Optimization & Uncertainty in Learning Report
CS7641: Machine Learning
Fall 2025

1 Assignment Weight

The assignment is worth 12.5% of the total points.

Read everything below carefully as this assignment has changed term-over-term.

2 Objective

Most machine learning projects do not fail because a model is incapable of learning; they fail because we pull the wrong levers, in the wrong order, and then mistake coincidence for causation. This assignment asks you to stop treating your network as a black box. Our goal is to develop a clear, defensible understanding of how specific training choices drive specific outcomes. That means tracing cause to effect, not merely reporting a final accuracy.

Using the two neural networks you finalized in the SL Report as fixed backbones, you will interrogate three forces that shape modern training. First, you will apply **randomized optimization** (RO) to the last one to three layers within a strict parameter budget to examine its impact on robustness, tail behavior, and local minima, if any. Second, you will examine the **mechanics of Adam** through targeted ablations on the full network, separating the contributions of plain SGD, classical momentum, Nesterov momentum, Adam with and without bias correction, the β_1=0 (RMSProp-like) variant, and AdamW. This is not a tour of optimizers for its own sake; it is an investigation into which ingredients are actually responsible for speed-to-target, stability, and generalization on *your* models. Third, you will study **regularization**, run *exclusively with standard Adam using your best Part 2 hyperparameters* to quantify how choices like L2 weight decay, early stopping, dropout, label smoothing, and modality-appropriate augmentation move the *test* metric, often by more than optimizer tweaks alone.

Every comparison is conducted under fixed, reported budgets (gradient/function evaluations and wall-clock) so that differences are causal rather than accidental. You will conclude by composing a single, coherent training recipe from the elements that proved most effective on your data, and by explicitly accepting or rejecting your hypotheses with quantitative evidence. The intended outcome is a practitioner's decision rule you can carry into research or industry: when the model stumbles, you will know which knob to turn first, and why!

You will:

- Apply randomized optimization to the last 1–3 layers of your SL Report optimized neural networks under a strict parameter cap.
- Dissect Adam via ablations on the full network (SGD, Momentum, Nesterov, Adam baseline, Adam w/o bias correction, β_1=0, AdamW) to measure speed-to-target, stability, and generalization.
- Run a targeted regularization study using standard Adam with your best Part 2 hyperparameters (e.g., L2, early stopping, dropout, label smoothing, modality-appropriate augmentation) to quantify test-set impact.
- Integrate the winning pieces into a single best training recipe and explicitly accept or reject your hypotheses with quantitative evidence.

3 Procedure

You will reuse the **same two datasets** from your SL Report. Design **exactly two** supervised learning tasks in total, **one per dataset**. Each task may be classification or regression. Keep the **SL Report network architecture fixed** as your baseline backbone; only the regularization experiments in Part 3 may introduce or remove regularization modules, and any such changes must be explicitly documented. For each dataset, maintain fixed train/validation/test splits and select appropriate metrics (e.g., accuracy/F1/AUROC for classification; MAE/MSE/R^2 for regression). Be prepared to explain *why* each dataset is interesting from an optimization and generalization perspective (e.g., class imbalance, ill-conditioning, non-smooth activations, spurious correlations).

You will systematically explore the optimization landscape and write a rigorous analysis of your findings. You do not need to implement deep learning frameworks or optimizers from scratch; however, you must design the experiments, set budgets, analyze diagnostics, and defend your conclusions. Concretely:

- You may program in any language you wish and are allowed to use any library, **as long as it was not written specifically to solve this assignment**. Code repositories, notebooks, or AutoML templates created for CS7641 assignments ("plug-and-play" solutions) are **not allowed**. With many ML-friendly languages available, I recommend Python for consistency. However, we have plenty of support for other languages.
- TAs must be able to recreate your experiments on a standard Linux machine if necessary.
- The analysis you provide in the report is paramount and the focus of the assignment.

Part 1: Randomized Optimization on the last several layers

Goal: evaluate whether search-based methods can improve the tail of your A1 network.

- **Freeze:** freeze all but the last 1–3 layers so the total trainable parameters for Randomized Optimization (RO) are $\leq \sim$50k (RO on $\gg$100k typically stalls).
- **Objective:** validation loss (include regularization terms if you are evaluating a "with regularization" condition).
- **Algorithms (all required):** Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA).
- **Design disclosures:** specify RHC restart policy and step-size schedule; SA initial temperature, decay factor, and step-size cooling; GA population size, selection, crossover, mutation rate, and whether elitism is used; define the weight-perturbation distribution/scale and its adaptation schedule.
- **Accounting:** count every objective call as a *function evaluation*; adhere to the budgets defined in the Budgets section.

Part 2: Adam ablations on the full network

Train the same backbone with the following optimizers:

- **SGD (no momentum)**
- **SGD + Momentum**
- **Nesterov momentum**
- **Adam (baseline)**
- **Adam (no bias correction)**
- **Adam with** $\beta_1 = 0$ (RMSProp-like)
- **AdamW** (decoupled weight decay)

Required analyses (per dataset):

- **Time/steps to a fixed validation-loss threshold** ℓ (define once and use consistently).
- **Sensitivity heatmaps** over (α, β_1) and (α, β_2) on coarse grids.
- **Stability across many seeds** (variance bands or median±IQR).
- **Generalization gap** (train vs. validation at budget).

Part 3: Regularization study (standard Adam only)

Constraint: run **all** regularization experiments with **standard Adam** using the **best hyperparameters from Part 2** (learning rate, β_1, β_2, ε, batch size). Do not switch to AdamW or other variants in this part.

Required techniques (do all five):

1. **L2 weight decay (coupled):** select and justify a search strategy; report the chosen setting clearly.
2. **Early stopping:** define and justify a stopping criterion (e.g., a patience rule); keep max epochs fixed so gains are not from extra training.
3. **Dropout:** insert at appropriate layers of your A1 architecture (document precisely where); select and justify dropout rates.
4. **Target smoothing/noise:** use label smoothing for classification or an appropriate noise model for regression; select and justify magnitudes.
5. **Modality-appropriate augmentation/input noise:** choose a modest, sensible policy per modality and justify why it fits the dataset.

Presentation (per dataset): Baseline Adam (no added regularization), best single regularizer (Adam), and best combination (Adam). Keep training budgets identical across these comparisons and make explicit how much regularization moved the *test* metric relative to optimizer ablations.

Presentation (per dataset):

- **Baseline:** standard Adam with no added regularization (use the Part 2 best hyperparameters).
- **Best single regularizer (Adam):** the single technique that helped most under the fixed budget.
- **Best combination (Adam):** the best combination of regularizers you found under the same budget.

Make the conclusion explicit: how much did regularization move the *test* metric relative to the optimizer ablations? Keep training budgets identical across these comparisons.

Part 4: Integrated Best Combination (Extra Credit)

Optional (up to +5 pts). If you choose to complete Part 4, you must **integrate all three components** and report how the final recipe compares to the rest of your results.

What to do (all required):

- **Optimizer:** use **standard Adam** with the **best hyperparameters from Part 2** (same batch size).
- **Regularization:** apply the **best combination from Part 3**, with exact values and layer placements.
- **RO fine-tuning:** apply your **best-performing RO algorithm from Part 1** to the **same last 1–3 layers** under the **same parameter cap** used in Part 1.

Constraints (to keep this a composition, not a new search):

- Do not introduce new optimizers or fresh wide hyperparameter sweeps.
- Limit RO fine-tuning to $\leq$**10%** of the Part 1 RO function-evaluation budget and $\leq$**5** small interaction trials (e.g., dropout rate ± 0.1).

- Keep seeds, hardware class, data splits, and batch size consistent with earlier parts.

What to report explicitly: whether RO fine-tuning on top of the best Adam + regularization **improves test metrics** beyond the best from Parts 1–3; the **compute trade-off** (test metric vs. total evaluations/time); and any stability effects you observed (e.g., variance across seeds). Hypothesis resolution: a concise paragraph that accepts/rejects your hypotheses and explains *why* the integrated recipe wins or does not (evidence from the table and curves)

4 What is required as you go through your analysis

The following applies to both datasets, for all three parts (RO, Adam ablations, Regularization), and the Extra Credit composition, if attempted.

Data & targets (per dataset)

- Declare the target (classification or regression) and justify your metrics (e.g., Accuracy/F1/AUROC; MAE/MSE/R^2). For imbalance, include PR-AUC and report the prevalence baseline; calibrate probabilities if you threshold.
- Use the **same fixed** train/validation/test splits from the SL Report; document preprocessing and prevent leakage.

Methodology & splits

- One held-out test used once at the end; tune only on train/validation (CV or hold-out).
- Set and log all random seeds; record exact splits and any deterministic flags.
- Keep the **SL Report backbone architecture fixed**; only Part 3 may add/remove regularization modules (document precisely).

Compute accounting & fairness

- **Report** gradient evaluations (SGD/Adam-family), function evaluations (RO), and wall-clock on the same hardware class.
- Respect assignment budgets; clearly mark any over-budget runs and exclude them from head-to-head tables.
- Keep batch size, hardware class, and data splits consistent across methods unless a change is essential (justify and log).

Required figures/tables (per dataset)

- **Loss vs. compute:** validation loss vs. wall-clock *and* vs. evaluations (gradient or function).
- **Stability bands (Part 2):** median ± IQR over many seeds for optimizer trajectories.
- **Sensitivity heatmaps (Part 2):** (α, β_1) and (α, β_2) for Adam variants (coarse grids).
- **RO progress (Part 1):** best-so-far objective vs. function evaluations (log-x allowed), with operator settings in the caption.
- **Regularization sweep (Part 3):** test metric across individual regularizers and the best combination, under identical training budgets.
- **Summary table:** Method, Best Val Loss, Test Metric, Time to ℓ, #Grad Evals, #Func Evals, Updates.
- **If doing Extra Credit (Part 4):** final comparison panel and a frontier plot (test metric vs. total compute).

Part-specific essentials

- **Part 1 (RO on last layers):** freeze all but the last 1–3 layers (report which; give parameter count $\leq\sim$50k). Run RHC, SA, GA. Disclose restart / temperature / schedule settings, GA population / selection / crossover / mutation / elitism, and the perturbation distribution / scale and any adaptation.
- **Part 2 (Adam ablations on full net):** run SGD (no momentum), SGD+Momentum, Nesterov, Adam (baseline), Adam without bias correction, Adam with β_1=0 (RMSProp-like), and AdamW. Report time/steps to a fixed ℓ, stability across seeds, sensitivity heatmaps, and generalization gap.
- **Part 3 (Regularization with standard Adam only):** use standard Adam with the **best** Part 2 hyperparameters; evaluate L2 weight decay, early stopping, dropout (document placements), target smoothing/noise, and modality-appropriate augmentation/input noise. Present baseline (no extra reg), best single regularizer, and best combination under identical training budgets; quantify test-set impact.
- **Part 4 (Extra Credit, integrated best):** compose best optimizer (standard Adam, Part 2), best regularization combo (Part 3), and **best RO fine-tuning** (Part 1) on the same last layers/parameter cap. Compare against the best from Parts 1–3; report compute trade-offs and resolve hypotheses.

Analysis & conclusion

- Make claims only with budget-matched evidence (include dispersion, not just means). Explain failures (divergence, plateaus) and attribute causes.
- End with a **decision rule** per dataset: when to prioritize optimizer settings vs. regularization vs. RO; state limits of transfer.

5 Experiments and Analysis

External sources & literature (required)

- Include outside sources beyond course materials, with at least **two peer-reviewed references**.
- Use these to **justify choices** (metrics for imbalance, threshold selection, calibration, pruning, kernel/activation decisions, leakage controls, spatiotemporal validation) and to **interpret results** (e.g., why PR-AUC is preferred, why pruning reduces variance).
- Acceptable styles: **MLA, APA, or IEEE**, pick one and be consistent in-text and in the bibliography.

Interpret, don't summarize

- Every figure/table needs takeaway tied to algorithm behavior & data. Avoid repeating the legend; explain what the figure/table *means*.

Analysis writeup is limited to 8 pages.

The page limit includes your citations. Citations must be in IEEE, MLA, or APA format. Anything past 8 pages will not be read. Please keep your analysis concise while still covering the requirements of the assignment. As a final check during your submission process, download the submission to double-check everything looks correct on Canvas. Try not to wait until the last minute to submit as you will only be tempting Murphy's Law.

In addition, your report must be written in LaTeX on Overleaf.

You can create an account with your Georgia Tech email (e.g., `gburdell3@gatech.edu`). When submitting your report, you are required to include a **READ-ONLY** link to the Overleaf Project. If a link is not provided in the report or Canvas submission comment, 5 points will be deducted from your score. Do not share the project directly with the Instructor or TAs via email. For a starting template, please use the IEEE Conference template.

6 Updated Datasets for Fall 2025

The following datasets are required for the Fall 2025 cohort. Each semester these datasets may change to avoid overused "toy" data. These are small- to mid-sized, real-world datasets that support rigorous analysis. Treat each dataset *separately* so you can compare algorithmic behavior across distinct domains. If access to the original download is limited, copies are posted on Canvas. If these datasets are not used, you will receive a zero for the assignment.

- **Hotel Booking Demand (H1/H2).** Kaggle Repository: *Hotel booking demand*.
  Source/paper: N. C. António, A. Almeida, and L. Nunes (2019), "Hotel booking demand datasets," *Data in Brief* 22:41–49. doi:10.1016/j.dib.2018.11.126.
  Scope & size: Bookings from a city hotel (H2) and a resort hotel (H1) in Portugal; ∼119,000 rows and ∼32 features.
  Suggested targets & tasks:
  - **Classification**: `is_canceled` (cancellation prediction); `is_repeated_guest`; other classification via `assigned_room_type` vs. `reserved_room_type` (binary "upgraded?" or multiclass).
  - **Regression**: `adr` (average daily rate) and `lead_time`. Optional derived target: total nights = `stays_in_week_nights` + `stays_in_weekend_nights`.
- **US Accidents (since 2016).** Kaggle Repository: *US Accidents (since 2016)*.
  About: Large, countrywide traffic accident records from 49 U.S. states, enriched with weather/points-of-interest and temporal context. Licensed CC BY-NC-SA 4.0.
  Size/features: Millions of rows (versioned) with ∼47 attributes (timestamps, geolocation, distance, weather, road features, day/night, etc.).
  Suggested targets & tasks:
  - **Classification**: `Severity` (ordinal 1–4). You may treat it as multiclass classification; if you choose ordinal-aware methods, justify briefly.
  - **Regression**: incident duration (derive: `End_Time - Start_Time`) or `Distance(mi)` at onset.

Scale requirement (read this). The **US Accidents** dataset is intentionally large. *Prototype small, submit large.* It is fine to start with a stratified subset for EDA and coarse tuning (e.g., 5–20% of rows), but your **final experiments, figures, and cross-model comparisons must be run on at least 80% of the available rows after cleaning and leakage controls**, with the clear goal of using 100%. You must **report exact row counts and percentages** at each stage (raw → cleaned → train/test). Any subsampling must be *reproducible* (fixed seed) and *stratified* on the target (and, for US Accidents, respectful of time/geography to avoid spatiotemporal leakage). **If you use less than 80% in any final experiment, you must include a brief *Compute Justification*** describing your hardware (CPU/GPU/RAM), observed runtime/memory limits, and mitigation steps attempted (e.g., efficient dtypes, chunked/streaming pipelines, sparse/ordinal encodings, linear/scalable baselines, or use of cluster/cloud resources). **If we find you used fewer than 1,000,000 rows (i.e., 15% of a typical current release), we will treat that as cherry-picking and deduct -50% from the report score.**

7 Acceptable Libraries

Here are **examples** of acceptable libraries. You may use others if they meet the assignment conditions (no code written specifically to solve this assignment; reproducible on a standard Linux machine; budgets and methods disclosed). **AutoML services/templates are not allowed.** Hyperparameter tools may be used only as *orchestration/logging* under the stated budget caps (disable automatic pruning/early-stopping that would violate fairness), and you must report the true number of function/gradient evaluations.

Core ML (Python)

- **scikit-learn** — pipelines, CV, metrics, calibration, model selection.
- **imbalanced-learn** — resampling, class weights for imbalanced targets.
- **scikit-learn-extra** — additional algorithms/utilities when relevant.
- **SciPy** (`scipy.stats`, `scipy.sparse`) — distributions, utilities for preprocessing and analysis.

Deep Learning (for NNs)

- **PyTorch** (*Lightning optional for training loop hygiene*); **torchvision**, **torchtext** for datasets/augmentations.
- **TensorFlow 2 / Keras**.
- **JAX + Flax/Equinox** (*advanced users*); **Optax** for optimizer variants.
- **torchmetrics / torchinfo** — metrics and model summaries.

*Optimizer rule for this assignment: use only the optimizers specified in Procedure Part 2 (SGD, Momentum, Nesterov, Adam variants, AdamW). In Part 3, use **standard Adam** with your best Part 2 hyperparameters.*

Randomized Optimization (RO)

- **mlrose-hiive** (Python) — RHC, SA, GA; `https://pypi.org/project/mlrose-hiive/`
- **ABAGAIL** (Java) — RHC/SA/GA; `https://github.com/pushkar/ABAGAIL`
- **pyperch** (Python) — search utilities; `https://github.com/jlm429/pyperch`
- **DEAP** (Python) — evolutionary algorithms (GA tooling).
- **Nevergrad** (Python) — derivative-free optimization toolbox (use only for GA/RHC/SA-like operators you disclose).

Note: stick to the required algorithms (RHC, SA, GA). If you use these libraries, document operator choices (restarts, temperature/cooling, population/selection/crossover/mutation/elitism) and count every objective call as a function evaluation.

Data & Visualization

- **pandas / polars** — tabular wrangling; **NumPy** — arrays/linear algebra.
- **matplotlib / seaborn / plotly / altair** — plotting; **yellowbrick** — diagnostic viz for ML.
- **ggplot2** (R) — if you prefer R for figures (ensure reproducibility on Linux).
- **albumentations** (images), lightweight **imgaug** or torchvision transforms — modality-appropriate data augmentation.

Experiment tracking & configuration (optional)

- **Weights & Biases / MLflow** — experiment logging. Provide offline exports or static logs; do not require graders to use paid SaaS.
- **Hydra / OmegaConf** — configuration management; helps make runs reproducible.
- **Optuna / Ray Tune** — *allowed for orchestration only* within the assignment's hyperparameter trial caps; disable pruning/schedulers that alter fairness; report true eval counts.

8 Submission Details

All scored assignments are due by the time and date indicated. Here "time and date" means Eastern Time (ET). **Canvas displays times in your local time zone if your profile is set correctly; grading deadlines are enforced in ET.** Please double check your settings and assignments for the exact due dates to mark your calendars appropriately.

All assignments will be due at 11:59:00 PM ET on the final Sunday of the unit. Since we will not be looking at the assignments until morning, submissions received by **7:59:00 AM ET** the next morning are accepted **without penalty**.

Late penalty. After the grace window, the score is reduced **20 points per calendar day**. (Example: submitted Monday 9:00 AM ET → −20; Tuesday 9:00 AM ET → −40.) Treat **11:59 PM ET** as your real deadline; allow time for upload and verification on Canvas.

You will submit **two PDFs**:

1. **OL_Report_{GTusername}.pdf** — your report. Your document must be written in LaTeX using Overleaf.
2. **DOCSTRING_{GTusername}.pdf** — includes:
   (a) A **READ-ONLY** link to your Overleaf project.
   (b) A **GitHub commit hash** (single SHA) from the final push of your report/code.
   (c) **Exact run instructions** to reproduce results on a standard Linux machine (environment, commands, data paths/URLs, and seeds).

- Include the READ-ONLY Overleaf link in the report or Canvas submission comment. **Do not send email invitations.**
- Use the **GT Enterprise GitHub** for all course-related code. This must be the actual commit hash, not a general link.
- Provide sufficient instructions and pointers to retrieve code and data (URLs where appropriate). You need not vendor entire libraries; ensure we can reproduce your results on a standard Linux machine.

For a starting template, we recommend using the IEEE Conference template[1].

Only your **latest submission** will be graded. Please double-check that **both PDFs** are submitted.

Formal writing requirement (bullets). Your *report* is a formal technical paper. Bullet writing is not formal writing and will be treated as a draft. Use paragraphs and integrated prose. In your final report, any list environment (`itemize`, `enumerate`, or `description`) with more than **two** items in any section will incur a **50% deduction on the overall report score**. Keep enumerations to ≤ 2 items or convert them to narrative prose.

9 Feedback Requests

When your assignment is scored, you will receive feedback explaining your errors and successes in some level of detail. This feedback is for your benefit, both on this assignment and for future assignments. It is considered a part of your learning goal to internalize this feedback. We strive to give meaningful feedback with a human interaction at scale. We have a multitude of mechanisms behind the scenes to ensure grading consistency with meaningful feedback. This can be difficult; however, sometimes feedback isn't as clear as you need. If you are confused by a piece of feedback, please start a private thread on Ed and we will help clarify. When you make a private request, please use the 'Feedback Request - OL' tag on Ed for this cycle.

Change for Fall 2025. Reviewer Response. In an effort to learn and grow assignment-to-assignment, we will provide a mechanism to edit and respond to your feedback. We will call this the *Reviewer Response.* You will have one week from the assignment grade being posted to edit and provide a two-page maximum response with both edits made and reviewer feedback. You will need to reasonably respond and edit your initial paper submission to improve your paper in good faith. This requires you to incorporate all the missed items from your feedback, rather than just a written rebuttal or disagreement. If all items are not met, the reviewer response will not be complete and no additional points will be awarded. Both the initial submission, revised submission, and two-page response will be needed for a proper Reviewer Response. If satisfied, you will receive half of the missed points back for the assignment. For example, if the initial grade was a 70/100, if everything is satisfied for the Reviewer Response, there will be 15 points added resulting in an 85/100. Further examples will be provided when the assignment grades are posted.

[1] `https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn`

10 Plagiarism and Proper Citation

The easiest way to fail this class is to plagiarize. **Using the analysis, code, or graphs of others in this class is considered plagiarism**. The assignments are designed to force you to immerse yourself in the empirical and engineering side of ML that one must master to be a viable practitioner and researcher. It is important that you understand why your algorithms work and how they are affected by your choices in data and hyperparameters. The phrase "as long as you participate in this journey of exploring, tuning, and analyzing" is key. We take this very seriously and you should too.

What is plagiarism?

If you copy any amount of text from other students, websites, or any other source without proper attribution, that is plagiarism. The most common form of plagiarism is copying definitions or explanations from Wikipedia or similar websites. We use an anti-cheat tool to find out which parts of the assignments are your own and there is a near 100 percent chance we will find out if you copy or paraphrase text or plots from online articles, assignments of other students (even across sections and previous courses), or website repositories.

What does it mean to be original?

In this course, we care very much about your analysis. It must be original. Original here means two things: 1) the text of the written report must be your own and 2) the exploration that leads to your analysis must be your own.

It is well known that for this course we do not care about code. We are not interested in your working out the edge cases in k-NN, or proving your skills with Python. While there is some value in implementing algorithms yourselves in general, here we are interested in your grokking the practice of ML itself. That practice is about the interaction of algorithms with data. As such, the vast majority of what you're going to learn in order to master the empirical practice of ML flows from doing your own analysis of the data, hyperparameters, and so on; hence, you are allowed to use ML code from libraries but are not allowed to use code written explicitly for this course, particularly those parts of code that automate exploration. You will be tempted to just run said code that has already been overfit to the specific datasets used by that code and will therefore learn very little.

Information on citations:

If you refer to information from a third-party source or paraphrase another author, you need to cite them right where you do so and provide a reference at the end of the document [Col]. Furthermore, "if you use an author's specific word or words, you must place those words within quotation marks and you must credit the source." [Wis]. It is good style to use quotations sparingly. Obviously, you cannot quote other people's assignments and assume that is acceptable. Citing is not a get-out-of-jail-free card: you cannot directly copy text flippantly, cite it all, and then claim it's not plagiarism just because you cited it. Too many quotes of more than, say, two sentences will be considered plagiarism and a terminal lack of academic originality.

All citations need to be in IEEE, MLA, or APA format.

Your README file will include pointers to any code and libraries you used.

If we catch you...

We report all suspected cases of plagiarism to the Office of Student Integrity. Students who are under investigation are not allowed to drop from the course in question, and the consequences can be severe, ranging from a lowered grade to expulsion from the program.

Update for Fall 2025: Use of LLMs and Generative AI

We treat AI based assistance the same way we treat collaboration with people. You may discuss ideas and seek help from classmates, colleagues, and AI tools, but all submitted work must be your own. For this course a large language model is any model with more than one billion parameters. These tools can increase productivity, but they are aids, not substitutes for your skills. The goal of reports is synthesis of analysis, not merely getting an algorithm to run. Even if you locate code elsewhere or generate code with an AI tool, you must apply it thoughtfully to your data and write the analysis yourself.

Every submission must include an AI Use Statement. List the tools used and what they assisted with and confirm that you reviewed and understood all assisted content. If an editor or platform surfaces suggestions, you may keep them only with disclosure and verification. You are not required to disable assistants. Be aware that coding editors often include assistants by default. Google Colab and Visual Studio Code commonly surface them. Also check any writing or grammar platforms you use. Overleaf through Georgia Tech is acceptable.

Allowed with disclosure. Brainstorming, outlining, grammar and clarity edits, code generation, code refactoring, and debugging.

Not allowed. Submitting AI written analysis, conclusions, or figures as your own. Fabricating results or citations. Paraphrasing AI or prior work to evade checks. Uploading course materials or private data to public tools.

Example Statement at the very end of the report before References. "AI Use Statement. I used ChatGPT and Visual Studio Code Copilot to brainstorm and outline sections of the report, generate and refactor small code snippets, debug a pandas indexing issue, and edit grammar and clarity throughout. I reviewed, verified, and understand all assisted content."

Verification will rely on provenance and reproducibility. We will not ask for live coding. We will review platform histories, including Overleaf revision timelines and copy and paste patterns, and Git commit and branch history. We may request your repository with full history, Overleaf project history, notebooks, logs, and intermediate outputs, and we may run reproducibility checks. Provide run instructions, environment details, random seeds, and the scripts used to generate figures and tables. Similarity and AI indicators, including Turnitin, are corroborating signals only and are not used in isolation.

If we suspect a violation, we will document evidence, notify you, and request artifacts. Cases may be referred to the Office of Student Integrity (OSI). Undisclosed or prohibited use is academic misconduct. This policy may be updated during the term. When in doubt, always use proper citations, and ask clarifying questions on Ed.

How to attribute & cite

Hotel Booking Demand: Data: Hotel booking demand (H1/H2), António, Almeida & Nunes (2019), *Data in Brief* 22:41–49, doi:10.1016/j.dib.2018.11.126.
US Accidents: Data: US Accidents (since 2016), Sobhan Moosavi et al., CC BY-NC-SA 4.0, retrieved from Kaggle US Accidents.

How to cite peer-reviewed sources in-text.

You may use **MLA**, **APA**, or **IEEE**; pick one style and stay consistent across the paper (in-text and references). Examples using the hotel paper:

- **APA**: "... (António, Almeida, & Nunes, 2019)." or "António et al. (2019) ..."
- **MLA**: "... (António, Almeida, and Nunes 2019)." or "António, Almeida, and Nunes argue ..."
- **IEEE**: "... [AAN19]." (numbered, bracketed citations like [1]; reference list ordered by first appearance)

Include the full reference entry in your bibliography. In LaTeX, use BibTeX/BibLaTeX with an appropriate style (e.g., `style=apa` or `style=ieee`). *Tip:* In Google Scholar, click "Cite" → "BibTeX" to copy a starter entry, then verify authors, capitalization, and DOI.

11 Frequently Asked Questions (FAQ)

The assignment is open-ended, but expectations are concrete. Use this FAQ to avoid common pitfalls.

Scope & setup

Q: Can I change datasets or the backbone architecture from the SL Report?
A: No for both. Use the **same two datasets** and keep the **SL backbone fixed**. Only *Part 3* may introduce/remove regularization modules (e.g., dropout, normalization) and you must document the exact changes.

Q: Do I really need exactly one task per dataset?
A: Yes—two tasks total (one per dataset), each either classification or regression. Pick metrics that fit each task (e.g., AUROC/PR-AUC for imbalanced classification; MAE/MSE/R^2 for regression).

Q: What counts as data leakage?
A: Any operation using information outside the training split (e.g., normalizing with stats computed on test/val; including post-outcome columns). Fit all preprocessing on training only; apply to val/test.

Q: How many seeds?
A: Use **enough seeds to demonstrate variance and stability**. As a rule of thumb, **3–5 seeds at minimum**; more are welcome if they add insight. Fix and log `python`/`numpy`/framework seeds and any deterministic flags. Report **median and IQR** on required curves/tables; if you run ≥ 5 seeds, also report **mean $\pm$ 95% CI** where appropriate.

Budgets, accounting, and fairness

Q: What is a "gradient evaluation" vs. a "function evaluation"?
A: A *gradient evaluation* is one backward pass (one parameter update for mini-batch methods). A *function evaluation* is one objective computation on the **entire validation split** (accumulated over microbatches still counts as one).

Q: Do I include wall-clock time?
A: Yes. Report wall-clock on the **same hardware class**. If you change hardware mid-way, clearly separate results (not recommended).

Q: Why not report epochs?
A: Epochs are not comparable across batch sizes and datasets. Use evaluations (grad/func) and wall-clock for fair cross-method comparisons.

Q: Can I exceed the budgets?
A: You can *run* over, but those runs must be clearly marked and **excluded** from head-to-head tables and conclusions.

Q: Can I use hyperparameter tools (Optuna, Ray Tune, W&B sweeps)?
A: Only as orchestration/logging within the trial caps; **disable** pruning/ASHA-style schedulers that silently change budgets. Report the true number of evaluations.

Part 1 (Randomized Optimization) specifics

Q: Can I do RO on the entire network?
A: No. Restrict to the **last 1–3 layers** with $\leq\sim$**50k** trainable parameters (report the exact count). If your tail exceeds this, reduce k or restructure the head (without changing earlier layers).

Q: Which RO algorithms are required?
A: **RHC**, **SA**, and **GA**. You must disclose: (i) RHC restart policy and step-size schedule; (ii) SA initial temperature, cooling schedule, and any step-size cooling; (iii) GA population size, selection method, crossover, mutation rate, and whether you use elitism; and (iv) the perturbation distribution/scale and how it adapts.

Q: What is the RO objective?
A: The **validation loss** computed with the model in `eval` mode (dropout off; batch norm uses stored running stats). If you evaluate a "with regularization" condition, include those terms consistently in the loss. Count **every** objective computation as one function evaluation.

Q: Can I use a mini-batch or a subsampled validation set to speed RO?
A: Default is the **full validation split per evaluation**. If you use a fixed subsample for candidate *screening*, disclose it, fix its seed, and re-rank finalists on the *full* validation set at regular intervals. Always report results based on the full validation objective.

Q: How should I handle stochastic layers and augmentation during RO?
A: Use `model.eval()` for every evaluation: disable dropout and any stochastic augmentation. For batch norm, do *not* update running stats during RO; use stored statistics to keep the objective deterministic.

Q: My RO is slow. What can I do without violating rules?
A: Reduce the number of unfrozen layers, shrink the tail (e.g., narrower last MLP block), or use coarser operator settings (larger steps, smaller populations) while **keeping budgets comparable**. Do *not* change datasets, splits, or the backbone.

Q: Can I mix in gradient steps during RO?
A: No. RO in Part 1 is **zero-order**. Do not interleave gradient updates. (Gradient-based methods belong to Part 2.)

Q: Are bounds, clipping, or constraints allowed?
A: Yes, but **declare them** and keep them **constant across algorithms**. If you use weight clipping, the same bounds must apply to RHC/SA/GA.

Q: What if a candidate yields NaNs or infs?
A: Count it as a **function evaluation**, discard the candidate, and record the failure rate. If failures cluster, reduce step sizes or add bounds.

Q: Can I cache evaluations or evaluate candidates in parallel?
A: Yes to both. Caching is recommended (identical candidates occur in GA). Parallelization does not change how you **count** function evaluations.

Q: Warm start—allowed or not?
A: If you choose to warm-start the tail (e.g., from the SL backbone trained with Adam), disclose it and apply the *same* initialization policy across RHC/SA/GA. Do not spend additional gradient budget inside Part 1.

Q: How do I stop an RO run?
A: Use a **budget-based** stop (max evaluations) and optionally a **plateau rule** (no improvement after m evaluations). If you use a plateau rule, it must be **identical across algorithms** and **still** count the attempted evaluations.

Caveats & landmines:

- **Non-deterministic objective:** forgetting `eval` mode (dropout on, BN updating) makes results irreproducible.
- **Unequal budgets:** comparing RHC at 50k evals vs. GA at 10k is invalid; keep function-eval budgets equal.
- **Unreported operator details:** "we used GA" without population/selection/crossover/mutation/elitism is not acceptable.
- **Dimension blow-up:** tuning $\gg$100k parameters with RO will stall; respect the $\leq\sim$50k cap and report the count.
- **Changing loss mid-run:** adding/removing regularization terms or augmentation during the RO run invalidates comparisons.
- **Hidden gradient use:** any gradient steps, schedulers, or optimizer calls inside Part 1 violate the zero-order requirement.
- **Different initializations per algorithm:** seeds or init states must be matched (or explicitly compared) to avoid confounding.
- **Selective reporting:** show full progress curves (best-so-far objective vs. evaluations) and failure rates, not just final winners.

Part 2 (Adam ablations) specifics

Q: Which optimizers do I need to run?
A: **SGD (no momentum)**, **SGD+Momentum**, **Nesterov**, **Adam (baseline)**, **Adam (no bias correction)**, **Adam with β_1=0 (RMSProp-like)**, and **AdamW**. Part 3 depends on your *best standard Adam* hyperparameters, so you **must** include standard Adam in Part 2.

Q: How do I pick ℓ for "time/steps to threshold"?
A: Choose a validation-loss threshold that is **reachable** by most methods under budget on each dataset. Fix it *once* and use it for all Part 2 comparisons; state it clearly in the paper. If a method fails to reach ℓ within budget, record it as "> budget" and include it in plots/tables (do not drop it).

Q: What do the heatmaps show and how do I build them fairly?
A: Coarse grids over (α, β_1) and (α, β_2) for Adam-family methods to reveal stability regions and brittle zones. Use the **same data split and training budget per grid point**. Keep batch size and augmentation fixed. Report any divergent runs explicitly.

Q: What does "Adam (no bias correction)" mean in practice?
A: Use the Adam update but **disable the bias-correction terms** for m_t and v_t (i.e., use m_t and v_t directly, not $\hat{m}_t$ or $\hat{v}_t$). Many libraries do not expose this flag; if you re-implement, **verify** equivalence on a toy model before running full experiments and document how you disabled the correction. It can always help to have something you know you can debug.

Q: "Adam with β_1=0 (RMSProp-like)", is that exactly RMSProp?
A: Not necessarily. Setting β_1=0 removes momentum, leaving per-parameter scaling via v_t; this behaves *similarly* to RMSProp under some settings, but details (e.g., ε, decay form) can differ. Report your ε and decay parameters.

Q: How should I choose learning rates (and momentum/betas) across optimizers?
A: Use **small, budgeted sweeps per optimizer** (e.g., a short grid or line search) under the same gradient-eval cap. Record the best setting used for head-to-head. Do not reuse one optimizer's best LR for another. For SGD+Momentum/Nesterov, sweep both LR and momentum; for Adam-family, sweep LR with small grids over (β_1, β_2) as required by the heatmaps.

Q: Can I use LR schedules or warm-up?
A: Yes, but keep the **schedule form consistent** across optimizers (e.g., constant LR for all, or the same cosine/step schedule for all). If you use warm-up, apply a comparable warm-up across methods and include it in your gradient-eval counts. State schedules explicitly.

Q: What about weight decay in Part 2?
A: **Keep regularization minimal and consistent** in Part 2. If you include L2 with Adam, that is *coupled* weight decay; **AdamW** implements *decoupled* decay and is already included as a separate ablation. Do not accidentally use decoupled decay in "Adam (baseline)."

Q: Batch size, accumulation, and mixed precision—any rules?
A: Keep **batch size fixed** across ablations unless a change is essential to avoid OOM; justify and log any deviation. If you use gradient accumulation, count the underlying micro-batch backward passes as gradient evaluations. Mixed precision is fine; keep hardware class constant and report it.

Q: What stability metrics should I report?
A: At minimum, validation-loss trajectories with **median and IQR** over 3–5 seeds, and time/steps-to-ℓ distributions. If you run ≥ 5 seeds, also report **mean $\pm$ 95% CI** for the final test metric. Log divergence/NaN rates where relevant.

Q: Can I early-stop in Part 2?
A: Use early stopping **only** for the purpose of measuring *time/steps to reach* ℓ. Do **not** change max-epoch budgets across optimizers; early stopping must not create hidden compute advantages. Report whether/when early stopping triggered.

Q: How do I ensure fair initialization?
A: Use the **same initialization and seeds** for all optimizers on a given dataset/architecture. If an optimizer needs a different LR warm-up to avoid instability, keep the initial weights identical and report the change explicitly.

Q: What if an optimizer diverges?
A: Count all attempted steps toward the gradient-eval budget; mark the run as divergent and include it in stability summaries. Reduce LR or adjust β values only within your allowed sweep, not after the fact.

Q: What belongs in captions/tables for Part 2?
A: For each plot/table, include: optimizer name; LR schedule; key hyperparameters used (LR, momentum or $\beta_1, \beta_2, \varepsilon$); batch size; seeds; total gradient evaluations; wall-clock; and whether bias correction was on/off.

Caveats & landmines (we dock for these):

- **Inconsistent batch sizes or augmentation** across optimizers without a clear justification.
- **Mixed schedules** (e.g., cosine for Adam, constant for SGD) without making them consistent or explaining the choice.
- **Calling Adam "baseline" but using decoupled decay** (that is AdamW behavior).
- **Using different initial weights** across optimizers on the same dataset/architecture.
- **Cherry-picking a single best seed** instead of reporting dispersion (IQR/CI).
- **Not counting warm-up or accumulation steps** in gradient-eval totals.
- **Dropping failed/divergent runs** from stability analyses.
- **Changing ℓ mid-way** or defining it differently per optimizer.

Part 3 (Regularization on standard Adam) specifics

Q: Can I switch to AdamW or change the optimizer in Part 3?
A: No. Part 3 **must** use **standard Adam** with your **best Part 2 hyperparameters** (learning rate, β_1, β_2, ε, batch size). The goal is to *isolate* regularization effects; changing the optimizer would confound attribution.

Q: Which regularizers are required?
A: L2 weight decay (coupled), early stopping, dropout (document placements), target smoothing/noise (label smoothing for classification; an appropriate noise model for regression), and modality-appropriate augmentation/input noise. You **choose and justify** the settings—no prescribed grids.

Q: Do I include a "best single" and a "best combo"?
A: Yes, on **each dataset**. Present (i) baseline Adam (no extra reg), (ii) best single regularizer, and (iii) best combination, all under **identical training budgets**. Quantify gains on the *test* metric with dispersion across seeds.

Q: How do I avoid over-crediting early stopping?
A: Keep **max epochs fixed** across conditions; disclose your patience and `min_delta`. Early stopping should not silently change compute budgets elsewhere. Report whether/when stopping triggered.

Q: How should I apply L2 (coupled weight decay) fairly?
A: Keep L2 as a **coupled** penalty in the Adam loss (this is *not* AdamW). Apply it consistently across parameter groups; if you exclude biases or norm parameters, say so and keep that rule constant across conditions. **Do not** switch to decoupled decay for "Adam baseline."

Q: Can I retune Adam's LR when I add regularization?
A: No. Part 3 fixes the **Adam hyperparameters** to your best from Part 2. If a regularizer destabilizes training, **adjust the regularizer** (e.g., reduce dropout rate) rather than retuning the optimizer.

Q: Train/eval modes and stochasticity?
A: Use `model.train()` during training and `model.eval()` for validation/test. Disable stochastic augmentation on val/test; dropout should be **off** in eval. Keep BatchNorm/LayerNorm behavior standard (BN uses running stats in eval).

Q: What should I log to make Part 3 credible?
A: For each regularizer: (i) exact setting(s) and placements, (ii) loss components used during training/validation, (iii) whether early stopping fired and at which epoch, and (iv) the compute accounting (updates/grad evals, wall-clock). Include seed dispersion (median $\pm$ IQR).

Q: How do I compare "best single" vs "best combo" fairly?
A: Use the **same training budget**, same seeds (or matched seed sets), same data splits, same batch size, and the same Adam hyperparameters across both; only the regularizer choices differ.

Q: What if a regularizer helps validation but hurts test?
A: Report it. Discuss overfitting to the validation protocol or policy mismatch (e.g., augmentation too strong). Do not hide regressions.

Q: Can I stack normalization changes with dropout?
A: Yes, but changes to normalization **are regularization choices** and must be confined to Part 3 and documented (on/off; layer type; placement). Be careful with BN statistics when adding/removing dropout around BN layers.

Presentation checklist (per dataset, Part 3):

- Baseline Adam (no extra regularization) vs. best single vs. best combination—**budget-matched**, with mean/median and dispersion over seeds.
- A bar/line figure showing **test** metric across regularizers and the combo; captions must list key settings (e.g., dropout location/rate; whether biases/norms were excluded from L2; augmentation policy summary; patience/`min_delta` for early stopping).
- A short paragraph attributing the observed gains to mechanisms (e.g., "label smoothing reduced overconfidence on minority classes," "dropout improved stability under small data," "L2 reduced effective capacity and helped generalization").

Caveats & landmines (we dock for these):

- **Changing Adam hyperparameters** in Part 3 (LR, betas, ε, batch size) instead of adjusting the

regularizer.

- **Accidentally using AdamW** (decoupled decay) while calling it "Adam baseline," or mixing coupled/decoupled decay across conditions.
- **Augmentation leakage** (applying it to validation/test) or using label-altering transforms without justification.
- **Unequal budgets** between "best single" and "best combo," or silently changing the number of updates/epochs.
- **Omitting placements/values** (e.g., not saying where dropout was inserted; not stating L2 exclusions).
- **Recomputing validation with dropout on** or with BN stats updating; validation/eval must be deterministic.
- **Cherry-picking** best seeds or dropping runs where the regularizer underperforms.

Part 4 (Integrated Best Combination) specifics

Q: What is the goal of Part 4?
A: To *compose* your best choices from Parts 1–3 into a single training recipe and show, with budget-aware evidence, whether the integrated approach outperforms the best individual pieces.

Q: What exactly must be included?
A: All three components are **required**: (1) **standard Adam** with your **best Part 2 hyperparameters** (same batch size), (2) the **best regularization combination** from Part 3 (exact values and placements), and (3) **RO fine-tuning** on the **same last 1–3 layers** and **same parameter cap** as Part 1 using your **best-performing RO algorithm**.

Q: When do I apply RO fine-tuning in Part 4?
A: Train with Adam + best regularizers to your standard budget, *then* freeze all earlier layers and run RO on the designated last layers. Evaluate on the **full validation split** each function evaluation (as in Part 1).

Q: Can I retune Adam or the regularizers in Part 4?
A: No. Part 4 is a **composition**, not a new search. You may make at most **5 small interaction trials** (e.g., dropout ± 0.1) and must keep Adam hyperparameters **exactly** as in Part 2 and regularizer choices **exactly** as in Part 3.

Q: How much RO budget can I use in Part 4?
A: At most **10%** of your Part 1 RO function-evaluation budget (per dataset). Count **every** validation objective as one function evaluation.

Q: Which checkpoint should I start RO from?
A: Use the **best validation** snapshot from Adam + regularizers (Part 3). State your selection rule (e.g., "lowest val loss"). Do not cherry-pick different starting points for different RO algorithms.

Q: How do I account for compute in Part 4?
A: Report (i) the **additional RO evals** used in Part 4 and (ii) the **total compute** for each pipeline you compare. The frontier plot (see below) must use a consistent "total compute" definition so readers see the trade-off.

Q: What if RO fine-tuning *hurts* validation or test?
A: Report it. Discuss why (e.g., overfitting the tail; too-large perturbations; insufficient plateau rule). You may abort an RO run via a **plateau stopping rule** that is **identical** to the one used in Part 1, but attempted evaluations must still be counted.

Q: Can I change which last layers are tuned in Part 4?
A: No. Use the **same** layers and parameter cap you validated in Part 1. Otherwise you are conducting a new search.

Q: What stability reporting is expected in Part 4?
A: Use the **same seed set** as in Parts 1–3 (3–5 seeds). Report median and IQR for the test metric; include mean $\pm$ 95% CI if you have ≥ 5 seeds. Consider a **paired improvement** plot (per-seed deltas) to show whether gains are consistent.

Q: How do I attribute causality in Part 4?
A: Provide a short mechanism-level explanation (e.g., "dropout reduced overfitting; L2 smoothed the land-

scape; RO escaped a flat tail on the classifier layer"). Tie claims to observed diagnostics (RO progress curve, regularization sweep, optimizer stability).

Caveats & landmines (we dock for these):

- **Changing Adam hyperparameters or regularizer choices** in Part 4 (must be identical to your Part 2/Part 3 winners).
- **Using different last layers or parameter caps** for RO than in Part 1.
- **Exceeding the 10% RO budget** or not counting all evaluations.
- **Cherry-picking starting checkpoints** or seeds per pipeline.
- **Reporting only the best seed** instead of dispersion and paired deltas.
- **Inconsistent preprocessing or augmentation** relative to Part 3.

Reproducibility, tooling, and LLMs

Q: Can I use experiment trackers (W&B/MLflow)?
A: Yes, but provide offline exports or static logs. Do not require graders to use paid SaaS. All runs must list evaluation counts and hardware.

Q: Can I use LLMs?
A: You may use them for *code suggestions/debugging/figure code* only if you **disclose and cite** model/provider and date, and describe usage. Do **not** use LLMs to write report prose.

Q: What if I find a bug late?
A: Fix it, re-run the affected comparisons under the same budgets, and document the correction and impact. Do not quietly replace plots.

Making and defending claims

Q: What constitutes sufficient evidence to claim "X is better"?
A: Budget-matched numbers, dispersion across seeds, and (when applicable) confidence intervals—**plus** a causal story tied to mechanisms (e.g., "bias correction sped early progress," "label smoothing improved calibration," "RO escaped a flat tail in the last layer").

Q: What does a good conclusion look like?
A: A decision rule per dataset (when to favor optimizer tweaks vs. regularization vs. RO), an integrated best recipe (if you did Part 4), and the limits of transfer (when your findings likely will/won't generalize).

12 Version Control

- v4.0 – 09/24/2025 – TJL updated Section 5 with 8-page limit and latex requirement. These are the exact same requirements from the SL Report.
- v3.0 – 09/03/2025 – TJL updated dataset usage and bullet point warnings.
- v2.0 - 08/20/2025 - TJL updated a majority of the writing. Reviewed the requirements and added FAQ.
- v1.0 - 08/17/2025 - TJL created and wrote the OL Report.

References

[AAN19] Nuno C. António, Ana Almeida, and Luis Nunes. "Hotel booking demand datasets". In: *Data in Brief* 22 (2019), pp. 41–49. DOI: 10.1016/j.dib.2018.11.126.

[Col] Williams College. *Citing Your Sources: Citing Basics*. URL: https://libguides.williams.edu/citing.

[Wis] University of Wisconsin - Madison. *Quoting and Paraphrasing*. URL: https://writing.wisc.edu/handbook/assignments/quotingsources.