Deep Learning Projects / Spring 2026

PROJECT DEADLINES AND REPORT GUIDELINES

Rules:
- Projects can be on teams of up to 3 students
- Students must write their reports independently (but they can write the code and run experiments together)
- The final project accounts for 35% of the grade.
- Projects should use elements learned in the course (just running someone else's code is not enough to count as a project)

Description deadline

1. Project description deadline: **03/03 (Tuesday) 6pm** on Gradescope
2. The initial project descriptions can be submitted one per group.
3. **Project description guidelines (project descriptions are part of the project grade):**
   (a) Project description: exactly 2-pages in ICLR 2026 format:
   `https://github.com/ICLR/Master-Template/raw/master/iclr2026.zip`
   (b) **Description must contain a title and student name + Purdue ID**
   (c) Section "Introduction": ~1/2 page to describe objective, clearly define the task (statistically, e.g., learn $p(x)$, learn $p(y|x)$ where train and test data have the same distribution, etc.)
   (d) Section "Dataset": ~1/4 page to describe dataset used and why it aligns with the task
   (e) Section "Deep learning method used": ~1/2 page to describe deep learning methods that will be used (or tried) and why it aligns with the task
   (f) Section "Related work": ~1/2 page to describe the related work and how it connects to the proposed work
   (g) Section "Expected results": ~1/4 page describing the expected results and metrics of success (how can we tell your project failed or has been successful?)

Complete project submission deadline

1. **Final project report due: 04/23 (Thursday) 6pm** on Gradescope.
2. Final project reports must be written individually to ensure everyone contributes and prevent one person from doing all the work.
3. **Final project reports guidelines:**
   (a) Final report must contain between 4 and 6 pages in ICLR 2026 format:
   `https://github.com/ICLR/Master-Template/raw/master/iclr2026.zip`
   (b) **Description must contain a title and student name + Purdue ID**
   (c) Section "Introduction": ~1 page to describe objective, clearly define the task (statistically, e.g., learn $p(x)$, learn $p(y|x)$ where train and test data have the same distribution, etc.)
   (d) Section "Dataset": ~1/4 page to describe dataset used and why it aligns with the task
   (e) Section "Proposed Approach": 2+ pages to describe deep learning methods that will be used (or tried); formally write down the **objective function**, describe why objective aligns with objective described in the introduction.
   (f) Section "Related work": ~1/2 page to describe the related work and how it connects to the proposed work
   (g) Section "Results": 1+ page describing the results and metrics of success.

SOME SUGGESTED TOPICS (FOR STUDENTS WITHOUT A TOPIC)

1. **Exploring better Question and Answering query methods**
   (a) A recently published benchmark (https://arxiv.org/pdf/2410.12537) in question answering (QA), shows existing approaches struggle to achieve strong results. This presents an opportunity for a project: the goal is not necessarily to surpass state-of-the-art performance, but rather to explore novel ways of modifying neural network architectures or training methods and document their impact on model behavior in your report. From alternative GNNs, new attention mechanisms, to unconventional loss functions. This project could be a great fit for those of you interested in LLMs and graphs.

2. **Enhancing Bayesian Sampling Procedures for Large Language Models (LLMs)**
   (a) Consider a Large Language Model (LLM) as a fundamental proposal distribution, where $x \sim q(x)$ represents a sentence or sequence of tokens. The LLM generates text based on a given temperature T, which controls the diversity and coherence of the output. Specifically:
      i. For a fixed temperature T, the LLM produces a sequence of tokens (a sentence) as its output.
      ii. As the temperature T increases, the generated samples exhibit greater diversity, but also become more prone to producing nonsensical or low-quality text.
   (b) To refine the sampling process, one can integrate a rejection sampling procedure, leveraging a classifier to evaluate the quality of the LLM's outputs. The classifier determines whether each generated sample is accepted or rejected based on its coherence and relevance.
   (c) There are multiple possible projects:
      i. Can techniques such as Hamiltonian Monte Carlo help in this search? Could one devise a good Metropolis-Hastings procedure?
      ii. Could we design an adaptive layer that takes the LLM's pre-softmax outputs as input and generates proposals that are more likely to be accepted by the classifier. By developing such an adaptation mechanism, we aim to improve the efficiency and effectiveness of Bayesian sampling procedures for LLMs, ultimately enhancing the quality and diversity of generated text.

3. **Exploring Conditional Invariances for Enhanced Robustness in Supervised Neural Network Learning: Investigating Data-Dependent Group Symmetries**
   (a) This topic considers *conditional invariances*, where the choice of invariance (G-invariance or G-equivariance) applied to the neural network is adaptively determined based on the characteristics of the input data. For instance, image classification of birds will be G-invariant to rotations, but image classification of "6" and "9" will not be invariant to rotations. By examining how different group symmetries can be leveraged to improve the robustness of neural networks, this project aims to contribute to the development of more resilient and effective supervised learning models. The project focuses on designing and evaluating conditional invariance mechanisms that can dynamically adjust to varying input data.

4. **Graph Neural Networks for Combinatorial Optimization**
   (a) Combinatorial optimization problems such as set covering, facility location, and combinatorial auctions are typically solved via branch-and-bound algorithms in Mixed-Integer Linear Programming (MILP) solvers. A key bottleneck in these solvers is the variable selection (branching) decision at each node of the search tree. Traditional heuristics like strong branching are effective but computationally expensive. Gasse et al. (NeurIPS 2019, `https://arxiv.org/abs/1906.01629`) proposed learning a branching policy using a Graph Convolutional Neural Network (GCN) that operates on the variable-constraint bipartite graph representation of the MILP. The GCN is trained via imitation learning from the strong branching expert, learning to mimic expert variable selections at a fraction of the computational cost. The learned policy generalizes to problem instances significantly larger than those seen during training. Scavuzzo et al. (NeurIPS 2022, `https://arxiv.org/abs/2205.11107`) introduced tree MDPs for RL-based branching with improved credit assignment. Beyond branching, GNNs have been applied to cutting plane selection (Li et al., NeurIPS 2023, `https://arxiv.org/abs/2311.05650`), node selection (Labassi et al., NeurIPS 2022, `https://arxiv.org/abs/2210.16934`), and primal heuristics via neural diving (Nair et al., `https://arxiv.org/abs/2012.13349`).
   (b) There are multiple possible projects:

   i. Reproduce the GCN branching approach of Gasse et al. and compare imitation learning against RL-based alternatives such as SORREL (AAAI 2025, `https://arxiv.org/abs/2412.15534`), which shows that RL can match imitation learning even when trained from suboptimal demonstrations.
   ii. Implement a GNN-based cutting plane selector that learns which cuts to add at each round of the branch-and-cut algorithm. Li et al. (NeurIPS 2023) showed that learning to configure separators can accelerate SCIP by up to 72% on certain benchmarks.
   iii. Investigate the representational limits of message-passing GNNs for MILP tasks. Chen et al. (ICLR 2023, `https://arxiv.org/abs/2210.10759`) proved that standard GNNs cannot distinguish certain feasible and infeasible MILPs, while a follow-up (NeurIPS 2024, `https://arxiv.org/abs/2402.07099`) showed that higher-order GNNs overcome this limitation.

5. **Graph Foundation Models (GFMs)** Graph Foundation Models (GFMs) and Relational Transformers (RT) (`https://arxiv.org/abs/2510.06377`) promise unified intelligence for relational data by pretraining on diverse databases. However, they struggle with heterogeneity (distinct attribute domains) and limited zero-shot generalization to unseen attribute spaces. STAGE (`https://github.com/snap-stanford/stage-gnn`) addresses this by encoding statistical dependencies (e.g., order statistics, conditional dependencies) instead of raw attribute values. This project proposes STAGE-RT, an extension of RT that integrates STAGE's order statistics encoding to enable robust zero-shot transfer across graphs with heterogeneous node attributes. This approach enhances RT's Relational Attention with statistical inductive biases, improving generalization to unseen attribute domains while preserving scalability and multi-table relational modeling.
   (a) Reproduce relational transformer `https://github.com/snap-stanford/relational-transformer` over a data subset (using the entire data requires a lot of compute).
   (b) Apply the order statistics encoding of ordered sets prescribed on STAGE `https://github.com/snap-stanford/stage-gnn` to the relational transformer.