APSY 780 – Computational Methods for AI models, Class Number: 9115, Credits: 3
Spring 2025, Mon & Wed 13:10 PM – 14:30 PM, Lecture Center 15

Instructor: Gaurav Malhotra
Office Location: SS 387
Office Hours: Mon 2:45—4:45PM
E-Mail: gmalhotra@albany.edu

Course Description and Learning Objectives:

This course provides students with fundamental programming skills and methodologies essential for developing cognitive computational models, particularly machine learning models such as convolutional neural networks, transformers, and deep reinforcement learning models. We will explore practical tools (e.g., version control using *GitHub*) and fundamental programming concepts (e.g., object-oriented programming). Additionally, students will gain hands-on experience in model development and the opportunity to connect these models to their own research endeavours.

Support

Office hours: If you are struggling with any course content, or would like to talk about a personal issue that is interfering with your learning, please come and see the instructor during the office hours. Office hours may also be very useful for clarifying any details of the the mini-projects before submitting them. If you're unable to make the office hours, please come and see me after the class meeting or email me and we will work out an alternative time.

Support for students with disabilities:

Reasonable accommodations will be provided for students with documented physical, sensory, systemic, medical, cognitive, learning and/or mental health (psychiatric) disabilities. If you believe you have a disability and require accommodation in this class, please register with Disability Access and Inclusion Student Services (DAISS). You can contact DAISS at daiss@albany.edu, 518-442-5501 or www.albany.edu/disability. Once you have registered with DAISS, they will provide you with an accommodation letter that you can send to your instructors to receive your approved accommodation.

Course Structure:

The course comprises of twelve modules listed below. Each module contains a **mini-project** that we will collaboratively create during our class. These mini-projects are built around research questions from cognitive psychology and requires you to learn as aspect of Python programming. The idea is that as you build these projects, you will iteratively acquire the skills required to create your own project using AI tools (such as PyTorch). There is a wide range of topics covered including perception, memory, attention, decision-making, and psycholinguistics. I have listed the mini-projects below, but we will revise this list as we go along based on the skills needed and interests of the class.

Each module will be covered in three stages:

Stage 1 (in-class): In-class demos, where we will collaboratively do some live coding (led by the instructor). These demos will focus on aspects of the programming language (Python), frameworks (such as PyTorch) and libraries (such as numpy and scipy) required to carry out the mini-project.

Stage 2 (in-class): In-class mini-project, where one student will lead the development of the code for the mini-project. Everyone will contribute and discuss how to the creation of this mini-project.

Stage 3 (take home): An extension to the mini-project, which will be completed by each student at home based on what they learn during the demos (stage 1), and creation of the in-class mini-project (stage 2).

To assist learning, each mini-project will be accompanied by some useful resources. You can obviously look at (at least some of) the resources before the class, but you are also free to consult these during the class and when you work on your mini-project extension. I have identified useful references for the first few mini-projects and I'll be updating this list as we go along.

Assignments & Evaluation

The course assessment will be based on the take-home "extensions" to the mini-project. As the name implies, each extension will ask you to build upon the in-class mini-project. This will challenge you to acquire additional skills and enhance your proficiency in creating AI models. Your final grade will be evaluated as follows:

- Extensions to mini-projects (12 total, lowest 2 dropped): 10% x 10 = 100%
- *Note 1: Deadlines: You will have 1 week from when each extension is handed out to complete it.*
- *Note 2:* In the event that the course covers fewer than 12 modules (likely), the quantity of assignments and their point values will be adjusted so that the total is 100%.

Modules:

Note: I will be revising these modules as we go along, based on the progress we are making.

Module 1: Stroop Effect & Data Analysis Basics

Psychological Motivation: Selective attention and interference (Stroop, 1935). Slower responses when word meaning conflicts with ink color.

Programming Skills: Python basics, Pandas, CSVs, plotting.

Mini-Project (in-class): Simulate Stroop RT data; visualize congruent vs. incongruent distributions.

Extension (take home): Extend the simulation to multiple participants, each with slightly different average reaction times. Then plot both individual-level Stroop effects and the group average.

Expected Output: A set of plots showing within-participant variability and an aggregated group plot, highlighting that Stroop effects are robust across individuals but noisy at the trial level.

Helpful Resources:

- VanderPlas – Whirlwind Tour of Python
- Stroop (1935): Studies of interference in serial verbal reactions

Module 2: Memory Recall Simulation & Data Structures

Psychological Motivation: Memory accuracy over repeated trials; forgetting vs. learning curves.

Programming Skills: Lists, dicts, indexing, Matplotlib.

Mini-Project (in-class): Simulate memory recall accuracy across participants and trials; visualise forgetting curves.

Extension (take-home): Add a condition with distractor interference (e.g., background noise, or overlapping word lists). The goal is to see how interference accelerates forgetting relative to the baseline.

Expected Output: A comparative plot showing recall accuracy across conditions, with steeper forgetting in the interference condition.

Module 3: Selective Attention & Toy Attention Mechanism

Psychological Motivation: Cueing effects in attention (Posner tasks).

Programming Skills: NumPy arrays, random sampling, matrix math.

Mini-Project: Implement a toy attention mechanism that weights valid and invalid cues and see how cue validity affects accuracy.

Extension (take-home): Introduce random noise into cue signals so that sometimes cues are misleading or weak. Vary the noise level systematically to see how it changes performance.

Expected Output: A graph showing accuracy as a function of cue noise, illustrating robustness and failure points of attentional systems.

Module 4: Decision-Making and Drift Diffusion Models

Psychological Motivation: Evidence accumulation in binary choice; reaction times as markers.

Programming Skills: Random walks, loops, simulation logic.

Mini-Project: Implement a drift diffusion model; simulate RT distributions.

Module 5: Reinforcement Learning in Bandit Task

Psychological Motivation: Exploration vs. exploitation in human decision-making.

Programming Skills: Tabular Q-learning, epsilon-greedy choice.

Mini-Project: Simulate 2-armed bandit learning with rewards; visualize learning curves.

Module 6: Neural Networks & Pattern Learning

Psychological Motivation: How perceptrons classify patterns; analogy to human categorisation.

Programming Skills: PyTorch basics, feedforward nets, backprop.

Mini-Project: Train a small FFNN on MNIST digits.

Module 7: Sequence Learning with RNNs

Psychological Motivation: Working memory & sequence prediction.

Programming Skills: Recurrent nets, hidden state, training loops.

Mini-Project: Train RNN to predict toy sequences; compare to human recall.

Module 8: Visual Categorisation with CNNs

Psychological Motivation: Object recognition in vision; hierarchical filters (Hubel & Wiesel).

Programming Skills: Convolutions, pooling, CNNs in PyTorch.

Mini-Project: Train CNN on MNIST; visualise learned filters.

Module 9: Deep RL in Bandits

Psychological Motivation: Neural correlates of dopamine and learning.

Programming Skills: DQN with PyTorch, replay buffers.

Mini-Project: Train DQN in bandit; compare to tabular RL.

Module 10: Hybrid Symbolic + Neural Models

Psychological Motivation: Debates over rule-based vs. statistical cognition.

Programming Skills: Parsing CSV knowledge bases; hybrid predictions.

Mini-Project: Symbolic 'is-a' reasoning with KB; add neural predictions.

Module 11: Deep CNNs & Human Vision Confusions

Psychological Motivation: Compare human visual confusion patterns with CNNs.

Programming Skills: Training deeper CNNs, confusion matrices.

Mini-Project: Train CNN on MNIST; compare errors to toy human confusion data.

Module 12: Transformers & Psycholinguistics

Psychological Motivation: Sentence processing, garden-path sentences, surprisal theory.

Programming Skills: HuggingFace transformers; log-loss as surprisal.

Mini-Project: Compute surprisal for sentences; compare to human difficulty