during training and deployment.

**Learning a Metric Space for Task-Relevant Actions**

* TCNs use **triplet loss** to structure the embedding space:
  + **Pulls together** temporally aligned frames from different viewpoints.
  + **Pushes apart** visually similar frames that are temporally distant.

**Leveraging Pretrained Features to Reduce Data Needs**

* Instead of learning from scratch, TCNs **build on pretrained deep networks** (e.g., ResNet, ImageNet).
* This enables efficient feature extraction with fewer labeled demonstrations.

While **TCNs do not directly infer IRL reward functions**, they provide a structured embedding space

**Temporal Cycle-Consistency Learning (TCC) (***Temporal Cycle-Consistency Learning****)***

One of the biggest challenges in video-based IRL is aligning multiple demonstrations when execution speeds, viewpoints, and motion styles vary. Traditional methods like keypoint tracking and feature matching struggle with these inconsistencies, leading to misaligned reward inferences.

TCC solves this problem by learning temporally aligned embeddings for video frames in a self-supervised manner:

* Learns per-frame correspondences across multiple video demonstrations.
* Captures temporal structure, ensuring actions align despite differences in appearance or speed.
* Improves state representations by normalizing execution styles, making reward inference more stable.

How TCC Works?

1. **Nearest-Neighbor Matching:** For each frame in a video, TCC finds its closest match in another demonstration using learned embeddings.
2. **Cycle-Consistency Loss:** The method ensures that if a frame A in one video maps to frame B in another, then B should map back to A, enforcing temporal consistency.
3. **Training Process:** TCC uses self-supervised learning with a neural network encoder to refine these embeddings over time.

Why it can be useful?

* **Aligns Video Demonstrations:** Ensures different expert trajectories match temporally, making reward extraction more accurate.
* **Reduces Noise in State Representations:** Helps IRL models generalize better across variations in human execution styles.
* **Enhances Feature Representations:** Provides structured embeddings that improve reward function estimation from video data.

**XIRL (***XIRL: Cross-embodiment Inverse Reinforcement Learning****)***

Another major challenge in IRL is transferring skills from **one embodiment to another**, such as teaching a robot to imitate human actions when their physical structures differ significantly.

XIRL (Cross-Embodiment Inverse Reinforcement Learning) addresses this by leveraging TCC to learn embodiment-invariant visual representations:

* Learns **task-relevant representations** rather than direct physical actions, allowing generalization across embodiments.
* Extracts a **reward function from human videos** without requiring expert action labels.
* Uses a **self-supervised learning approach** to align similar task phases across different embodiments.

**How XIRL Works?**

1. **Training the Encoder (φ):** A neural network extracts feature embeddings from video frames while TCC ensures alignment across different embodiments.
2. **Computing the Goal Embedding (g):** The goal state is defined as the average embedding of the final frames of multiple expert demonstrations.
3. **Defining the Reward Function:**
   * The agent's current state embedding is compared to the goal embedding.
   * The reward is computed as the negative distance between these embeddings, encouraging the agent to align with expert demonstrations.

**How XIRL Fits into IRL:**

1. **Learning an Embodiment-Invariant Representation:** Uses TCC to align video frames across different embodiments, ensuring task progression is preserved.
2. **Defining a Reward Function from Video Demonstrations:** Instead of traditional IRL reward recovery, XIRL defines rewards based on similarity to goal embeddings extracted from human demonstrations.
3. **Using the Reward in IRL or RL:** The learned reward function can be used in standard reinforcement learning frameworks to guide policy learning