

(0) Your own project from outside of EECS – only for PhD students outside EECS (and their team members)

Groups that choose this option will need to bring deep learning techniques to bear on new applications and data that are brought to the class by students from other departments. In particular, you will need to describe a problem in a new domain to which deep learning techniques from this class can be applied, and try doing it! This can include “modernization” wherein newer ideas (e.g. Muon, state-space models, hybrid attention, etc.) are incorporated into existing DL methods in the field.

Important Note - This project option **must** be led by a graduate student (with a **faculty** adviser) outside of EECS who takes full responsibility for disciplinary expertise as well as providing both data and anything else required to work on it. The 182/282A course staff can give comments and point to deep learning directions, but have no domain expertise in your domain or knowledge of your existing techniques. We encourage graduate students with such projects to propose to use EdStem to help recruit interested students in the class to their project team, including undergraduates. Students from the EECS department are allowed to be on such teams as long as the project is led by a graduate student outside of EECS.

Project Proposal

- Please write a **3-4** page proposal on your project idea; including a high-level overview of the problem domain, what are the existing/traditional non-deep-learning approaches to solving the specific problem, what your investigation will be, code bases you'll build upon, your training/test data, and your resource budget — an awareness of how much compute power you can muster and use is very important. Note, there is no limit here on compute and if the graduate student has access to additional compute resources you are free to use them. This must include the following sections
- Please include all the group members in your proposal, but only one person needs to submit the proposal. Typically this will be the non-EECS graduate student leading the team.

Final Report Requirements:

- Must have a coherent story that clearly lays out the question being investigated, cites relevant literature while being a self-contained treatment, and shows clear and systematic analysis/exploration.
- For example, you really need to provide loss curves for training and validation sets and give comparisons as needed to tell your story.
- Provide a link to your github repo with all the code used for this project as well as trained model checkpoints sufficient to efficiently replicate all plots and tables in your paper.
 - i) If you are using any data that can't be shared, it is your responsibility to argue explicitly why what you are providing is enough for the peer reviewers to actually check and comment on your work.

(1) Theoretical/empirical study of optimizers in deep learning and hyperparameter transfer

There has been a contemporary resurgence in the interest in optimizers for deep learning with the spectacular success and adoption of the Shampoo/Muon families. Read the September 2025 paper "[Fantastic Pretraining Optimizers and Where to Find Them](#)" by Wen, et. al., look at the references and related literature and propose and attack a problem in this space. The beautiful recent [Thinking Machines blog post on Manifold Muon](#) by Jeremy Bernstein is another good starting point to search for problems of interest. The muP related space also remains interesting, where we'd like to call particular attention to recent work arguing that Standard Parameterization (SP) works far better than one might expect given the muP arguments, how muP arguments/scaling don't properly deal with modern state-space models, the idea of Telescoping and partial-shot optimization (experiments at different smaller scales) instead of zero-shot as well as work establishing results for mixture-of-experts models as well as other architectures. Explorations of line-search based ideas (see the last example in the 2024 [Anthology paper](#) by Bernstein and Newhouse and go digging from there) and how to eliminate the learning-rate entirely as a hyperparameter that needs separate tuning are also welcome. In addition, because of the importance of wall-clock time, explorations of low-level CUDA optimizations and interactions with finite-precision arithmetic are also welcome. See also this [comparison paper](#) or the following two works on unit-mup: ([Blake et. al, "u- \$\mu\$ P: The Unit-Scaled Maximal Update Parametrization"](#), [Naryan et. al, "unit Scaling: Simple and Scalable FP8 LLM Training"](#))

There are many different directions that are possible here and everything you propose will presumably involve at least a partial replication of what has already been done. The nature of this project invites combination with Project Category (2) as we mention later.

Project Proposal

- Please write a **one-two** page proposal on your project idea; including a high-level overview of your investigation, code bases you'll build upon, how you will generate training/test data, and your resource budget — an awareness of how much compute power you can muster and use is very important.
- Please include all the group members in your proposal, but only one person needs to submit the proposal.

Final Report Requirements:

- Must have a coherent story that clearly lays out the question being investigated, cites relevant literature while being a self-contained treatment, and shows clear and systematic analysis/exploration.
- Must cite relevant work to put your work into an intellectual context.
- For example, you need to provide loss curves for training and validation and compare as needed to tell your story. (Here, it can be fine to partner up with other project teams in this theme to share baselines, etc.) **Meaningful error-bars are required.**
- Provide a link to your github repo with all the code used for this project as well as trained model checkpoints sufficient to efficiently replicate all plots and tables in your paper.

(2) Theoretical/empirical study of in-context learning

Read the late 2022 paper "[What Can Transformers Learn In-Context? A Case Study of Simple Function Classes](#)" by Garg, Tsipras, Liang, and Valiant, look at some follow-up work like the ones listed below, and propose your own investigation that follows that theme and leverages things that you have learned. There are many different directions that are possible here and everything you propose will presumably involve at least a partial replication of what has already been done. For example, you can investigate other simple function classes that have not already been done (e.g. kernelized linear models), compare different architectural choices (e.g. transformer variants including those based on faster attention mechanisms, alternatives like state-space models, etc... Or take the architectural choices that ended up winning the nanoGPT speedrun), as well as questions related to the coverage of training vs test data (e.g. an appropriate distributional shift type of question) or even focus on questions of optimizers and hyper-parameter tuning (for example, exploring muP in this space).

Some relevant works:

- [von Oswald et al, "Transformers learn in-context by gradient descent"](#)
- [Akyurek et al, "What learning algorithm is in-context learning? Investigations with linear models"](#)
- [Zhang et al, "Trained Transformers Learn Linear Models In-Context"](#)
- [Zhang et al, "Training Dynamics of In-Context Learning in Linear Attention"](#)
- [Singh et al, "Strategy Coopetition Explains the Emergence and Transience of In-Context Learning"](#)
- [Bondaschi et al, "From Markov to Laplace: How Mamba In-Context Learns Markov Chains"](#)
- [Daniels et al, "Decomposing Prediction Mechanisms for In-context Recall"](#)

Project Proposal

- Please write a **one-two** page proposal on your project idea; including a high-level overview of your investigation, code bases you'll build upon, how you will generate training/test data, and your resource budget — an awareness of how much compute power you can muster and use is very important.
- Please include all the group members in your proposal, but only one person needs to submit the proposal.

Final Report Requirements:

- Must have a coherent story that clearly lays out the question being investigated, cites relevant literature while being a self-contained treatment, and shows clear and systematic analysis/exploration.
- Must cite relevant work to put your work into an intellectual context.
- For example, you need to provide loss curves for training and validation and compare as needed to tell your story. (Here, it can be fine to partner up with other project teams in this theme to share baselines, etc.) **Meaningful error-bars are required.**
- Provide a link to your github repo with all the code used for this project as well as trained model checkpoints sufficient to efficiently replicate all plots and tables in your paper.

(3) Interpretability

While we expect most student groups taking option (2) above to be training your models from scratch, this option could involve working with pretrained models in some way.

Interpretability is a broad area where what we're trying to do is understand in more detail how/why a trained neural net is behaving the way that it does. This is quite broad, and you can see <https://www.neelnanda.io/mechanistic-interpretability/getting-started> for a pointer to many resources. One way that one can test whether one has found an actual mechanism is to figure out an intervention based on that mechanism that would alter behavior. (For fun, take a look at <https://www.anthropic.com/news/golden-gate-claude>.) For another recent example, take a look at <https://arxiv.org/abs/2502.03708>. (The “linear representation hypothesis” is what suggests this family of interventions and explorations — and so any work that further confirms, challenges, or extends this hypothesis is an example of what is welcome in a project.)

There are now multiple open-source models that have also released pretraining checkpoints. Those checkpoints can be used to explore many things, including the emergence/stability of mechanisms as well as the transferability of interventions across training.

There is also interpretability work that either trains or theoretically analyzes small toy models on algorithmic tasks (often with known algorithmic solutions) to try and better understand the solutions that models converge to and how they develop their capabilities. For example see:

- [Elhage et al, "A Mathematical Framework for Transformer Circuits"](#)
- [Elhage et al, "Toy Models of Superposition"](#)
- [Kunin et al, "Alternating Gradient Flows: A Theory of Feature Learning in Two-layer Neural Networks"](#)
- [Nanda et al, "Progress measures for grokking via mechanistic interpretability"](#)
- [Daniels et al, "Decomposing Prediction Mechanisms for In-context Recall"](#)
- [Michaud et al, "Quantization Model of Neural Scaling"](#)
- [Arous et al, "Learning quadratic neural networks in high dimensions: SGD dynamics and scaling laws"](#)

Project Proposal:

- Please write a **one-two** page proposal on your project idea; including a high level overview of your investigation, your base pretrained models, code bases, datasets, theoretical setting, and/or resource budget.
- Please include all the group members in your proposal, but only one person needs to submit the proposal.

Final Report Requirements:

- Must have a coherent story and be written as a more systematic investigation instead of a mere demo or hack.
- Must cite relevant work and put your own project into context.
- Provide a link to your github repo with all the code used for this project. Everything needs to be reproducible.