AuthorMix: Modular Authorship Style Transfer via Layer-wise Adapter Mixing

Anonymous ACL submission

Abstract

The task of authorship style transfer involves rewriting text in the style of a target author while preserving the meaning of the original text. Existing style transfer methods train a single model on large corpora to model all target styles at once: this high-cost approach offers limited flexibility for target-specific adaptation, and often sacrifices meaning preservation for style transfer. In this paper, we propose AuthorMix: a lightweight, modular, and interpretable style transfer framework. We train individual, style-specific LoRA adapters on a small set of high-resource authors, allowing the rapid training of specialized adaptation models for each new target via learned, layer-wise adapter mixing, using only a handful of target-style training examples. AuthorMix outperforms existing, SoTA style-transfer baselines—as well as GPT-5.1—for low-resource targets, achieving the highest overall score and substantially improving meaning preservation.[1]

1 Introduction

Generating text that matches a targeted writing style is crucial for many applications such as personalized tutors, language learning, and conversational agents, among others (Xu et al., 2025). An important aspect of fostering user engagement is properly adjusting the style of the generated text. This is typically referred to as *authorship style transfer* and framed as a rewriting task in which a given text is rewritten to match the style of a specific target author (Patel et al., 2024; Liu et al., 2024a).

While style transfer methods have versatile use cases, they face two major challenges: (i) the rewritten text must preserve the original meaning, while (ii) the style must change substantially enough to match the target style. These challenges are further compounded in low-resource settings where examples of the target style are limited.

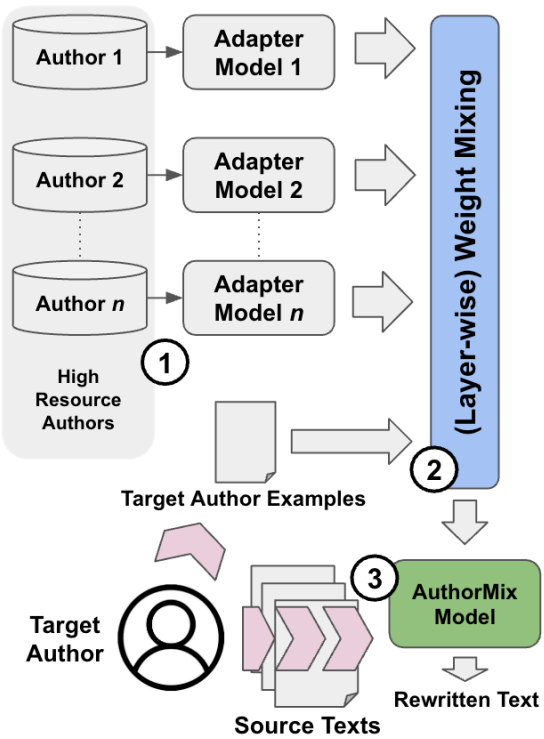

Figure 1: Overview of AuthorMix. (1) We train individual LoRA adapters for each high-resource author. (2) Scalar mixing weights are optimized to jointly maximize target style and meaning preservation. (3) The resulting model rewrites any text into the target style.

To address these challenges, many methods first rewrite the source text into a neutral style and then rewrite it into the target style (Liu et al., 2024a; Patel et al., 2024). Other approaches avoid the need for a neutral pivot style by adjusting the style directly in the embedding space, utilizing pre-trained style embeddings (Horvitz et al., 2024b; Riley et al., 2021; Horvitz et al., 2024a). All existing methods involve on training a single, general-

[1] Code and data will be made publicly available.

Approach	MIS	Output
Original	1.000	"I want to tell you something," I said: "I want to tell you all."
ASTRAPOP	0.271	I want to tell you,' he said.
TinyStyler	0.800	I wanted to tell you all, and tell you all, and tell you all, and tell you all, and tell you all, and tell you all.
STYLL	0.731	I have a burning desire to share with you everything, to lay it all out on the table."
GPT 5.1$_{\text{16-shot}}$	0.768	I broke in at last. "I have something to say," I began; "I would lay it bare before every one of you."
AuthorMix (Ours)	0.905	"Listen, I have something to say to you," I declared. "I have a message for all of you," I exclaimed.
Neutral	0.830	The speaker expressed a desire to convey information, stating that they wished to disclose everything.

Table 1: Outputs of different style-transfer methods for a single sentence along with their mutual implication score (MIS) that captures how well the meaning is preserved. See analysis in Appendix D.5.1

purpose model—an approach that often struggles to adapt to new rewrite styles (see Table 1).

In contrast, we propose AuthorMix, a lightweight, modular approach involving two stages of training. In the first stage, we train individual LoRA adapters (Hu et al., 2022) for a few high-resource authors (styles). For each target style, we then construct a model by learning (layer-wise) mixing weights over these adapters using a small set of target-style examples.

In comparison to existing style-transfer methods that require extensive pre-training or intermediate neutral-style text, AuthorMix directly produces a model tailored to a specific target style under a lightweight training regimen (see Figure 1). Moreover, our layer-wise weight mixing improves interpretability by revealing which *high-resource authors* contribute at which layer. Finally, our weight mixing objective explicitly balances style transfer and meaning preservation, encouraging the model-generated rewrites' semantic faithfulness to the source text (see Table 2).

Experiments demonstrate that AuthorMix outperforms current SoTA rewrite models—and GPT-5.1—at meaning preservation, while matching the current SoTA for style transfer in low-resource settings. We further find that layer-wise mixing with weights learnt via reinforcement-learning (RL) substantially outperforms static adapter-wise mixing; identifying layer-specific, gradient-based mixing as a promising direction for future work in model merging. Our contributions are as follows:

1. AuthorMix, a lightweight, modular, and interpretable authorship style transfer framework capable of efficiently producing a style transfer model for any given target author.
2. Extensive evaluations comparing AuthorMix to current SoTA style transfer models on 100 source–target author pairs, demonstrating that AuthorMix more effectively preserves semantic content in low-resource settings than existing approaches without sacrificing style-transfer capability.

Source: 'I will,' he said; and instantly went off through a gate, Lizzy continuing her way.
Output 1: 'I will,' he muttered, and darted off through the gate, while Lizzy continued on her way.
Output 2: "Agreed," he exclaimed, and at once strode through the gate, while Lizzy continued on her path.
Output 3: He replied, "I will," and instantly went off through a gate, while Lizzy continued on her way.
Output 4: He replied, then swiftly exited through a gate, while Lizzy continued on her path.
Output 5: "Yes, I will," he said, and hurried through the gate, while Lizzy went on her way.

Table 2: AuthorMix rewrites a single source sentence into different target author styles. Each output is produced by the same base model with different per-target mixing weights.

2 Related Work

Jin et al. (2022) broadly categorize style transfer methods into (pseudo-)parallel and non-parallel data methods. Whereas early parallel data works heavily rely on large parallel corpora (Xu et al., 2012) (Krishna et al., 2020), later works increasingly focus on low-resource scenarios with only a few available target style texts.

The lack of large parallel corpora is often mitigated via intermediate neutral style texts (i.e., pseudo-parallel data) which are then used as in-context learning examples (Patel et al., 2024) or even for policy optimization (Liu et al., 2024a). While neutral style texts compensate for a lack of parallel data, using them can lead to a semantic drift and result in lower meaning preservation;

	STYLL	ASTRAPOP	TinyStyler	AuthorMix
Method	Few-shot prompting	Condition on few-shot SFT + DPO/CPO	Condition on Embedding SFT + self-distill	Adapter mixing
Trainset authors	–	All 10 high-resource authors	∼1M	∼4 authors
Neutralization	Yes	Yes	No	No
Adapt to new target	Change prompt	Change prompt	Change embedding	Optimize weights
Modular / decomposable	No	No	No	Yes
Interpretable	No	No	No	Yes (per-layer)
Training Cost	–	48 h	> 48 h	∼8.7 h

Table 3: Comparison of authorship style transfer approaches. See Appendix D.4 for Training cost details.

especially when they are also required during inference (and not only during training). To avoid neutralization, non-parallel data methods often operate in a style embedding space, aiming to directly adjust the style (Rivera-Soto et al., 2021) (Wegmann et al., 2022). One recent work that utilizes style embeddings is TinyStyler, that relies upon a multi-stage training pipeline to remove the reliance on neutral style text during inference (Horvitz et al., 2024b).

Two common shortcomings of existing methods is that they rely upon costly training (using thousands of author pairs or millions of instances) and that they train a single monolithic model to serve all target styles, making it difficult to extend to new target styles. AuthorMix alleviates these issues by providing a lightweight means to produce a specific target style model (see Table 3).

3 Methodology

We separate authorship style transfer into three steps, which induces modularity, eases extensibility, and allows us produce a target-style specific model (see Figure 1):

1. **Author Adapter Training**: We train individual style adapters for a few *high resource* authors where abundant text is available.
2. **(Layer-wise) Weight Mixing**: For a given *target* author, we use a few text examples to conduct (layer-wise) weight mixing of individual style adapters.
3. **Text Rewriting**: We use the resulting model during inference to rewrite any *source* text provided by a source author.[2]

[2]Note, that the target and source author might be the same (i.e., the user) in various real-world scenarios. However, this distinction is important for evaluation (see Section 4).

3.1 Author Adapter Training

We first train individual LoRA adapters (Hu et al., 2022) to rewrite source texts into the style[3] of a specific, high-resource target author a. Given a corpus $\mathcal{X}_a$ of texts from author a, we construct pseudo-parallel training pairs by paraphrasing each $x \in \mathcal{X}_a$ into a neutral version $\tilde{x}$—removing vocabulary choices, sentence rhythm, syntactic patterns—while preserving the underlying content and meaning, by prompting a Language Model (see Appendix B.1). We then fine-tune a separate LoRA adapter to reconstruct the original text x from the style-removed version $\tilde{x}$:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{X}_a|} \sum_{x \in \mathcal{X}_a} \frac{1}{|x|} \sum_{i=1}^{|x|} \log p_{\theta+\theta_a}(x_i | \tilde{x}, x_{:i})$$

where θ are the frozen base model parameters and θ_a are the LoRA parameters for author a. Given n high-resource authors $a_1, ..., a_n$, we train the set $\mathcal{A} = \{\theta_1, ..., \theta_n\}$ of corresponding author-specific adapters.

3.2 Weight Mixing

In our second step, we conduct a weighted linear mixing of the trained adapters $\theta^{(i)} \in \mathcal{A}$ into a single target-style adapter $\hat{\theta}$. While similar mixing approaches have been explored before with adapter-specific weights (Huang et al., 2024; Ilharco et al., 2023; Fisher et al., 2024), to the best of our knowledge we are the first to conduct a more fine-grained mixing using layer-specific weights. For a model with L layers, the jth layer $\hat{\theta}_j$ ($1 \leq j \leq L$) of the target-style adapter $\hat{\theta}$ is then the linear combination of weights $W_{i,j}$ across n adapters:

[3]By style, we refer to recurring characteristics of how an author writes, such as lexical choice, syntactic preferences, punctuation, formatting, and other surface-level linguistic patterns, rather than the underlying semantic content.

$$\hat{\theta}_j = \sum_{i=1}^{n} W_{i,j} \cdot \theta_j^{(i)}$$

We use $W \in \mathbb{R}^{n \times L}$ to refer to the matrix of mixing weights. Despite the computational overhead compared to an adapter-wise mixing, we conjecture that the greater expressiveness will help in accommodating author-specific characteristics that may be located at different layers. This is in line with findings in model interpretability research that attribute specific linguistic knowledge to different layers (Tenney et al., 2019; Fayyaz et al., 2021; Zhou and Srikumar, 2022).

Given a low-resource target style (author) t with a small set $\mathcal{X}_t$ of example texts written by t, we obtain W by optimizing with respect to a style-transfer-specific objective function $S(x_s, x_{s\to t}, \mathcal{X}_t)$ (the *Joint Score* of Section 4.5).

For each x_s in our set $\mathcal{X}_s$ of training examples, and each corresponding candidate re-write $x_{s\to t}$ of x_s, $S(x_s, x_{s\to t}, \mathcal{X}_t)$ is defined as the geometric mean of: (i) the *Toward Score* (Patel et al., 2024) $T(x_s, x_{s\to t}, \mathcal{X}_t)$ between the style embedding of $x_{s\to t}$ and the mean style embedding of $\mathcal{X}_t$, which measures the stylistic similarity between the candidate rewrite and the target author's example texts; and (ii) the Mutual Implication Score (Babakov et al., 2022) $MIS(x_s, x_{s\to t})$, which measures the semantic similarity between the original sentence x_s and the rewrite $x_{s\to t}$ (see Section 4.5).

$$S(x_s, x_{s\to t}, \mathcal{X}_t) = \sqrt{T(x_s, x_{s\to t}, \mathcal{X}_t) \times MIS(x_s, x_{s\to t})}$$

In both of our optimization approaches (Section 3.2.1 and Section 3.2.2), the base model and all LoRA parameters remain frozen, and only the mixing weights W are updated.

3.2.1 LoRAHub

In our first approach, following the LoRAHub framework (Huang et al., 2024), we use Nevergrad's Auto-Optimizer (NGOpt, Rapin and Teytaud (2018)). This meta-optimizer automatically selects an appropriate gradient-free optimization strategy for the given problem.

Under this approach, the model $\pi_{W^{(i)}}$ generates a candidate rewrite $x_{s\to t}$ for each $x_s \in \mathcal{X}_s$ at each iteration i. The weights $W^{(i+1)}$ for the next iteration are then optimized via NGOpt to maximize the objective $\mathcal{L}_{\text{LH}}$:

$$\mathcal{L}_{\text{LH}} = \sum_{x_s \in \mathcal{X}_s} S(x_s, x_{s\to t}, \mathcal{X}_t) + \lambda \cdot |W^{(i)}|$$

Following Huang et al. (2024), we incorporate the L1 regularization term $\lambda \cdot |W^{(i)}|$ to mitigate overly large weights.

3.2.2 GRPO

We additionally consider directly learning W using the Group Relative Policy Optimization (GRPO; Shao et al. (2024)) reinforcement learning (RL) algorithm, with our joint score $S(x_s, x_{s\to t}, \mathcal{X}_t)$ serving as the reward function.

To the best of our knowledge, this is the first application of RL-based policy optimization to learning adapter-based mixing weights. Prior work on adapter mixing has relied exclusively on gradient-free methods (Huang et al., 2024; Fisher et al., 2024) or fixed heuristics (Ilharco et al., 2023; Wortsman et al., 2022).

3.3 Adapter Subset Selection

While weight mixing can be conducted using all adapters available in $\mathcal{A}$, this does not scale well with an increasing number of adapters, especially for gradient-free mixing, which often relies on search-based algorithms. Moreover, considering irrelevant adapters for weight mixing can even hurt the performance of the resulting model (see our ablation study in Section 5.3). To address this, we restrict to the subset of $\mathcal{A}$ corresponding to the top-k most stylistically similar authors to the target style for weight mixing (Fisher et al., 2024). We quantify style similarity using the cosine distance between prototypical style embeddings e_a for high-resource author a and target style embeddings e_t. We compute e_a by randomly sampling a set of texts $\mathcal{X}'_a \subset \mathcal{X}_a$ and averaging the resulting embeddings:

$$e_a = \frac{1}{|\mathcal{X}'_a|} \sum_{i=1}^{|\mathcal{X}'_a|} \text{Emb}(x_i)$$

We similarly obtain a representative embedding for the target style t:

$$e_t = \frac{1}{|\mathcal{X}_t|} \sum_{i=1}^{|\mathcal{X}_t|} \text{Emb}(x_i)$$

Finally, we select the top-k most similar adapters based on the resulting ranking using the cosine distance $\cos(e_t, e_a)$ between all available authors and the target style.

4 Experimental Setup

Our experiments include six baselines as well as four different variations of AuthorMix (adapter-

and layer-wise mixing using LoRAHub and GRPO methods). We report averaged results across ten runs using different seeds for pseudo-random number generation.

4.1 Datasets

We utilize three disjoint sets of authors with English literary text from Project Gutenberg (Gerlach and Font-Clos, 2018):

- **High-resource authors.** Liu et al. (2024b) provide a collection of 10 high-resource authors with substantial amounts of available text (more than 2,000 texts each). We use these to train our individual adapter models (see Section 3.1).
- **Target authors.** We randomly sample 10 unique authors from Project Gutenberg with 16 target texts each. This mimicks a low-resource scenario where only a small amount of text is available for a given target style.
- **Source authors.** Finally, we randomly sample another 10 unique authors from Project Gutenberg as source authors, whose texts serve as input to be style-transferred ($\mathcal{X}_s$ in Section 3.2). For each source author, we randomly split their texts into a training set of 50 texts for RL-based weight mixing (Section 3.2.2) and reserve 16 texts for testing. Following Patel et al. (2024) and Liu et al. (2024a), we sample 16 test examples for each source-target pair, resulting in $10 \times 10 = 100$ evaluation pairs and a total of 1,600 test instances.

We provide detailed statistics for all 30 authors in Appendix C.1.

4.2 Baselines

We compare AuthorMix against various baselines including prompting-based, embedding-conditioned, and policy-optimization methods. For all baselines, we use the same hyperparameters as reported in their respective original papers unless otherwise noted. Table 3 summarizes the key differences between different style transfer methods.

- **Neutral Text Baseline.** This serves as a lower-bound and simply rewrites source texts into a neutral style using LLaMA-3.3-70B-Instruct.
- **Few-shot Prompting.** We prompt GPT 5.1 to perform style transfer and provide 16 target style texts as in-context examples alongside the source text to be transferred.
- **STYLL.** Patel et al. (2024) propose an in-context learning approach using pseudo-parallel texts. We replicate their work using 16 in-context examples and respective style descriptors (see Appendix C.2 for more details).
- **TinyStyler.** Horvitz et al. (2024b) propose a method that utilizes style embeddings during training and is capable of style transfer without requiring pseudo-parallel text during inference. We directly use their benchmarked released checkpoint. To ensure a fair comparison against single-instance baselines, we do not generate multiple candidates with a re-ranking.
- **ASTRAPOP.** Liu et al. (2024a) investigate supervised fine-tuning (SFT) as well as direct preference optimization (DPO) to train models for style transfer. Due to the lack of publically-available model weights, we re-train their approach with a same amount of data (20k samples). Specifically, we used 2,000 texts from each of the 10 high-resource authors and trained ASTRAPOP with LLaMA 3.1 8B Instruct as the base model. We report results for both the SFT-only variant and the full, DPO-optimized model.

4.3 AuthorMix Variants

AuthorMix conducts a target-specific weight mixing either via LoRAHub; see Section 3.2.1 or GRPO; see Section 3.2.2. To better assess the impact of each weight mixing variant as well as the impact of our adapter subset selection (Section 3.3), we evaluate four variants:

- **LoRAHub (adapter-wise)**. The adapter-wise weight mixing method proposed by Huang et al. (2024).
- **LoRAHub (layer-wise)**. The LoRAHub framework extended to layer-wise mixing.
- **GRPO (adapter-wise)**. We utilize GRPO to learn single mixing weights for each adapter.
- **GRPO (layer-wise)**. We utilize GRPO to learn layer-wise mixing weights for each adapter.

We further evaluate all AuthorMix configurations for different $k \in \{2, ..., 10\}$ of most stylistically similar authors.

4.4 Base Model and Hyperparameters

We use LLaMA-3.1-8B-Instruct (Touvron and others, 2023) as the base model for AuthorMix. All high-resource author adapters are implemented as LoRA (Hu et al., 2022) modules and fine-tuned using the LLaMA-Factory framework (Zheng et al., 2024). The neutral paraphrases that are used for adapter training are generated using a LLaMA-3.3-70B-Instruct model with zero shot prompting (Appendix B.1). All experiments were conducted on a high-performance computing cluster with $4 \times$ NVIDIA H100/A100 GPUs.[4] We provide detailed hyperparameters in Appendix C.3.

4.5 Evaluation Metrics

Following Patel et al. (2024), we report three primary metrics:

- **Toward.** Measures how far the transferred output has moved toward the target author's style, as a fraction of the maximum possible movement. Given source text x_s, target author t, and transferred text $x_{s\to t}$ with respective style embeddings e_s, e_t, and $e_{s\to t}$, the toward score is computed as:

$$\text{Toward} = \frac{\max(Sim(e_{s\to t}, e_t) - Sim(e_s, e_t), 0)}{1 - Sim(e_s, e_t)},$$

  where Sim is the angular similarity $Sim(u, v) = 1 - \arccos(u \cdot v/(\|u\| \cdot \|v\|))/\pi$ over style embeddings from STAR (Huertas-Tato et al., 2023) — a RoBERTa-large encoder pre-trained on 4.5M texts from 70K authors (including authors from Project Gutenberg).[5]

- **Semantic preservation (MIS).** The mutual implication score (MIS, Babakov et al. (2022)) evaluates bidirectional entailment between the source and transferred texts using an NLI model. MIS ranges between 0 and 1 and has been shown to correlate well with human judgments of meaning preservation in style transfer tasks (Babakov et al., 2022; Patel et al., 2024).

Method	Toward ↑	MIS ↑	Joint ↑
Neutral Text†	0.01	0.79	0.05
Few-shot (GPT-5.1)	0.08	0.81	0.20
STYLL	0.07	0.68	0.16
ASTRAPOP-SFT	<u>0.16</u>	0.63	0.29
ASTRAPOP-DPO	**0.17**	0.63	0.29
TinyStyler	<u>0.16</u>	0.75	<u>0.31</u>
AuthorMix			
LoRAHub, AW (k=3)	0.11	<u>0.85</u>	0.25
LoRAHub, LW (k=8)	0.12	**0.87**	0.27
GRPO, AW (k=7)	0.13	0.78	0.29
GRPO, LW (k=4)	<u>0.16</u>	0.83	**0.34**

Table 4: Main results averaged over 100 source–target author pairs. ↑ = higher is better. **Bold** = best, <u>underlined</u> = second best. †Reference bound (neutralized text, no style transfer). AW = adapter-wise mixing; LW = layerwise mixing. Secondary metrics (CoLA, Away) are reported in Table 7.

- **Joint.** The geometric mean of Toward and MIS: $\text{Joint} = \sqrt{\text{Toward} \times \text{MIS}}$, capturing the trade-off between style transfer and meaning preservation.

We also report Away (Patel et al., 2024) and CoLA (Warstadt et al., 2019) as secondary evaluation metrics in Appendix C.4: Away is more relevant to source-style obfuscation than to target-style transfer, while CoLA mainly serves as a fluency safeguard for adapter mixing. AuthorMix maintains high fluency (CoLA = 0.83–0.87), and its Away scores are on par with other methods (0.64-0.73).

5 Results & Analysis

Table 4 reports results for AuthorMix and all baselines, averaged across all 100 source–target author pairs. We report the best-performing k (for adapter subset selection) for each AuthorMix configuration and provide the full results across all k in Figure 4 (detailed results are provided in Table 7). The results show that **AuthorMix achieves the best joint score** (0.34) with layer-wise GRPO weight mixing ($k = 4$), followed by TinyStyler (0.31) and ASTRAPOP-DPO (0.29). With respect to level of granularity for the weight mixing, we find that a fine-grained, layer-wise mixing consistently outperforms an adapter-wise weight mixing.

[4] The CPU architectures varied depending on the used node.

[5] LUAR-MUD (Rivera-Soto et al., 2021) was not suitable for our selected author set; see Gaafar et al. (2025) for discussion.

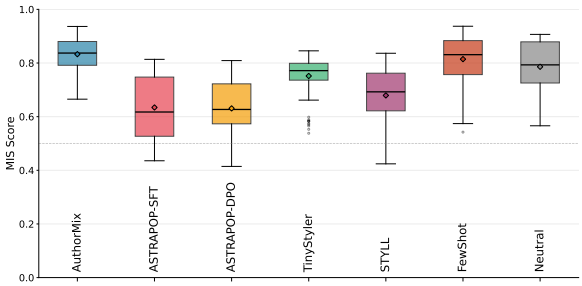

Figure 2: Distribution of MIS scores across all 100 source–target pairs for each method. Boxes show interquartile range (IQR); diamonds mark the mean; whiskers extend to 1.5×IQR. The dashed line at 0.5 indicates the threshold below which semantic equivalence breaks down.

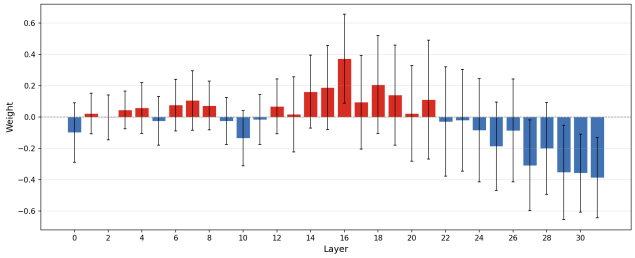

Figure 3: Global average of learned layer weights across all 10 target authors and all k values for GRPO layerwise mixing. Red bars indicate positive weights (adapter contribution is used); blue bars indicate negative weights (adapter is suppressed). Vertical lines indicate ± 1 standard deviation across target authors. A per-target-author heatmap is provided in Figure 8.

5.1 Other Metrics

Both AuthorMix and TinyStyler operate directly on source text without a neutralization step, yet AuthorMix achieves a substantially higher MIS (0.83 vs. 0.75) while matching TinyStyler's Toward score (both 0.16). Methods that *do* utilize neutral text during inference (ASTRAPOP, STYLL) achieve comparable or higher Toward scores, but at the cost of lower MIS ($\sim$ 0.63). While the MIS differences may appear modest numerically, Table 1 illustrates how scores around 0.75 already introduce noticeable meaning changes. Figure 2 shows the distribution of MIS scores across all instances and random seeds. AuthorMix has a consistently high MIS score with low variance. In contrast, all author style transfer methods either have a lower mean and median MIS and/or a larger variation. We provide detailed distributions and method-wise variances in Appendix D.2.

With respect to fluency, AuthorMix variants achieve consistently high CoLA scores of 0.83–0.87, comparable to all other baselines except both TinyStyler variants (which scores 0.68). This is also reflected in the TinyStyler example in Table 1. The low fluency of TinyStyler might be attributed to its informal Reddit text training data (Horvitz et al., 2024b) (in contrast to other methods trained on Project Gutenberg data) and the resulting domain mismatch might further contribute to the fluency gap when applying this model to literary text. Overall, our results suggest that AuthorMix with a modular, per-target weight mixing achieves a competitive target style transfer with a substantially higher meaning preservation compared to methods that train large, monolithic models.

5.2 LoRAHub vs. GRPO

A key design choice in AuthorMix is the selection of the weight mixing method. Comparing GRPO against LoraHub under identical conditions (same k) and similar compute budget reveals that across all configurations, **GRPO achieves higher Toward scores**; with larger differences for layerwise mixing (vs. author-wise mixing). Although **LoRAHub variants achieve a higher MIS** (0.85–0.87 vs. 0.78–0.83), they trade-off suboptimally with the Toward score (as reflected in their lower joint scores). We can further see that GRPO benefits substantially more compared to LoRAHub from a low number k of high-resource author adapters (Figure 4).

Finally, we analyze the individually learned weights in each layer, and visualize the averaged weights for GRPO and LoRAHub separately. Figure 3 shows the global average of learned layer weights across all 10 target authors and all k configurations (k=2–10) for GRPO layerwise mixing. Interestingly, we find that GRPO learns structured layerwise patterns, with the middle layers (14–21) receiving the strongest positive weights, while early layers (0–10) show small mixed-sign weights and deep layers (24–31) are predominantly negative, suppressing adapter contributions at these levels. The high standard deviation across target authors reflects that different target authors require different layer compositions, reinforcing the need for per-target optimization. A per-target-author breakdown is provided in Figure 8. In contrast, LoRAHub assigns near-zero weights across most layers (see Appendix D.3.1), indicating that it is not suited well for a layer-wise mixing. We conjecture that the substantially larger search space

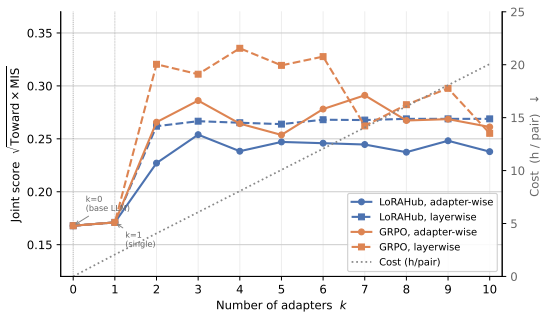

Figure 4: Joint score (left axis) and wall-clock training cost in hours per target (right axis) vs. number of related authors k for all four AuthorMix variants. $k = 0$ is the base LLaMA-3.1-8B-Instruct (zero-shot, no adapter); $k = 1$ is a single best-matching adapter. All scores averaged over 100 source–target pairs.

(compared to adapter-wise mixing) results in an optimization problem that is too difficult to solve well using the gradient-free methods available in NGOpt. We leave investigating a mixed-integer linear programing approach using more sophisticated solvers such as Gurobi (Gurobi Optimization LLC, 2024) for future work.

5.3 Impact of Number of Adapters k

We further conduct an ablation study to better understand the impact of using different k high-resource author adapters. Figure 4 shows the results for all four AuthorMix variants. We can see that all scores sharply rise from k=1 (single adapter, $\approx$ 0.17 joint score) to k=2 (0.23–0.32), confirming that mixing even two adapters is substantially better than using a single one. We further observe that GRPO layer-wise peaks at k=4 (Joint = 0.34) and remains competitive through k=6 before degrading at $k \geq 7$. In contrast, LoRAHub variants plateau earlier around k=3 and remain consistent for increasing k. This aligns with findings by Fisher et al. (2024) who report that mixing more than five adapters hurts grammaticality and findings by Huang et al. (2024) who observe that mixing more adapters expands the search space, leading to a higher variance. We conclude that identifying a good subset of top-k stylistically similar authors is essential as it reduces the search space and ensures that each adapter contributes a meaningful stylistic signal.

5.4 Analyzing Resource Consumption

We further provide an analysis on the resource consumption for each methods, highlighting the efficiency of AuthorMix (see Table 3). For instance, on comparable compute infrastructure, ASTRAPOP requires ∼48h of LoRA SFT + DPO training on all 10 high-resource authors (Liu et al., 2024a). TinyStyler is pre-trained using ∼8M texts from over 1M users (Horvitz et al., 2024b). In contrast, AuthorMix selects a small, targeted subset of authors and adapts to any new target by learning a total of only ∼128 scalar mixing weights which takes ∼40 minutes on a single GPU and requires no model retraining. Moreover, new adapters can be added independently to $\mathcal{A}$ with LoRA SFT requiring ∼2h of training on a single GPU.

AuthorMix provides an easily extensible alternative to large, monolithic models where producing a target style specific model incurs a cost of ∼ 40 min. Our results indicate that this per-target investment leads to a genuine per-target optimization —maximizing style transfer and meaning preservation for each target style.

6 Conclusion

We introduced AuthorMix, a modular framework for low-resource authorship style transfer that composes a target style specific LoRA adapter via optimized mixing weights. Unlike prior methods that train a single, monolithic model on large corpora, AuthorMix builds a modular adapter library from a small number of high-resource authors and dynamically adapts them to any new target. Our analysis shows that our novel GRPO-based method for learning a weight mixing results in interpretable, layer-wise weights; revealing that most stylistically important information is centered around the middle layers.

In future work, we plan to explore more sophisticated solvers for gradient-free weight mixing (as the ones implemented in LoRAHub failed to find meaningful weights). Moreover, the improved results for layer-wise mixing (over adapter-wise mixing) indicate that there is a high untapped potential which would also intrinsically improve interpretability (using the mixing weights). We believe that studying more advanced model merging strategies such as TIES-Merging (Yadav et al., 2023) or learned gating mechanisms may yield further improvements and are directly compatible with our framework. Since AuthorMix is modular, it naturally extends to cross-task style transfer by combining style adapters with task-specific ones (e.g., for summarization or dialogue) to produce stylized outputs for tasks beyond paraphrasing, and to scale larger, more diverse adapter libraries.

7 Limitations

Our evaluation is conducted on a single domain (Project Gutenberg literary texts) and a single base model (LLaMA-3.1-8B-Instruct). While the framework is architecture-agnostic, we have not yet demonstrated generalization to other base models (e.g., Qwen) or to domains with substantially different stylistic characteristics such as informal social media text or technical writing. All authors in our dataset are historical nonfiction/fiction writers from a similar time period; the method's effectiveness for authors with highly distinctive or contemporary styles remains to be validated.

The adapter library currently contains 10 high-resource authors. Scaling to a larger and more diverse library could improve coverage of a wider range of target styles, but it would also increase the one-time cost of adapter training ($\sim$2hrs). However, this cost is incurred only once, and the same high-resource adapters can then be reused across multiple target authors. In return, AuthorMix provides a more interpretable approach to authorship style transfer. In addition, although AuthorMix requires only about 40 minutes of weight learning for each new target, this per-target adaptation cost is not zero, unlike in baselines that adapt through prompt or embedding changes at no additional cost. Inference cost, however, remains unchanged across all AuthorMix variants, regardless of k, because the mixed adapters are merged into the base model; it is therefore comparable to that of baselines built on the same LLaMA-3.1-8B-Instruct backbone.

Finally, our evaluation relies on automatic metrics (MIS, CoLA, Toward). The STAR model used for evaluation also guides adapter selection and weight optimization, mirroring the design of StyleRemix (Fisher et al., 2024), which likewise uses its style scores for both axis selection and optimization. Weight learning uses source-author texts disjoint from the test set. While MIS correlates well with human judgments of meaning preservation (Babakov et al., 2022), a human evaluation and validation with an independent style metric would further strengthen the findings.

8 Ethical Considerations

Authorship style transfer presents important ethical risks, as models that reproduce a person's writing style could be used for impersonation, deception, or other forms of misuse. In this work, we study the task in the context of publicly available literary texts and position it as a research problem aimed at understanding and improving personalized text adaptation, rather than enabling unauthorized imitation. Nevertheless, improved style transfer quality may also increase misuse potential. We therefore emphasize that any downstream deployment should include safeguards such as transparency about generated content, restrictions on modeling private or living individuals without consent, and careful consideration of potential harms related to identity, authorship, and misuse.

References

Nikolay Babakov, David Dale, Varvara Logacheva, and Alexander Panchenko. 2022. A Large-Scale Computational Study of Content Preservation Measures for Text Style Transfer and Paraphrase Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 300–321.

Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. 2023. AdapterSoup: Weight Averaging to Improve Generalization of Pretrained Language Models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2009–2025.

Mohsen Fayyaz, Ehsan Aghazadeh, Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2021. Not All Models Localize Linguistic Knowledge in the Same Place: A Layer-wise Probing on BERToids' Representations. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 375–388, Punta Cana, Dominican Republic.

Jillian Fisher, Skyler Hallinan, Ximing Lu, Mitchell L Gordon, Zaid Harchaoui, and Yejin Choi. 2024. StyleRemix: Interpretable Authorship Obfuscation via Distillation and Perturbation of Style Elements. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Moemen Gaafar, Anurag Sarkar, and Matthew Guzdial. 2025. Few-shot Style-Conditioned LLM Text Generation via Latent Interpolation.

Martin Gerlach and Francesc Font-Clos. 2018. A Standardized Project Gutenberg Corpus for Statistical Analysis of Natural Language and Quantitative Linguistics. *Entropy* 22.

Gurobi Optimization LLC. 2024. Gurobi Optimizer Reference Manual.

Carolin Holtermann, Markus Frohmann, Navid Rekabsaz, and Anne Lauscher. 2024. What the Weight?! A Unified Framework for Zero-Shot Knowledge Composition. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1138–1157, St. Julian's, Malta.

Zachary Horvitz, Ajay Patel, Chris Callison-Burch, Zhou Yu, and Kathleen McKeown. 2024a. Paraguide: Guided diffusion paraphrasers for plug-and-play textual style transfer. In *Proceedings of the AAAI conference on artificial intelligence*, pages 18216–18224.

Zachary Horvitz, Ajay Patel, Kanishk Singh, Chris Callison-Burch, Kathleen McKeown, and Zhou Yu. 2024b. TinyStyler: Efficient Few-Shot Text Style Transfer with Authorship Embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13376–13390.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition. In *First Conference on Language Modeling*.

Javier Huertas-Tato, Alejandro Martin, and David Camacho. 2023. Understanding Writing Style in Social Media with a Supervised Contrastively Pre-trained Transformer. *arXiv preprint arXiv:2310.11081*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing Models with Task Arithmetic. In *International Conference on Learning Representations (ICLR)*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Look At? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP*.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep Learning for Text Style Transfer: A Survey. *Computational Linguistics* 48(1):155–205.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating Unsupervised Style Transfer as Paraphrase Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 737–762.

Shuai Liu, Shantanu Agarwal, and Jonathan May. 2024a. Authorship style transfer with policy optimization. *arXiv preprint arXiv:2403.08043*.

Xinyue Liu, Harshita Diddee, and Daphne Ippolito. 2024b. Customizing Large Language Model Generation Style using Parameter-Efficient Finetuning. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 412–426, Tokyo, Japan.

Ajay Patel, Nicholas Andrews, and Chris Callison-Burch. 2024. Low-Resource Authorship Style Transfer: Can Non-Famous Authors Be Imitated? In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 487–503.

Jeremy Rapin and Olivier Teytaud. 2018. Nevergrad — A Gradient-Free Optimization Platform.

Parker Riley, Noah Constant, Mandy Guo, Girish Kumar, David Uthus, and Zarana Parekh. 2021. TextSETTR: Few-Shot Text Style Extraction and Tunable Targeted Restyling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3786–3800, Online.

Rafael A. Rivera-Soto, Olivia Elizabeth Miano, Juanita Ordonez, Barry Y. Chen, Aleem Khan, Marcus Bishop, and Nicholas Andrews. 2021. Learning Universal Authorship Representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 913–919, Online and Punta Cana, Dominican Republic.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy.

Hugo Touvron and others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Weizhong Wang, Chao Du, Binhang Yuan, Zirui Liu, and Pengcheng He. 2024. LoRA Soups: Merging LoRAs for Practical Skill Composition Tasks. *arXiv preprint arXiv:2410.13025*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics* 7:625–641.

Anna Wegmann, Marijn Schraagen, and Dong Nguyen. 2022. Same author or just same topic? towards content-independent style representations. *arXiv preprint arXiv:2204.04907*.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time. In *International Conference on Machine Learning*, pages 23965–23998, PMLR.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for Style. In *Proceedings of COLING 2012*, pages 2899–2914, Mumbai, India.

Yiyan Xu, Jinghao Zhang, Alireza Salemi, Xinting Hu, Wenjie Wang, Fuli Feng, Hamed Zamani, Xiangnan He, and Tat-Seng Chua. 2025. Personalized Generation In Large Model Era: A Survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24607–24649.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Yichu Zhou and Vivek Srikumar. 2022. A Closer Look at How Fine-tuning Changes BERT. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061, Dublin, Ireland.

A Related Work

A.1 Adapter Merging and Composition

LoRA (Hu et al., 2022) enables parameter-efficient fine-tuning, and composing multiple adapters has become an active research area. AdapterFusion (Pfeiffer et al., 2021) introduces a two-stage approach that first trains task-specific adapters independently, then learns to compose them via a learned attention mechanism — separating knowledge extraction from knowledge composition. AdapterSoup (Chronopoulou et al., 2023), (Holtermann et al., 2024) show that weight-averaging domain-specific adapters, selected by similarity to the test input, improves out-of-domain generalization without any additional training. LoRAHub (Huang et al., 2024) introduces gradient-free optimization over a library of task-specific LoRA modules, learning scalar mixing weights from a few examples to achieve competitive cross-task generalization. StyleRemix (Fisher et al., 2024) applies a similar composition idea to text style: it trains LoRA modules on distinct stylistic axes (formality, sentence length, lexical complexity) and combines them with optimized adapter-level weights to steer outputs toward desired style dimensions for authorship obfuscation. More recently, LoRA Soups (Wang et al., 2024) shows that optimally weighted concatenation of LoRAs outperforms naive averaging on skill composition tasks. Our work builds directly on this paradigm: rather than composing task or style-axis adapters, we compose *author-style* adapters and optimize per-target mixing weights to steer generation toward a specific individual's writing style.

B Methodology

B.1 Pseudo-Parallel Data Generation

For each high-resource author's text, we generate a neutral paraphrase using LLaMA-3.3-70B-Instruct with the prompt: *"Paraphrase the following text into a neutral, encyclopedic tone, removing stylistic markers while keeping meaning unchanged."* The resulting (neutral, original) pairs serve as supervised fine-tuning data for the author adapters.

C Experimental Setup

C.1 Author Lists

Project Gutenberg is an open repository of literary works by authors whose writings have entered the public domain[6]. Its eBooks may be used for a wide range of purposes, including the creation of derivative works, reports, performances, and research. All texts used in this work were obtained from Project Gutenberg and are drawn from English-language authors.

High-resource (adapter) authors. Mark Twain, Virginia Woolf, Vernon Lee, Charlotte Perkins Gilman, George Orwell, Jane Austen, Nathaniel Hawthorne, Oscar Wilde, P. G. Wodehouse, and Samuel Richardson (obtained from (Liu et al., 2024b)[7]).

Low-resource target authors. Arthur Conan Doyle, Joseph Conrad, Jack London, E. M. Forster, F. Scott Fitzgerald, Rudyard Kipling, Robert W. Chambers, Agatha Christie, Edith Wharton, and L. M. Montgomery.

Source authors. L. Frank Baum, Charlotte Brontë, Lewis Carroll, Charles Dickens, Thomas Hardy, G. A. Henty, John Ruskin, Walter Scott, Henry B. Wheatley, and David Widger.

C.2 Baseline Details

We evaluate STYLL on GPT-3.5turbo since the GPT-3 endpoint used in (Patel et al., 2024) is deprecated by OpenAI, and GPT-3.5-turbo is the closest available model. We did not include BLOOM-7B based on STYLL, since MIS is in low range.

C.3 Base Model and Hyperparameters

C.3.1 LoRA Adapter Configuration

All adapters are fine-tuned using LLaMA-Factory (Zheng et al., 2024) with the following configuration: LoRA rank 8, $\alpha = 16$, applied to all linear target modules, input cutoff length of 256 tokens, and maximum output length of 256 tokens. *high-resource author* adapters and AuthorMix trained using following template: Listing 1 (no other additional input like previous works). For weight optimization, merging, and inference we largely follow the hyperparameter choices of StyleRemix (Fisher et al., 2024) (e.g., top-$p = 0.95$, temperature $\tau = 1.0$).

C.3.2 Weight Learning Details

For each target author, we learn the layerwise mixing weights using GRPO (Section 3.2.2) on the held-out source-author texts: each candidate

[6] https://www.gutenberg.org/policy/license.html
[7] https://github.com/cauchy221/StyleTunedLM

```
{
    "instruction": "Paraphrase",
    "input": "By 2020 and beyond this
number may have increased even further.",
    "output": "it is possible that this
number will increase even later by 2020."
}
```

Listing 1: Example input-output pair used in adapter fine-tuning for AuthorMix.

Hyperparameter	Value
Optimizer	NGOpt
Optimization steps	250
Weight initialization	0
Weight bounds	[−1.5 to +1.5]
Top-p sampling	0.95
Temperature τ	1.0
seed	42

Table 5: LoRAHub, Gradient-free optimization hyperparameters.

weight configuration generates a batch of style-transferred texts, which are scored against the target author's reference texts using the joint reward (Section 3.2.1). This is conceptually similar to the policy optimization stage of (Liu et al., 2024a) however, in our case the base model and all adapter parameters are frozen — only the $m \times L$ scalar mixing weights are tuned (e.g., $4 \times 32 = 128$ weights with $k = 4$ related authors and LLaMA-3.1-8B's 32 transformer layers). For each input, the model samples a group of G candidate outputs, and the mixing weights are updated using relative reward rankings within each group.

C.3.3 Gradient-Free Optimization Hyperparameters

For the gradient-free learning (Section 4.3), we use NGOpt (Rapin and Teytaud, 2018) as provided by LoraHub (Huang et al., 2024) with the configurations provided in Table 5. We use the comparable compute budgets to match with GRPO based weight learning (40mints).

C.3.4 GRPO Hyperparameters

We use hyper parameters as given in Table 6. we set β controls the optional KL regularization to zero.

Hyperparameter	Value
Learning rate	0.02
Optimization steps	300
Top-p sampling	0.95
Temperature τ	1.0
Weight initialization	0
Random seeds	42

Table 6: GRPO optimization hyperparameters.

C.4 On Away and CoLA as Secondary Metrics

Away score. The Away metric measures how far the transferred text has moved *away* from the source author's style, expressed as a fraction of the source–target distance (Patel et al., 2024). We exclude it from the Joint score for two reasons. First, Away is under-determined: many stylistically different outputs can achieve the same Away value, since any text that is equidistant from the source in the embedding space scores identically — regardless of whether it moved toward the target or in an unrelated direction. This makes Away a weak signal for evaluating style *transfer* quality. Second, a high Away score primarily matters in **authorship obfuscation** settings — such as those targeted by StyleRemix (Fisher et al., 2024) — where the explicit goal is to hide the source author's identity. In standard style transfer, the objective is to approach the target style (captured by Toward) while preserving content (captured by MIS); a text can succeed on both without fully erasing source-style traces, which is acceptable in most practical applications.

CoLA score. Large language models, especially instruction-tuned variants like LLaMA-3.1-8B-Instruct, generally produce fluent and grammatical text, making fluency a less discriminative metric in recent work. However, (Fisher et al., 2024) (who trained on Base LLaMa model) observe that adapter merging — which is central to our approach — can introduce fluency degradation, particularly when merging five or more adapters. Since AuthorMix composes multiple LoRA adapters via learned weights, we include CoLA as a safeguard to verify that the merging process does not degrade linguistic acceptability. Our results confirm that AuthorMix maintains high CoLA scores (0.83–0.87), comparable to

Method	k	Toward ↑	MIS ↑	Joint ↑	CoLA ↑	Away ↑
Neutral Text†	—	0.01	0.79	0.05	**0.89**	**0.91**
Zero-shot (LLaMA-8B)	—	0.07	0.81	0.17	0.88	0.81
Few-shot (GPT-5.1)	—	0.08	0.81	0.20	0.72	0.65
STYLL (GPT-3.5)	—	0.07	0.68	0.16	0.88	0.86
STYLL (GPT-5.1)	—	0.15	0.62	0.27	0.71	0.71
ASTRAPOP-SFT	—	0.16	0.63	0.29	0.80	0.63
ASTRAPOP-DPO	—	**0.17**	0.63	0.29	0.83	0.78
TinyStyler	—	0.16	0.75	0.31	0.68	0.66
TinyStyler-Sim	—	0.16	0.80	0.33	0.68	0.63
Single adapter (k=1)	1	0.06	0.86	0.17	0.73	0.46
AuthorMix — LoRAHub, adapter-wise						
	2	0.10	0.84	0.23	0.86	0.74
	3	0.11	0.85	0.25	0.85	0.71
	4	0.10	0.84	0.24	0.85	0.73
	5	0.11	0.84	0.25	0.84	0.70
	6	0.11	0.85	0.25	0.83	0.67
	7	0.11	0.86	0.24	0.81	0.61
	8	0.10	0.84	0.24	0.84	0.69
	9	0.11	0.86	0.25	0.82	0.62
	10	0.10	0.85	0.24	0.81	0.63
AuthorMix — LoRAHub, layerwise						
	2	0.11	0.86	0.26	0.83	0.65
	3	0.12	0.86	0.27	0.83	0.64
	4	0.12	0.86	0.27	0.83	0.65
	5	0.12	0.86	0.26	0.83	0.65
	6	0.12	**0.87**	0.27	0.83	0.64
	7	0.12	0.86	0.27	0.83	0.64
	8	0.12	**0.87**	0.27	0.83	0.64
	9	0.12	**0.87**	0.27	0.83	0.64
	10	0.12	**0.87**	0.27	0.83	0.64
AuthorMix — GRPO, adapter-wise						
	2	0.12	0.79	0.27	0.87	0.75
	3	0.13	0.78	0.29	0.86	0.72
	4	0.12	0.77	0.26	0.85	0.73
	5	0.11	0.75	0.25	0.86	0.77
	6	0.13	0.78	0.28	0.88	0.75
	7	0.13	0.78	0.29	0.87	0.73
	8	0.12	0.76	0.27	0.86	0.75
	9	0.12	0.76	0.27	0.86	0.76
	10	0.12	0.73	0.26	0.86	0.77
AuthorMix — GRPO, layerwise						
	2	0.15	0.83	0.32	0.85	0.69
	3	0.14	0.83	0.31	0.85	0.71
	4	0.16	0.83	**0.34**	0.83	0.67
	5	0.15	0.82	0.32	0.84	0.72
	6	0.15	0.82	0.33	0.83	0.71
	7	0.11	0.80	0.26	0.83	0.77
	8	0.12	0.82	0.28	0.84	0.76
	9	0.13	0.80	0.30	0.83	0.77
	10	0.11	0.79	0.26	0.82	0.80

Table 7: Full results for all baselines and all AuthorMix configurations across $k \in \{2, ..., 10\}$ related authors. All scores averaged over 100 source–target pairs. **Bold** = best within each AuthorMix variant. †Reference bound.

non-merged baselines, indicating that the learned weight composition does not harm fluency.

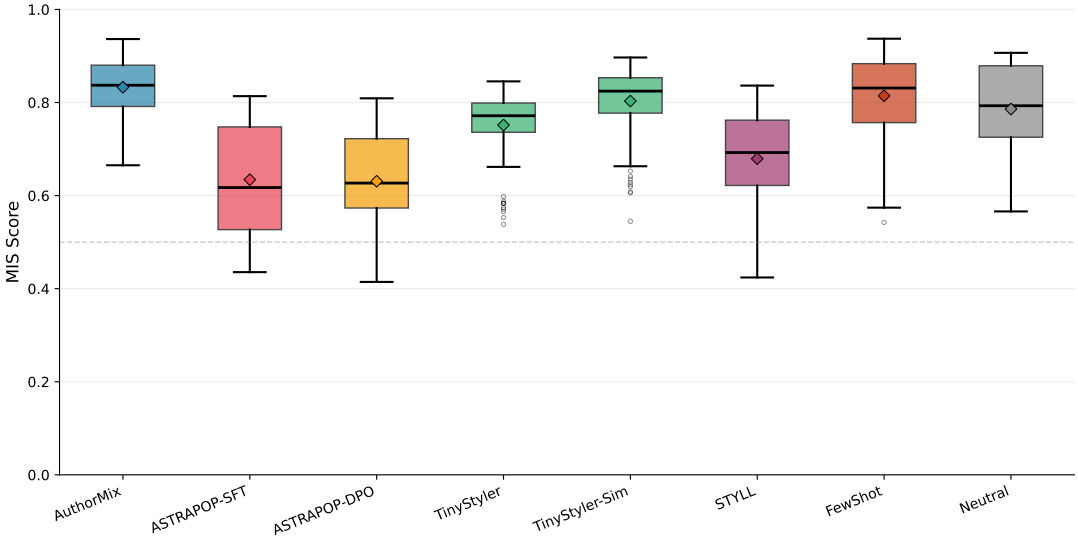

Figure 5: Box plot of MIS score distributions across all 100 source–target pairs for each method (including TinyStyler-Sim). Boxes show the interquartile range; diamonds mark the mean; whiskers extend to 1.5×IQR; circles are outliers. The dashed line at 0.5 indicates the threshold below which semantic equivalence breaks down.

D Results & Analysis

D.1 Full Results Table

Table 7 reports the complete results including TinyStyler-Sim and all AuthorMix configurations across $k \in \{1, ..., 10\}$.

D.2 MIS Score Distributions

Figure 5 and Figure 6 show the full MIS score distributions across all 100 source–target pairs, including TinyStyler-Sim post-publication variant. The KDE plot (Figure 6) annotates each method with its mean (μ) and standard deviation (σ).

AuthorMix achieves the highest mean MIS ($\mu = 0.83$) with the lowest variance ($\sigma = 0.059$) among all style-transfer methods — nearly half the standard deviation of ASTRAPOP-SFT ($\sigma = 0.115$) and ASTRAPOP-DPO ($\sigma = 0.107$). This confirms that AuthorMix not only preserves meaning better on average but does so *consistently* across source–target pairs.

TinyStyler-Sim ($\mu = 0.80$, $\sigma = 0.072$) improves over TinyStyler ($\mu = 0.75$, $\sigma = 0.071$) by adding further candidate generation and filtering candidates for semantic similarity, narrowing the gap to AuthorMix. However, AuthorMix still achieves a +4% higher mean with lower variance, and without requiring any candidate filtering step.

ASTRAPOP variants show the widest spread, with density mass extending well below the 0.5 threshold (dashed line in Figure 5), confirming that aggressive style transfer via DPO/SFT frequently comes at the cost of meaning. STYLL ($\mu = 0.68$, $\sigma = 0.099$) occupies a middle ground, while FewShot ($\mu = 0.81$) and Neutral ($\mu = 0.79$) score high on MIS but apply minimal style transfer.

D.3 Gradient-Based vs. Gradient-Free Weight Learning

D.3.1 Layer Weight Analysis

Figure 8 shows the per-layer adapter weights learned by GRPO for a single target author (Robert W. Chambers, k=4). The weights are far from uniform: adapter PGW (P. G. Wodehouse) receives strong positive weights throughout, while adapters VW (Virginia Woolf) and GO (George Orwell) are actively suppressed with negative weights (down to -1.0) at deep layers 24–31 — the layers that tend to encode discourse-level and syntactic patterns (Jawahar et al., 2019). The early layers (0–10) show moderate mixed-sign weights, and the middle layers (14–21) are dominated by strong positive weights for PGW and VW. This is principled: following the task-arithmetic framework (Ilharco et al., 2023) and TIES-Merging (Yadav et al., 2023), LoRA adapters act as task vectors in parameter space, and subtracting a task vector steers the model *away* from the style that adapter encodes.

Figure 9 breaks down the GRPO layerwise weights by k (number of adapters). The pattern described in the main text — positive middle layers, negative deep layers — is consistent across all k

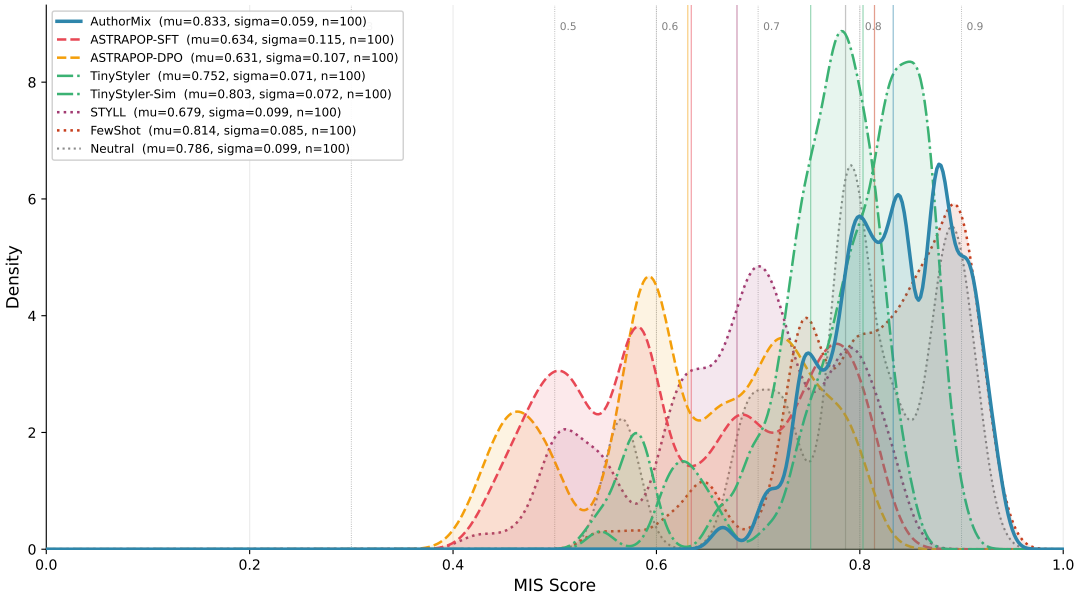

Figure 6: Kernel density estimate (KDE) of MIS score distributions for all methods. Legend shows the mean (μ), standard deviation (σ), and number of evaluation pairs ($n = 100$) for each method. Vertical lines mark the mean of each distribution.

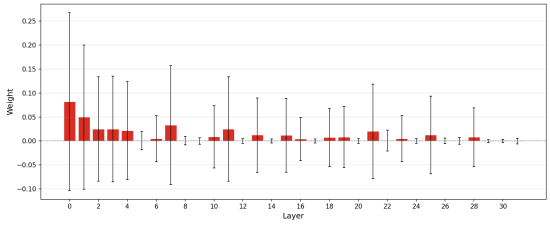

Figure 7: LoRAHub layerwise: global average of learned layer weights across all 10 target authors and all k values. All weights are near zero, indicating that gradient-free optimization does not produce structured layerwise patterns. Vertical lines indicate ± 1 standard deviation.

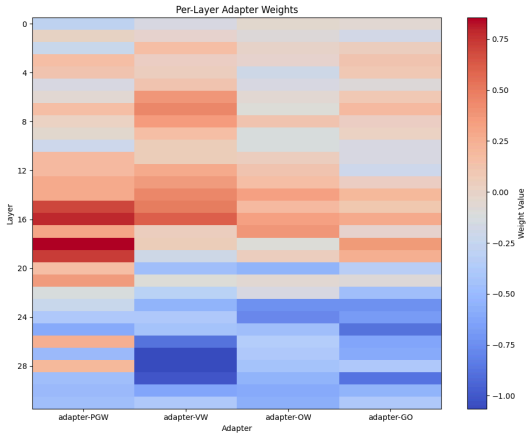

Figure 8: Per-layer adapter weights learned by GRPO for target author Robert W. Chambers (k=4). Each column is one of the top-k=4 style-similar related authors (PGW, VW, OW, GO); each row is a transformer layer (0–31). Red = positive weight (adapter is drawn upon); blue = negative (adapter is suppressed).

values (k=2–10), confirming that this structure is a robust property of GRPO optimization rather than an artifact of a specific k.

In contrast, Figure 7 shows the LoRAHub (gradient-free) layerwise weights averaged across all k values and target authors. The weights are near zero across all layers, with no discernible structure — confirming that gradient-free optimization struggles to exploit the layerwise search space effectively.

D.4 Analyzing Resource Consumption

Wall-Clock Training Cost: To facilitate fair efficiency comparisons, we report wall-clock training cost *per target author* — the total compute required before the system can serve a single new target individual. This is the practically relevant unit: it measures how quickly a user can be onboarded.

For AuthorMix, the cost to adapt to one target author has two components:

1. **Adapter training (one-time).** Each *high-resource author*'s LoRA adapter is fine-tuned independently in $\sim$2 h on a single A100 GPU. With k related authors, this costs $k \times 2$ h. These adapters are trained once and reused for any future target that selects them — so this cost is shared across targets. For the *first*

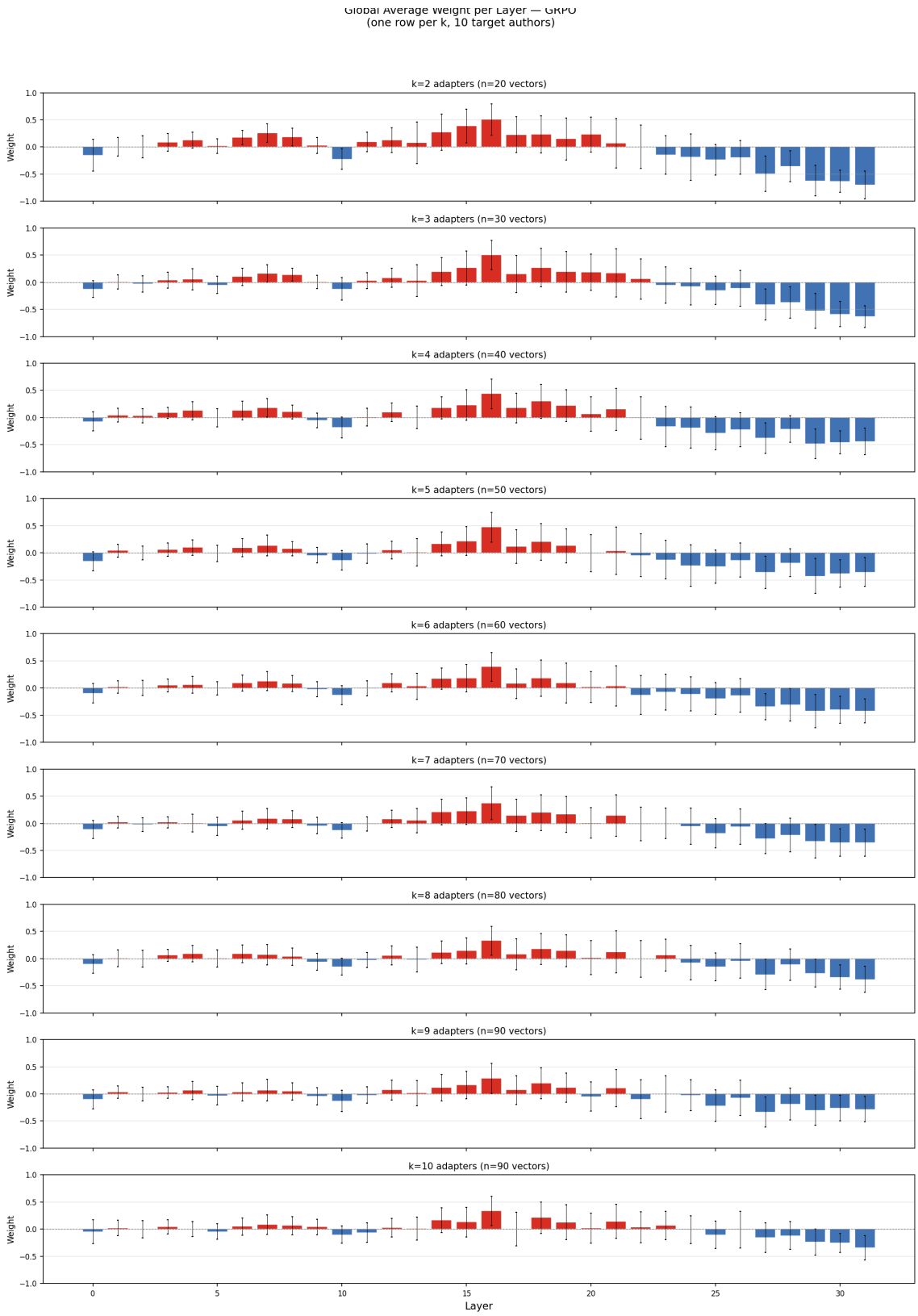

Figure 9: GRPO layerwise: global average of learned layer weights per k (k=2–10), averaged across all 10 target authors. Each subplot shows one k value. Red = positive weight; blue = negative weight. Vertical lines indicate $\pm$ 1 standard deviation across target authors.

target using a given adapter set it must be paid in full; for subsequent targets that reuse the same adapters, it is zero.

2. **Weight learning (per target).** Learning the mixing weights via GRPO takes ∼40 min per target author on a single A100 GPU. This is the only cost that recurs for each new target.

3. Inference cost remains unchanged across all AuthorMix variants, regardless of k, since the mixed adapters are merged into the base model; it is therefore comparable to that of baselines built on the same LLaMA-3.1-8B-Instruct backbone.

The total cost for the first target at $k = 4$ is therefore $4 \times 2 + 0.67 \approx 8.7$ h. Each additional target that reuses the same adapter library requires only ∼40 min.

For the baselines, the entire training cost must be invested before *any* target can be served:

- **ASTRAPOP-SFT:** ∼10 h of full SFT fine-tuning (one-time).
- **ASTRAPOP-DPO:** ∼48 h of SFT + DPO training (one-time).
- **TinyStyler:** reconstruction pre-training on ∼8M texts, full SFT, distillation. Despite using a small 800M-parameter backbone, TinyStyler's training cost remains high (>48 GPU h) because it requires full-model fine-tuning (reconstruction plus self-distillation) and is likely higher than both ASTRAPOP and our method.
- **STYLL / Few-shot:** No training cost (inference-only via API).

Once trained, baseline methods adapt to new targets at zero additional cost (by changing the prompt or embedding). AuthorMix instead pays ∼40 min per new target — but requires **5.5× less compute** than ASTRAPOP-DPO and far less dataset than TinyStyler to reach the first target.

D.5 Output Examples

D.5.1 MIS output analysis

Table 1 shows outputs of different style-transfer methods for a single sentence along with their mutual implication score (MIS) that captures how well the meaning is preserved. The original conveys three key elements: the speaker ("I"), the intent ("to tell something"), and the addressee ("you all"). Only AuthorMix preserves all three. ASTRAPOP changes the speaker to "he" and drops the group address; STYLL preserves the speaker but shifts "you all" to "everything," losing the direct address; TinyStyler preserves the content but degenerates into ungrammatical repetition. GPT-5.1 retains all three elements but adds embellishment not present in the original ("I broke in at last," "lay it bare"). The neutral baseline depersonalizes the speaker entirely ("the speaker," "they") and removes the addressee.

D.6 Qualitative Examples

Table 8 by fixing a single target author (Fitzgerald) and varying the source text, revealing how each method handles different inputs when targeting the same style.

Target Style Reference texts (Fitzgerald): “That’s what you ought to do,” she exclaimed triumphantly.“, “I can just see you,” she stormed, “letting him back you down!”, “Sit down and wait,” suggested the lieutenant nonchalantly….	
Method	**Output (target: Fitzgerald)**
Source (Widger):	‘I cannot do it,’ said Ethelberta; ‘and I will not.’
AuthorMix	“No, I won’t do it,” Ethelberta declared firmly. “Nor will I, under any circumstances.”
ASTRAPOP-DPO	“I couldn’t do it,” Ethelberta said, “and I don’t want to do it.
TinyStyler	Ethelberta said, “I cannot do it, and I will not do it.”
TinyStyler-Sim	Ethelberta said, ‘I cannot do it, and I will not.’
STYLL	Ethelberta declared she couldn’t and wouldn’t even try.
Few-shot (GPT-5.1)	“I can’t do it,” Ethelberta declared; “and I won’t.”
Source (Widger):	‘Well; better late than never, if so be he’ll have her now.‘
AuthorMix	“Better late than never, I suppose, and if he’s going to have her, it’s just as well he does it now.”
ASTRAPOP-DPO	“Yes, it’s better for him to have her now, even if it is late.
TinyStyler	‘Well, better late than never, if he has her now.’
TinyStyler-Sim	‘Well, better late than never, if he’ll have her now, if he doesn’t have her now…‘
STYLL	(Roguishly) Better late than never, right?
Few-shot (GPT-5.1)	“Well, it’s just as well he’s come around at last—provided he’s still willing to take her.”
Source (Widger):	Darton held his breath till he said with treacherous solicitude, ‘Try again—’tis coyness.‘
AuthorMix	He held his breath, and then whispered with a treacherous mixture of concern and flirtation, ‘Try again, it’s just coyness.‘
ASTRAPOP-DPO	Darton kept quiet until he spoke, as if she had been coy about it and he was urging her to be bold again.
TinyStyler	Darton was a little squirming in his seat until he said with treacherous solicitude, ‘Try again.’
TinyStyler-Sim	Darton held his breath until he said with a treacherous solicitude, “Try again—‘tis coyness.“
STYLL	Darton kept mum until he finally broke the silence, hinting that their hesitance was just a coy act, and egging them on to have another go.
Few-shot (GPT-5.1)	Darton, still holding his breath, murmured with a deceitful gentleness, “Try once more—it’s only shyness.”
Source (Widger):	One of them actually said that you must be fifty to have got such an experience.
AuthorMix	“If you’ve had an experience like that, you must be at least fifty,” he said bluntly.
ASTRAPOP-DPO	“He said a man’s got to be fifty years old to have that much experience.”
TinyStyler	One of them said you had to be fifty to have such an experience, smh.
TinyStyler-Sim	One of them said that you must be 50 to have had such an experience …
STYLL	It was said that you needed to be half a century old to have gathered so much experience.
Few-shot (GPT-5.1)	One of them even ventured the opinion that I must be fifty years old to have managed such an adventure.

Table 8: Multiple source sentences rewritten into Fitzgerald’s style by all methods. AuthorMix produces stylistically distinct outputs while preserving meaning across different source inputs for the same target author.