Trustworthy Machine Learning, Spring 2025 *Mahmood Sharif*

# Course Projects

*Version No.: 1*

## 1 Introduction

This document contains instructions for the course projects and suggests some project ideas. You may select one of the projects proposed in this document, or, alternately, pitch your own project ideas to the instructor for approval.

## 2 Important Notes

- You need to form teams consisting of **two or (preferably) three members** to work on the projects. All members are expected to contribute equally. Students should register their teams via Moodle, under the Groups activity.
- Team registration and project selection should be completed by **June 5$^{\text{th}}$, 2025, 09:59**.
- All project descriptions contain references to related work. Please make sure to read the related work carefully before initiating the hands-on work. You may want to follow Keshav's advice on how to read academic research papers [12], especially if you have limited experience doing so.
- We recommend that you use a private GitHub repositories for version control and collaboration between the team members.
- To work on your projects, you should consider taking advantage of the school's SLURM cluster.[1]

## 3 Intermediate report

To ensure that everyone is making sufficient progress and provide feedback to all teams, teams are required to submit intermediate reports. The intermediate reports should contain the following sections: abstract, introduction, related work, work plan, and references. The work plan section should roughly describe how you intend to tackle the main problem in your project, while the other sections are in line with those that will appear in the final technical report (Sec. 4.1). The intermediate report should be **two to four pages long, and, similarly to the technical report, needs to be typeset in LaTex and formatted using the ACM double-column template** (see instructions on Moodle). Reports should be uploaded to Moodle toward the end of the semester, by **June 23$^{\text{rd}}$, 2025, 09:59**. Afterwards, all the teams will meet the instructor (∼10 minutes each) to discuss their projects.

## 4 Submission

The final submission consists of two primary parts: A technical report and code.

[1] See `https://www.cs.tau.ac.il/system/slurm`.

### 4.1 Technical Report

The technical report will serve as a textual summary of your work. We recommend that you structure it as follows:

1. **Abstract**: A concise summary of the report (∼200 words or fewer).
2. **Introduction**: Motivate the problem that your work seeks to address, and describe how it does so at a high level.
3. **Related work**: Describe prior efforts and where your work fits in (e.g., how it complements prior efforts).
4. **Technical Approach**: Detail and justify your technical approach, including the tools you used and the design choices you made.
5. **Results**: Here you need to present your experiment design (e.g., dataset, baselines, and analyses methods) followed by the evaluation results (e.g., reporting on appropriate metrics) to demonstrate the value of your work.
6. **Discussion**: In this section, you should discuss the broader consequences of your project and findings, the potential limitations of the work, and possible future work and extensions you envision.
7. **Conclusion**: Wrap up your work to leave the reader with the main takeaways.
8. **Contributions**: Shortly describe what were the contributions of each team member.
9. **References**: The bibliography cited from within the main body of the report should be placed at the end.

The report should be **four to eight pages long, typeset in `LaTex`, and formatted using the ACM double-column conference template** (see instructions on Moodle). The final submission should contain both the source and the PDF.

### 4.2 Code

The code should contain your complete implementation. Additionally, you should include a `README.md` file (i.e., a Markdown file) specifying detailed execution instructions (commands, dependencies, etc.).

### 4.3 How to Submit

You should submit a compressed tarball titles `<groupname>.tar.gz` (where `<groupname>` is a placeholder for your actual group's name), containing the code and the report. To create the compressed tarball, first make a directory called `<groupname>/` with two sub-directories: `code/` (containing your implementation and `README.md`) and `report/` (containing both the PDF and source in LaTex). Subsequently, execute the command `tar -czvf <groupname>.tar.gz <groupname>/`.

The final submissions should be uploaded to Moodle by **September 20$^{\text{th}}$, 2025, 23:59**.

## 5 Proposed Project

This section describes the proposed projects.

### 5.1 Scalable hardware attacks on large language models (LLMs)

Hardware-level bit-flip attacks mainly target classifiers and focus on quantized, low-precision models [18]. While attacks tailored for full-precision models have been proposed, these remain inefficient (e.g., due to solving mixed-integer programs) [3]. Accordingly, existing methods may be inadequate for attacking (generative) LLMs, for instance, to attain goals such as circumventing alignment. In this project, your aim will be to generalize established approaches to enable efficient bit-flip attacks targeting full-precision LLMs while flipping as few bits as possible and maintaining the model's overall utility.

### 5.2 Attacking and defending tabular models with LTNs

Generating adversarial examples against tabular models in a generic manner raises a profound question: To realistically assess models' adversarial robustness, how can we ensure that attacks in the feature space (i.e., in the space of inputs provided to models) can be implemented in the problem space? [5, 22] For instance, how can we ensure that evasive features produced against a phishing-page detector can be implemented as an actual phishing page? Prior work has tackled this problem in an ad hoc manner, through mining domain constraints from the training set and incorporating SMT solvers in attacks to project adversarial examples on constraints and ensure these are satisfied. However, projection operations are often costly and may conflict with the misclassification objective. Consequently, attacks may suffer from limited success and may incur significant run-time overheads, hindering their inclusion in adversarial training schemes. This project aims to address these shortcomings through logic tensor networks (LTNs) [2] that would be incorporated in attacks for better efficiency and attack success rates.

### 5.3 Robustness verification for tabular models

Verification techniques often tackle scalability issues, preventing them from proving adversarial robustness for large models or large attack budgets [14]. In certain domains, such as for tabular models, however, we may have access to information about the input domain that can help us bound the search space for adversarial examples and improve verifiers' scalability. For instance, domain constraints on the feature space [5, 22] can be used to constrain the input space beyond the attack budget, limiting the search space to sound adversarial examples, thus enabling robustness verification for larger attack budgets. In this project, you will explore this direction to help devise more precise and scalable verifiers for tabular models.

### 5.4 Attacking agents

Previous work has shown than agents based on LLMs and Vision Language Models (VLMs) are vulnerable to attack [1, 13]. For instance, a web agent may be manipulated into sending credit card numbers to adversaries or targeting users' contacts by phishing emails. In this project, you aim will be to explore extensions of these attacks to additional agents [17, 16] and propose new attack vectors with novel objectives and under new assumptions.

### 5.5 Assessing the robustness of new model types

Recent work has proposed a powerful set of model types, including Large Language Diffusion Models (LLaDA) [15] that match autoregressive LLMs on a variety of tasks, and biology-inspired Continuous Thought Machines (CTMs) [7] that synchronize processing across neurons and excel at classification and reasoning tasks. At the same time, the trustworthiness of these model types remains unknown. For instance, LLaDA models may produce harmful outputs [24] and CTMs may be even more vulnerable to adversarial examples than traditional Transformer- and CNN-based models [4]. This project aims to characterize the adversarial robustness of these new types of models.

### 5.6 Evaluating the robustness of ML-powered networking systems

Recent efforts from the networking community have proposed various ML-powered algorithms for optimizing networked systems. Notably, Rashelbach et al. proposed systems based on neural networks for packet classification [20] and longest prefix matching [19] with high scalability. In this project, you will study the reliability of these approaches through crafting worst-case inputs to evaluate their performance compared to traditional methods.

### 5.7 Irreversible backdoor attacks against foundation models

Backdoors against foundation models are pernicious, posing risk to a wide variety of downstream models built on top of the foundation models [21]. Nonetheless, prior work has also shown that excessive fine-tuning on downstream data can nullify the effectiveness of such backdoors [6]. As part of this project, you will explore means to make the backdoor more resilient against such defenses through drawing on ideas from none-fine-tunable learning [8].

### 5.8 Truth Serum: Adversarial suffixes to uncover memorization

LLMs often memorize their training data seen during pre-training or fine-tuning, thus potentially exposing sensitive secrets during deployment [28]. As part of this effort, you will explore a new attack vector to expose memorization. Specifically, you will study the possibility of creating a "Truth Serum," an optimized prefix that, when prepended to data seen during training, leads LLMs to complete the training samples verbatim.

### 5.9 Mitigating brand bias

Similarly to gender and ethnic biases, LLMs often exhibit biases w.r.t. brands. For instance, an LLM may favor one brand (e.g., Coca Cola) compared to another (e.g., Pepsi), presenting more positive sentiment toward this brand and having higher likelihood to include it in outputs [11]. This project aims to study the prevalence of such biases, adopt (debiasing) mitigations from gender and ethnic biases to prevent brand bias [23], and extend them to enable a finer-grain control over LLMs outputs (i.e., "surgical" bias-editing) that strikes better balance between utility and bias.

### 5.10 Iterative gradient leakage attacks

Data-reconstruction and label-leakage attacks against federated learning typically leverage updates from a single iteration to reconstruct inputs or extract labels [9, 27]. This raises the question of

whether such attacks would become significantly more potent and accurate by leveraging updates that clients share across multiple iterations, especially closer to convergence. Your goal in this project would be to extend attacks to multiple-iteration settings and study how attack success improves as a function of the number of client updates.

### 5.11 Additional Project Ideas

- Clean-label poisoning to enable privacy attacks against inliers [25].
- Robustness via an activation-sparsity-oriented training [26].
- Fingerprinting NLP models as part of reconnaissance attacks.
- Robustness-aware pruning of models.
- Studying the effect of universal embedding spaces on corpus poisoning [10].

## References


[1] L. Aichberger, A. Paren, Y. Gal, P. Torr, and A. Bibi. Attacking multimodal OS agents with malicious image patches. *arXiv*, 2025.

[2] S. Badreddine, A. d. Garcez, L. Serafini, and M. Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[3] J. Bai, B. Wu, Z. Li, and S.-t. Xia. Versatile weight attack via flipping limited bits. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[4] Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are Transformers more robust than CNNs? In *Proc. NeurIPS*, 2021.

[5] M. Ben-Tov, D. Deutch, N. Frost, and M. Sharif. CaFA: Cost-aware, feasible attacks with database constraints against neural tabular classifiers. In *Proc. IEEE S&P*, 2024.

[6] G. Cui, L. Yuan, B. He, Y. Chen, Z. Liu, and M. Sun. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. In *Proc. NeurIPS*, 2022.

[7] L. Darlow, C. Regan, S. Risi, J. Seely, and L. Jones. Continuous thought machines. *arXiv*, 2025.

[8] J. Deng, S. Pang, Y. Chen, L. Xia, Y. Bai, H. Weng, and W. Xu. SOPHON: Non-fine-tunable learning to restrain task transferability for pre-trained models. In *Proc. IEEE S&P*, 2024.

[9] N. Gat and M. Sharif. Harmful bias: A general label-leakage attack on federated learning from bias gradients. In *Proc. AISec*, 2024.

[10] R. Jha, C. Zhang, V. Shmatikov, and J. X. Morris. Harnessing the universal geometry of embeddings. *arXiv*, 2025.

[11] M. Kamruzzaman, H. M. Nguyen, and G. L. Kim. "Global is good, local is bad?": Understanding brand bias in LLMs. In *Proc. EMNLP*, 2024.

[12] S. Keshav. How to read a paper. *ACM SIGCOMM Computer Communication Review*, 37(3):83–84, 2007.

[13] A. Li, Y. Zhou, V. C. Raghuram, T. Goldstein, and M. Goldblum. Commercial LLM agents are already vulnerable to simple yet dangerous attacks. *arXiv*, 2025.

[14] Marabou Developers. Marabou. `https://github.com/NeuralNetworkVerification/Marabou`.

[15] S. Nie, F. Zhu, Z. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and C. Li. Large language diffusion models. *arXiv*, 2025.

[16] OpenInterpreter Developers. Openinterpreter: An agent-computer-interface project from seattle, washington. `https://www.openinterpreter.com/`.

[17] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Proc. NeurIPS*, 2024.

[18] A. S. Rakin, Z. He, and D. Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proc. ICCV*, 2019.

[19] A. Rashelbach, I. de Paula, and M. Silberstein. Neurolpm-scaling longest prefix match hardware with neural networks. In *Proc. MICRO*, 2023.

[20] A. Rashelbach, O. Rottenstreich, and M. Silberstein. A computational approach to packet classification. In *Proc. SIGCOMM*, 2020.

[21] L. Shen, S. Ji, X. Zhang, J. Li, J. Chen, J. Shi, C. Fang, J. Yin, and T. Wang. Backdoor pre-trained models can transfer to all. In *Proc. CCS*, 2021.

[22] T. Simonetto, S. Ghamizi, and M. Cordy. TabularBench: Benchmarking adversarial robustness for tabular deep learning in real-world use-cases. In *Proc. NeurIPS*, 2024.

[23] S. Singh, S. Ravfogel, J. Herzig, R. Aharoni, R. Cotterell, and P. Kumaraguru. Representation surgery: Theory and practice of affine steering. In *Proc. ICML*, 2024.

[24] A. Souly, Q. Lu, D. Bowen, T. Trinh, E. Hsieh, S. Pandey, P. Abbeel, J. Svegliato, S. Emmons, O. Watkins, et al. A strongreject for empty jailbreaks. In *Proc. NeurIPS*, 2024.

[25] F. Tramèr, R. Shokri, A. San Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proc. CCS*, 2022.

[26] G. Vardi, G. Yehudai, and O. Shamir. Gradient methods provably converge to non-robust networks. In *Proc. NeurIPS*, 2022.

[27] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proc. CVPR*, 2021.

[28] J. Zhang, D. Das, G. Kamath, and F. Tramèr. Membership inference attacks cannot prove that a model was trained on your data. In *Proc. SaTML*, 2025.