MINT-DFC: Meta-Learning In-Context with Transformers Across Diverse Function Classes

Vardhan Shorewala (3037822634), Vikram Iyer (3037663579), Ashok Devireddy (3037562714), Soam Desai (3037834100), Saaketh Kanduri (3039230846)

Overview and Motivation. In-context learning (ICL) lets large language models adapt to unseen tasks from examples in their input, without changing parameters. Garg et al. [1] showed that a Transformer can master individual function classes (e.g. linear, tree-based) via ICL. We now ask whether *one Transformer* can tackle multiple function types by inferring each task structure from a prompt (i.e. "meta-meta-learning"). This is relevant for real-world usage, where models like GPT-3 or GPT-4 encounter diverse tasks. Our goal: train a single Transformer on linear, polynomial (up to cubic), two-layer neural networks, and decision trees, then see if it matches specialized baselines while discovering class distinctions internally.

Research Goals.

- **Unified ICL:** Develop one model that handles linear, polynomial, neural net, and decision tree tasks.
- **Task Identification:** No class labels are given; the Transformer must discern each function type from in-context examples alone.
- **Meta-Learning Analysis:** Determine if the model implements distinct internal algorithms (e.g. approximate least squares vs. split-based rules) and examine whether it rapidly reduces error as k (number of examples) grows.
- **Performance vs. Specialists:** Compare to least squares (linear/polynomial), small neural nets (two-layer tasks), and CART/XGBoost (trees). Check if multi-task training induces synergies or trade-offs.

Methodology. We build on the open-source framework from Garg et al. [1]:

- *Model:* A GPT-2–style decoder-only Transformer, 12 layers, 8 heads, $\approx$ 25–30M parameters, trained from scratch with no language pretraining.
- *Function Classes:*
  1. Linear: $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$.
  2. Polynomials: $f(x) = ax^2 + bx + c$ (extendable to cubic).
  3. Two-Layer Nets: $f(\mathbf{x}) = W_2 \,\mathrm{ReLU}(W_1 \,\mathbf{x})$, $W_1, W_2 \sim \mathcal{N}(0, 1)$.
  4. Decision Trees: Depth-4 trees with random splits and constant leaves.
- *Prompt Format:* Each training example is k input-output pairs $(x_i, f(x_i))$, plus a query input x_q. The Transformer predicts $f(x_q)$ under MSE loss.
- *Mixed-Task Training:* Each batch samples tasks from all function classes. We adopt a curriculum: begin with linear/polynomial, then add neural nets and trees. We track per-class validation error to ensure balanced progress.
- *Compute:* One NVIDIA A100 (40GB). Sequence lengths up to $\sim$ 200 tokens, batch size $\sim$ 64. We expect $\sim$ 500k steps ($\sim$ 1–2 days of training).

Formal Setup. Let $f \sim \mathcal{F}$ be a function drawn from a distribution over functions $\mathcal{F} = \bigcup_i \mathcal{F}_i$, where each $\mathcal{F}_i$ represents a different function class (e.g. linear, neural net). A single training example consists of:

$$P = \{(x_1, f(x_1)), \ldots, (x_k, f(x_k)), x_q\}$$

The Transformer T_θ receives the sequence P and outputs a prediction $\hat{y}_q = T_\theta(P)$, aiming to minimize:

$$\mathcal{L} = \mathbb{E}_{f \sim \mathcal{F}} \left[\left(f(x_q) - T_\theta(P) \right)^2 \right]$$

The challenge lies in generalizing across $\mathcal{F}_i$ without knowing i. Thus, T_θ must implement a meta-learner that internally infers which $\mathcal{F}_i$ generated the context and adapts its strategy accordingly.

Why Curriculum Works. Assume function complexity can be ordered: Linear < Polynomial < NN < Tree. Curriculum learning amounts to training the model first on $\mathcal{F}_1, \ldots, \mathcal{F}_j$ before introducing $\mathcal{F}_{j+1}$. Under reasonable assumptions (e.g. task similarity or compositionality), early training helps bootstrap an internal representation that accelerates convergence on harder classes. We monitor validation loss $\mathcal{L}_{\mathcal{F}_i}^{\text{val}}$ per class and promote new tasks only if $\mathcal{L}_{\mathcal{F}_i}^{\text{val}} < \epsilon$ for all $i \leq j$.

Evaluation Plan.

- *Per-Class vs. Specialized Baselines:* For each class, generate unseen tasks and compare the Transformer's predictions to known methods. Measure MSE as a function of k.
- *Adaptation Speed:* Plot learning curves of MSE vs. k. We expect fast error reduction for models that effectively infer function structure.
- *Mixed-Class Prompts:* Evaluate on uniformly sampled mixtures of classes. Test stability and absence of class-wise bias.
- *Meta-Learning Analysis:* Probe hidden states $\mathbf{h}_k$ after the kth input-output pair to detect internal signals of function class. Train diagnostic classifiers: $\phi(\mathbf{h}_k) \approx \text{class}(f)$.

Team, Timeline, and Compute. We are five students sharing one A100 GPU. Work divides into:

- Data generation and prompt construction.
- Transformer training code (task mixing, curriculum).
- Evaluation infrastructure (baselines, learning curves).
- Group syncs, experiments, and final report.

We will first validate the framework on a subset of classes, then expand to all tasks with curriculum. Final models will be evaluated using the outlined metrics.

Conclusion. This study extends single-class ICL to the multi-class regime. We ask: *Can one Transformer implement multiple internal learners, automatically selecting the right one given a prompt?* If successful, our results will show that a single model can match or exceed experts trained on specific function families. If not, we gain insight into interference and how to structure learning for generalist models. Either outcome pushes the frontier of meta-learning in large Transformer-based systems.

References

[1] S. Garg, et al. *What Can Transformers Learn In-Context? A Case Study of Simple Function Classes.* arXiv:2208.01066, 2022.

[2] Y. Guo, et al. *How Do Transformers Learn In-Context Beyond Simple Functions? A Case Study on Learning with Representations.* OpenReview, 2024.