# ****Serverless Edge Computing Function Offloading Pipeline using DRL****

## ****1. Problem Definition****

* **Goal:** Optimize function offloading in serverless edge computing (SEC) using **Multi-Agent PPO (MAPPO)**.
* **Challenges:**
  + Dynamic network conditions
  + Heterogeneous resource availability
  + Latency and energy trade-offs

## ****2. System Model & Feature Extraction****

### ****Function Characteristics (F):****

* wiw\_i: Required data size
* cic\_i: Required CPU cycles for computation
* mim\_i: Required memory
* did\_i: Execution deadline

### ****IoT Device State (G):****

* uiu\_i: Computation capacity of IoT device
* qiq\_i: Queue latency of the IoT device

### ****Edge Server State (H):****

* uieu^e\_i: Computation capacity of the edge server
* qeq^e: Queue latency of EFaaS

### ****Network State:****

* rier^e\_i: Data transmission rate
* tipt^p\_i: Transmission power
* hi,jh\_{i,j}: Channel gain between IoT device and base station
* σ2\sigma^2: Background noise variance
* Ni,jN\_{i,j}: Signal-to-Interference-plus-Noise Ratio (SINR)

## ****3. Latency Cost Considerations****

### ****Local Execution Latency (****LlL^l****)****

* til=ciui+qit^l\_i = \frac{c\_i}{u\_i} + q\_i
* If til≤dit^l\_i \leq d\_i, then cost = tilt^l\_i, otherwise penalty Υ\Upsilon

### ****Edge Execution Latency (****LeL^e****)****

* Transmission Latency: tie,tr=wiriet^{e,tr}\_i = \frac{w\_i}{r^e\_i}
* Container Startup Latency: tie,stt^{e,st}\_i
* Execution Latency: tie,com=ciuie+qet^{e,com}\_i = \frac{c\_i}{u^e\_i} + q^e
* Total Latency: tie=tie,tr+tie,st+tie,comt^e\_i = t^{e,tr}\_i + t^{e,st}\_i + t^{e,com}\_i
* If tie≤dit^e\_i \leq d\_i, then cost = tiet^e\_i, otherwise penalty Υ\Upsilon

### ****Overall System Cost****

* Lsys=∑i∈I(1−ai)Ll+aiLeL\_{\text{sys}} = \sum\_{i \in I} (1 - a\_i) L^l + a\_i L^e, where ai∈{0,1}a\_i \in \{0,1\} is the offloading decision.

## ****4. DRL-Based Optimization Approach****

### ****MDP Formulation****

* **State Space (S):**
  + St={F,G,H,uie,qe}S\_t = \{F, G, H, u^e\_i, q^e\}
* **Action Space (A):**
  + At={ai∣i∈I}A\_t = \{a\_i | i \in I\}, where ai∈{0,1}a\_i \in \{0,1\} (0 = local, 1 = offload)
* **Reward Function (R):**
  + R(St,At)=−LsysR(S\_t, A\_t) = -L\_{\text{sys}}

### ****Training Process****

1. **Environment Initialization:** Simulate IoT devices, edge servers, and network conditions.
2. **Action Selection:** Agents choose whether to offload functions using PPO policy.
3. **State Transition:** Environment updates based on resource utilization and network conditions.
4. **Reward Computation:** Compute latency costs and penalties.
5. **Policy Update:** Train MAPPO using actor-critic optimization.
6. **Convergence Check:** Stop training when rewards stabilize.

## ****5. Implementation Plan****

### ****5.1. Simulation Setup****

* **Environment:** Python-based SEC simulation
* **Task Classification:** XGBoost-based function classification
* **Communication Model:** SINR-based wireless channel simulation
* **Edge Server Emulator:** Simulated EFaaS environment

### ****5.2. DRL Framework****

* **Algorithm:** Multi-Agent PPO (MAPPO)
* **Neural Network:** Policy & Value Function with LSTM-based memory
* **Training Strategy:** Experience replay with target networks
* **Implementation:**
  + **Libraries:** PyTorch, Stable-Baselines3
  + **Framework:** Ray RLlib for multi-agent training

## ****6. Evaluation Metrics****

* **Latency Reduction:** Compare with heuristic-based offloading.
* **Energy Efficiency:** Measure power consumption.
* **Scalability:** Analyze system performance with increasing edge nodes.
* **QoS Satisfaction:** Check deadline fulfillment rate.

## ****7. Expected Outcome****

* Adaptive offloading decisions for varying workloads.
* Efficient SEC resource utilization.
* Reduced function execution latency.
* Improved energy efficiency of IoT devices.

This pipeline ensures a structured approach to implementing and evaluating your **DRL-based function offloading system in serverless edge computing.**