Extended Abstract

Is Exploration Helpful? Evaluating Transfer of Open-Ended Traces to Assignment-Based Code Edits

Tushar Dalmia, Karthik Seetharaman, Aaryan Shah

Motivation When students learn to code, they split their time between structured assignments and open-ended tinkering. We use data from Pencil Code [1] to capture both modes as full edit traces, with 825K assignment traces across 50 named tasks and 785K exploratory traces from untitled programs. If a model can learn something useful from the exploratory data, it is a free source of signal for modeling student behavior and hint generation for intelligent tutors. We ask whether pretraining on exploratory data improves a model's ability to predict student behavior on assignments.

Method & Implementation We frame student programming as a sequential decision problem where states are code snapshots and actions are line edits classified into a 9 class taxonomy from [1]. We compare training only on assignments (no-pretrain) and pretraining on exploratory data before fine-tuning on assignments (explore-pretrain). We also run a control (assign-pretrain) that pretrains on held out assignment data to test whether any benefit is specific to exploration or is a generic warm start. To isolate whether transfer requires sufficient model expressivity, we run a capacity sweep across a hand crafted feature MLP (Tier 1), a learned token embedding bag MLP (Tier 2), and a Transformer encoder over raw code tokens (Tier 3). We also implement a decoder-only Transformer for next snapshot prediction and a blinded human hint evaluation study. Finally, we conduct a curiosity analysis to compare edit entropy in exploratory vs. assignment traces and study whether filtering pretrain exploratory data by entropy improves behavior cloning. For most experiments, we use 5M pretrain pairs, 2M finetune pairs, 500k eval pairs, and run 3 random seeds.

Results **(1)** The next edit classification task shows that transfer is heavily dependent on model capacity. The delta between explore-pretrain and no-pretrain is -0.005, +0.010, +0.019 for Tier 1, Tier 2, and Tier 3, respectively, and the number of edit type classes with positive delta increases from 2 to 4 to 7 out of 9. At tier 3, the assign-pretrain control has a delta of +0.020 and 8/9 positive classes compared to no-pretrain, which demonstrates that the benefit of pretraining is generic and independent of exploratory vs. assignment. We also conduct a title ablation to determine which exploratory subsets produce positive transfer, finding that both the `first` and `untitled` subsets each uniquely transfer at a similar amount to combined data. **(2)** In the generation sweep, pretraining cuts validation loss by roughly 80% at 25K assignment pairs and modestly improves changed-line F1 edit accuracy, but the improvements lessen as the amount of assignment data increases, evening out by about 500K pairs. **(3)** A blinded qualitative hint analysis of 20 (state, action) pairs finds no significant preference between models (53% win rate for explore-pretrain compared to no-pretrain) and high subjectivity or overlap between hints. **(4)** In a curiosity analysis, exploratory traces show slightly higher trace edit type entropy (mean uncertainty 0.969 for exploratory traces vs. 0.909 for assignment traces) but no strong entropy decay over time (51.2% of exploratory traces decrease, which is near chance), disproving the hypothesis that student exploration may start out uncertain and become more pointed over time. Curiosity-filtered pretraining (top/bottom 25% by uncertainty, 2M rows) does not beat a 25% subsample of the combined exploratory pool at Tier 2, with pair-level F1 of 0.202 (full pool) vs. 0.196 (top 25%, worst). However, at Tier 3, full pool reaches 0.214 F1 ($\Delta = +0.003$ vs. no-pretrain, a slight improvement) while the top 25% bin is again the worst (0.208).

Discussion & Conclusion Exploratory code is useful pretraining data, but the benefit is broad and generic rather than unique to curiosity. Across models and tasks, exploratory pretraining improves soft distributional metrics such as trace-level Pearson correlation and generation validation loss, especially when assignment data is scarce. Its effect on precise single step behavior is weaker, with pair level F1, changed-line F1, and human hint preference are mostly flat or inconsistent. The assign-pretrain control and entropy filtering results suggest that the model benefits from extra student code pretraining, and not necessarily from high curiosity traces. Overall, exploration teaches the distribution and surface form of novice code more than the exact next edit. This may be valuable for distributional applications such as trace-level student modeling, curriculum analysis, or anomaly detection but offers limited gains for tutor or hint systems that depend on accurate single step predictions.

Is Exploration Helpful? Evaluating Transfer of Open-Ended Traces to Assignment-Based Code Edits

Tushar Dalmia
Department of Computer Science
Stanford University
tdalmia@stanford.edu

Karthik Seetharaman
Department of Computer Science
Stanford University
kvseet17@stanford.edu

Aaryan Shah
Department of Computer Science
Stanford University
aaryans@stanford.edu

Abstract

We study whether open-ended exploratory programming traces can serve as useful pretraining data for predicting student behavior on structured coding assignments. Using the Pencil Code dataset (825K assignment traces, 785K exploratory traces), we compare assignment-only training against exploratory pretraining followed by assignment fine-tuning across a capacity sweep of three model tiers: an MLP with hand crafted features, a learned token-embedding MLP, and a Transformer encoder. We find that transfer is null with hand-crafted features and is slightly positive when models process raw code tokens (macro Pearson $\Delta = +0.019$ at Tier 3, 7/9 edit classes non-negative). However, this transfer does not reach pair level macro F1, the single step accuracy metric, where the deltas are small and inconsistent across tiers. The next code snapshot generation experiment finds that exploratory pretraining reduces language modeling loss and improves edit accuracy, but an assignment pretraining control indicates that this benefit is a generic warm start rather than a gain specific to exploration data. A blinded human hint evaluation finds no significant preference between models. We conclude that exploratory data is a generic warm start. It teaches models the distribution and surface form of student code rather than the precise next edit, helps about as much as additional assignment data, and only when models have the capacity to read raw tokens.

1 Introduction

When students learn to program, they do not only work on assigned tasks. A substantial portion of their time is spent tinkering, such as writing small programs with no particular goal and changing numbers to see what happens. The Pencil Code platform captures both modes of student activity as complete edit traces, providing a rare window into the spectrum of student coding behavior.

From the perspective of intelligent tutoring systems, this exploratory data is appealing. It is abundant, requires no labeling, and captures a different mode of student engagement than assignment work. If a model pretrained on exploratory traces can better predict student behavior on assignments, this would provide a free source of signal for student modeling and hint systems.

We frame this question as a transfer learning problem using behavior cloning. Given a current code snapshot, we train models to predict the student's next edit, classified into a 9 class taxonomy from Ross et al. [1]. We also train a Transformer to predict the next code snapshot. Our primary

comparison is between two configurations: training only on assignments (no-pretrain) and pretraining on exploratory data followed by fine-tuning on assignments (explore-pretrain). The key question is whether explore-pretrain outperforms no-pretrain. In several experiments we also run a control that pretrains on a held out set of assignment data before fine-tuning (assign-pretrain), which tests whether any benefit is specific to exploration or is simply a generic warm start.

A natural concern is that the answer may depend on model capacity. Hand-crafted features may be too coarse to capture whatever signal exploratory data provides. To address this, we run a capacity sweep across three model tiers, from 11 hand-crafted features to learned token embeddings to a Transformer encoder over raw code tokens. We also conduct a human evaluation of model-generated hints.

Our main findings are as follows. In the next edit classification track, the most striking finding is that transfer is gated by representation capacity. With hand crafted features, the transfer is null and only emerges as a small positive signal once models process raw tokens. We also find that the benefit concentrates in soft, distributional signals such as trace level edit correlations, validation loss, and code fluency, while precise single-step prediction stays weak. Pair level macro F1 and hint preference are largely flat, and edit accuracy improves only modestly. We perform a controlled title ablation and an assignment pretraining control to reveal that the benefit is a generic pretraining warm start rather than something specific to exploratory content. In the generation experiment, exploratory pretraining is a strong warm start that sharply reduces validation loss when assignment data is scarce and modestly improves edit accuracy, but this advantage decays as we use more assignment data. We perform a blinded human hint evaluation and find no significant preference between explore-pretrain and no-pretrain, which is consistent with the weak single step results. Additionally, we perform a curiosity analysis and find that exploratory traces are slightly more diverse in edit types within a session but do not show structured entropy decay. Pretraining with data filtered by curiosity does not outperform using the full exploratory pool, with filtering based on high-uncertainty traces performing the worst, suggesting that the effect of just having more data outweighs the effect of filtering.

These results suggest that exploratory student code is a useful but generic source of pretraining signal. It teaches a model the distribution and surface form of student programming rather than the precise next move, and it helps about as much as pretraining on more assignment data. The largest gains are when assignment data is limited, and we often find that sampling from the full pool of exploratory data is similar to or beats filtered subsets by title and curiosity, respectively. Exploratory traces are thus most valuable as an abundant and label free warm start rather than as a uniquely informative source of data compared to assignments.

2 Related Work

Modeling student programming behavior. Ross et al. [1] introduced the Pencil Code dataset and a 9-class edit type taxonomy for characterizing student code edits. Their work focuses on predicting edit type distributions from assignment traces and evaluates models using trace-level Pearson correlations between predicted and actual edit type counts. They do not study transfer from exploratory to assignment data and instead only investigate assignment data in their work. Our project fills in this gap, making use of the exploratory data to study whether transfer effects exist. Other student programming datasets include BlueJ Blackbox [2] and Falcon Code [3], both of which contain code submissions from students alongside metadata, but not the same granularity of full edit traces and sequences used in Pencil Code.

Curiosity and intrinsic motivation in RL. Pathak et al. [4] introduces curiosity-driven exploration through prediction-error-based intrinsic rewards, showing that agents can learn useful behaviors without extrinsic reward signals. Burda et al. [5] extended this with Random Network Distillation, another technique to encourage agents to explore environments with curiosity. Liu and Abbeel [6] demonstrated that entropy maximization in latent spaces produces representations useful for downstream tasks. Our work asks a related but distinct question, not whether artificial curiosity helps an agent explore, but whether traces of real human curiosity (students exploring a coding environment) provide useful pretraining signal for predicting assignment behavior.

Transfer learning and pretraining in offline RL. Yang et al. [7] showed that unsupervised representation learning objectives improve offline RL policy performance. Prudencio et al. [8], write that surveys of offline RL frame the setting as learning from previous collected interactions available

in static datasets. Our work adapts this framing by treating the exploratory and assignment traces as distinct data sources within the transfer learning setup. This lets us explore whether exploratory behavior provides reusable information for downstream assignment prediction and whether that structure is dependent on model capacity.

3 Method

3.1 Problem formulation

We formulate student programming as a sequential decision problem. A state s_t consists of the current code snapshot at time t along with metadata such as the assignment title and timestamp. An action a_t is the edit that transforms s_t into s_{t+1}. A trace is the full sequence $(s_0, a_0, s_1, a_1, \ldots, s_T)$ representing one student's work on one program, which can be either exploratory or assignment.

The primary task is edit-type classification. Given s_t, we want to predict the edit type of a_t. Following Ross et al. [1], we classify each action into one of 9 edit types: `small_addition`, `large_addition`, `small_deletion`, `large_deletion`, `line_modification`, `number_change`, `color_change`, `function_addition`, and `comment_addition`.

We treat each student as an expert policy $\pi_{\exp}(a \mid s)$ and fit π_θ by maximizing the log-likelihood of demonstrated transitions:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}[\log \pi_\theta(a \mid s)],$$

with a categorical distribution over the 9 edit types. For generation, the objective

$$\log \pi_\theta(a \mid s) = \sum_t \log \pi_\theta(a_t \mid a_{<t}, s)$$

is an autoregressive product of per-token softmaxes. We use categorical MLE rather than ℓ_2 regression because student edits are highly multimodal at a given snapshot.

We evaluate four tracks using behavior cloning: (1) next edit-type classification across the three-tier capacity sweep; (2) next code snapshot generation with a decoder-only Transformer (3) blinded human evaluation of edit-type hints derived from the classifier and (4) curiosity metrics plus curiosity-filtered exploratory pretraining.

3.2 Training configurations

We hold an evaluation set of assignment data fixed and compare three configurations:

- **No-pretrain:** train only on assignment transition pairs (s_t, a_t). This is a baseline that uses limited assignment data.
- **Explore-pretrain:** pretrain on exploratory data, then fine-tune on assignment data. This is the transfer test and our main point of interest.
- **Assign-pretrain (control):** pretrain on a held-out set of assignment data, then fine-tune on a disjoint set of assignment data. This isolates whether any benefit is specific to exploratory content or is a generic warm start.

3.3 Capacity sweep

To test whether transfer depends on model capacity, we evaluate three architectures of increasing representational power.

Tier 1: Hand-crafted features + MLP. The state s_t is represented by 11 hand-crafted features, which are line counts, token counts, color/number literal counts, comment counts, function definition counts, and presence of control structures such as `for` and `if`. These are fed to a small 3-layer MLP with approximately 5K parameters.

Tier 2: Token embedding bag + MLP. The code is tokenized using a simple regex tokenizer which tracks alphanumeric runs and single-character punctuation, with a vocabulary of 10K tokens built from the training data. Token embeddings are then fed to a MLP totaling approximately 2.6M parameters. The maximum sequence length is 128 tokens.

Tier 3: Transformer encoder. The same tokenizer and vocabulary are used as in Tier 2. We use a Transformer encoder (d_model $= 256$, num_heads $= 4$, num_layers $= 4$, d_ff $= 512$, dropout $= 0.1$, max_seq_len $= 256$) followed by a linear classifier. This totals approximately 4.7M parameters. Training uses AdamW (lr=$3 \cdot 10^{-4}$ pretrain, 10^{-4} finetune, weight_decay=0.01) and a cosine learning rate schedule with a linear warmup for 5% of steps, then cosine decay to zero, updated per batch.

3.4 Generation experiments

In addition to edit-type classification, we evaluate next code snapshot generation, which is the task to, given the current snapshot s_t, autoregressively predict the next snapshot s_{t+1}. We use a GPT-style decoder-only Transformer (d_model $= 256$, num_heads $= 4$, num_layers $= 4$, max_seq_len $= 512$, weight tying) trained with the same categorical MLE objective over code tokens. Inputs and targets use a `__NL__` newline sentinel so line structure is preserved under tokenization.

We apply the same three training configurations (no-pretrain, explore-pretrain, assign-pretrain), fix exploratory pretraining at 5M pairs, and sweep assignment fine-tuning from 25K to 2M pairs. At decode time, we use top-p sampling (with p=0.9) and enforce a floor for minimum new tokens (`min_new_tokens`=25). Without this floor, models pretrained on exploratory data tend to terminate too early under greedy decoding because exploratory edits are often very short. We report validation cross-entropy loss and `changed_line_f1` (F1 over changed lines).

3.5 Evaluation metrics

We report both precise single-step metrics from one snapshot to the next as well as soft distributional metrics across traces, since exploratory pretraining may improve aggregate edit patterns without improving pair level accuracy.

Pair-level macro F1. This is our primary single-step accuracy metric. For some pair i, let $\hat{y}_i \in \{1, \ldots, 9\}$ be the predicted edit type and y_i be the ground truth. We compute per-class $\mathrm{F1}_c$ and average uniformly over the 9 classes:

$$\mathrm{F1}_{\mathrm{macro}} = \frac{1}{9} \sum_{c=1}^{9} \mathrm{F1}_c, \qquad \mathrm{F1}_c = \frac{2\,\mathrm{TP}_c}{2\,\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c}, \tag{1}$$

where TP indicates true positive, FP indicates false positive, and FN indicates false negative.

Trace-level Pearson r. For each edit type c and trace τ, let $n_{c,\tau}^{\mathrm{true}}$ and $n_{c,\tau}^{\mathrm{pred}}$ be the counts of class c in that trace. We form vectors over held-out traces and compute

$$r_c = \mathrm{Pearson}\big(\{n_{c,\tau}^{\mathrm{true}}\}_\tau,\ \{n_{c,\tau}^{\mathrm{pred}}\}_\tau\big), \tag{2}$$

then report per-class r_c and their unweighted mean following Ross et al. [1]. This measures whether the model captures the distribution which traces emphasize which edit types, even when individual pair predictions are noisy.

Generation metrics. We use the mean token cross-entropy loss on held-out next snapshots for our validation loss. For measuring how localized edits in each pair of snapshots are, let $\mathcal{L}(\mathrm{prefix}, \mathrm{code})$ be the multiset of lines added or deleted between two snapshots. Then, with predicted snapshot $\hat{s}_{t+1}$ and ground truth snapshot s_{t+1}, we obtain

$$\mathrm{changed\_line\_f1} = \frac{2\,|\mathcal{L}(s_t, \hat{s}_{t+1}) \cap \mathcal{L}(s_t, s_{t+1})|}{|\mathcal{L}(s_t, \hat{s}_{t+1})| + |\mathcal{L}(s_t, s_{t+1})|}, \tag{3}$$

where intersection is taken with multiset multiplicity (a line modification counts as one added and one deleted line).

Hint evaluation. We map the top classifier edit type $\hat{y}_i$ to a fixed hint template $h(\hat{y}_i)$. For instance, `number_change` becomes "Try adjusting one of your numbers". Raters blindly compare hints from no-pretrain vs. explore-pretrain on the same state. We report the explore-pretrain win rate among non-tie judgments.

Curiosity metrics. If students explore with curiosity, it is reasonable to expect high edit-type diversity early on during a student's exploration (trying different edits to see what happens to the output) that decays as they settle on an approach. We operationalize this intuition with a sliding-window entropy measure. For a trace with edit types $e_1, \ldots, e_T$ (where $T \geq 12$), we define length-W sliding windows $\mathcal{W}_j = (e_j, \ldots, e_{j+W-1})$ for $j = 1, \ldots, T - W + 1$, where $W = 10$. If the empirical edit-type distribution in $\mathcal{W}_j$ has probabilities $\hat{p}_k^{(j)}$ (the fraction of edits with type k) - then, the window entropy H_j is

$$H_j = -\sum_{k=1}^{9} \hat{p}_k^{(j)} \log \hat{p}_k^{(j)}. \tag{4}$$

Uncertainty is the mean window entropy, given by

$$\texttt{uncertainty} = \frac{1}{T - W + 1} \sum_{j=1}^{T-W+1} H_j. \tag{5}$$

To test for how exploration evolves during a trace, we use an entropy delta after splitting windows into early and late thirds:

$$\texttt{entropy\_delta} = \overline{H}_{\text{late}} - \overline{H}_{\text{early}}, \tag{6}$$

where $\overline{H}_{\text{early}}$ and $\overline{H}_{\text{late}}$ are means over the first and last $\lfloor (T - W + 1)/3 \rfloor$ windows. Negative `entropy_delta` indicates decreasing edit type diversity over the trace.

4 Experimental Setup

4.1 Dataset

The Pencil Code dataset consists of two splits hosted on Hugging Face:

- **Assignment:** 825,353 traces across 50 named assignments, yielding 17.8M transition pairs after parsing.
- **Exploratory:** 785,439 traces from untitled programs, yielding 26.2M transition pairs.

After some data quality filtering (removing traces with more than 200 snapshots or spanning more than 7 days, which are likely shared classroom accounts per Ross et al.), we pretrain on 5M pairs, finetune on 2M assignment pairs, and evaluate on a held out set of 500K assignment pairs from 25K traces, with all three sets disjoint by trace. For the generation task we fix the 5M pretraining budget and sweep the assignment fine-tuning budget from 25K to 2M pairs.

4.2 Dataset characterization

Exploratory traces differ from assignment traces in several ways. Exploratory traces are longer (median 18 vs. 12 snapshots, median 19.0 vs. 10.3 minutes), consist of smaller per-step edits on average (mean 3.38 vs. 4.84 changed lines), and skew toward parameter tweaking: `number_change` is 4 percentage points more common in exploratory data, while `line_modification` is 5.7 points less common. Assignment data spans 50 distinct named tasks with a long tail, while 89% of exploratory data is concentrated in `first` (a seeded starter program) and numbered `untitled` programs. Table 1 shows metrics comparing the datasets of exploratory and assignment traces.

4.3 Training details

All classification experiments train with cross-entropy using inverse-frequency weights, and are evaluated with macro F1, which weights all 9 classes equally. We train the no-pretrain model for

Metric	Assignment	Exploratory	Ratio
Median snapshots per trace	12	18	1.50×
Median time span (min)	10.3	19.0	1.85×
Mean edit size (lines)	4.84	3.38	0.70×
Median code length (lines)	10	10	1.00×

Table 1: Dataset summary statistics. We use medians since the tails are extreme.

4 epochs. For the explore-pretrain and assign-pretrain configurations we use 2 pretraining epochs and then 2 finetuning epochs. The generation task fixes the 5M pretraining budget, and sweeps the assignment fine-tuning budget from 25K to 2M pairs with a fixed 500K held-out assignment evaluation set. All experiments were run on Modal with A10G GPUs.

5 Results

5.1 Edit type classification

All classification experiments use 5M pretraining pairs, 2M assignment fine-tuning pairs, and a fixed 500K held-out assignment evaluation set, which are all disjoint by trace. Table 2 summarizes the capacity sweep across all three tiers of models.

Table 2: Edit type classification capacity sweep. No-pretrain is trained on the 2M assignment set only, and explore-pretrain is trained on 5M exploratory rows and 2M assignment rows. Δ Pearson is the macro trace-level Pearson delta (explore-pretrain minus no-pretrain). The last column counts the 9 edit classes with non-negative trace delta.

Tier	Model	No-pretrain F1	Explore-pretrain F1	Δ Pearson	Positive Classes
1	Hand features + MLP	$.200 \pm .002$	$.193 \pm .001$	$-.005$	$2/9$
2	Token embed + MLP	$.234 \pm .001$	$.240 \pm .001$	$+.010$	$4/9$
3	Transformer encoder	$.240 \pm .003$	$.244 \pm .002$	$+.019$	$7/9$

The table demonstrates that Tier 1 shows no transfer. With 11 hand-crafted features, explore-pretrain produces a pair-level F1 of 0.193, slightly below no-pretrain at 0.200. The trace-level macro Pearson delta is -0.005, with only 2 of 9 edit classes showing positive transfer. The one notable win is `large_deletion` at $+0.049$, but this is offset by losses across the remaining common classes. Thus, hand crafted features do not represent whatever structure exploratory data provides.

Tier 2 starts to show emerging transfer. The token-embedding MLP improves no-pretrain F1 to 0.234 (a 17% relative gain over Tier 1) and explore-pretrain to 0.240. The pair-level delta is $+0.005$, which is modest but positive. The trace-level macro delta is $+0.010$ with 4/9 positive classes. The biggest gains are on the rare structural classes, such as `comment_addition` ($+0.058$) and `function_addition` ($+0.043$). We can thus conclude that token embeddings begin to capture code patterns that exploratory data provides, but the signal is not yet consistent across classes.

By contrast, tier 3 shows the clearest transfer. The Transformer encoder achieves the highest absolute performance (0.240 ± 0.003 F1 for no-pretrain) and the most consistent transfer signal. The pair level F1 delta remains small ($+0.004$), but 7 of 9 edit classes show positive trace-level Pearson deltas, with the Pearson delta at $+0.019$. The capacity gradient across tiers is monotone, increasing from 2/9 to 4/9 to 7/9 positive classes. Table 3 shows the per-class trace-level results for Tier 3, where we can see the highest gain in rare structural additions such as `function_addition` and `comment_addition`. Overall, representation capacity gates whether exploratory data transfers at all.

5.2 Title ablation

The exploratory data consists of two program types. There is "first" (52%, a seeded starter program) and "untitled" (48%, freeform exploration). We ablate each 5M row subset to test whether the transfer signal depends on one or both. We finetune on 2M assignment rows and evaluate on the same 500K held out assignment set in each ablation.

Table 3: Tier 3 trace-level Pearson r.

Edit type	No-pretrain	Explore-pretrain	Δ
line_modification	$.813 \pm .006$	$.822 \pm .002$	$+.009$
number_change	$.726 \pm .010$	$.738 \pm .006$	$+.012$
small_addition	$.704 \pm .009$	$.696 \pm .006$	$-.008$
color_change	$.516 \pm .026$	$.533 \pm .007$	$+.017$
large_addition	$.502 \pm .019$	$.522 \pm .006$	$+.020$
small_deletion	$.484 \pm .007$	$.503 \pm .006$	$+.019$
function_addition	$.454 \pm .027$	$.528 \pm .006$	$+\mathbf{.074}$
large_deletion	$.330 \pm .009$	$.336 \pm .012$	$+.006$
comment_addition	$.228 \pm .043$	$.248 \pm .030$	$+.020$

Table 4: Title ablation (Tier 3, 5M pretrain, 2M finetune, 500K eval). No-pretrain F1 is 0.240 ± 0.003 across all conditions.

Exploratory pretrain set	Explore-pretrain F1	Δ Pearson	Positive / 9
All (first + untitled)	$.244 \pm .002$	$+0.019$	$7/9$
First only	$.245 \pm .001$	$+0.008$	$6/9$
Untitled only	$.245 \pm .003$	$+0.022$	$7/9$

The no-pretrain baseline is 0.240 ± 0.003 for all conditions since the finetune and eval sets are the same. All three exploratory subsets produce positive trace-level transfer. The "untitled" subset alone achieves the strongest trace delta ($+0.022$, 7/9 positive), while "first" alone is weakest ($+0.008$, 6/9) but still positive. The combined set falls between them ($+0.019$, 7/9). Each exploratory subset independently contributes to transfer, and neither is necessary on its own. The transfer benefit thus appears to come from pretraining volume rather than from any specific type of exploratory content.

5.3 Assign-pretrain control

To test whether the transfer benefit is specific to exploratory data, we run a control where we pretrain on 5M assignment rows, finetune on 2M disjoint assignment rows, and evaluate on the same 500K held-out set. The assign-pretrain control produces a macro trace delta of $+0.020$ with 8/9 positive classes and a pair-level F1 delta of $+0.011$, which is comparable to or slightly better than the exploratory conditions. The classification transfer effect is a generic pretraining warm start rather than something unique to exploratory data.

5.4 Next code snapshot generation

In addition to next edit type classification, we also evaluate the next code snapshot generation task. We fix the pretraining budget at 5M pairs and sweep the assignment finetuning budget, comparing no-pretrain, explore-pretrain, and assign-pretrain, which are all evaluated on the same 500K held out assignment pairs. This is done to isolate whether any benefits are specific to exploratory data or just come from a warm start. Table 5 reports the results.

Table 5: Generation results with 5M pretraining pairs and sweep over assignment finetuning budget.

	val loss (lower better)			**changed-line F1** (higher better)		
Finetune	no pretrain	explore	assign	no pretrain	explore	assign
25K	1.496	0.244	0.195	0.174	0.186	0.188
100K	0.735	0.222	0.202	0.163	0.200	0.185
500K	0.236	0.208	0.198	0.166	0.188	0.171
2M	0.203	0.197	0.193	0.167	0.177	0.180

Figure 1 shows that explore-pretrain provides a strong warm start. At the smallest 25K fine-tuning budget it cuts validation loss from 1.50 to 0.24. As assignment data grows to 2M pairs, no-pretrain catches up and the gap nearly closes, at 0.203 vs. 0.197. Thus, pretraining helps when the target data

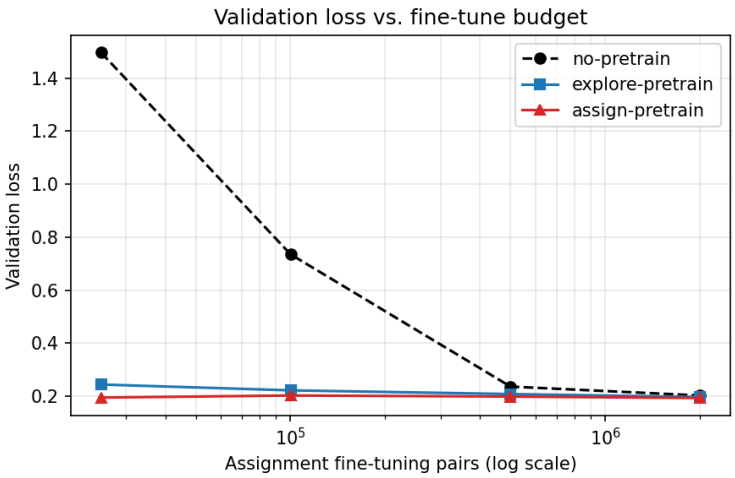

Figure 1: Val loss against assignment finetuning budget. No-pretrain starts much higher when assignment data is scarce and converges to the two pretraining curves by 2M pairs. Explore-pretrain and assign-pretrain overlap throughout, which shows the warm start is generic rather than specific to exploratory data. Each point at 25K, 100K, and 500K is a single seed per condition. The 2M point is averaged over 3 seeds and has std < 0.002. We omit error bars because the smaller budgets are single seed and the curve separation is much larger than noise at the seed level.

is scarce. We also find modest improvements in edit accuracy. Changed-line F1 exceeds no pretrain at every finetuning budget, with the largest effects at lower budgets.

We also ran a control that does assignment pretraining, which shows that the benefit of the warm start is not specific to exploratory data. Exploratory and assignment pretraining perform similarly on changed-line F1, with 0.177 vs. 0.180 at 2M rows, respectively. Exploratory data is thus a useful source for additional pretraining data, but would be most beneficial when assignment data is scarce.

5.5 Qualitative Hint Analysis

We convert the classifier's top edit type prediction into a hint, providing a nudge rather than the answer. For instance, `number_change` would be mapped to the hint "Try adjusting one of your numbers". Each held out state receives two hints, one from no-pretrain and one from explore-pretrain. The hints are randomized to A/B and we, the authors, blindly choose our preference or mark a tie.

Of 50 held out states, 20 are actually discriminative with the two models predicting different next edits and thus producing different hints. On the other 30, hints are identical and are not useful for preferential ratings.

Table 6: Blinded hint evaluation (20 discriminative states, 3 raters).

Rater	Explore-pretrain wins	No-pretrain wins	Tie
Rater 1	10	9	1
Rater 2	8	7	5
Rater 3	10	9	1
Total	**28**	**25**	**7**

Among the non-ties, we see that explore-pretrain is preferred on 28/53 = 53%, which is close to chance. Pairwise agreement between us is 55% to 65%, and we agree unanimously on only 9 of 20 items. Thus, the two models' hints cannot often be told apart, and this null result is consistent with the flat pair level F1 in the classification task. We posit that the edit types are somewhat overlapping (e.g., color_change vs. line_modification) or hard to delineate (e.g., small_addition vs. large_addition).

Table 7 shows three representative discriminative states with both model hints and the actual next edit.

Table 7: Three discriminative hint examples with rater preferences.

Current code	No-pretrain hint	Explore-pretrain hint	Actual edit	Rater pref.	Explore-pretrain correct?
`recolor [...], (ir, opt, ...) -> red: ir; green: opt*.2 + xrayle*.8; blue: xrayhe`	"Try experimenting with a different color."	"Try adjusting one of your numbers, like a size, angle, or distance."	`number_change`	Explore-pretrain (3/3)	Yes
`recolor [...], (ir, optical, radio, xray) -> red: optical; green: ir; blue: radio`	"Try modifying one of the lines you've already written."	"Try adjusting one of your numbers, like a size, angle, or distance."	`number_change`	Split (1 vs 2)	Yes
`dot purple, 100; pd(); fd 100; rt 90`	"Try adding a line or two to build on what you have so far."	"Try experimenting with a different color."	`small_addition`	No-pretrain (3/3)	No

The first two examples show the explore-pretrain model correctly identifying that a student working on a numeric color blending program is likely to tweak coefficients next. By contrast, the no-pretrain model misclassifies this as a generic line modification or color change. These two cases come from different programs (keplersnova and supernova) and the explore-pretrain model predicts the correct edit type in both, suggesting it has learned that numeric changes are the dominant next action in this kind of code. In the third example, we have a different situation where the student has a 4 line snapshot and adds a pen command, which is a small addition rather than a color experiment. No-pretrain correctly predicts this while explore-pretrain guesses a color change.

These examples illustrate the overlapping nature of the hint task. A student adding `pen purple, 10` to a program could be interpreted as a color change, an addition, or a parameter tweak. The models often disagree on cases like this, which is part of why rater agreement is only 55% to 65%. In the numeric blending programs we looked at, explore-pretrain appears better calibrated to the specific coding domain of those assignments. Even then, one of us preferred no-pretrain on the second example, which captures how subjective these judgments are without a reference solution.

5.6 Curiosity analysis

For the curiosity analysis, we ask whether exploratory traces exhibit more curiosity-type tendencies such as higher edit type diversity and entropy decay as a student narrows down their exploration. Then, we rank exploratory traces by this diversity signal and measure if pretraining on the highest-uncertainty subset improves assignment behavior cloning more than pretraining on the full exploratory pool. We answer the first with a trace-level entropy analysis (Phase 1) and the second with curiosity-filtered explore-pretrain at Tier 2 and Tier 3 (Phase 2).

Phase 1 scores 676K traces (294K assignment, 382K exploratory) using the trace `uncertainty` metric discussed earlier. We find that exploratory traces are slightly more diverse in edit types within a session. The mean uncertainty is 0.969 ± 0.290 vs. 0.909 ± 0.337 for assignment. Thus, exploratory data have slightly higher edit-type diversity than assignment traces, but this difference is small. Early-vs.-late entropy (`entropy_delta`) is small (-0.020 exploratory, -0.044 assignment), and entropy decreases in only 51.2% of exploratory traces and 52.8% of assignment traces, which is near chance. Thus, student exploration looks more like uniformly varied tinkering than open-ended curiosity decreasing over time.

Phase 2 tests whether this signal identifies better pretraining data. We rank exploratory traces by uncertainty and take the top and bottom 25% traces (6,115 each of 24,461 exploratory traces). We compare no-pretrain, explore-pretrain on a 25% subsample of the full exploratory pool, and

explore-pretrain on the top 25% and bottom 25%. Table 8 reports the pair-level macro F1 for each configuration at both Tier 2 and Tier 3 model capacities.

Table 8: Curiosity filtering for explore-pretrain (1 seed, 2M rows).

Exploratory pretrain set	Tier 2 F1	Tier 3 F1
No-pretrain	.206	.210
Combined exploratory pool	.202	**.214**
Bottom 25% uncertainty	.199	.210
Top 25% uncertainty	.196	.208

At Tier 2, we find that pair-level F1 is flat or worse for every explore-pretrain variant relative to no-pretrain at 0.206 F1. The combined exploratory pool reaches 0.202 F1, above both filtered bins, and pretraining on the top 25% by uncertainty is worst at 0.196 F1. The ranking is the same at Tier 3, with full pool, bottom 25%, and top 25% at 0.214, 0.210, and 0.208, respectively. However, increasing the model capacity from Tier 2 to Tier 3 slightly changes the results. The combined explore-pretrain now slightly exceeds the no-pretrain baseline (0.214 vs. 0.210, $\Delta = +0.003$), while the top and bottom bins match or underperform relative to no-pretrain. The trace-level macro Pearson at Tier 3 shows the same pattern, with the deltas compared to no-pretrain being $\Delta = +0.014$ for combined, $\Delta = -0.009$ for top, and $\Delta = -0.013$ for bottom.

Overall, we find that curiosity filtering does not help at either tier compared to using a subsample of the combined exploratory pool. Ranking traces by edit-type entropy does not result in better pretraining data as the top bin is consistently worst on pair level F1 for both Tier 2 and Tier 3. We can conclude that entropy captures local distributional diversity within a trace, but not a better behavior cloning objective, which is consistent with the flat pair-level F1 and hint preference we report earlier.

6 Discussion

Capacity gates transfer. The most striking finding is that the same pretraining recipe produces null results with hand-crafted features and a small positive signal once the model sees raw tokens. The boundary sits between Tier 1 (hand features) and Tier 2 (learned embeddings). This suggests that the transferable information in exploratory data is encoded at the token level (specific keywords, number patterns, code idioms) rather than at the level of coarse program statistics. The hand-crafted features were simply unable to represent whatever structure exploratory data provides.

Distributional vs. precise transfer. Across experiments, exploratory pretraining most clearly improves softer, distributional metrics such as trace-level Pearson and validation loss. Its effect on precise, single-step metrics is weaker and uneven. Pair-level F1 and hint preference stay flat, and changed-line F1 improves only modestly and only as much as the assign-pretrain control. This suggests that exploration teaches the model the distribution of student code and the surface form of programs more than the specific next move a student will make on a particular assignment.

This distinction is important for applications. For hint systems that need to predict the right next step, exploratory pretraining does not offer much measurable benefit. For applications that operate at the distributional level, such as identifying students whose edit patterns deviate from the norm, the trace-level signal may be more actionable.

Transfer is a generic warm start, not specific to exploration. The controlled title ablation shows that each exploratory subset independently produces positive trace-level transfer. When we compared these results to the assignment-pretrain control, we also saw comparable transfer. We see similar results on the generation experiment. At low data budgets, the pretrained model has a massive advantage in language modeling (validation loss 0.28 vs. 1.52). However, this advantage slowly decreases and is negligible by 500K pairs, and the assign-pretrain control achieves a similar warm start to the explore-pretrain method. Taken together, we can see that these results indicate that the transfer benefit may not be specific to exploratory content or to the diversity of the exploratory distribution. In this case, exploratory data may be useful because it is abundant and freely available, not that it has any net new structural value compared to assignment data.

Implications for curiosity in education. Our curiosity analysis found that exploratory traces are slightly more diverse but do not exhibit structured novelty. The mean entropy difference (0.97 vs. 0.91) is small, and entropy does not systematically decay over time, contrary to our original expectations. This suggests that what students do when exploring is better described as "uniformly varied tinkering" than as curiosity-driven exploration in the formal RL sense. This has implications for whether curiosity-weighted pretraining objectives (giving more weight to high-novelty edits) would help. Our results suggest that the diversity itself, rather than any structured curiosity signal, is what matters.

Limitations. The transfer effects we observe are small. Our trace-level Pearson deltas of $+0.010$ to $+0.020$ are larger than the observed seed-level variations but modest in practice. In addition, the controlled comparison uses a fixed 5M pretraining budget, so we do not characterize how transfer scales with pretraining volume. Larger or smaller pretraining corpora might shift the capacity boundary. Our generation model is relatively small, and larger models or different architectures might reveal different transfer dynamics. The hint evaluation is limited to 20 discriminative items with 3 raters, which gives limited statistical power.

Future work. There are several possibilities for future work based on our findings and limitations. Currently, we only conduct experiments with up to 5M pretraining pairs and models up to 4.7M parameters, but we could scale up to larger models and more data volume to measure if transfer increases. In addition, we currently only use the classification track to generate a single-step prediction and a corresponding hint, but a deployed tutor would be rolled out over multiple steps. Future work could investigate if errors over multiple steps compound over time. We could also experiment with a larger hint generation study for the next snapshot task on top of the classification task. Lastly, we find that naive behavior cloning may be too weak to extract exploration-specific structure, but that the curiosity metrics we use such as edit type diversity and entropy deltas are not significantly higher in exploratory data. More work could be done to calculate other curiosity metrics in code space such as forward model surprise to see if they differ, and then use these as pretraining signal.

7 Conclusion

We conducted a systematic evaluation of whether exploratory student programming traces can serve as useful pretraining data for downstream assignment tasks. Across a capacity sweep of three model tiers, two downstream tasks (classification and generation), a data scaling experiment, and a human hint evaluation, we find a consistent pattern: exploratory pretraining acts as a generic warm start. It moves soft, distributional signals and buys data efficiency when assignment data is scarce, but it offers little that additional assignment data does not.

The transfer effect is identifiable but small, and gated by model capacity. It is invisible with hand crafted features and present with learned token representations. It is also not specific to exploratory data. Each exploratory subset transfers on its own, and an assign-pretrain control transfers just as well, which shows the benefit is a generic pretraining warm start. In generation, pretraining sharply reduces validation loss when assignment data is scarce and modestly improves edit accuracy, but this advantage decays as we obtain more assignment data. A blinded human evaluation of hints finds no significant preference between models.

These results have practical implications. For hint systems that depend on single step accuracy, exploratory data provides no measurable benefit, and for next step prediction more broadly its benefit is small and roughly even against pretraining on additional assignment data. For applications that operate at the distributional level, such as trace-level anomaly detection or clustering students by mix of edit types, the small positive signal may be worth capturing. Future work should explore whether stronger transfer objectives (e.g., prediction error weighting, continued language model pretraining, or curriculum based data mixing) can push the distributional gains into single step accuracy, and whether larger models exhibit qualitatively different transfer dynamics.

8 Team Contributions

- **Tushar Dalmia:** Implemented the Tier 1 behavior cloning baseline. Developed the next snapshot generation track and worked on the Tier 3 Transformer, including the scarcity sweep

and debugging of the decoding pipeline such as the `__NL__` sentinels and `min_new_tokens` fix. Designed and coordinated the blinded hint evaluation study and analyzed the qualitative results.

- **Aaryan Shah:** Implemented the Tier 2 token-embedding MLP model. Ran the 5M scaling experiments, as well as the 5M-2M-500k title ablation study for comparing assignment vs. exploratory pretraining.
- **Karthik Seetharaman:** Implemented the Tier 3 Transformer encoder. Developed and computed the curiosity metrics (edit-type entropy, entropy decay analysis), along with leading the curiosity experiments and findings.

Changes from proposal. Our proposal outlined three main directions: a multi-model comparison across different policy architectures, a curiosity-based pretraining objective inspired by Pathak et al. [4], and standard behavior cloning evaluation. In practice, we replaced the multi-model comparison with a capacity sweep across three tiers of the same BC framework, which turned out to be more informative for isolating the role of representation capacity and model expressivity in transfer. The curiosity-based pretraining objective was not implemented as a training objective. Instead, we conducted a correlational analysis of curiosity-like properties in exploratory traces and found that the formal curiosity signal was weak, which informed our decision to focus on naive BC transfer instead of curiosity-based pretraining.

We also added three components not in the original proposal. First, we added a next-snapshot generation track with a scarcity sweep. Second, following feedback from the milestone, we added a blinded human hint evaluation for qualitative analysis. Third, following feedback from the poster session, we also evaluated an assign-pretrain control to investigate whether effects were unique to exploratory pretraining data or generic across any source of pretraining data. These additions strengthened the paper by providing converging evidence from different tasks and evaluation modalities.

9 AI Tools Disclosure

We wrote the core model code ourselves to understand the underlying principles, such as the Tier 1, Tier 2, and Tier 3 classifier architectures, the decoder-only generation model, and the cross entropy objective. We used AI as an aid for standard supporting code such as the data loaders and Modal compute infrastructure. We also used AI to format the LaTeX tables in this report.

References

[1] Alexis Ross, Megha Srivastava, Jeremiah Blanchard, and Jacob Andreas. Modeling student learning with 3.8 million program traces, 2025. Accepted to AIED 2026.

[2] Neil Christopher Charles Brown, Michael K"olling, Davin McCall, and Ian Utting. Blackbox: A large scale repository of novice programmers' activity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 223–228. ACM, 2014.

[3] Adrian de Freitas, Joel Coffman, Michelle de Freitas, Justin Wilson, and Troy Weingart. FalconCode: A multiyear dataset of python code samples from an introductory computer science course. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 938–944. ACM, 2023.

[4] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 2017.

[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[6] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems*, volume 34, pages 18459–18473, 2021.

[7] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11784–11794. PMLR, 2021.

[8] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, 2024.