# *Course Project*

**Code of Honor.** All external resources used in the project, including research papers, open-source repositories, datasets, and any content or code generated using AI tools, e.g., ChatGPT, GitHub Copilot, Claude, Gemini, must be *clearly cited* in the final submission. The final report must also include *a clear breakdown of individual group member contributions.* Any lack of transparency in the use of external resources or in reporting group contributions will be considered academic dishonesty and will significantly impact the final evaluation.

| | |
|---|---|
| **Topic** | Cache Management via RL |
| **Category** | Resource Allocation |

Objective Design and implement an RL agent for cache management in a simulated system. The agent should learn eviction and admission strategies to minimize cache misses given a stream of content requests. The project aims to compare RL-based caching policies with classical baselines such as Least Recently Used (LRU) and Least Frequently Used (LFU).

Motivation Caching is a fundamental component of modern computer systems and networks, from CPU memory hierarchies to content delivery networks (CDNs). Traditional policies such as LRU and LFU are simple but cannot adapt to complex or changing access patterns. RL provides a framework for learning adaptive caching strategies that optimize hit rates under dynamic conditions [3, 4, 1, 2]. This project exposes students to a research-oriented application of RL in computer systems, with opportunities to design environments and reward functions tailored to performance and fairness.

Requirements The final submission should address the following requirements while the details can be freely decided by the group members.

1. Implementation: in this respect, you should
   - design a simple cache simulation with a stream of item requests,
   - implement an RL agent (e.g., DQN, PPO) that learns cache eviction/admission decisions, and
   - implement baseline caching strategies (LRU, LFU) for comparison.
2. Environment modification: to ensure originality, you **must** extend the environment with **at least one** of the following modifications:
   - introduce non-stationary request patterns (e.g., popularity shifts),
   - add heterogeneous costs for cache misses (e.g., penalty depending on item type),
   - simulate cache capacity fluctuations, or
   - introduce multiple cache nodes with limited coordination.
3. Evaluation: the final project should report key evaluation metrics in the modified environment. In this respect, the results should

- compare RL-based caching with baseline policies,
- analyze hit ratio, latency, and fairness across request types, and
- report robustness under dynamic request distributions.

4. The results should be elaborated through
    - ablation experiments, e.g., different reward formulations (hit rate vs latency), and
    - providing discussions on trade-offs between adaptivity and complexity.

Milestones The following milestones are to be accomplished through semester.

1. Literature Review and Setup
    - Review RL applications in caching and memory management.
    - Design a simple cache simulator and finalize environment modifications.
2. Implementation
    - Implement baseline caching policies (LRU, LFU).
    - Implement RL agent (DQN or PPO) and validate on stationary request distributions.
    - Train agent on modified (non-stationary, constrained) environments.
3. Evaluation and Analysis
    - Collect and plot caching performance metrics (hit ratio, latency).
    - Compare RL agent vs baselines across workloads.
    - Perform ablation experiments (reward variations, workload variations).
4. Final Report and Presentation

Submission Guidelines The main body of work is submitted through Git. In addition, each group submits a final paper and gives a presentation. In this respect, please follow these steps.

- Each group must maintain a Git repository, e.g., GitHub or GitLab, for the project. By the time of final submission, the repository should have
    - Well-documented codebase
    - Clear `README.md` with setup and usage instructions
    - A `requirements.txt` file listing all required packages or an `environment.yaml` file with a reproducible environment setup
    - Demo script or notebook showing sample input-output
    - *If applicable,* a `/doc` folder with extended documentation
- A final report (maximum *5 pages*) must be submitted in a PDF format. The report should be written in the provided formal style, including an abstract, introduction, method, experiments, results, and conclusion.
  **Important:** Submissions that do not use template are considered *incomplete.*
- A 5-minute presentation (maximum *5 slides including the title slide*) is given on the internal seminar on Week 14, i.e., *Dec 1 to Dec 5,* by the group. For presentation, any template can be used.

Final Notes While planning for the milestones please consider the following points.

1. Creativity in modifying the request model and reward functions is encouraged.
2. Training should remain feasible by keeping cache size small and request sequences short.
3. Students are expected to balance research-oriented exploration with achievable implementation.
4. Teams are expected to manage their computing needs and are advised to perform early tests to estimate runtime and training feasibility. As graduate students, team members can use facilities provided by the university, e.g., ECE Facility. Teams are expected to inform themselves about the limitations of the available computing resources and design accordingly.

## References

[1] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE communications surveys & tutorials*, 21(4):3133–3174, 2019.

[2] Zhe Wang, Jia Hu, Geyong Min, Zhiwei Zhao, and Zi Wang. Agile cache replacement in edge computing via offline-online deep reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, 35(4):663–674, 2024.

[3] Chen Zhong, M Cenk Gursoy, and Senem Velipasalar. A deep reinforcement learning-based framework for content caching. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2018.

[4] Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin. Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network*, 32(6):50–57, 2018.