Course Syllabus
Jump to Today
Welcome to CS 190N!
Course Overview
Objective
This course will focus on learning problems for networking, i.e., how network protocols or network operators make their decisions at different granularities (e.g., network, TCP, application, etc.) to keep networks safe and performant. How these decisions are made right now, and how one can replace existing heuristics-based decision-making with ML-based learning models. We will learn about the challenges of applying machine learning to learning problems in networking and how we can resolve them. Specifically, we will learn how different stages in ML pipelines work, which includes data collection, data pre-processing and feature extraction, model selection, training, evaluation, analysis, etc., and how to make changes to individual stages to develop models that are not only performant but reliable (i.e., trustworthy, explainable, robust, etc.). In this course, we will get our hands dirty working with real data and developing/analyzing different learning models. This course will organically build up on the topics covered in CS 176A (Computer Networks) and CS 5A and B (Data Science).

Approach
In CS 176A, we covered formal principles broadly applicable to different networks (e.g., packet switching) and read about various protocols enabling different hosts to communicate. However, these principles and protocols have not evolved much in the past few decades. In contrast, this topic is still evolving. Therefore, instead of following a conventional textbook, I will be publishing new content as course notes, synthesizing ideas presented in research papers from the top networking, security, and ML conferences (e.g., USENIX NSDI, USENIX Security, ACM SIGCOMM, ACM CCS, NeurIPS, etc.) to cover the topics in this course.

Prerequisite
Before taking this course, students must take CS 176A and be familiar with networking and operating systems basics. Also, it will help if the students have already taken CS 5 A/B.

Course Structure
We will have two lectures and one discussion section every week. The lectures will cover various ML4Nets concepts and their applications, while the discussion sections will provide hands-on opportunities to master the ML4Nets pipeline. The course entails regular quizzes and programming assignments, and the grading is based on them, along with a term project. There are no mid-ter or end-term exams for this course.

Lectures
The lectures will cover different components of the ML pipeline.

Quizzes
This course will have two quizzes, testing the understanding of concepts covered in the lectures and discussion sections.

Assignments
This course will have three programming assignments, giving students hands-on experience with different ML4Nets pipeline components.

Class Participation
This course will assign 10 % of the total grade points based on active participation in- and post-lecture discussions (over Piazza). Regularly attending lectures and engaging in these discussions is highly encouraged, as they are an integral part of the learning process and can significantly impact your final grade. You are required to proactively report any plans to miss the lecture or discussion section.

Term Project
This course will entail a term project. One key aspect of this class is to have students gain hands-on experience developing and analyzing ML models. To do this, each student will need to propose a term project. The teaching staff will provide feedback on the proposal and track the progress of each student. We will provide all the required computing resources to each team for their project.

Team

This is a team effort, and you can have up to five members on your team. Given the amount of effort, I will encourage you to create a team of three. Please consider using Piazza to find your teammates. You are required to report the team name and its members by October 25.

Proposal
Your research proposal will report the problem statement and motivation and give a rough idea of the related work and critical insights. More concretely, the project proposal must clearly mention the following aspects of the project:

What is the motivation for the problem?
What is the exact specification of the problem?
What are some existing approaches to this problem?
What are some existing datasets that you can work on?
What is the novelty of your project? New problem? New approach? New dataset?
How are you going to implement your approach, what are your metrics of success, and what's your evaluation plan?
This document should be at most two pages of content; you may use extra pages for bibliographic references. Your proposal must follow the formatting requirements mentioned above. Your instructor will read your plan in detail and provide feedback. Based on your instructor's feedback, you will refine your proposal. This document is due by Nov 1.

You can consider the following strategies for your projects: (1) identify a new learning problem and develop a reasonable solution to the problem, leveraging the closed-loop ML pipeline, (2)

identify an existing learning problem and solution and explore how you can reproduce the original results (using the provided publicly accessible dataset) and explore its vulnerability to underspecification issues, (3) identify an existing learning problem, and explore how to leverage PINOT and netUnicorn to curate new (representative) data, and then either employ an existing ML-based solution or develop a new one. Explore the generalizability of trained ML models. For approaches (2) and (3), there will be bonus credits for iteratively collecting new data to improve the model's generalizability.

You are free to explore new and existing learning problems (and their solutions) on your own. But, for starters, here are some ideas.

Existing learning problems and related ML-based solutions:

Video streaming applications, like YouTube or Netflix, download video in chunks. In this project, you would use the packets (and related metadata) from the past few video chunks to predict how long it will take to download the next chunks. For example, if the network is slow, the next chunk might take longer to download. This kind of prediction can help streaming services ensure that users don’t experience interruptions like rebuffering. The Puffer project at Stanford provides tools and insights that could help guide this project.
Quality of Experience (QoE) refers to how a user perceives the quality of their connection while using video streaming (e.g., YouTube, Netflix) or video conferencing (e.g., Zoom, Microsoft Teams). This project involves using machine learning to infer users' QoE based on network performance metrics. Tools like NetMicroscope Links to an external site. and Requet Links to an external site.(for streaming) and IMC'23 Links to an external site. (for conferencing) provide examples of how this might be done. The challenge here is to accurately predict how users experience the quality of their connection based on underlying network data.
When watching a video online, the streaming service dynamically adjusts the video quality (bitrate) based on the network conditions to prevent buffering. This project would involve learning an optimal policy for selecting the best bitrate using machine learning. Several papers have explored this, including RPC (not ML) Links to an external site., Pensieve Links to an external site., and OBOE Links to an external site.. The challenge here is to maximize video quality without causing buffering, especially under fluctuating network conditions.
When data flows over a network, it’s not always clear which user, device, or application is responsible for generating the packets. This project would involve using machine learning techniques to classify packets, i.e., to map them to their respective users, devices, or applications. There are existing tools like Neural packet classification Links to an external site. and nPrintML Links to an external site. that do this, but improvements can be made, especially as more data becomes encrypted. You can refer to the paper "A Look Behind the Curtain: Traffic Classification in an Increasingly Encrypted Web Links to an external site." for deeper insights. You can also consider a subset of results from these papers as part of your term project.
New learning problems:

When data is transferred over the Internet, it uses a protocol called TCP (Transmission Control Protocol). There are different "flavors" or "variants" of TCP (e.g., Reno, Cubic), each with its

own way of managing data transmission. In this project, you would take the first few packets of a flow (e.g., the first 5, 10, or 100 packets) and try to infer which TCP variant is being used. The challenge is that you only see a small part of the flow, but you need to make a prediction based on patterns in the packets (such as timing, size, etc.). This is useful for network operators trying to understand how data is being sent and optimize performance.

When users run a speedtest to measure their Internet speeds, the test runs for a certain duration. But is it possible to predict the final speed (upload or download) using just the first few packets? This project involves predicting QoS (Quality of Service) metrics like speed from the first k packets of the speedtest flow. The goal is to determine how early the test can be stopped while still getting an accurate speed measurement. This would save time and resources, especially on networks with limited bandwidth.

YouTube (or similar streaming platforms) often estimates network bandwidth to decide on the video quality that users should receive. This project involves understanding the rules or heuristics YouTube might use to make this decision. Since YouTube's exact algorithms are proprietary, you can use open datasets (like from the PINOT project or netUnicorn) to infer similar rules. Alternatively, you can simplify the problem by using the Puffer project's video player to replicate YouTube's behavior in a more controlled setting.

YouTube adjusts video quality based on network conditions using ABR (Adaptive Bitrate) algorithms. The challenge here is to reverse-engineer or explain these decision rules. Since the exact rules are unknown, an alternative approach is to create a black-box machine learning model that mimics YouTube's decisions and then use explainability tools to interpret the model's decision-making. This allows you to understand how video quality adjustments are made in real time.

This project focuses on building a machine learning model that not only predicts user QoE but also explains its decisions. You would train two models: one for making predictions and another for explaining which features (network data) are most important for these predictions. This approach can be applied to several problems, including video streaming or traffic classification. The idea is to make the system transparent, so users or network operators can understand how and why certain QoE levels are predicted.

netMicroscope is a tool for predicting QoE, but it hasn't been extensively tested under real-world conditions. This project involves using the netReplica tool to generate realistic network conditions and then testing netMicroscope's performance. The goal is to create a variant of the original model that can handle real-world network conditions more accurately. You could also experiment with pre-trained network foundation models to improve netMicroscope's performance.

Using anonymized data from the campus network, this project involves curating a dataset of YouTube traffic. The challenge is to quantify the differences in network features (like packet size, and inter-packet timing) across various network conditions. Once the dataset is curated, you could apply domain-adaptation techniques to improve the model's robustness in handling different types of networks.

This project involves using the Puffer video player and netReplica to generate a labeled dataset of QoE metrics for video streaming. The idea is to explore how various network conditions (simulated by netReplica) impact QoE metrics like buffering, video quality, and load times. The dataset you generate can be used for future QoE prediction tasks and to study the relationship between network conditions and user experience.
In this project, you will use the labeled QoE dataset you created (or an existing one) to refine the data collection process. Using the Trustee tool, you will analyze how the current model makes decisions, identify underspecification issues (e.g., when the model relies on shortcuts), and modify netReplica's configuration to generate better training data. This iterative process will improve the model until it is free from obvious underspecification issues.

This project involves learning a policy that minimizes the number of active measurements needed to estimate coarse-grained QoE metrics (e.g., the number of rebuffering events during a video). The intuition is that for high-speed, stable networks, fewer measurements are needed, while for slower, more variable networks, more measurements are required. The goal is to balance prediction accuracy with minimal data collection (i.e., how many chunks to download and measure).
For all the above projects, you can replace video streaming applications like YouTube with video conferencing applications like Zoom, Google Meet, or Microsoft Teams. The core ideas remain the same, but the challenges may differ because of the interactive, real-time nature of video conferencing.

REPORT
You must use this style file Links to an external site. for writing the report. The final report must be at least three pages long, excluding references. It is encouraged to include the following components in your reports (not necessarily in this order): abstract, introduction (motivation, task definition, your novel contributions), related work, your technical approach, such as math formulation of the problem, algorithms, theorems (if any), experiments, discussion, and conclusion.

This report is due by December 13.

Execution
It is required that each group create a GitHub repo for their project with a README file, which provides instructions on how to reproduce different results in the report. These instructions should be as descriptive and self-sufficient as possible. This step will help the instructor assess how well your team executed different proposed ideas. Your project report should provide a pointer to this GitHub repository.

Checkpoints
Given how critical this term project is for this course. We want to make sure that we have some additional checkpoints beyond what I discussed above.

0. (0 pts) Send the names of your teammates to the instructor over email (cc both TAs) by Oct 25.
1. (5 pts) Understand related works / Write Proposal (3 weeks) done by Nov 1.
2. (5 pts) Setup data collection with PINOT/netUnicorn (2 weeks) done by Nov 15.
3. (5 pts) Replicate code/train model using new data (2 weeks) done by Nov 29.
4. (20 pts) Evaluation/Analysis/Write Report (2 weeks) done by Dec 13.

For each of these checkpoints, we will require you to submit a small update (not more than 200 words) that lets us know how you are progressing on the term project and whether you are on track or need any help. That way, we can help you better to make consistent progress and calibrate the expectations if/when needed. We will provide more instructions on how to prepare updates for each of the checkpoints listed above.

Student Evaluation
The grading for this course is structured around quizzes, programming assignments, and mid/end-term examinations, as shown below.

Assignment Type Weight
Quizzes (2) 15%
Programming Assignments (3) 40%
Term Project 35%
Class Participation
10%

Course Summary:
Date Details Due
Tue Oct 29, 2024 Assignment Programming Assignment 1 due by 11:59pm
Fri Nov 1, 2024 Assignment Project Proposal due by 11:59pm
Fri Nov 8, 2024 Assignment Programming Assignment 2 due by 11:59pm
Fri Nov 15, 2024 Assignment Project Checkpoint 1 due by 11:59pm
Assignment Quiz 1 due by 11:59pm
Fri Nov 29, 2024 Assignment Project Checkpoint 2 due by 11:59pm
Fri Dec 6, 2024 Assignment Programming Assignment 3 due by 11:59pm
Fri Dec 13, 2024 Assignment Project Final Report due by 11:59pm
Assignment Quiz 2 due by 11:59pm
Assignment Participation