Matching Features, Not Tokens: Energy-Based Fine-Tuning of Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Cross-entropy (CE) training provides dense and scalable supervision for language models, but it optimizes next-token prediction under teacher forcing rather than sequence-level behavior under model rollouts. We introduce a feature-matching objective for language-model fine-tuning that targets sequence-level statistics of the completion distribution, providing dense semantic feedback without requiring a task-specific verifier or preference model. To optimize this objective efficiently, we propose *energy-based fine-tuning* (EBFT), which uses strided block-parallel sampling to generate multiple rollouts from nested prefixes concurrently, batches feature extraction over these rollouts, and uses the resulting embeddings to perform an on-policy policy-gradient update. We present a theoretical perspective connecting EBFT to KL-regularized feature-matching and energy-based modeling. Empirically, across Q&A coding, unstructured coding, and translation, EBFT matches RLVR and outperforms SFT on downstream accuracy while achieving a lower validation cross-entropy than both methods.

1 Introduction

Cross-entropy (CE) training under teacher forcing is the standard approach for pre-training, continued/mid-training, and supervised fine-tuning (SFT) of large language models. Its next-token objective provides an extremely *dense* learning signal, is stable under massively parallel optimization, and admits efficient implementations at scale [18, 5]. However, the same teacher-forcing setup introduces a distribution shift: during training, the model conditions on ground-truth prefixes, while at deployment time, it must condition on *its own* generations. Errors early in a generated sequence alter the conditioning context for subsequent predictions, causing later tokens to be sampled from distributions the model was rarely trained on [3, 19]. Braverman et al. [4] quantify this: the expected conditional entropy of the k-th generated token grows with k even for models with low training perplexity, showing that token-level supervision does not guarantee well-calibrated sequence-level behavior.

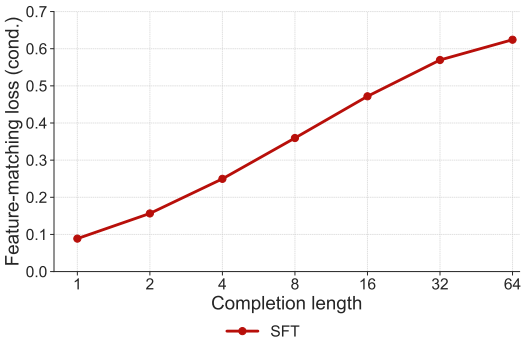

Figure 1: **Feature-matching loss grows with completion length.** Conditional feature-matching loss (lower is better) as a function of completion length for Qwen2.5-1.5B fine-tuned with SFT on OpenCodeInstruct [1]. Although this increase is expected even under a perfect model due to growing feature variance, part of the degradation reflects SFT's inability to calibrate the model's rollout distribution over long horizons.

A natural way to measure this divergence is to compare statistics of model-generated and ground-truth completions in a feature space. We formalize this in Section 2.1 as a *conditional feature-matching loss*; Figure 1 shows that under SFT this loss grows with completion length, motivating a method that targets it directly.

RL fine-tuning [25, 31] addresses this mismatch by optimizing sequence-level rewards under the model's own rollouts, enabling direct behavioral control. However, its effectiveness depends on access to a reliable reward function or verifier, as in reinforcement learning with verifiable rewards (RLVR; [21, 32]). These reward signals may be unavailable, noisy, or misaligned with desired behavior in many open-ended tasks. Even when a reliable reward exists, RL optimizes a scalar signal and does not directly target distributional calibration of the rollout distribution. In our experiments, we observe this tradeoff concretely: RLVR improves downstream performance at the cost of worsening both the validation cross-entropy and the feature-matching loss introduced above.

Beyond RLVR, a related class of methods incorporates *partial rollouts* at training time [37, 16, 11]. Since these rollouts are typically too short or incomplete to be scored by a verifier, these methods introduce surrogate rewards – commonly the model's own log-probabilities or token-overlap similarity between a sampled continuation and a reference. While useful in practice, neither type of surrogate provides calibration guarantees: self-likelihood rewards reinforce already high-probability samples without necessarily improving coverage, and similarity-based measures can improve the chosen metric without improving likelihood or calibration.

We propose an alternative that replaces these heuristic surrogate rewards with a principled objective: we directly optimize the feature-matching loss, using it not just as a diagnostic but as the training signal itself. A frozen feature network ϕ, initialized from the pre-trained model, embeds concatenated prompt–completion sequences, and the generator p_θ is fine-tuned to match the resulting feature moments using a REINFORCE-style gradient estimator on partial rollouts (see Figure 4). The resulting training signal is dense, operates at the sequence level, requires no task-specific reward or verifier, and — unlike the surrogate-reward methods above — optimizes a proper scoring rule under sufficiently rich features. We call this approach *Energy-Based Fine-Tuning* (EBFT): under a KL-regularized view, the feature-matching objective implicitly defines an energy function over sequences, with the optimal policy taking the form of an exponential tilt of the base model (Section E).

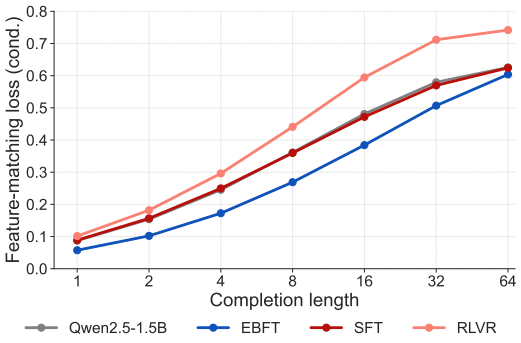

Figure 2: **EBFT achieves the lowest feature-matching loss across all completion lengths.** Despite training with a rollout horizon of only 8 tokens, EBFT's gains persist and grow at longer completions. RLVR worsens this loss relative to the base model.

Contributions. We introduce a feature-matching loss for language model fine-tuning that targets sequence-level statistics of the rollout distribution, and propose Energy-Based Fine-Tuning (EBFT) as a practical method to optimize it. We provide a theoretical perspective connecting EBFT to KL-regularized energy-based models (Section E). Empirically, we observe the following across Q&A coding, unstructured coding, and translation datasets:

1. EBFT achieves the lowest feature-matching loss across all completion lengths (Figure 2). Despite training with a rollout horizon of only 8 tokens, its gains persist and grow at longer generations, indicating genuine distributional calibration. In contrast, RLVR worsens this loss relative to the base model.
2. On downstream performance, EBFT consistently outperforms SFT and is competitive with RLVR, despite requiring no task-specific reward or verifier.
3. On validation cross-entropy, EBFT improves over SFT across all tasks, even though SFT explicitly optimizes this objective (Figure 3). RLVR, by contrast, substantially degrades validation perplexity.
4. EBFT can be applied in non-verifiable settings where RLVR is inapplicable. For instance, when training on raw code scraped from GitHub, EBFT yields substantial gains over SFT.

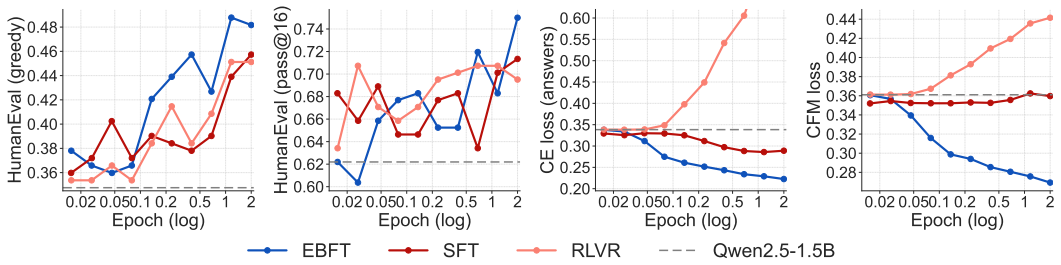

Figure 3: **EBFT improves downstream performance without sacrificing distributional calibration.** From left to right, we plot HumanEval accuracy (greedy and pass@16), validation cross-entropy (CE), and conditional feature-matching (CFM) loss over training for Qwen2.5-1.5B fine-tuned on OpenCodeInstruct [1]. SFT improves cross-entropy and CFM loss but lags on downstream accuracy. RLVR improves downstream accuracy but substantially degrades both calibration metrics relative to the base model (dashed line). EBFT achieves the best results across all four metrics, avoiding this tradeoff. CE and CFM losses are computed on a 1k-samples held-out subset of OpenCodeInstruct.

2 Language Modeling with Feature Matching

2.1 The feature-matching loss

Given vocabulary $\mathcal{V}$ and ground truth distribution p over contexts $c \in \mathcal{V}^*$ and completions $y \in \mathcal{V}^G$ of length G, and a language model p_θ, the feature-matching loss function is:

$$\mathcal{L}_{\mathrm{FM}}(\theta) := \quad \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi(c{:}\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi(c{:}y)]\big\|^2\Big], \tag{1}$$

where $c:y$ denotes concatenation and $\phi : \mathcal{V}^* \to \mathbb{R}^d$ is the feature map. We use the short-hand $\phi_c(y) := \phi(c{:}y)$. Since $\mathcal{L}_{\mathrm{FM}}$ depends on the unknown data moment $\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]$, it cannot be directly estimated from ground-truth pairs (c, y). A bias-variance decomposition lets us write the feature-matching loss in terms of the *conditional feature-matching loss*:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathcal{L}_{\mathrm{CFM}}(\theta) - \mathbb{E}_{c\sim p}\Big[\mathrm{Var}[\phi_c(y)|c]\Big], \tag{2}$$

where $\mathcal{L}_{\mathrm{CFM}}(\theta) := \mathbb{E}_{(c,y)\sim p}\Big[\big\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \phi_c(y)\big\|^2\Big]$, $\mathrm{Var}[\phi_c(y)|c] = \mathbb{E}_{y\sim p(\cdot|c)}\Big[\big\|\phi_c(y) - \mathbb{E}_{y'\sim p(\cdot|c)}[\phi_c(y')]\big\|^2\Big]$.

The offset $\mathbb{E}_{c\sim p}[\mathrm{Var}[\phi_c(y)|c]]$ is independent of θ and captures the per-context variability of the ground-truth features. Since $\mathcal{L}_{\mathrm{FM}}$ has optimal value zero, $\mathcal{L}_{\mathrm{CFM}}$ equals this offset at optimality. $\mathcal{L}_{\mathrm{CFM}}$ admits an unbiased estimator from ground-truth pairs (c, y); we plot these estimates in Figures 1 and 2. Note that $\mathcal{L}_{\mathrm{FM}}$ and $\mathcal{L}_{\mathrm{CFM}}$ mirror the (conditional) flow-matching loss functions for continuous generative modeling [22].

If the feature map ϕ is rich enough that matching feature moments implies matching distributions, then feature-calibration is equivalent to $p_\theta = p$ and $\mathcal{L}_{\mathrm{FM}}$ is a *strictly proper scoring rule* [14]. In other words, under a sufficiently expressive feature map, minimizing $\mathcal{L}_{\mathrm{FM}}$ is guaranteed to recover the true conditional distribution: it cannot be fooled by a model that matches some statistics while diverging elsewhere.

Relationship between feature-matching and cross-entropy. In practice, we optimize a mixed objective that combines feature matching with standard next-token cross-entropy (CE):

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{FM}}(\theta) + \gamma\,\mathcal{L}_{\mathrm{CE}}(\theta), \qquad \gamma \geq 0.$$

Both losses *share* the same unique minimizer, the ground-truth distribution $p_\theta = p$, and so does their combination $\mathcal{L}$. However, their landscapes differ: $\mathcal{L}_{\mathrm{FM}}$ penalizes deviations from $p(y|c)$ symmetrically, while $\mathcal{L}_{\mathrm{CE}}$ penalizes underestimation more heavily than overestimation. More importantly, using longer completions (e.g., $G \in \{4, 8, 16\}$) with richer features lets $\mathcal{L}_{\mathrm{FM}}$ target sequence-level statistics that the token-level CE loss is blind to.

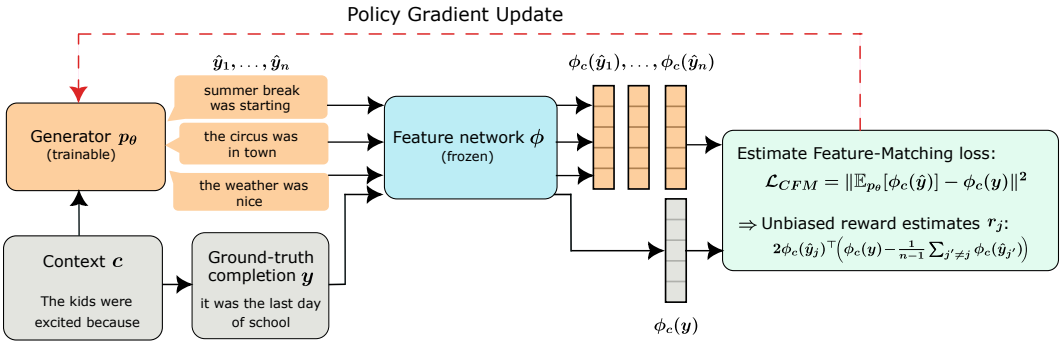

Figure 4: **Overview of Energy-Based Fine-Tuning (EBFT).** For each context c, the generator p_θ samples n completions. A frozen feature network ϕ embeds each prompt–completion pair, producing features $\phi(c : \hat{y}_j)$ for the sampled completions and $\phi(c : y)$ for the ground truth. Each completion receives a feature-matching reward measuring alignment with the ground-truth feature moment, and the generator is updated via REINFORCE with an RLOO baseline.

Constructing the feature map. We instantiate ϕ as a frozen *feature network* obtained by copying p_θ at initialization. Given a concatenated sequence $c : y$, we take the concatenation of intermediate activations at different depths of the feature network, normalize each block to unit L^2 norm, and concatenate them to form $\phi(c : y)$. In all experiments, we use layers at depths 25%, 50%, and 75%; the intuition is that earlier layers capture low-level information, final layers are biased toward next-token prediction, and middle layers carry semantic and structural information. We hypothesize that such high-dimensional feature maps are close to satisfying the richness condition above.

2.2 Feature-matching rewards and on-policy training

This subsection derives an unbiased REINFORCE estimator for $\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta)$ and describes the practical training recipe summarized in Algorithm 1.

Gradient estimation via REINFORCE. Since $\mathcal{L}_{\mathrm{FM}}$ and $\mathcal{L}_{\mathrm{CFM}}$ differ by a constant independent of θ per (2), their gradients coincide: $\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta)$. Hence, it suffices to estimate the gradient of the per-example loss

$$\mathcal{L}_{\mathrm{CFM}}(\theta; c, y) = \left\|\mathbb{E}_{\hat{y} \sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \phi_c(y)\right\|^2, \tag{3}$$

which satisfies $\nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{(c,y)\sim p}[\nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta; c, y)]$. Using the product rule of differentiation and the widely used identity $\nabla_\theta \mathbb{E}_{\hat{y}\sim p_\theta}[g(\hat{y})] = \mathbb{E}_{\hat{y}\sim p_\theta}[g(\hat{y})\, \nabla_\theta \log p_\theta(\hat{y} \mid c)]$ yields a REINFORCE gradient

$$\nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta; c, y) = -\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\nabla_\theta \log p_\theta(\hat{y}|c)\, r(\hat{y}, c)\right],$$

where the reward is

$$r(\hat{y}, c) = \underbrace{2\phi_c(\hat{y})^\top \phi_c(y)}_{\text{alignment term}} - \underbrace{2\phi_c(\hat{y})^\top \mathbb{E}_{\tilde{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\tilde{y})\right]}_{\text{diversity term}}. \tag{4}$$

We obtain an unbiased estimator of this gradient by sampling $n > 1$ completions $(\hat{y}_j)_{j=1}^n$ from $p_\theta(\cdot|c)$ and computing

$$\frac{1}{n}\sum_{j=1}^{n} \nabla_\theta \log p_\theta(\hat{y}_j|c) r_j, \quad \text{where} \quad r_j = 2\phi_c(\hat{y}_j)^\top \phi_c(y) - \frac{2}{n-1}\sum_{j' \neq j} \phi_c(\hat{y}_j)^\top \phi_c(\hat{y}_{j'}). \tag{5}$$

As in REINFORCE, it is possible to reduce the variance of this gradient by subtracting from r_j a baseline which is independent from $\hat{y}_j$. We use REINFORCE leave-one-out (RLOO), but must account for the fact that r_j already depends on the other completions; see Section F for the derivation of the REINFORCE gradient and RLOO baseline.

Algorithm 1 One EBFT training iteration.

1: **Input:** input prompt c; ground-truth completion y; n: samples per prompt; generation length G; *generator* p_θ; *feature network* ϕ
2: **Generation:** sample n rollouts of length G from the actor: $(\hat{y}_j)_{j=1}^n \sim p_\theta(\cdot \mid c)$
3: **Feature network embeddings:** compute the ground truth feature vector $\phi_c(y)$ and the rollout feature vectors $\left(\phi_c(\hat{y}_j)\right)_{j=1}^n$. Whiten the features as in (6) if needed.
4: **Feature-matching reward:** For $j = 1 : n$, compute

$$r_j = 2\phi_c(\hat{y}_j)^\top \phi_c(y) - \frac{2}{n-1} \sum_{j'=1, j' \neq j}^{n} \phi_c(\hat{y}_j)^\top \phi_c(\hat{y}_{j'}),$$

and the RLOO baseline b^j as in (90) in Section F.
5: **Actor update:** update p_θ with an RLOO update across $j = 1, \ldots, n$.

Energy-Based Fine-Tuning (EBFT) training recipe. Algorithm 1 summarizes one EBFT iteration. EBFT uses two models: a *generator* p_θ, which is the model that we want to fine-tune, and a *feature network* ϕ. In what follows, we only train the generator; we keep the feature network *frozen*. Given a pair (c, y) of ground truth context and completion of length G, we generate n completions $(\hat{y}_j)_{j=1}^n$ of the same length, and we feed the concatenated sequences $c{:}y$ and $(c{:}\hat{y}_j)_{j=1}^n$ through the feature network to obtain the feature vectors $\phi_c(y)$ and $(\phi_c(\hat{y}_j))_{j=1}^n$, which we use to get the rewards $(r_j)_{j=1}^n$ and the RL gradient following equation (5).

In practice, we introduce additional implementation details which affect how the method is instantiated: an optional *alignment-biased reward* that adjusts the fidelity–diversity trade-off, an efficient *strided block-parallel rollout scheme* for collecting many on-policy samples, and *feature whitening* to improve the conditioning of the feature space. We describe the first two below, and discuss whitening in the following subsection.

Adding an alignment bias. In some settings, one may prefer higher-fidelity samples at the cost of reduced diversity. This can be achieved by scaling the diversity term of the reward in (4) by a factor $\alpha \in (0, 1)$, which biases the objective toward closer alignment with the ground-truth features. We describe this variant in Section D and include experiments in Section I.1.

Strided block-parallel rollouts. To obtain many on-policy rollouts per training sequence efficiently, we use a strided block-parallel decoding scheme implemented with a custom attention mask (introduced by Quiet-STaR [37]). At a high level, this amortizes prefix computation and enables batched feature-network evaluation across many anchored prompts extracted from the same sequence. We give details and an example in Section G.

Feature matching with whitening. When the feature map ϕ has correlated or anisotropic directions, some dimensions can dominate the feature-matching loss. We address this with a whitened variant. For each context c and sampled completions $(\hat{y}_j)_{j=1}^n$, we estimate the second-moment matrix $\hat{\Sigma}_c = \frac{1}{n} \sum_{j=1}^n \phi_c(\hat{y}_j)\phi_c(\hat{y}_j)^\top$ and define whitened features

$$\tilde{\phi}_c(z) = (\hat{\Sigma}_c^\dagger)^{1/2} \phi_c(z), \tag{6}$$

where $\dagger$ denotes the Moore–Penrose pseudoinverse. Additional details are provided in Section C

2.3 Connection to energy-based models

Energy-based view. To further motivate EBFT, we show that under KL regularization, feature matching admits an energy-based interpretation. Adding a KL term to the feature-matching objective with reference distribution $q(\cdot|c)$ yields

$$\min_{\rho} \; \mathbb{E}_{c \sim p}\Big[\big\| \mathbb{E}_{\rho(\cdot|c)}[\phi_c(y)] - \mathbb{E}_{p(\cdot|c)}[\phi_c(y)] \big\|^2 \; + \; \tfrac{1}{\beta} D_{\mathrm{KL}}(\rho(\cdot|c) \,\|\, q(\cdot|c)) \Big], \tag{7}$$

whose solution is an exponential tilt of the base distribution, $\rho^\star(y|c) \propto q(y|c) \, \exp(-\chi_c^\top \phi_c(y))$, for some context-dependent $\chi_c \in \mathbb{R}^d$ (see Theorem E.3). This is precisely the maximum-likelihood

Method	Q&A Coding						Unstructured Coding					
	CE	FM	greedy	pass@1	pass@4	pass@16	CE	FM	greedy	pass@1	pass@4	pass@16
Base	0.338	0.361	0.484	0.424	0.606	0.715	0.631	0.369	0.473	0.419	0.596	0.702
Warm start	0.301	0.344	0.483	0.440	0.611	0.723	0.499	0.317	0.508	0.458	0.638	0.743
SFT	0.289	0.315	0.483	0.455	0.617	0.728	0.501	0.321	0.504	0.467	0.644	0.747
EBFT	0.207	0.258	**0.548**	0.510	0.659	**0.771**	0.499	0.320	**0.548**	**0.524**	**0.664**	**0.769**
EBFT (ws.)	**0.190**	**0.255**	0.534	0.508	0.658	0.756	**0.481**	**0.312**	0.536	0.514	0.659	**0.769**
RLVR	0.774	0.442	0.535	0.510	0.660	0.752	–	–	–	–	–	–
RLVR (ws.)	0.389	0.402	0.524	**0.529**	**0.662**	0.749	–	–	–	–	–	–

Method	CE	FM	Translation (COMET)				Translation (BLEU)			
			greedy	best-of-1	best-of-4	best-of-16	greedy	best-of-1	best-of-4	best-of-16
Base	1.870	0.637	0.644	0.611	0.701	0.745	0.074	0.124	0.186	0.231
Warm start	2.647	0.695	0.711	0.691	0.759	0.793	0.158	0.169	0.233	0.279
SFT	1.782	0.690	0.717	0.696	0.761	0.795	0.160	0.172	0.235	0.280
EBFT	**1.670**	**0.578**	0.725	0.713	0.765	0.795	0.182	0.194	0.244	0.283
EBFT (ws.)	1.671	0.580	**0.734**	**0.724**	**0.772**	**0.800**	0.185	0.197	**0.247**	**0.286**
RLVR	2.454	0.641	0.697	0.691	0.735	0.761	0.176	0.194	0.226	0.248
RLVR (ws.)	2.311	0.718	0.724	0.718	0.759	0.781	**0.195**	**0.210**	0.245	0.269

Table 1: **EBFT outperforms SFT and matches or exceeds RLVR on downstream metrics, while achieving the best distributional calibration across all tasks.** Best results per method on Q&A coding, unstructured coding, and translation. CE: validation cross-entropy; FM: feature-matching loss (both lower is better). "ws.": warm-started from an SFT checkpoint. RLVR is inapplicable to unstructured coding where no verifier exists; EBFT still yields substantial gains over SFT in this setting. See Table 6 for per-benchmark results and Section 3 for full experimental details.

problem for an energy-based model with energy $E(y, c) = \chi_c^\top \phi_c(y)$, motivating the term *energy-based fine-tuning*. EBFT does not parameterize or learn χ explicitly; it directly optimizes the generator via feature-matching gradients. See Section E for the full derivation.

Calibration view. The same exponential-tilt solution arises as the KL projection of $q(\cdot|c)$ onto the moment constraint $\mathbb{E}_{\rho(\cdot|c)}[f(y, c)] = m$, yielding $p_\chi(y|c) \propto q(y|c) \exp(\chi_c^\top f(y, c))$. Braverman et al. [4] use this with a *scalar* statistic $f(y, c) = -\log p_\theta(y|c)$ to correct entropy-rate drift in language model generations; EBFT applies the same correction with $f(y, c) = -\phi_c(y)$, enforcing *high-dimensional* moment constraints in a semantically rich feature space.

3 Experimental protocol

3.1 Tasks and metrics

We evaluate EBFT on both *verifiable* settings (where RLVR applies) and *non-verifiable* settings (where only SFT is typically applicable). For all tasks, we train on subsets of the full datasets to enable controlled comparisons under a fixed compute budget.

Coding tasks. We consider two complementary training regimes: (a) Q&A coding uses a 100k-sample subset of OpenCodeInstruct [1], consisting of natural-language programming prompts paired with reference solutions, and (b) unstructured coding uses a 40k-sample subset of SwallowCode [13], containing raw Python code without explicit instructions. The former is verifiable via unit tests; the latter is not.

We evaluate on HumanEval [2], MBPP [8], and MultiPL-E [7], reporting greedy accuracy (temperature 0) as well as `pass@1`, `pass@4`, and `pass@16` accuracy at temperature 0.6. HumanEval and MBPP are in-distribution for Q&A coding but out-of-distribution for unstructured code, since SwallowCode lacks explicit prompts. We evaluate MultiPL-E on eight languages (C++, JavaScript, TypeScript, Rust, C#, Go, PHP, and Java); since all training data is Python-only, it tests cross-lingual transfer.

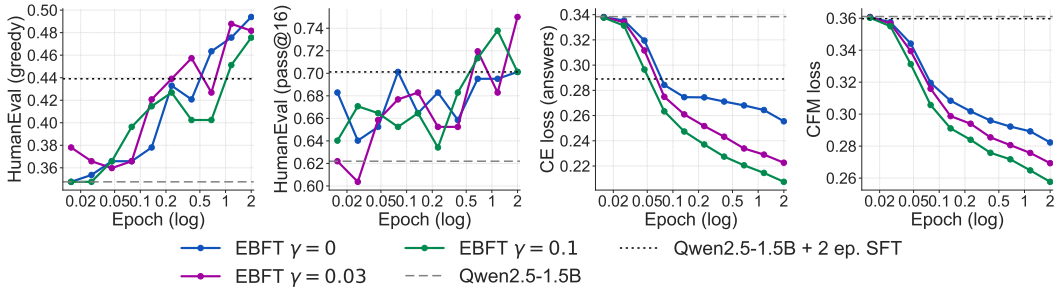

Figure 5: **The CE regularization weight γ controls cross-entropy reduction without affecting downstream performance or feature-matching loss.** Even pure feature matching ($\gamma = 0$) surpasses SFT on cross-entropy, confirming that the two objectives are aligned rather than in tension.

Translation. We train on a 100k subset of ALMA-Human-Parallel [34, 35], consisting of human-curated parallel sentence pairs. Following Xu et al. [34], we use WMT'22 as our primary evaluation benchmark, which covers news and general-domain translation. To test out-of-distribution robustness, we additionally evaluate on MTNT [24], consisting of noisy Reddit comments with typos, slang, and code-switching, and OpenSubtitles [23], containing short, informal movie and TV dialogue (both OOD relative to ALMA). We report COMET scores in the main text and BLEU in the appendix. For best-of-k evaluation ($k \in \{1, 4, 16\}$, temperature 0.6), we report the per-instance maximum aggregated over the test set.

We also track validation CE and feature-matching loss on 1k held-out samples (OpenCodeInstruct, ALMA), which are central to our analysis.

3.2 Baselines and methods

We evaluate three methods: (a) standard CE fine-tuning (SFT); (b) RLVR, where the reward is whether the generated code passes all unit tests for Q&A coding, and BLEU score for translation; and (c) EBFT with a frozen feature network. All methods are initialized from the base pre-trained model (Qwen2.5-1.5B [27] for coding and Llama3.2-1B [15] for translation). All EBFT runs use whitening as described in Equation (6). We run all methods for 2 epochs. As an additional variant, we report results for EBFT and RLVR initialized from a *warm-start* checkpoint obtained after one epoch of SFT, followed by one epoch of EBFT or RLVR. We include warm-start because RLVR requires it for competitive performance (Section 4). Hyperparameter details are provided in Section H.

4 Experimental results

We evaluate EBFT against SFT and RLVR on Q&A coding, unstructured coding, and translation. The main finding is that EBFT consistently matches or exceeds RLVR on downstream accuracy while achieving the best cross-entropy and feature-matching losses across all tasks — avoiding the tradeoff between task performance and distributional quality that characterizes RLVR. Table 1 summarizes the best results per method; Figures 3 and 12 show training dynamics for representative runs; Figures 5–6 report ablations. Full results across hyperparameter sweeps are provided in Section I.

4.1 Main results

EBFT matches RLVR and outperforms SFT on downstream accuracy. On Q&A coding (Table 1 and Figure 3), EBFT outperforms SFT by a wide margin across all decoding strategies (e.g., greedy: 0.548 vs 0.483, pass@16: 0.771 vs 0.728) and matches or exceeds RLVR (greedy: 0.548 vs 0.535, pass@16: 0.771 vs 0.752), despite not using any correctness signal. On unstructured code (Table 1), where RLVR is inapplicable, EBFT similarly outperforms SFT across all metrics (pass@1: 0.524 vs 0.467, pass@16: 0.769 vs 0.747). On translation (Table 1), EBFT outperforms both SFT and RLVR on COMET scores across all decoding strategies (e.g., greedy: 0.725 vs 0.717 for SFT and 0.697 for RLVR).

EBFT achieves lower cross-entropy than SFT, while RLVR degrades it. A striking finding is that EBFT reduces the validation cross-entropy more than SFT, even though SFT explicitly optimizes

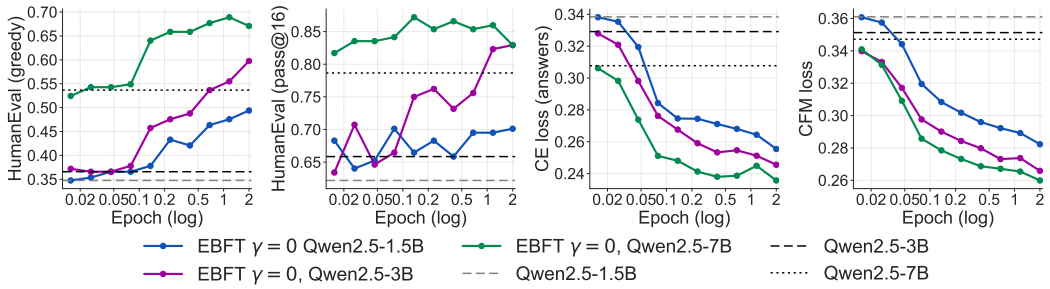

Figure 6: **EBFT improvements are consistent across model scales.** HumanEval accuracy, validation cross-entropy, and CFM loss over training for EBFT ($\gamma = 0$) applied to Qwen2.5-1.5B, 3B, and 7B. Each model uses a frozen copy of itself as the feature network. Dashed lines indicate base model performance. All three scales show substantial and qualitatively similar improvements across all four metrics, with no sign of diminishing returns.

this objective. On Q&A coding, EBFT achieves a validation CE of 0.207 compared to 0.289 for SFT (Table 1) and Figure 3 shows that this gap widens steadily over training. On translation, EBFT similarly outperforms SFT on CE (1.670 vs 1.782). On unstructured code, the two methods are comparable (0.499 vs 0.501). We attribute this counterintuitive result to EBFT with whitening approximately optimizing a relaxation of the χ^2 divergence, which is locally equivalent to the KL divergence when the model is close to the data distribution (see Section 2.2). RLVR exhibits the opposite behavior: its validation CE increases throughout training (Figure 3), reaching 0.774 on Q&A coding and 2.454 on translation — both substantially worse than the base model (0.338 and 1.870, respectively). This confirms that reward-driven optimization can improve downstream accuracy at the cost of severely degrading the model's language modeling quality, a tradeoff that EBFT avoids entirely.

EBFT achieves the lowest feature-matching loss. While it is natural to expect improvements on the feature-matching metric that EBFT directly optimizes, the margins are informative. On Q&A coding, EBFT achieves a feature-matching loss of 0.258, compared to 0.315 for SFT and 0.442 for RLVR (Table 1). RLVR not only fails to improve this metric but actively worsens it relative to the base model (0.361), and Figure 3 shows that this degradation accelerates over training. On translation, EBFT achieves the largest improvement (0.578 vs 0.690 for SFT and 0.641 for RLVR), while on unstructured code EBFT and SFT are comparable (0.320 vs 0.321). As shown in Figure 2, this improvement holds across all completion lengths and extends well beyond the 8-token rollout horizon used during training, suggesting that EBFT improves calibration of the rollout distribution broadly rather than overfitting to the training sequence length.

The CE loss coefficient γ improves CE loss without sacrificing downstream performance or feature matching. Figure 5 compares EBFT with $\gamma \in \{0, 0.03, 0.1\}$ on Q&A coding. The three settings achieve nearly identical feature-matching loss trajectories and comparable downstream performance on HumanEval, but differ markedly in how fast the validation cross-entropy decreases: larger γ drives it down faster, with $\gamma = 0.1$ reaching a CE of approximately 0.21 compared to 0.25 for $\gamma = 0$. All three settings surpass the 2-epoch SFT baseline on cross-entropy, confirming that even pure feature matching ($\gamma = 0$) reduces CE more effectively than directly optimizing it. The absence of any tension between these objectives is expected from a theoretical standpoint: as mentioned in Section 2.1, $\mathcal{L}_{\text{FM}}$ and $\mathcal{L}_{\text{CE}}$ share the same minimizer (the ground-truth distribution p), so optimizing feature matching naturally drives the cross-entropy down as well. The role of γ is simply to control how aggressively the CE loss is minimized, at no cost to calibration or downstream accuracy.

EBFT generalizes better than SFT to out-of-distribution benchmarks. On out-of-distribution coding languages (MultiPL-E benchmark performance in Table 6), SFT *degrades* performance relative to the base model (greedy: 0.465 vs 0.506), while EBFT yields improvements (0.524). On translation, EBFT outperforms both SFT and RLVR on the noisy MTNT benchmark (greedy COMET: 0.737 vs 0.703 and 0.705), while performing comparably on OpenSubtitles.

4.2 Ablations

Feature network ablations: mean pooling and removing whitening hurt most; random weights hurt slightly; a larger feature network has little effect. Figure 9 ablates key feature network design choices on Q&A coding at $\gamma = 0$. The default configuration (last-token features with whitening from a frozen copy of the 1.5B generator) achieves the best downstream performance and lowest feature-matching loss. Mean pooling and removing whitening cause the largest degradations, while random feature network weights hurt only modestly, indicating that pre-trained representations are helpful but not essential. Perhaps surprisingly, replacing the 1.5B feature network with a frozen Qwen2.5-7B produces similar results, suggesting that naively scaling the feature network does not yield additional gains.

EBFT improvements scale consistently across model sizes. To assess whether EBFT's benefits persist at larger scales, we run EBFT with $\gamma = 0$ using Qwen2.5-1.5B, 3B, and 7B as both actor and feature networks, each initialized from the respective base checkpoint. As shown in Figure 6, downstream improvements are consistent across model sizes: greedy HumanEval scores increase from approximately 0.49 (1.5B) to 0.60 (3B) to 0.69 (7B), with each model improving substantially over its respective base performance (0.35, 0.37, and 0.55). The same figure shows that both validation cross-entropy and feature-matching losses decrease faster and reach lower absolute values at larger scales, while preserving the same monotonic ordering across runs. These results suggest that EBFT's mechanism, which matches rollout feature statistics to ground-truth statistics, transfers predictably across model scales.

4.3 Qualitative analysis

Across both code and translation, EBFT outputs are more *semantically faithful* to the prompt and more *cleanly formatted*. We provide representative generations from HumanEval and MTNT translation in Sections I.2 and I.3; here we summarize the main patterns.

Each method exhibits a characteristic failure mode. SFT typically produces structurally reasonable outputs but misses subtle prompt requirements. For instance, when asked to count overlapping substring occurrences, SFT advances by the full substring length and misses overlaps; when asked to return the greatest integer satisfying a condition, SFT returns the first one it finds instead. RLVR often generates plausible logic but fails at the execution level: the generated code calls helper functions like `is_prime` without defining them, or includes prose explanations interleaved with code, preventing execution; translations begin with a reasonable output but then drift into multilingual tag lists (e.g., appending "Português: ...", "Spanish: ...") and truncate mid-word. EBFT avoids both failure modes, producing self-contained executable code and clean single-sentence translations that preserve the source meaning. These patterns suggest that the feature-matching objective encourages outputs that are both semantically faithful and cleanly formatted.

5 Conclusion

We introduced Energy-Based Fine-Tuning (EBFT), a method that fine-tunes language models by matching feature statistics of on-policy rollouts to those of ground-truth completions. Across Q&A coding, unstructured coding, and translation, EBFT consistently outperforms SFT and matches RLVR on downstream accuracy, while achieving the best cross-entropy and feature-matching losses. Notably, EBFT reduces cross-entropy more than SFT despite not directly optimizing it. Unlike RLVR, EBFT requires no task-specific reward or verifier, making it applicable in non-verifiable settings where RLVR cannot be used.

EBFT connects classical ideas from moment matching and distribution alignment with modern language model training. By operating in a feature space rather than over tokens or scalar rewards, it provides a flexible mechanism for shaping sequence-level behavior. However, EBFT is rollout-based and therefore slower per update than standard SFT, making it most suitable as a fine-tuning stage applied after cross-entropy training. It also requires a frozen feature network and has so far been evaluated on models up to 7B parameters with short rollout horizons. Scaling both axes, and exploring learned or adaptive feature networks, are promising directions for future work. More broadly, we view feature matching as a complementary training signal that may help bridge likelihood-based training and rollout-based optimization.

References

[1] Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. Opencodeinstruct: A large-scale instruction tuning dataset for code llms. *arXiv preprint arXiv:2504.04030*, 2025.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[4] Mark Braverman, Xinyi Chen, Sham M. Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models. *arXiv preprint arXiv:1906.05664*, 2019.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Locke Cai and Ivan Provilkov. Escaping the verifier: Learning to reason via demonstrations. *arXiv preprint arXiv:2511.21667*, 2025. URL https://arxiv.org/abs/2511.21667.

[7] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. Multiple: A scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691, 2023.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[9] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

[10] Carles Domingo-Enrich, Alberto Bietti, Marylou Gabrié, Joan Bruna, and Eric Vanden-Eijnden. Dual training of energy-based models with overparametrized shallow neural networks. *arXiv preprint arXiv:2107.05134*, 2022.

[11] Qingxiu Dong, Dong Li, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

[12] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Classical structured prediction losses for sequence to sequence learning. In *ACL 2018*, 2018.

[13] Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Ohi, Masaki Kawamura, Taishi Nakamura, Takumi Okamoto, Shigeki Ishida, Kakeru Hattori, Youmi Ma, Hiroya Takamura, Rio Yokota, and Naoaki Okazaki. Rewriting pre-training data boosts llm performance in math and code, 2025. URL https://arxiv.org/abs/2505.02881.

[14] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[15] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[16] Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Rlp: Reinforcement as a pretraining objective. *arXiv preprint arXiv:2510.01265*, 2025.

[17] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[18] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[19] Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *arXiv preprint arXiv:1610.09038*, 2016.

[20] Siheng Li, Kejiao Li, Zenan Xu, Guanhua Huang, Evander Yang, Kun Li, Haoyuan Wu, Jiajia Wu, Zihao Zheng, Chenchen Zhang, Kun Shi, Kyrierl Deng, Qi Yi, Ruibin Xiong, Tingqiang Xu, Yuhao Jiang, Jianfeng Yan, Yuyuan Zeng, Guanghui Xu, Jinbao Xue, Zhijiang Xu, Zheng Fang, Bo Chao Wang, Qibin Liu, Xiaoxue Li, and Yangyu Tao. Reinforcement learning on pre-training data. *arXiv preprint arXiv:2509.19249*, 2025.

[21] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[23] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, 2016.

[24] Paul Michel and Graham Neubig. Mtnt: A testbed for machine translation of noisy text. *arXiv preprint arXiv:1809.00388*, 2018.

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[26] Aaditya Vikram Prasad, Connor Watts, Jack Merullo, Dhruvil Gala, Owen Lewis, Thomas McGrath, and Ekdeep Singh Lubana. Features as rewards: Scalable supervision for open-ended tasks via interpretability. *arXiv preprint arXiv:2602.10067*, 2026.

[27] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[28] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, and Christopher D Manning. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[29] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2016.

[30] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[32] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[33] Yusong Wu, Stephen Brade, Teng Ma, Tia-Jane Fowler, Enning Yang, Berker Banar, Aaron Courville, Natasha Jaques, and Cheng-Zhi Anna Huang. Generative adversarial post-training mitigates reward hacking in live human-ai music interaction. *arXiv preprint arXiv:2511.17879*, 2025. doi: 10.48550/arXiv.2511.17879. URL https://arxiv.org/abs/2511.17879.

[34] Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674*, 2023.

[35] Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation, 2024.

[36] Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-based diffusion language models for text generation. In *ICLR 2025*, 2024. arXiv:2410.21357.

[37] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

[38] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[39] Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

[40] Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, Qizheng Zhang, Lin Chen, Fanghao Shao, Bo Xue, Yunchong Song, Zhenjie Yang, Ganqu Cui, Ning Ding, Jianfeng Gao, Xiaodong Liu, Hongyuan Zhou, and Zhouhan Mei. Flowrl: Matching reward distributions for llm reasoning. *arXiv preprint arXiv:2509.15207*, 2025.

A Related Work

Most language model training pipelines remain centered on next-token maximum likelihood (MLE), with reinforcement learning (RL) typically applied as a post-training step. RLHF-style methods optimize sequence-level rewards while regularizing toward a reference policy, often via a KL constraint [9, 25], and preference-optimization approaches such as DPO can be interpreted as reward maximization under a similar regularization [28]. Earlier sequence-level training methods likewise augment cross-entropy training with REINFORCE-style updates, but continue to rely on token-level supervision [29, 12].

Recent work has explored using RL earlier in training or framing pretraining objectives in RL terms. RLP [16], Reinforcement Pre-Training (RPT) [11], and RLPT [20] introduce rewards tied to reasoning traces, information gain, or next-segment prediction. However, in all cases the reward signal is ultimately derived from next-token likelihood or correctness on the pretraining stream, rather than from a distinct semantic objective. Similarly, FlowRL proposes matching the full reward distribution to encourage diversity, but still defines rewards through likelihood-based or task-specific signals [40].

Closely related are methods that derive reward signals from internal model representations rather than external verifiers. Generative Adversarial Post-Training (GAPT) employs a co-evolving discriminator to mitigate reward hacking in interactive generation [33]. RARO [6] uses a relativistic discriminator within an inverse reinforcement learning framework to recover implicit rewards from expert reasoning demonstrations. Concurrently, RLFR [26] trains lightweight probes on internal model activations to detect hallucinated claims and uses the probe output as a reward signal for reinforcement learning. While motivated by different applications, all three methods reduce rich representations to learned, task-specific scalar rewards. In contrast, our approach avoids learned reward models entirely, instead optimizing a fixed, vector-valued feature-matching objective that directly aligns rollout and data distributions in a general-purpose feature space.

Alternative generative frameworks aim to move beyond left-to-right likelihood training. Energy-Based Diffusion Language Models (EDLM) and related energy-based approaches operate at the sequence level [36], but focus on modeling the data distribution itself rather than defining a feature-space alignment objective for an autoregressive policy. Embedding-based similarity has been widely used for evaluation (e.g., BERTScore [38]) and occasionally optimized via RL for metric-driven fine-tuning [30], but not as a general replacement for teacher-forced token prediction.

In contrast, our approach decouples training from surface-form tokens entirely. We define rewards via a learned feature network and optimize an autoregressive policy using REINFORCE to match generated and ground-truth text in embedding space. This provides dense, semantic feedback that does not depend on next-token log loss, enabling sequence-level optimization that directly targets meaning rather than token reconstruction.

B The feature-matching loss profile and its optimal behavior

Figures 1 and 2 show the conditional feature-matching loss defined in (2), plotted against the completion length G. We refer to the function that maps G to the corresponding (conditional) feature-matching loss value as the *(conditional) feature-matching loss profile*. To compute conditional feature-matching loss values, we extract ground-truth pairs (c, y) from long ground-truth token sequences by *selecting a strided set of prefixes of the sequence as the contexts c, and the ensuing windows of length G as the completions y*. The bias-variance decomposition (2) directly implies that the minimum value of $\mathcal{L}_{\mathrm{CFM}}$ is $\mathbb{E}_{c\sim p}\big[\mathrm{Var}[\phi_c(y)|c]\big]$. The following lemma shows that $\mathbb{E}_{c\sim p}\big[\mathrm{Var}[\phi_c(y)|c]\big]$ is non-decreasing with the completion length G.

Lemma B.1 (The optimal conditional feature-matching profile)**.** *Consider the assumptions*

(a) $\phi_c(y) := \phi(c{:}y)$ depends on c and y only through their concatenation $c{:}y$, which follows the construction in Section 2.1.

(b) Context-completion pairs (c, y) are selected/sampled from long ground-truth sequences as described above, such that for a fixed completion length G, concatenations $c : y$ are equally distributed to contexts c. Neglecting edge effects that stem from the long ground-truth

sequences being finite, this holds for example if contexts are sampled as random prefixes of the ground-truth sequence.

Then, the optimal conditional feature-matching loss profile $\mathbb{E}_{c\sim p}\big[\mathrm{Var}[\phi_c(y)|c]\big]$ *is non-decreasing with* G *and admits the bound*

$$\mathbb{E}_{c\sim p}\big[\mathrm{Var}[\phi_c(y)|c]\big] \leq \mathrm{Var}_{c\sim p}[\phi(c)]. \tag{8}$$

Proof. Consider completion lengths $1 \leq G' \leq G$. Let $y, \hat{y}$ denote sequences of length G, $y', \hat{y}'$ completions of length G', and $y'', \hat{y}''$ completions of length $G'' = G - G'$, which means that we write the optimal conditional feature-matching loss at completion lengths G' and G as $\mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y'\sim p(\cdot|c)}[\phi(c{:}y')|c]\big]$ and $\mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y\sim p(\cdot|c)}[\phi(c{:}y)|c]\big]$, respectively. Observe that

$$\begin{aligned}\mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y\sim p(\cdot|c)}[\phi_c(y)|c]\big] &= \mathbb{E}_{(c,y)\sim p}\Big[\big\|\phi_c(y)\big\|^2\Big] - \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big\|^2\Big] \\ &= \mathbb{E}_{c\sim p}\Big[\big\|\phi(c)\big\|^2\Big] - \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y''\sim p(\cdot|c)}\big[\mathbb{E}_{y'\sim p(\cdot|c:y'')}[\phi(c{:}y''{:}y')]\big]\big\|^2\Big]\end{aligned} \tag{9}$$

Here, the second equality holds because $c : y$ is equally distributed to c by Assumption *(b)*, and using the tower property of expectation together with the decomposition $y = y'' : y'$. By Jensen's inequality, we have that

$$\begin{aligned}&\mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y''\sim p(\cdot|c)}\big[\mathbb{E}_{y'\sim p(\cdot|c:y'')}[\phi(c{:}y''{:}y')]\big]\big\|^2\Big] \leq \mathbb{E}_{c\sim p}\Big[\mathbb{E}_{y''\sim p(\cdot|c)}\Big[\big\|\mathbb{E}_{y'\sim p(\cdot|c:y'')}[\phi(c{:}y''{:}y')]\big\|^2\Big]\Big] \\ &= \mathbb{E}_{(c,y'')\sim p}\Big[\big\|\mathbb{E}_{y'\sim p(\cdot|c:y'')}[\phi(c{:}y''{:}y')]\big\|^2\Big] = \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y'\sim p(\cdot|c)}[\phi(c{:}y')]\big\|^2\Big]\end{aligned} \tag{10}$$

Plugging this back into the right-hand side of (9) yields

$$\begin{aligned}\mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y\sim p(\cdot|c)}[\phi_c(y)|c]\big] &\geq \mathbb{E}_{c\sim p}\Big[\big\|\phi(c)\big\|^2\Big] - \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y'\sim p(\cdot|c)}[\phi(c{:}y')]\big\|^2\Big] \\ &= \mathbb{E}_{(c,y)\sim p}\Big[\big\|\phi_c(y)\big\|^2\Big] - \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y'\sim p(\cdot|c)}[\phi(c{:}y')]\big\|^2\Big] \\ &= \mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y'\sim p(\cdot|c)}[\phi_c(y')|c]\big],\end{aligned} \tag{11}$$

which concludes the proof that the optimal conditional feature-matching loss is non-decreasing with the completion length G. To prove the bound (8), we apply Jensen's inequality in the opposite direction:

$$\mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big\|^2\Big] \geq \big\|\mathbb{E}_{(c,y)\sim p}[\phi_c(y)]\big\|^2, \tag{12}$$

and plugging this into (9) yields

$$\begin{aligned}&\mathbb{E}_{c\sim p}\big[\mathrm{Var}_{y\sim p(\cdot|c)}[\phi_c(y)|c]\big] \leq \mathbb{E}_{(c,y)\sim p}\Big[\big\|\phi_c(y)\big\|^2\Big] - \big\|\mathbb{E}_{(c,y)\sim p}[\phi_c(y)]\big\|^2 \\ &= \mathrm{Var}_{(c,y)\sim p}[\phi(c{:}y)] = \mathrm{Var}_{c\sim p}[\phi(c)]\end{aligned} \tag{13}$$

□

C Feature matching with whitening

This section motivates a *whitened* variant of feature matching by connecting standard cross-entropy training to a local χ^2 objective. Then, we relax a variational formulation of the χ^2 divergence by restricting the function space to a generalized linear model space corresponding to a chosen feature map. The resulting optimization problem admits a closed form and corresponds to the feature-matching loss with whitening, which amounts to premultiplying the feature vectors by the inverse of the matrix of second moments. However, in practice we only have access to a low-rank approximation of this matrix, which means that we can only compute a pseudo-inverse. We describe different empirical loss variants that we tried, including the one that we used to obtain the results in the main paper.

C.1 Relating cross-entropy training to a χ^2 divergence objective

Fix a context c and consider completions $y \in \mathcal{V}^G$ (for some completion length G). Let $p(\cdot \mid c)$ denote the ground-truth conditional distribution over completions and $p_\theta(\cdot \mid c)$ the model distribution. Cross-entropy training minimizes the conditional KL divergence

$$D_{\mathrm{KL}}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) = \sum_{y \in \mathcal{V}^G} p(y \mid c) \log \frac{p(y \mid c)}{p_\theta(y \mid c)}. \tag{14}$$

Rewriting (14) as an expectation under p_θ gives

$$D_{\mathrm{KL}}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) = \sum_{y \in \mathcal{V}^G} p_\theta(y \mid c) \, \frac{p(y \mid c)}{p_\theta(y \mid c)} \log \frac{p(y \mid c)}{p_\theta(y \mid c)}. \tag{15}$$

The first-order Taylor expansion of $x \mapsto x \log x$ around $x = 1$ is

$$x \log x = (x-1) + \tfrac{1}{2}(x-1)^2 + O\big((x-1)^3\big). \tag{16}$$

Plugging (16) into (15) yields

$$\begin{aligned} D_{\mathrm{KL}}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) &= \tfrac{1}{2} \sum_{y \in \mathcal{V}^G} p_\theta(y \mid c) \Big(\tfrac{p(y|c)}{p_\theta(y|c)} - 1\Big)^2 + \sum_{y \in \mathcal{V}^G} p_\theta(y \mid c)\, O\Big(\Big(\tfrac{p(y|c)}{p_\theta(y|c)} - 1\Big)^3\Big) \\ &= \tfrac{1}{2}\, D_{\chi^2}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) + \sum_{y \in \mathcal{V}^G} p_\theta(y \mid c)\, O\Big(\Big(\tfrac{p(y|c)}{p_\theta(y|c)} - 1\Big)^3\Big), \end{aligned} \tag{17}$$

where the χ^2 divergence is

$$D_{\chi^2}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) := \sum_{y \in \mathcal{V}^G} \frac{\big(p(y \mid c) - p_\theta(y \mid c)\big)^2}{p_\theta(y \mid c)} = \mathbb{E}_{Y \sim p_\theta(\cdot|c)}\Big[\Big(\tfrac{p(Y|c)}{p_\theta(Y|c)} - 1\Big)^2\Big]. \tag{18}$$

When $p(\cdot \mid c) \approx p_\theta(\cdot \mid c)$ (so the ratio p/p_θ is close to 1), the remainder term in (17) can be neglected, and we obtain the local approximation

$$D_{\mathrm{KL}}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big) \approx \tfrac{1}{2}\, D_{\chi^2}\big(p(\cdot \mid c) \,\|\, p_\theta(\cdot \mid c)\big). \tag{19}$$

C.2 Relaxing a variational formulation of the χ^2 divergence to a linear feature class

The representation (18) makes explicit that D_{χ^2} corresponds to an L^2 discrepancy in the space of one-hot features. Next, we want to express the χ^2 divergence, or rather a relaxation of it, in terms of generic feature maps. For that, we consider a variational representation of the χ^2 divergence.

Lemma C.1 (Variational representation of the chi-squared divergence)**.** *Let P and Q be probability measures on a measurable space $\mathcal{Y}$ such that $P \ll Q$, and write their Radon–Nikodym derivative as $r(y) = \frac{dP}{dQ}(y)$. Then*

$$D_{\chi^2}(P\|Q) = \sup_{f \in L^2(Q),\, \mathbb{E}_{Y\sim Q}[f(Y)]=0} \Big\{ 2\,(\mathbb{E}_{Y\sim P}[f(Y)] - \mathbb{E}_{Y\sim Q}[f(Y)]) - \mathbb{E}_{Y\sim Q}[f(Y)^2] \Big\}, \tag{20}$$

Moreover, the supremum is attained at $f^\star(y) = r(y) - 1$.

Proof. Define $g(y) = \frac{dP}{dQ}(y)$. Note that $\mathbb{E}_Q[r(Y)] = 1$ and $\chi^2(P\|Q) = \mathbb{E}_Q\big[(g(Y)-1)^2\big]$. For any $f \in L^2(Q)$,

$$2\,(\mathbb{E}_P[f(Y)] - \mathbb{E}_Q[f(Y)]) - \mathbb{E}_Q[f(Y)^2] = 2\,\mathbb{E}_Q[(g(Y)-1)f(Y)] - \mathbb{E}_Q[f(Y)^2] = \mathbb{E}_Q\big[2(g(Y)-1)f(Y) - f(Y)^2\big]. \tag{21}$$

Completing the square pointwise gives $2(g-1)f - f^2 = -(f-g+1)^2 + (r-1)^2$, hence

$$2\,(\mathbb{E}_P[f(Y)] - \mathbb{E}_Q[f(Y)]) - \mathbb{E}_Q[f(Y)^2] = \mathbb{E}_Q[(g(Y)-1)^2] - \mathbb{E}_Q[(f(Y)-g(Y)+1)^2] \le \mathbb{E}_Q[(g(Y)-1)^2], \tag{22}$$

with equality iff $f = g - 1$. Therefore,

$$\sup_{f \in L^2(Q)} \left\{ 2\left(\mathbb{E}_P[f(Y)] - \mathbb{E}_Q[f(Y)]\right) - \mathbb{E}_Q[f(Y)^2] \right\} = \mathbb{E}_Q[(g(Y)-1)^2] = D_{\chi^2}(P\|Q). \quad (23)$$

where the optimum is achieved at $f(x) = g(x) - 1$. □

We particularize Lemma C.1 in the language model setting. Let $\varphi : \mathcal{V}^G \to \mathbb{R}^d$ be a feature map over completions (in our setting, the natural choice is $\varphi(y) = \phi_c(y)$, the feature-network embedding of the concatenated sequence). Restricting the supremum in (20) to generalized linear model $f_w(y) = w^\top \varphi(y)$ yields the relaxation

$$D_{\chi^2}(P\|Q) \geq \sup_{w \in \mathbb{R}^d} \left\{ 2\left(\mathbb{E}_{Y\sim P}\left[w^\top \varphi(Y)\right] - \mathbb{E}_{Y\sim Q}\left[w^\top \varphi(Y)\right]\right) - \mathbb{E}_{Y\sim Q}\left[(w^\top \varphi(Y))^2\right] \right\}. \quad (24)$$

We can rewrite the expression in the supremum as follows:

$$2\left(\mathbb{E}_{Y\sim P}\left[w^\top \varphi(Y)\right] - \mathbb{E}_{Y\sim Q}\left[w^\top \varphi(Y)\right]\right) - \mathbb{E}_{Y\sim Q}\left[\left(w^\top \varphi(Y)\right)^2\right] = 2w^\top(\mu_P - \mu_Q) - w^\top \Sigma_Q w, \quad (25)$$

where $\mu_P = \mathbb{E}_{Y\sim P}[\varphi(Y)]$, $\mu_Q = \mathbb{E}_{Y\sim Q}[\varphi(Y)]$ and $\Sigma_Q = \mathbb{E}_{Y\sim Q}\left[\varphi(Y)\varphi(Y)^\top\right]$. When Σ_Q is invertible, the supremum in (24) is attained at

$$\hat{w} = \Sigma_Q^{-1}(\mu_P - \mu_Q), \quad (26)$$

and the optimal value is

$$\sup_w \{2w^\top(\mu_P - \mu_Q) - w^\top \Sigma_Q w\} = (\mu_P - \mu_Q)\,\Sigma_Q^{-1}(\mu_P - \mu_Q). \quad (27)$$

Thus, assuming that $\Sigma_{p_\theta(\cdot|c)} = \mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]$ is invertible, we define the whitened feature matching loss as

$$\begin{aligned}\mathcal{L}_{\mathrm{WFM}}(\theta) = \mathbb{E}_{c\sim p}\Big[&\left(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\right)^\top \\ &\times \mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]^{-1}\left(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\right)\Big],\end{aligned} \quad (28)$$

Observe that the dependence of $\mathcal{L}_{\mathrm{WFM}}(\theta)$ on θ is through $\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})]$ as in $\mathcal{L}_{\mathrm{FM}}(\theta)$, but also through $\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]^{-1}$. While applying the REINFORCE argument to estimate the gradient for the former can be done as in Section 2, the approach breaks down for the latter. We decide to disregard the gradient with respect to $\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]^{-1}$, which amounts to the following loss:

$$\begin{aligned}\mathcal{L}_{\mathrm{WFM}}(\theta) = \mathbb{E}_{c\sim p}\Big[&\left(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\right)^\top \\ &\times \mathrm{stopgrad}\left(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]\right)^{-1}\left(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\right)\Big],\end{aligned} \quad (29)$$

C.3 Dealing with non-invertible second moment matrices $\Sigma_{p_\theta(\cdot|c)}$: a first approach

The matrix $\Sigma_{p_\theta(\cdot|c)} = \mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\left[\phi_c(\hat{y})\phi_c(\hat{y})^\top\right]$ and especially its empirical version $\hat{\Sigma}_{p_\theta(\cdot|c)} = \frac{1}{n}\sum_{j=1}^n \phi_c(\hat{y}_j)\phi_c(\hat{y}_j)^\top$ are often not invertible, in particular when the feature dimension d is high and/or the number of samples per prompt n is low. In particular, the rank of the empirical matrix is upper-bounded by the number of samples per prompt, meaning that it is never invertible when $n < d$, which is usually the case. Thus, we need to solve $\sup_w\{2w^\top(\mu_P - \mu_Q) - w^\top \Sigma_Q w\}$ for general positive semidefinite Σ_Q. More generally, consider maximizing a functional of the form $f(w) = 2\langle w, b\rangle - \langle w, \Sigma w\rangle$. Let $\Sigma = \sum_{i=1}^r \lambda_i u_i u_i^\top$ be an eigendecomposition, and write $w = \sum_i \alpha_i u_i$ and $b = \sum_i \beta_i u_i$. Then

$$f(w) = 2\sum_i \alpha_i \beta_i - \sum_i \lambda_i \alpha_i^2. \quad (30)$$

Splitting into nonzero and zero eigenvalues yields the dichotomy:

(i) If b has any component in $\ker(\Sigma)$ (i.e., there exists i with $\lambda_i = 0$ and $\beta_i \neq 0$), then $\sup_w f(w) = +\infty$ and there is no maximizer in $\mathbb{R}^d$.

(ii) If $b \perp \ker(\Sigma)$, i.e. $b \in \mathrm{Im}(\Sigma)$, then the supremum is finite and equals

$$\sup_w f(w) = \sum_{\lambda_i > 0} \frac{\beta_i^2}{\lambda_i} = b^\top \Sigma^\dagger b, \tag{31}$$

where $\Sigma^\dagger = \sum_{\lambda_i>0} \frac{1}{\lambda_i} u_i u_i^\top$ is the Moore–Penrose pseudoinverse.

In our case, $b = \mu_P - \mu_Q$ and $\Sigma = \Sigma_Q$. While $\mu_Q \perp \ker(\Sigma_Q)$ by construction, in general μ_P will have non-zero components in $\ker(\Sigma)$, and in that case $\sup_w\{2w^\top(\mu_P - \mu_Q) - w^\top \Sigma_Q w\} = +\infty$. This is not surprising given that when the inequality (24) holds with equality, which is the case for one-hot feature maps, $\sup_w\{2w^\top(\mu_P - \mu_Q) - w^\top \Sigma_Q w\} = \chi^2(P\|Q)$, and it is easy to see that $\chi^2(P\|Q) = +\infty$ when the support of P is larger than the support of Q. To obtain a finite value, it is convenient to replace μ_P by the projection $\mathrm{Pr}_{\mathrm{Im}(\Sigma_Q)}\mu_P$. Then, by equation (31) we obtain

$$\sup_w\{2w^\top(\mathrm{Pr}_{\mathrm{Im}(\Sigma_Q)}\mu_P - \mu_Q) - w^\top \Sigma_Q w\} = (\mathrm{Pr}_{\mathrm{Im}(\Sigma_Q)}\mu_P - \mu_Q)^\top \Sigma_Q^\dagger (\mathrm{Pr}_{\mathrm{Im}(\Sigma_Q)}\mu_P - \mu_Q) \tag{32}$$

$$= (\mu_P - \mu_Q)^\top \Sigma_Q^\dagger (\mu_P - \mu_Q), \tag{33}$$

where the last equality holds because $\ker(\Sigma_Q^\dagger) = \ker(\Sigma_Q)$, and that $\mu_P = \mathrm{Pr}_{\mathrm{Im}(\Sigma)}\mu_P + \mathrm{Pr}_{\ker(\Sigma)}\mu_P$. Hence, the following loss function is numerically robust:

$$\mathcal{L}_{\mathrm{WFM}}(\theta) = \mathbb{E}_{c\sim p}\Big[\big(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big)^\top \times \mathrm{stopgrad}\big(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\big[\phi_c(\hat{y})\phi_c(\hat{y})^\top\big]\big)^\dagger \big(\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big)\Big], \tag{34}$$

It is easy to compute the gradient of this loss through the framework of Section 2; in the computation of the population reward $r(\hat{y}, c)$ in (4) we simply replace the features $\phi_c(y)$ by the whitened features:

$$\tilde{\phi}_c(y) = (\Sigma^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(y), \qquad \Sigma_{p_\theta(\cdot|c)} = \mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\big[\phi_c(\hat{y})\phi_c(\hat{y})^\top\big], \tag{35}$$

and in practice, we compute the reward r_j in (5) using $\hat{\Sigma}_{p_\theta(\cdot|c)}$ instead of $\Sigma_{p_\theta(\cdot|c)}$:

$$\tilde{\phi}_c(y) = (\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(y), \qquad \hat{\Sigma}_{p_\theta(\cdot|c)} = \frac{1}{n}\sum_{j=1}^n \phi_c(\hat{y}_j)\phi_c(\hat{y}_j)^\top. \tag{36}$$

Above, $(\Sigma^\dagger)^{1/2}$ denotes the square root of the pseudo-inverse of Σ, i.e. if Σ admits the eigenvalue decomposition $\Sigma = \sum_{i=1}^d \lambda_i u_i u_i^\top$, then $(\Sigma^\dagger)^{1/2} = \sum_{i=1,\lambda_i>0}^d \lambda_i^{-1/2} u_i u_i^\top$. In practice, using a function to compute the singular value decomposition of is more numerically stable that using a function to compute the eigenvalue decomposition.

C.4 Variants of the whitened feature-matching loss with better empirical performance

Let us write the reward r_j explicitly under whitening:

$$r_j = \underbrace{2\phi_c(\hat{y}_j)^\top \hat{\Sigma}^\dagger_{p_\theta(\cdot|c)}\phi_c(y)}_{\text{alignment term } \mathrm{AT}_j} - \underbrace{\frac{2}{n-1}\sum_{j'\neq j}\phi_c(\hat{y}_j)^\top \hat{\Sigma}^\dagger_{p_\theta(\cdot|c)}\phi_c(\hat{y}_{j'})}_{\text{diversity term } \mathrm{DT}_j}. \tag{37}$$

Suppose that the features $\big(\phi_c(\hat{y}_j)\big)_{j=1}^n$ of the generated completions are ordered such that repeated completions are arranged consecutively, and that there are exactly K different feature vectors among $\big(\phi_c(\hat{y}_j)\big)_{j=1}^n$, with multiplicities $(n_k)_{k=1}^K$, such that $\sum_{k=1}^K n_k = n$. In this section, we make the following assumptions, which hold in practice:

(i) The feature dimension d is larger or equal than the number of generated completions n. This holds in our experiments, because d is on the order of thousands, while we take $n = 4$.

(ii) The K different feature vectors within $(\phi_c(\hat{y}_j))_{j=1}^n$ are linearly independent. This happens with very high probability in our experiments, also as a consequence of $d \gg n$.

For $1 \leq k \leq K$, let $j_k = \sum_{k'=1}^{k-1} n_{k'} + 1$. Hence, the list of feature vectors $\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$ contains each instance in $\big(\phi_c(\hat{y}_j)\big)_{j=1}^n$ with multiplicity one. We define the matrices

- $\Phi \in \mathbb{R}^{d\times n}$ as the matrix whose columns are $(\phi_c(\hat{y}_j))_{j=1}^n$, i.e. $\Phi_{\cdot j} = \phi_c(\hat{y}_j)$,
- $\hat{\Phi} \in \mathbb{R}^{d\times K}$ as the matrix whose columns are $\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$,
- $\bar{\Phi} \in \mathbb{R}^{d\times K}$ as the matrix whose columns are $\big(\sqrt{n_k}\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$,
- $\tilde{\Phi} = ((\Phi\Phi^\top)^\dagger)^{1/2}\Phi \in \mathbb{R}^{d\times n}$,

And the vectors

- $\psi = \phi_c(y) \in \mathbb{R}^d$,
- $\tilde{\psi} = ((\Phi\Phi^\top)^\dagger)^{1/2}\psi \in \mathbb{R}^d$,
- $x^{(\psi)} = \hat{\Phi}^\dagger\psi \in \mathbb{R}^K$, which is the vector of coefficients of the orthogonal projection of ψ onto $\operatorname{span}\big(\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K\big)^\perp$ with respect to the basis $\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$,

We can reexpress the alignment and diversity terms in (37) with respect to $\tilde{\Phi}$ and $\tilde{\psi}$:

$$\mathrm{AT}_j = 2n\phi_c(\hat{y}_j)^\top(\Phi\Phi^\top)^\dagger\phi_c(y) = 2n\tilde{\Phi}_{\cdot j}^\top\tilde{\psi}, \tag{38}$$

$$\mathrm{DT}_j = \frac{2n}{n-1}\sum_{j'\neq j}\phi_c(\hat{y}_j)^\top(\Phi\Phi^\top)^\dagger\phi_c(\hat{y}_{j'}) = \frac{2n}{n-1}\sum_{j'\neq j}\tilde{\Phi}_{\cdot j}^\top\tilde{\Phi}_{\cdot j'}. \tag{39}$$

The following lemma, proven in Section C.4.1, characterizes the inner products $\tilde{\Phi}_{\cdot j}^\top\tilde{\Phi}_{\cdot j'}$ and $\tilde{\Phi}_{\cdot j'}^\top\tilde{\psi}$, and the norm of $\tilde{\psi}$.

Lemma C.2. *Recall that n_{k_j} is the multiplicity of the completion $\hat{y}_j$ within $\big(\phi_c(\hat{y}_j)\big)_{j=1}^n$. The inner products between the columns $(\tilde{\Phi}_{\cdot j})_{j=1}^n$ are given by*

$$\tilde{\Phi}_{\cdot j}^\top\tilde{\Phi}_{\cdot j'} = 1/n_{k_j} \qquad \textit{for all } j' \textit{ such that } \phi_c(\hat{y}_j) = \phi_c(\hat{y}_{j'}) \textit{ (in particular } \|\tilde{\Phi}_{\cdot j}\| = 1/\sqrt{n_{k_j}}\textit{)}, \tag{40}$$

$$\tilde{\Phi}_{\cdot j}^\top\tilde{\Phi}_{\cdot j'} = 0 \qquad \textit{for all } j' \textit{ such that } \hat{y}_j \neq \hat{y}_{j'}. \tag{41}$$

And we have that

$$\tilde{\Phi}_{\cdot j}^\top\tilde{\psi} = \frac{x_j^{(\psi)}}{n_{k_j}}, \qquad \textit{for all } 1 \leq j \leq n\textit{, and,} \qquad \|\tilde{\psi}\|^2 = \sum_{k=1}^K \frac{(x_k^{(\psi)})^2}{n_k}. \tag{42}$$

Thus, under the conditions of Lemma C.2,

$$\mathrm{AT}_j = \frac{2nx_j^{(\psi)}}{n_{k_j}}, \tag{43}$$

$$\mathrm{DT}_j = \frac{2n}{n-1}\sum_{j'\neq j}\mathbf{1}[\phi_c(\hat{y}_j) = \phi_c(\hat{y}_{j'})]\frac{1}{n_{k_j}} = \frac{2n(n_{k_j}-1)}{(n-1)n_{k_j}} = \frac{2(1-1/n_{k_j})}{1-1/n}. \tag{44}$$

where $x_j^{(\psi)}$ is the j-th component of $x^{(\psi)}$.

Observe that when $\phi_c(y)$ is equal to $\phi_c(\hat{y}_{j_k})$ for some $1 \leq k \leq K$, $x^{(\psi)}$ is the k-th vector of the canonical basis of $\mathbb{R}^K$, which means that $\mathrm{AT}_j = \frac{2n}{n_{k_j}}$ for all j such that $\phi_c(y)$ is equal to $\phi_c(\hat{y}_j)$, and zero otherwise.

When $\phi_c(y)$ is different from all the vectors in $\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$, the norm of the projection of $\phi_c(y)$ onto $\operatorname{span}\big(\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K\big)$ is usually significantly smaller than the norm of $\phi_c(y)$, because this subspace is

smaller than the ambient dimension as $K \leq n \leq d$, and this is accentuated the smaller n is. Observe that in optimizing the empirical rewards $r_j = \mathrm{AT}_j - \mathrm{DT}_j$, the model balances increasing the alignment term AT_j and decreasing the diversity term DT_j, and the specific trade-off is determined by the relative sizes of both terms. Since for small n the alignment terms $(\mathrm{AT}_j)_{j=1}^n$ are small because $x^{(\psi)}$ is abnormally small, the model focuses on decreasing DT_j as opposed to strongly improving AT_j. Experimentally, this translates to moderate improvements in downstream performance. To achieve more solid boosts in downstream performance, we tested the following alternative approaches:

- *Variant (i)*: **Normalizing the whitened features of the generations and the ground truth in the alignment term**. As a result, even when $x^{(\psi)}$ is small, the alignment term is not. Namely, we set the diversity term DT_j as in (44), and the alignment term AT_j as follows:

$$\mathrm{AT}_j = \frac{2\phi_c(\hat{y}_j)^\top \hat{\Sigma}^\dagger_{p_\theta(\cdot|c)} \phi_c(y)}{\|(\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(\hat{y}_j)\| \|(\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(y)\|} = \frac{2\tilde{\Phi}_{\cdot j}^\top \tilde{\psi}}{\|\tilde{\Phi}_{\cdot j}\| \|\tilde{\psi}\|} = \frac{2x_j^{(\psi)}}{\sqrt{n_{k_j} \sum_{k=1}^K \frac{(x_k^{(\psi)})^2}{n_k}}}. \tag{45}$$

  Observe that we can write the vector of alignment terms $(\mathrm{AT}_j)_{j=1}^n$ as

$$(\mathrm{AT}_j)_{j=1}^n = \frac{2(x_j^{(\psi)}/\sqrt{n_{j_k}})_{j=1}^n}{\|(x_j^{(\psi)}/\sqrt{n_{j_k}})_{j=1}^n\|}. \tag{46}$$

- *Variant (ii)*: **Normalizing the whitened features of the generations and the ground truth in the alignment and diversity terms**. We set AT_j as in (45), and DT_j as follows:

$$\begin{aligned}\mathrm{DT}_j &= \frac{2}{n-1} \sum_{j' \neq j} \frac{\phi_c(\hat{y}_j)^\top \hat{\Sigma}^\dagger_{p_\theta(\cdot|c)} \phi_c(\hat{y}_{j'})}{\|(\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(\hat{y}_j)\| \|(\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(\hat{y}_{j'})\|} = \frac{2}{n-1} \sum_{j' \neq j} \frac{\tilde{\Phi}_{\cdot j}^\top \tilde{\Phi}_{\cdot j'}}{\|\tilde{\Phi}_{\cdot j}\| \|\tilde{\Phi}_{\cdot j'}\|} \\ &= \frac{2}{n-1} \sum_{j' \neq j} \mathbf{1}[\phi_c(\hat{y}_j) = \phi_c(\hat{y}_{j'})] \frac{\sqrt{n_{k_j} n_{k_{j'}}}}{n_{k_j}} = \frac{2(n_{k_j} - 1)}{n-1}.\end{aligned} \tag{47}$$

- *Variant (iii)*: **Normalizing the whitened features of the ground truth in the alignment term**. We set DT_j as in (44) and AT_j as follows:

$$\mathrm{AT}_j = \frac{2\phi_c(\hat{y}_j)^\top \hat{\Sigma}^\dagger_{p_\theta(\cdot|c)} \phi_c(y)}{\|(\hat{\Sigma}^\dagger_{p_\theta(\cdot|c)})^{1/2}\phi_c(y)\|} = \frac{2\tilde{\Phi}_{\cdot j'}^\top \tilde{\psi}}{\|\tilde{\psi}\|} = \frac{2x_j^{(\psi)}}{n_{k_j} \cdot \frac{1}{\sqrt{n_{k_j}}} \cdot \sqrt{\sum_{k=1}^K \frac{(x_k^{(\psi)})^2}{n_k}}} = \frac{2x_j^{(\psi)}}{n_{k_j} \sqrt{\sum_{k=1}^K \frac{(x_k^{(\psi)})^2}{n_k}}}, \tag{48}$$

Experimentally, **variant (i) offers the best performance, and is the one that we use in all the whitening experiments we report in this paper.**

Observe that when we use whitening (with or without any of these variants), we are not explicitly minimizing a particular loss function on θ, as our REINFORCE-style reward does not take into account that the second-moment matrix $\hat{\Phi}_{p_\theta(\cdot|c)}$ and the normalization factors depend on θ. In the figures of Section I, apart from the non-whitened feature matching loss, we report the proxy quantity:

$$\frac{1}{n} \sum_{j=1}^n \left(\mathrm{AT}_j - \frac{1}{2}\mathrm{DT}_j\right) \tag{49}$$

We refer to this quantity as the "feature-matching loss with whitening". Ignoring the dependence of the second-moment matrix and the normalization constants on θ, we view whitened feature matching as trying to decrease this loss.

C.4.1 Proof of Lemma C.2

And we define the matrix $B \in \mathbb{R}^{K \times n}$ as the matrix whose k-th row has value $1/\sqrt{n_k}$ on all positions from $j_k = \sum_{k'=1}^{k-1} n_{k'} + 1$ to $j_{k+1} - 1 = \sum_{k'=1}^{k} n_{k'}$ (both included), and value zero on all remaining

positions. Observe that the rows of B constitute an orthonormal set, i.e. $BB^\top = \mathrm{Id} \in \mathbb{R}^{K\times K}$, and that by construction,

$$\Phi = \bar{\Phi} B. \tag{50}$$

Thus,

$$\tilde{\Phi} = ((\bar{\Phi} B B^\top \bar{\Phi}^\top)^\dagger)^{1/2} \bar{\Phi} B = ((\bar{\Phi}\bar{\Phi}^\top)^\dagger)^{1/2}\bar{\Phi} B. \tag{51}$$

Next, we inspect $((\bar{\Phi}\bar{\Phi}^\top)^\dagger)^{1/2}\bar{\Phi}$. Observe that by assumptions *(i)* and *(ii)* above, the rank of $\bar{\Phi}$ is K. Let $\bar{\Phi} = U\Sigma V^\top$ be the thin singular value decomposition of $\bar{\Phi}$, i.e. $U \in \mathbb{R}^{d\times K}$ has orthonormal columns, $V \in \mathbb{R}^{K\times K}$ is an orthogonal matrix, and $\Sigma \in \mathbb{R}^{K\times K}$ is a diagonal matrix with strictly positive numbers on the diagonal (in general the singular values are non-negative, but since $\bar{\Phi}$ has rank K, all of them must be positive). Then, by the definitions of the square root and the pseudo-inverse

$$((\bar{\Phi}\bar{\Phi}^\top)^\dagger)^{1/2}\bar{\Phi} = ((U\Sigma^2 U^\top)^\dagger)^{1/2} U\Sigma V^\top = U\Sigma^{-1}U^\top U\Sigma V^\top = UV^\top \tag{52}$$

Plugging this into the right-hand side of (51) yields

$$\tilde{\Phi} = UV^\top B. \tag{53}$$

And

$$\tilde{\Phi}^\top\tilde{\Phi} = B^\top V U^\top U V^\top B = B^\top B, \tag{54}$$

which means that for $j, j' \in [n]$, $\tilde{\Phi}_{\cdot j}^\top \tilde{\Phi}_{\cdot j'}$, which is the (j, j')-th component of $\tilde{\Phi}^\top\tilde{\Phi}$, is equal to the (j, j')-th component of $B^\top B$, which is $B_{\cdot j}^\top B_{\cdot j'}$. Since $B_{\cdot j}$, (resp. $B_{\cdot j'}$) has a single non-zero entry $1/\sqrt{n_{k_j}}$ in position k_j (resp. $1/\sqrt{n_{k_{j'}}}$ in position $k_{j'}$), equalities (40) and (41) follow.

Let us define the diagonal matrix $\bar{B} \in \mathbb{R}^{K\times K}$ with diagonal values $\big(1/\sqrt{n_k}\big)_{k=1}^K$, such that we can express the matrix $\hat{\Phi}$ with columns $\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K$ as $\bar{\Phi}\bar{B}$. By construction of $x^{(\psi)}$, we have the following:

$$\psi = \hat{\Phi} x^{(\psi)} + \psi' = \bar{\Phi}\bar{B}x^{(\psi)} + \psi', \tag{55}$$

where ψ' is the orthogonal projection of ψ onto $\mathrm{span}\big(\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K\big)^\perp$. Hence, using equation (52), and the fact that $\mathrm{span}((U_{\cdot k})_{k=1}^K) = \mathrm{span}\big(\big(\phi_c(\hat{y}_{j_k})\big)_{k=1}^K\big)$,

$$((\Phi\Phi^\top)^\dagger)^{1/2}\psi = ((\bar{\Phi}\bar{\Phi}^\top)^\dagger)^{1/2}\big(\bar{\Phi}\bar{B}x^{(\psi)} + \psi'\big) = UV^\top\bar{B}x^{(\psi)} + U\Sigma^{-1}U^\top\psi' = UV^\top\bar{B}x^{(\psi)}. \tag{56}$$

Hence, using (53) and (56) yields

$$\Phi^\top((\Phi\Phi^\top)^\dagger)^{1/2}((\Phi\Phi^\top)^\dagger)^{1/2}\psi = BVU^\top UV^\top\bar{B}x^{(\psi)} = B^\top\bar{B}x^{(\psi)} = \tilde{B}x^{(\psi)} \tag{57}$$

where we define $\tilde{B} \in \mathbb{R}^{K\times n}$ as the matrix whose k-th row has value $1/n_k$ on all positions from $j_k = \sum_{k'=1}^{k-1} n_{k'} + 1$ to $j_{k+1} - 1 = \sum_{k'=1}^{k} n_{k'}$ (both included), and value zero on all remaining positions. And

$$\psi^\top((\Phi\Phi^\top)^\dagger)^{1/2}((\Phi\Phi^\top)^\dagger)^{1/2}\psi = (x^{(\psi)})^\top\bar{B}^\top VU^\top UV^\top\bar{B}x^{(\psi)} = \|\bar{B}x^{(\psi)}\|^2 = \sum_{k=1}^K \frac{(x_k^{(\psi)})^2}{n_k}. \tag{58}$$

D Feature matching with alignment bias

In some applications we prefer *more reference-aligned* samples, even at the cost of reduced diversity. We capture this tradeoff by scaling the target moment: for $\alpha \in (0, 1]$, define the loss with alignment bias as

$$\begin{aligned}\mathcal{L}_{\mathrm{FM}}^\alpha(\theta) &= \alpha\mathbb{E}_{c\sim p}\big[\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|^2\big] \\ &= \alpha\mathcal{L}_{\mathrm{FM}}(\theta) - \underbrace{2(1-\alpha)\mathbb{E}_{c\sim p}\big[\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})]^\top\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big]}_{\text{alignment bias}}.\end{aligned} \tag{59}$$

The additional term explicitly encourages alignment between model and data features, making the objective *mode-seeking* as α decreases (typically improving accuracy and faithfulness but reducing diversity). Operationally, this corresponds to multiplying the diversity terms in (4) and (5) by α, since the analogs of equations (2) and (3), (4), and (5) are

$$\mathcal{L}^{\alpha}_{\mathrm{CFM}}(\theta) = \alpha\mathbb{E}_{c\sim p}\big[\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \tfrac{1}{\alpha}\phi_c(y)\|^2\big], \tag{60}$$

$$\mathcal{L}^{\alpha}_{\mathrm{CFM}}(\theta; c, y) = \alpha\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \tfrac{1}{\alpha}\phi_c(y)\|^2, \tag{61}$$

$$r(\hat{y}, c) = 2\phi_c(\hat{y})^\top\phi_c(y) - 2\alpha\phi_c(\hat{y})^\top\mathbb{E}_{\tilde{y}\sim p_\theta(\cdot|c)}\big[\phi_c(\tilde{y})\big], \tag{62}$$

$$r_j = 2\phi_c(\hat{y}_j)^\top\phi_c(y) - \frac{2\alpha}{n-1}\sum_{j'\neq j}\phi_c(\hat{y}_j)^\top\phi_c(\hat{y}_{j'}). \tag{63}$$

Note that when $\alpha \neq 1$, $\mathcal{L}^{\alpha}_{\mathrm{FM}}$ is not a proper scoring rule, meaning that that it is not minimized by the ground truth distribution p, and this can be seen in practice. In Section I.1 we present experiments in which we sweep over α and γ on all the tasks we consider: we consider the values $\alpha \in \{0, 0.5, 1\}$. We conclude that taking α smaller helps in reducing the (unbiased) feature-matching loss faster, at the cost of a slower decrease of the CE loss, with stark differences in behavior depending on the value of γ. In particular, when $\gamma = 0.1$, the CE loss decreases similarly fast for $\alpha \in \{0, 0.5, 1\}$, but when $\gamma = 0$, taking $\alpha \in \{0, 0.5\}$ causes the CE loss to diverge, while for $\alpha = 1$ the CE loss still decreases monotonically, albeit at a slower rate than with higher α. Hence, the CE term can help in stabilizing feature matching with alignment bias.

E Feature matching with KL regularization

The feature matching loss with KL regularization with respect to a model π reads

$$\begin{aligned}\mathcal{L}_{\mathrm{FMKL}}(\theta) &= \mathbb{E}_{c\sim p}\big[\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|^2 + \tfrac{1}{\beta}D_{\mathrm{KL}}\big(p_\theta(\cdot|c)\|\pi(\cdot|c)\big)\big] \\ &= \mathbb{E}_{c\sim p}\big[\|\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}[\phi_c(\hat{y})] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|^2 + \tfrac{1}{\beta}\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\big[\log p_\theta(\hat{y}|c) - \log\pi(\hat{y}|c)\big]\big]\end{aligned} \tag{64}$$

Observe that since this loss function decouples across different contexts c, the optimal p_θ satisfies the following for all c:

$$p_\theta(\cdot|c) = \arg\min_{\rho\in\mathcal{V}^G}\big\{\tfrac{1}{\beta}D_{\mathrm{KL}}\big(\rho\|\pi(\cdot|c)\big) + \|\mathbb{E}_{y\sim\rho}[\phi_c(y)] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|_2^2\big\}. \tag{65}$$

Next, we will characterize the distribution p_θ that minimizes this loss function. The following section contains some preliminary results.

E.1 Energy-based models with RKHS function classes

The following duality theorem relates two optimization problems which underlie energy-based models for which the energy class is a ball of the RKHS induced by a feature map ϕ.

Theorem E.1 (Thm. 2, Domingo-Enrich et al. [10]). *Let π be a base measure over a measurable space $\mathcal{Y}$, and $\tilde{\beta} > 0$. Let $\varphi : \mathcal{Y} \to \mathbb{R}^d$ for some $d \geq 1$ be a feature map, and $v \in \mathbb{R}^d$. Consider the two problems*

$$\min_{\rho\in\mathcal{P}(\mathcal{Y})}\ \tfrac{1}{\tilde{\beta}}D_{\mathrm{KL}}(\rho\|\pi) + \|\mathbb{E}_{Y\sim\rho}[\varphi(Y)] - v\|_2, \tag{66}$$

and

$$\max_{\substack{h\in\mathbb{R}^d\\ \|h\|_2\leq 1}}\ -v^\top h - \tfrac{1}{\tilde{\beta}}\log\mathbb{E}_{Y\sim\pi}\big[\exp\big(-\tilde{\beta}\varphi(Y)^\top h\big)\big]. \tag{67}$$

The problems (66) and (67) are convex. The problem (67) is the Fenchel dual of problem (66), and strong duality holds. Moreover, the solution $\rho^\star$ of (66) is unique and satisfies

$$\frac{d\rho^\star}{d\pi}(y) = \frac{1}{Z_{\tilde{\beta}}}\exp\big(-\tilde{\beta}\varphi(y)^\top h^\star\big), \tag{68}$$

where $h^\star$ is a solution of (67) and Z_β is a normalization constant.

And the following equivalence between minimization problems holds:

Theorem E.2 (Prop. 3, Domingo-Enrich et al. [10])**.** *Consider the problem*

$$\min_{\rho\in\mathcal{P}(\mathcal{Y})} \beta^{-1} D_{\mathrm{KL}}(\rho\|\pi) + \|\mathbb{E}_{Y\sim\rho}[\varphi(Y)] - v\|_2^2, \tag{69}$$

Problems (66) and (69) are equivalent in the following sense: if $\rho_1^\star$ is a solution of (66) for $\tilde{\beta}$, then it is also a solution of (69) for

$$\beta = \big(2\|\mathbb{E}_{Y\sim\rho_1^\star}[\varphi(Y)] - v\|_2\big)^{-1}\tilde{\beta}, \tag{70}$$

provided that $\|\mathbb{E}_{Y\sim\rho_1^\star}[\varphi(Y)] - v\|_2$ is non-zero. Conversely, if $\rho_2^\star$ is a solution of (69) for β, then it is also a solution of (66) for

$$\tilde{\beta} = 2\|\mathbb{E}_{Y\sim\rho_2^\star}[\varphi(Y)] - v\|_2\beta. \tag{71}$$

E.2 Feature matching with KL regularization as an implicit energy-based model

Theorem E.3. *Consider the KL-regularized objective*

$$\min_{\rho} \ \mathbb{E}_{c\sim p}\Big[\big\|\mathbb{E}_{\rho(\cdot|c)}[\phi_c(y)] - \mathbb{E}_{p(\cdot|c)}[\phi_c(y)]\big\|^2 + \tfrac{1}{\beta} D_{\mathrm{KL}}(\rho(\cdot|c)\,\|\,q(\cdot|c))\Big], \tag{72}$$

where $\beta > 0$ controls the strength of the regularization. the solution to (72) has the form of an exponential tilt of the base distribution,

$$\rho^\star(y|c) \propto q(y|c)\,\exp\big(-\chi_c^\top\phi_c(y)\big),$$

for a context-dependent vector $\chi_c \in \mathbb{R}^d$ chosen to optimize:

$$\max_{\|\chi\|_2\le\tilde{\beta}} \big\{-\big(\tfrac{1}{\alpha}-1\big)\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top\chi + \mathbb{E}_{y\sim p(\cdot|c)}\big[\log\rho_\chi(y|c)\big]\big\}, \tag{73}$$

where $\rho_\chi(y|c) \propto q(y|c)\,\exp\big(-\chi^\top\phi_c(y)\big)$, for a $\tilde{\beta} > 0$ that depends on β. Two values of α admit specific interpretations:

- *For pure EBFT ($\alpha = 1$), the optimal χ is the **maximum likelihood estimate**: When $\alpha = 1$, the problem (73) simplifies to:*

$$\max_{\|\chi\|_2\le\tilde{\beta}} \mathbb{E}_{y\sim p(\cdot|c)}\big[\log\rho_\chi(y|c)\big], \tag{74}$$

  This corresponds to the maximum likelihood loss function for an energy-based model with energy function $E(y,c) = \chi^\top\phi_c(y)$.

- *For $\alpha^+ = 0$, the optimal χ has the **same direction as the ground-truth mean feature** $\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]$: When $\alpha = 0^+$, the problem (73) is equivalent to*

$$\max_{\substack{\chi\in\mathbb{R}^d\\ \|\chi\|_2\le\tilde{\beta}}} -\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top\chi, \tag{75}$$

  which has optimal solution

$$\chi^\star = -\tilde{\beta}\frac{\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]}{\big\|\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\big\|_2}. \tag{76}$$

Thus, for $\alpha \in (0,1)$, the solution $\chi^\star$ of (73) interpolates between the maximum likelihood estimate and the rescaled ground-truth mean feature.

Proof. Given a context c, and a completion length G, let us apply Theorem E.1 and Theorem E.2 by setting $\mathcal{Y} = \mathcal{V}^G$, $\varphi(y) = \phi_c(y)$, $\pi = \pi(\cdot|c)$, and $v = \frac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|^2$. Then, the problems (66), (69) and (67) take the form

$$\min_{\rho\in\mathcal{P}(\mathcal{Y})} \tfrac{1}{\tilde{\beta}} D_{\mathrm{KL}}\big(\rho\|\pi(\cdot|c)\big) + \|\mathbb{E}_{y\sim\rho}[\phi_c(y)] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|_2, \tag{77}$$

$$\min_{\rho\in\mathcal{P}(\mathcal{Y})} \tfrac{1}{\beta} D_{\mathrm{KL}}\big(\rho\|\pi(\cdot|c)\big) + \|\mathbb{E}_{y\sim\rho}[\phi_c(y)] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|_2^2, \tag{78}$$

$$\max_{\substack{h\in\mathbb{R}^d\\ \|h\|_2\le 1}} -\tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top h - \tfrac{1}{\tilde{\beta}}\log\mathbb{E}_{y\sim\pi(\cdot|c)}\big[\exp\big(-\tilde{\beta}\phi_c(y)^\top h\big)\big], \tag{79}$$

Problem (78) is the KL-regularized feature-matching objective with alignment bias, if we absorb the constant α accompanying $\|\mathbb{E}_{y\sim\rho}[\phi_c(y)] - \frac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|_2^2$ into the constants β. The (unique) solution (77) of $\rho^\star$ and the solution $h^\star$ of (79) are related by the equation

$$\rho^\star(y) = \frac{\pi(y|c)\exp\big(-\tilde{\beta}\phi_c(y)^\top h^\star\big)}{\sum_{y'}\pi(y'|c)\exp\big(-\tilde{\beta}\phi_c(y')^\top h^\star\big)}, \tag{80}$$

and $\rho^\star$ is also the (unique) solution of (78) provided that

$$\tilde{\beta} = 2\|\mathbb{E}_{Y\sim\rho^\star}[\varphi(Y)] - \tfrac{1}{\alpha}\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]\|_2\beta. \tag{81}$$

Observe that the problem (78) is equal to the problem (65), which means that (77)-(79) characterize the optimal $p_\theta(\cdot|c)$ when $\tilde{\beta}$ is chosen according to (81).

Next, we focus on the problem (79). We define $\mathcal{E}_{\tilde{\beta}}$ as the following class of energy functions:

$$\begin{aligned}\mathcal{E}_{\tilde{\beta}} &= \{E : \mathcal{V}^G \to \mathbb{R} \,|\, \exists \chi \in \mathbb{R}^d, \text{s.t.} \|\chi\|_2 \le \tilde{\beta}, \text{and}\, \forall x \in \mathcal{V}^G,\ E(x) = \chi^\top\phi_c(x)\} \\ &= \{E : \mathcal{V}^G \to \mathbb{R} \,|\, \exists h \in \mathbb{R}^d, \text{s.t.} \|h\|_2 \le 1, \text{and}\, \forall x \in \mathcal{V}^G,\ E(x) = \tilde{\beta}h^\top\phi_c(x)\},\end{aligned} \tag{82}$$

and given $\chi \in \mathbb{R}^d$, we define $\rho_\chi, \rho_h^{(\tilde{\beta})} \in \mathcal{P}(\mathcal{V}^G)$ as

$$\rho_\chi(y) = \frac{\pi(y|c)\exp\big(-\chi^\top\phi_c(y)\big)}{\mathbb{E}_{y'\sim\pi(\cdot|c)}\big[\exp\big(-\chi^\top\phi_c(y')\big)\big]}, \qquad \rho_h^{(\tilde{\beta})}(y) = \frac{\pi(y|c)\exp\big(-\tilde{\beta}\chi^\top\phi_c(y)\big)}{\mathbb{E}_{y'\sim\pi(\cdot|c)}\big[\exp\big(-\tilde{\beta}\chi^\top\phi_c(y')\big)\big]}. \tag{83}$$

We rewrite the problem (79) as

$$\begin{aligned}&\max_{\substack{h\in\mathbb{R}^d\\ \|h\|_2\le 1}} -\big(\tfrac{1}{\alpha}-1\big)\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top h - \mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top h - \tfrac{1}{\tilde{\beta}}\log\mathbb{E}_{y\sim\pi(\cdot|c)}\big[\exp\big(-\tilde{\beta}\phi_c(y)^\top h\big)\big] \\ &= \max_{\substack{h\in\mathbb{R}^d\\ \|h\|_2\le 1}} -\big(\tfrac{1}{\alpha}-1\big)\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top h + \frac{1}{\tilde{\beta}}\mathbb{E}_{y\sim p(\cdot|c)}\Big[\log\frac{\rho_h^{(\tilde{\beta})}(y)}{\pi(y|c)}\Big] \\ &= \max_{\substack{h\in\mathbb{R}^d\\ \|h\|_2\le 1}} -\big(\tfrac{1}{\alpha}-1\big)\mathbb{E}_{y\sim p(\cdot|c)}[\phi_c(y)]^\top h + \frac{1}{\tilde{\beta}}\mathbb{E}_{y\sim p(\cdot|c)}\Big[\log\rho_h^{(\tilde{\beta})}(y)\Big] + \text{const.}\end{aligned} \tag{84}$$

Writing the right-hand side problem in terms of χ instead of h, and multiplying the objective by $\tilde{\beta}$, yields the problem in (73). □

F Computing the REINFORCE gradient and the RLOO baseline for EBFT

We derive the REINFORCE gradient first. Rewriting $\mathcal{L}_{\mathrm{FM}}(\theta; c, y)$ explicitly in terms of p_θ, and using that $\hat{y}$, $\tilde{y}$ play a symmetric role, we have the following:

$$\begin{aligned}&\nabla_\theta\mathcal{L}_{\mathrm{FM}}(\theta; c, y) \\ &= \nabla_\theta\Big(\sum_{\hat{y},\tilde{y}} p_\theta(\hat{y}|c)p_\theta(\tilde{y}|c)\phi_c(\hat{y})^\top\phi_c(\tilde{y}) - 2\sum_{\hat{y}} p_\theta(\hat{y}|c)\phi_c(\hat{y})^\top\phi_c(y)\Big) \\ &= \sum_{\hat{y},\tilde{y}}\Big(\nabla_\theta\log p_\theta(\hat{y}|c) + \nabla_\theta\log p_\theta(\tilde{y}|c)\Big)p_\theta(\hat{y}|c)p_\theta(\tilde{y}|c)\phi_c(\hat{y})^\top\phi_c(\tilde{y}) - 2\sum_{\hat{y}}\nabla_\theta\log p_\theta(\hat{y}|c)p_\theta(\hat{y}|c)\phi_c(\hat{y})^\top\phi_c(y) \\ &= 2\mathbb{E}_{\hat{y},\tilde{y}\sim p_\theta(\cdot|c)}\Big[\nabla\log p_\theta(\hat{y}|c)\phi_c(\hat{y})^\top\phi_c(\tilde{y})\Big] - 2\mathbb{E}_{\hat{y}\sim p_\theta(\cdot|c)}\Big[\nabla\log p_\theta(\hat{y}|c)\phi_c(\hat{y})^\top\phi_c(y)\Big].\end{aligned} \tag{85}$$

Next, we derive the RLOO baseline. Define

$$T_1^{(j)} = 2\phi_c(\hat{y}_j)^\top\phi_c(y), \qquad T_2^{(j)} = \frac{2}{n-1}\sum_{j'=1, j'\neq j}^{n}\phi_c(\hat{y}_j)^\top\phi_c(\hat{y}_{j'}), \tag{86}$$

which means that we can rewrite the REINFORCE gradient (5) as

$$-\frac{1}{n}\sum_{j=1}^{n}\nabla\log p_\theta(\hat{y}_j|c)\big(T_1^{(j)}-T_2^{(j)}\big). \tag{87}$$

We want to use a baseline $b^{(j)}$ to reduce the variance of the gradient estimate (87). That is, the estimate with baseline $b^{(j)}$ reads

$$-\frac{1}{n}\sum_{j=1}^{n}\nabla\log p_\theta(\hat{y}_j|c)\big(T_1^{(j)}-T_2^{(j)}-b^{(j)}\big). \tag{88}$$

For the baselined gradient estimate to be unbiased, we need that $b^{(j)}$ is independent of $\hat{y}_j$.

A naive RLOO baseline would be $b^{(j)}=\frac{1}{N-1}\sum_{j'=1,j'\neq j}^{N}\big(T_1^{(j')}-T_2^{(j')}\big)$, i.e. simply averaging the rewards for all rollouts except the j-th one. However, the terms $T_2^{(j')}$ are not independent of $\hat{y}_j$, which means that this baseline is not independent of $\hat{y}_j$. To obtain an independent baseline, we need to replace $T_2^{(j')}$ by $T_2^{(j',j)}$, defined as

$$\begin{aligned}T_2^{(j',j)}&=\frac{2}{n-2}\sum_{j''=1,\,j''\neq j',j}^{n}\phi_c(\hat{y}_{j''})^\top\phi_c(\hat{y}_{j'})=\frac{2}{n-2}\Big(\sum_{j''=1,j''\neq j'}^{n}\varphi(c{:}\hat{y}_{j''})^\top\varphi(c{:}\hat{y}_{j'})-\varphi(c{:}\hat{y}_j)^\top\varphi(c{:}\hat{y}_{j'})\Big)\\&=\frac{n-1}{n-2}T_2^{(j')}-\frac{2}{n-2}\varphi(c{:}\hat{y}_j)^\top\varphi(c{:}\hat{y}_{j'}).\end{aligned} \tag{89}$$

Thus, the baseline that we end up with is:

$$\begin{aligned}b^{(j)}&=\frac{1}{n-1}\sum_{j'=1,j'\neq j}^{n}\big(T_1^{(j')}-T_2^{(j',j)}\big)\\&=\frac{1}{n-1}\sum_{j'=1,j'\neq j}^{n}\Big(T_1^{(j')}-\Big(\frac{n-1}{n-2}T_2^{(j')}-\frac{2}{n-2}\phi(\hat{y}_j)^\top\phi(\hat{y}_{j'})\Big)\Big)\\&=\frac{1}{n-1}\sum_{j'=1,j'\neq j}^{n}T_1^{(j')}-\frac{1}{n-2}\sum_{j'=1,j'\neq j}^{n}T_2^{(j')}+\frac{1}{n-2}T_2^{(j)}.\end{aligned} \tag{90}$$

G Details on the strided parallel rollout procedure

Sampling from a single, unstructured sequence provides only one supervision point and is a major bottleneck for EBFT, particularly because each sample must be embedded via forward passes through a separate feature network. Instead, we treat a single training sequence as a source of multiple nested prompts by identifying many anchor points along the text. Sampling from these points sequentially is prohibitively expensive, so we implement a novel parallel generation pipeline that simultaneously samples from different anchor points in one forward pass, similar to the custom attention mask approach introduced by Quiet-STaR [37]. Given a starting sequence $x_{0:T-1}$ of length T, a stride s, and a generation length G, we construct a set of nested prompts by segmenting $x_{0:T-1}$ every s tokens. This yields $B=\lfloor\frac{T-G}{s}\rfloor$ nested prompts. For each prompt $c_b=x_{0:bs}\quad(b=1,\dots,B)$, we take the next G tokens in the original sequence x as the ground-truth continuation y_b, yielding $\{(c_b,y_b)\}_{b=1}^{B}$ ground truth context and completion pairs. Additionally, from each prompt, we sample a continuation $\hat{y}_b$ of length G. Using our custom mask, we can sample one token from each prefix in just one forward pass. We can then obtain $\{\hat{y}_b\}_{b=1}^{B}$ length G model completions in G forward passes. The resulting BG generated tokens are appended in generation order; for example, with $B=3$ and $G=2$:

$$[\hat{y}_{1,0},\hat{y}_{2,0},\hat{y}_{3,0},\ \hat{y}_{1,1},\hat{y}_{2,1},\hat{y}_{3,1}].$$

This interleaving supports an efficient reshape into per-block windows for downstream scoring and feature extraction. In particular, we exploit the same strided structure to compute features for all generated blocks (and their ground-truth counterparts) with a single batched call to the feature network, followed by a reshape/indexing step to recover per-block embeddings.

Figure 7 shows a sketch of the strided parallel rollout procedure for a sequence of length $L = 12$, stride $S = 4$ and completion length $G = 4$, which means that $B = \lfloor (12 - 4)/4 \rfloor = 2$ context-completion pairs are used: $c_1 = (t_i)_{i=0}^{3}$, $y_1 = (t_i)_{i=4}^{7}$; and $c_2 = (t_i)_{i=0}^{7}$, $y_2 = (t_i)_{i=8}^{11}$. The ground truth sequence is in blue, and the generated completions are in red and green: $\hat{y}_1 = (t_{i\,a})_{i=4}^{7}$, $\hat{y}_2 = (t_{i\,b})_{i=8}^{11}$.

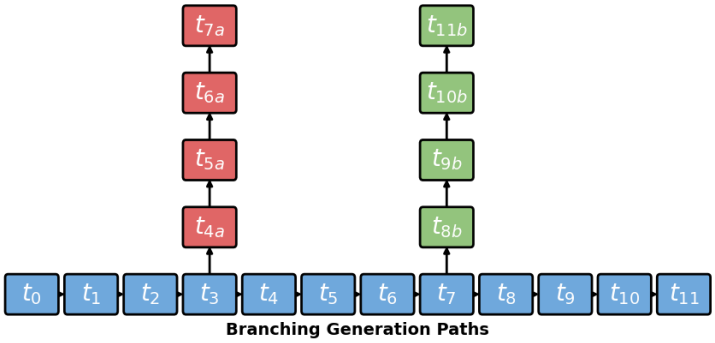

Figure 7: Strided parallel rollouts for a sequence of length $L = 12$, stride $S = 4$ and completion length $G = 4$.

The algorithm to obtain the generated completions $\hat{y}_1$ and $\hat{y}_2$ involves four model calls to p_θ. Figure 7 shows a sketch of the generated tokens using the strided parallel rollout procedure, and Section G shows the custom attention matrix for the fourth call (the horizontal and vertical lines show the three top left custom matrices used for the first three calls).

$$
-\left[\begin{array}{cccccccc|cc|cc|cc}
0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty \\
\hline
0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & 0 & \infty & \infty & \infty & \infty \\
\hline
0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & 0 & \infty & 0 & \infty & \infty & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & 0 & \infty & 0 & \infty & \infty \\
\hline
0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & 0 & \infty & 0 & \infty & 0 & \infty \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & 0 & \infty & 0 & \infty & 0
\end{array}\right]
$$

Figure 8: Custom attention matrix A in for a sequence of length $L = 12$, stride $S = 4$ and completion length $G = 4$. When the entry A_{ij} is 0, the token in position i attends to the token in position j, and when it is $-\infty$ it does not.

H Additional Experimental Details

We build our EBFT method on top of the OpenRLHF [17] framework, as well as use the OpenRLHF implementation of SFT and GRPO for our baselines. We use an internal cluster of 80GB H100 GPUs to conduct SFT, RLVR, and EBFT training runs. For Q&A code, a single epoch of SFT training takes 0.5 hours to run on a single 80GB H100, whereas a single epoch of RLVR using `vllm` training takes roughly 28 hours to run on two 80GB H100s, and a single epoch of EBFT training with our under-optimized implementation (without `vllm`) takes roughly 36 hours.

Parameter	Value		
	Q&A Code	**Unstructured Code**	**Translation**
Rollout Batch Size	16		
Sequence Length	1024		
Completion Length	8	8	4
Stride	8	8	2
Actor Learning Rate	1×10^{-6}		
Temperature	0.6		
KL Coefficient	0		
Samples per Prompt	4		
Training Batch Size	`rollout_batch_size` $\times$ `samples_per_prompt` = 64		
Warmup	0.03		
Adam Betas	(0.9, 0.95)		
Num Epochs	1		

Table 2: Hyperparameter Configuration for EBFT runs.

Hyperparameter details for the SFT training runs are provided in Table 3 for the warmstarted models (initialization for EBFT/RLVR) and in Table 4 for the five-epoch SFT baseline runs. We sweep over learning rate scheduler as well as training batch size for each task.

Parameter	Value
Training Batch Size	64
Epochs	1
Max Length	2048
Learning Rate	1×10^{-5}
Scheduler	Warmup + Cosine Decay to 0.1$\times$ `lr`
Warmup	0.03

Table 3: Hyperparameter Configuration for SFT warmstart runs.

Parameter	Value		
	Q&A Code	**Unstructured Code**	**Translation**
Training Batch Size	{64, 128}		
Max Length	2048		
(Learning Rate, Scheduler)	{(5×10^{-6}, Warmup + Constant), (1×10^{-5}, Warmup + Cosine Decay)}		
Warmup	0.03		
Adam Betas	(0.9, 0.95)		
Num Epochs	5		

Table 4: Hyperparameter Configuration for SFT baseline runs.

Hyperparameters for RLVR training runs are provided in Table 5. For RLVR, we fix training to be online and determine rollout batch size by roughly equating number of examples seen per step across both RLVR and EBFT for each task, sweeping over two values.

Parameter	Value	
	Q&A Code	**Translation**
Rollout Batch Size	{32, 64}	{128, 256}
Prompt Max Length	1024	
Generate Max Length	1024	
Actor Learning Rate	1×10^{-6}	
Temperature	1.0	
KL Coefficient	0	
Samples per Prompt	8	
Training Batch Size	`rollout_batch_size` $\times$ `samples_per_prompt`	
Warmup	0.03	
Adam Betas	(0.9, 0.95)	
Num Epochs	1	

Table 5: Hyperparameter Configuration for RLVR baseline runs.

Q&A Coding

Method	CE Q&A	HumanEval				MBPP				Multipl-E			
		greedy	pass@1	pass@4	pass@16	greedy	pass@1	pass@4	pass@16	greedy	pass@1	pass@4	pass@16
Base	0.524	0.348	0.324	0.490	0.622	0.599	0.514	0.703	0.782	0.506	0.433	0.626	0.742
Warm start	0.571	0.415	0.385	0.540	0.665	0.595	0.527	0.691	0.774	0.440	0.408	0.602	0.731
SFT	0.408	0.457	0.427	0.578	0.713	0.576	0.558	0.701	0.790	0.465	0.406	0.596	0.723
EBFT	**0.326**	0.494	0.448	0.616	0.750	**0.650**	**0.603**	**0.728**	0.813	0.524	0.488	0.645	0.753
EBFT (ws.)	0.337	**0.512**	0.500	**0.642**	**0.756**	0.638	0.584	0.727	**0.817**	0.476	0.452	0.621	0.734
RLVR	0.806	0.451	0.443	0.602	0.695	0.623	0.583	0.722	0.794	**0.531**	**0.502**	**0.658**	**0.767**
RLVR (ws.)	0.713	0.482	**0.516**	0.640	0.738	0.607	0.596	0.712	0.782	0.484	0.475	0.632	0.729

Unstructured Coding

Method	HumanEval				MBPP			
	greedy	pass@1	pass@4	pass@16	greedy	pass@1	pass@4	pass@16
Base	0.348	0.324	0.490	0.622	0.599	0.514	0.703	0.782
Warm start	0.463	0.419	0.586	0.707	0.553	0.497	0.690	0.778
SFT	0.451	0.417	0.596	0.707	0.556	0.516	0.691	0.786
EBFT	0.500	0.465	0.610	**0.726**	**0.611**	**0.583**	**0.718**	**0.813**
EBFT (ws.)	**0.512**	**0.478**	**0.629**	**0.726**	0.572	0.550	0.704	**0.813**

Translation

Method	CE Q&A	WMT'22 - COMET				MTNT - COMET				OpenSubtitles - COMET			
		greedy	best-of-1	best-of-4	best-of-16	greedy	best-of-1	best-of-4	best-of-16	greedy	best-of-1	best-of-4	best-of-16
Base	2.567	0.649	0.611	0.711	0.757	0.627	0.590	0.679	0.724	0.658	0.630	0.712	0.753
Warm start	2.648	0.733	0.712	0.776	0.807	0.705	0.683	0.759	0.796	0.696	0.677	0.742	0.776
SFT	2.692	0.747	0.722	0.784	0.815	0.703	0.683	0.755	0.792	0.701	0.682	0.745	0.777
EBFT	**2.399**	0.740	0.725	0.777	0.804	0.737	0.728	0.778	0.808	0.700	0.691	0.742	0.775
EBFT (ws.)	2.451	**0.753**	**0.741**	**0.788**	**0.812**	**0.742**	**0.732**	**0.782**	**0.810**	**0.708**	0.699	**0.749**	**0.779**
RLVR	3.225	0.704	0.698	0.743	0.769	0.705	0.698	0.745	0.772	0.684	0.679	0.718	0.741
RLVR (ws.)	3.148	0.738	0.730	0.771	0.794	0.727	0.721	0.765	0.789	**0.708**	**0.703**	0.740	0.762

Method	WMT'22 - BLEU				MTNT - BLEU				OpenSubtitles - BLEU			
	greedy	best-of-1	best-of-4	best-of-16	greedy	best-of-1	best-of-4	best-of-16	greedy	best-of-1	best-of-4	best-of-16
Base	0.069	0.103	0.166	0.215	0.073	0.098	0.154	0.197	0.081	0.171	0.238	0.281
Warm start	0.185	0.165	0.235	0.286	0.159	0.139	0.200	0.244	0.129	0.204	0.264	0.307
SFT	0.198	0.172	0.242	0.294	0.152	0.139	0.198	0.242	0.130	0.205	0.264	0.305
EBFT	0.204	0.187	0.247	0.289	0.212	0.175	**0.221**	**0.258**	0.136	0.219	0.266	0.305
EBFT (ws.)	**0.217**	**0.200**	**0.253**	**0.297**	0.202	0.174	0.219	0.256	0.142	0.221	**0.270**	**0.309**
RLVR	0.192	0.185	0.223	0.250	0.202	0.173	0.206	0.228	0.135	0.223	0.249	0.267
RLVR (ws.)	0.215	0.206	0.246	0.275	**0.217**	**0.186**	0.220	0.243	**0.152**	**0.240**	0.269	0.289

Table 6: **Across all tasks, EBFT matches or outperforms both SFT and RLVR on downstream metrics while maintaining substantially lower cross-entropy, and warm starting improves performance for both EBFT and RLVR.** On Q&A coding, EBFT achieves the best scores on HumanEval and MBPP, while RLVR is competitive on MultiPL-E. On unstructured coding, EBFT dominates across all benchmarks. On translation, EBFT (ws.) achieves the highest COMET scores on nearly every benchmark and leads on WMT'22 BLEU. RLVR (ws.) is competitive on MTNT and OpenSubtitles BLEU. Warm starting benefits both methods across all tasks. Notably, RLVR consistently degrades cross-entropy relative to the base model (e.g. 3.225 vs. 2.567 on translation), whereas EBFT improves it, suggesting that EBFT better preserves the model's language modeling capabilities while improving task performance.

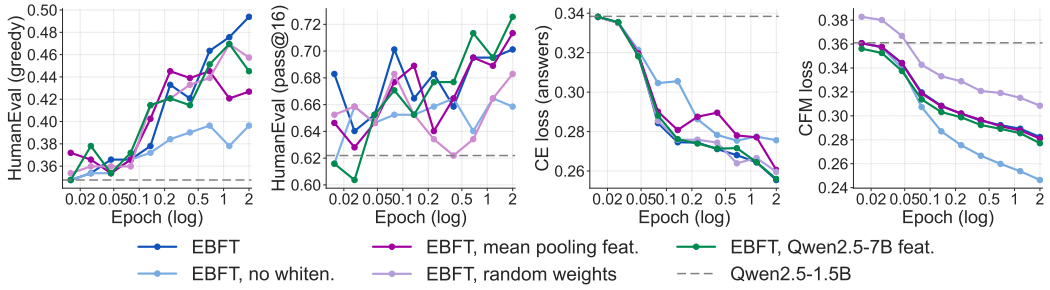

Figure 9: **Feature network ablations: whitening and last-token pooling matter most; scaling the feature network does not help.** HumanEval accuracy, validation cross-entropy, and CFM loss over training for EBFT ($\gamma = 0$) on Qwen2.5-1.5B with different feature network configurations. The default (last-token features with whitening from a frozen 1.5B copy) achieves the best downstream accuracy and CFM loss. Removing whitening and mean pooling cause the largest degradations. Random weights hurt only modestly, and replacing the 1.5B feature network with a frozen Qwen2.5-7B yields similar results, suggesting that pre-trained representations help but that naively scaling the feature network does not.

I Additional Experimental Results

I.1 Sweeping across γ, α and warm-starting

Figures 10, 11, 13 and 14 show 9 EBFT runs with $\alpha \in \{0, 0.5, 1\}$ and $\gamma \in \{0, 0.03, 0.1\}$ on the Q&A Coding, Unstructured Coding and Translation tasks, in which the models are initialized from the base Qwen2.5-1.5B and Llama3.2-1B, respectively. The observations below apply generally across tasks. We include additional observations about the behavior in particular settings in the captions of each figure.

Takeaways from the α, γ sweeps: $\alpha < 1$ is prone to instability when $\gamma = 0$, and increasing γ reduces the CE loss The choice $(\alpha, \gamma) = (1, 0)$ amounts to optimizing the pure feature-matching loss function $\mathcal{L}_{\mathrm{FM}}$; in this case the validation CE loss at a similar rate as for SFT, while the feature-matching loss decreases clearly faster, and the downstream performance is equal or better. The fact that pure FM beats SFT at reducing the CE loss may be attributed to FM with whitening optimizing a relaxation of the χ^2 divergence (see Section 2.2 and Section C). When $\gamma = 0$, and $\alpha \in \{0, 0.5\}$, the CE loss increases during training, which is not unexpected because the corresponding loss function is not a proper scoring rule: its minimizer is not the ground truth distribution p. In these settings, the FM loss decreases faster than when α is 1, even though we are optimizing a biased quantity, perhaps due to a bias-variance tradeoff of the gradient, and the downstream performance is slightly worse. Lastly, the CE loss gets reduced substantially with larger γ values both when α is 0.5 or 1.0, while the FM loss increases slightly with larger γ. The downstream performance is not affected substantially in either of the two cases.

Warm-starting: EBFT is more robust to weak initializations than RLVR Looking at Table 1, we can compare performance with and without warm-starting (running SFT for one epoch before initializing) for both EBFT and RLVR. Since both methods require sampling rollouts from the model, starting from a stronger model can in principle yield higher quality rollouts and improve RL gradients. However, the effect of warm-starting differs substantially between EBFT and RLVR. EBFT performs similarly with and without warm-starting, indicating that it is more robust to the quality of the initial model. In contrast, RLVR benefits heavily from warm-starting, and downstream performance and validation cross-entropy degrade significantly when initialized from weaker models. In summary, RLVR depends much more heavily on the capabilities of the initial model checkpoint. We hypothesize that this difference arises for two reasons. First, RLVR needs sufficiently accurate initial rollouts to produce a meaningful reward signal: poor initializations lead to sparse reward feedback. Second, RLVR introduces tension between reward maximization and maintaining low cross-entropy. In contrast, EBFT does not exhibit this conflict: it can simultaneously reduce the validation cross-entropy as much as SFT and improve downstream performance.

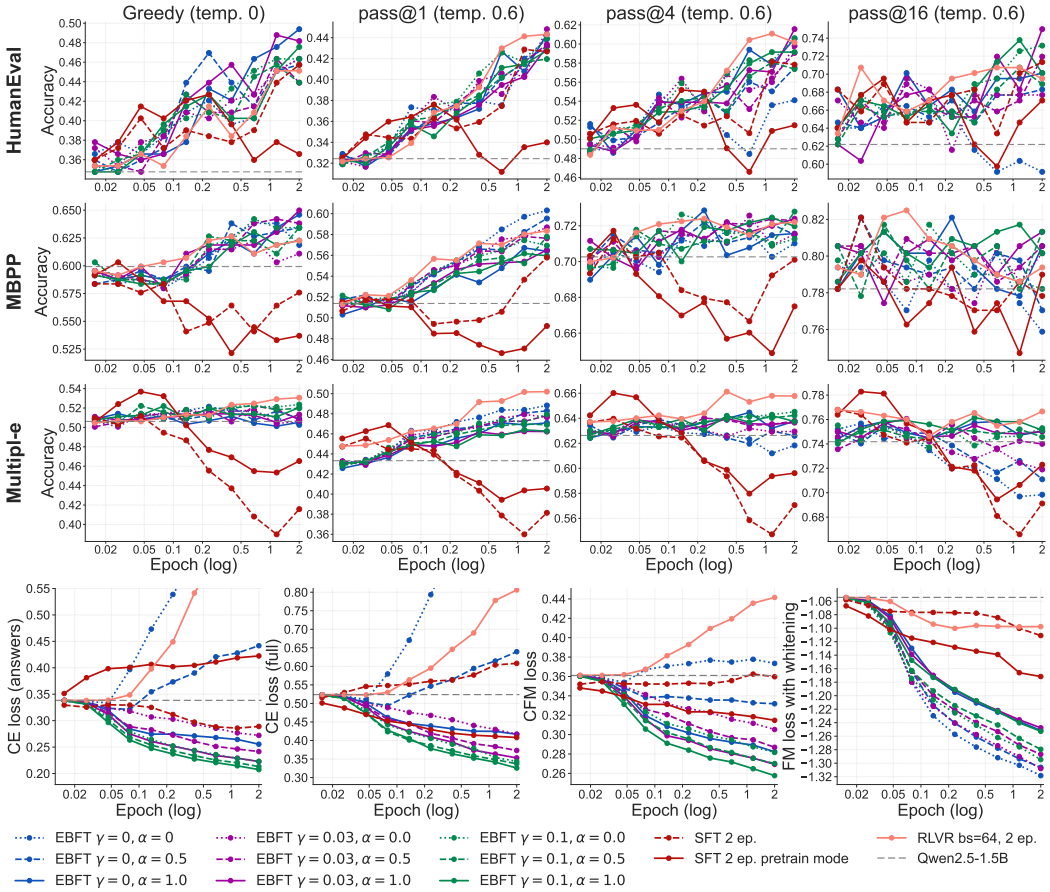

Figure 10: **On Q&A Coding, increasing the cross-entropy weight α consistently lowers both validation cross-entropy and feature-matching losses, while SFT on full sequences yields faster initial downstream gains that quickly degrade.** We sweep $\alpha \in \{0, 0.5, 1.0\}$ and $\gamma \in \{0, 0.03, 0.1\}$ for EBFT initialized from base Qwen2.5-1.5B. As a baseline, we compare against SFT trained on full sequences (solid red), whereas the main text reports SFT trained only on the answer. SFT on full sequences improves downstream performance faster early on but quickly deteriorates, and answer-level cross-entropy rises. Setting $\alpha = 0$ and $\gamma = 0$ (blue dotted) causes both cross-entropy and moment-matching losses to increase and leads to degraded pass@1 and pass@k scores, indicating that both terms are necessary for stable training. We also report feature-matching loss with and without whitening; the non-whitened variant tracks more closely with cross-entropy, which is why we use it for comparison. Overall, increasing α has limited effect on downstream metrics but helps decrease both the cross-entropy and moment-matching objectives. The validation set is a 1k-sample held-out subset of OpenCodeInstruct.

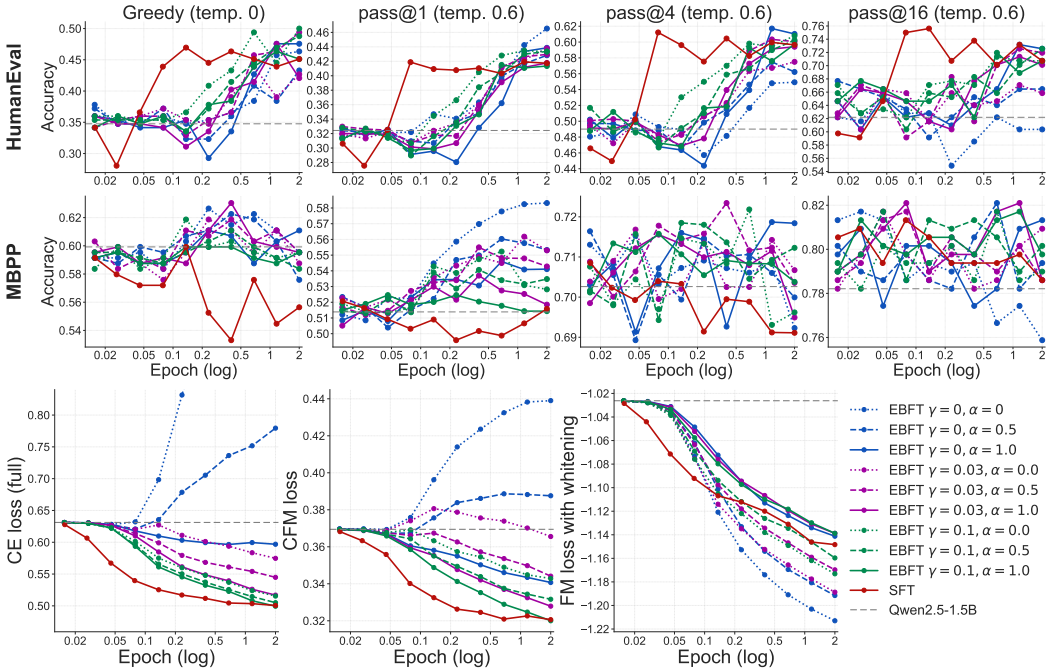

Figure 11: **On unstructured coding, EBFT achieves better downstream performance than SFT while reaching comparable levels on cross-entropy and feature-matching losses, even as downstream accuracy plateaus before these losses converge.** We sweep $\alpha \in \{0, 0.5, 1.0\}$ and $\gamma \in \{0, 0.03, 0.1\}$ for EBFT initialized from base Qwen2.5-1.5B. Cross-entropy and feature-matching losses continue to decrease throughout training even after downstream accuracy has plateaued, which we attribute to distribution shift between the training data and the downstream evaluation benchmarks. Setting $\alpha = 0$ (blue dotted) again leads to diverging cross-entropy, confirming the importance of the cross-entropy term. SFT (solid red) achieves the lowest cross-entropy and feature-matching losses but at the cost of weaker downstream performance compared to EBFT configurations with $\gamma > 0$. The validation set is a 1k-sample held-out subset of the unstructured coding data.

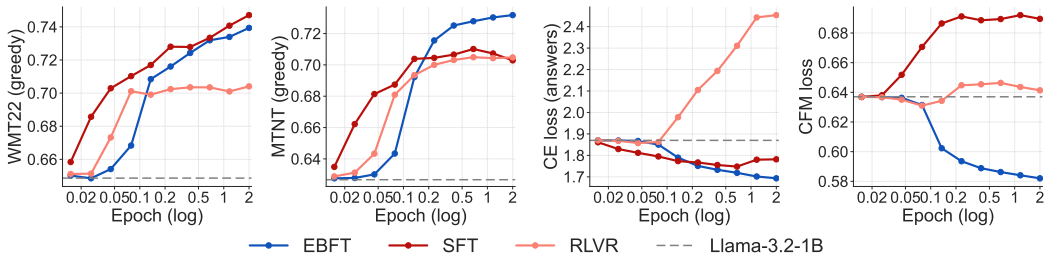

Figure 12: **On translation, EBFT outperforms both SFT and RLVR on downstream accuracy, cross-entropy, and feature-matching loss.** From left to right, we plot COMET scores on WMT22 and MTNT, validation cross-entropy, and CFM loss over training for Llama-3.2-1B fine-tuned on ALMA [34]. EBFT achieves the lowest CE and CFM losses and matches SFT on WMT22 while clearly outperforming it on MTNT. RLVR underperforms SFT on all four metrics, with cross-entropy rising well above the base model (dashed line).

I.2 Qualitative Analysis and Examples - Code

We present representative HumanEval generations produced by the final checkpoint of the 2-epoch runs for EBFT, SFT, and RLVR, along with the base Qwen-2.5-1.5B model. Across examples, EBFT generations more often accurately follow the prompt and are more reliably executable (complete, syntactically valid Python without extraneous scaffolding). By contrast, the base model frequently defaults to underspecified heuristics or incomplete solutions (e.g., using non-overlapping primitives such as `string.count`), while SFT and RLVR more often violate edge-case semantics or produce

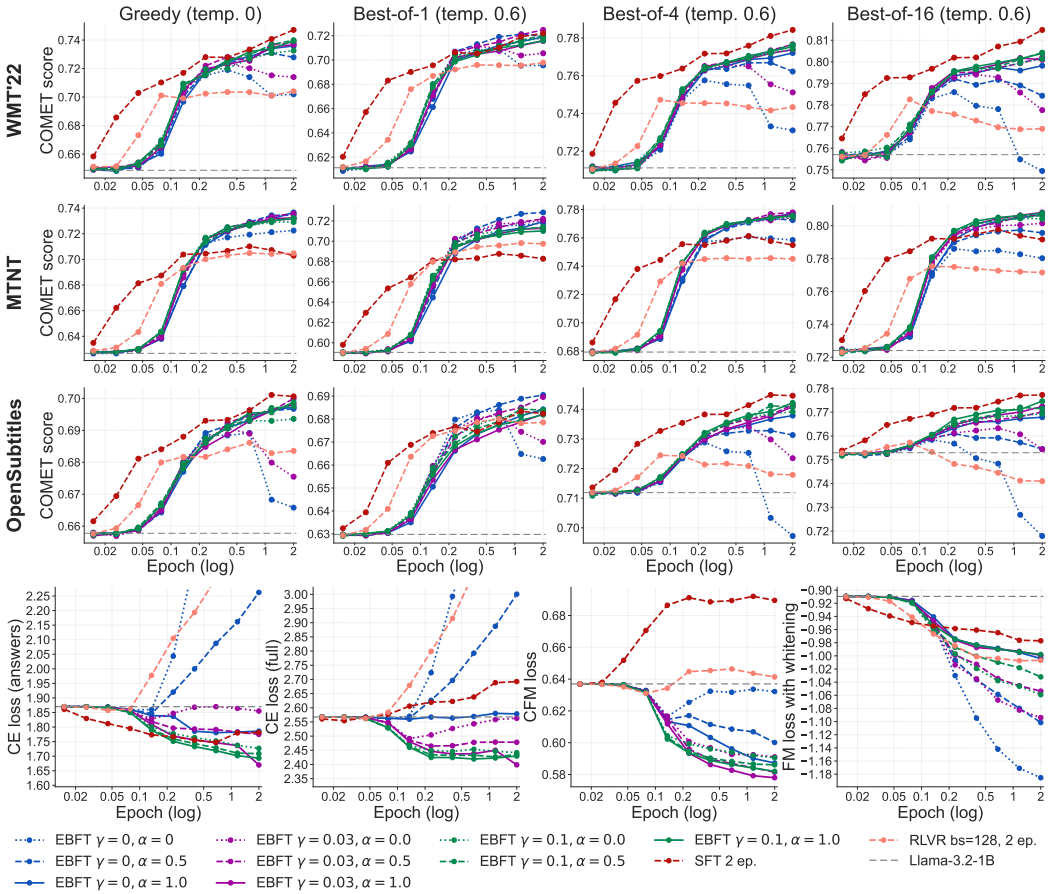

Figure 13: **Full Sweep: On translation, EBFT matches or exceeds RLVR on downstream COMET scores—outperforming it on MTNT—while consistently surpassing SFT across all benchmarks and evaluation settings.** We sweep $\alpha \in \{0, 0.5, 1.0\}$ and $\gamma \in \{0, 0.03, 0.1\}$ for EBFT initialized from base Llama-3.2-1B. On WMT'22, EBFT performs comparably to or slightly below RLVR, while on MTNT it clearly outperforms RLVR, and on OpenSubtitles the two are similar. EBFT surpasses SFT across all benchmarks and decoding strategies (greedy, best-of-1/4/16). Unlike in the coding setting, using $\alpha = 0$ and $\gamma = 0$ (blue dotted) leads to degraded downstream performance, indicating that both terms are important for translation. SFT is trained only on answers, as training on full sequences led to worse downstream results. On cross-entropy and feature-matching losses, EBFT achieves lower values than SFT, with larger γ configurations (green) reaching the best feature-matching levels. The legend is shared with the previous figures. The validation set is a 1k-sample held-out subset of ALMA.

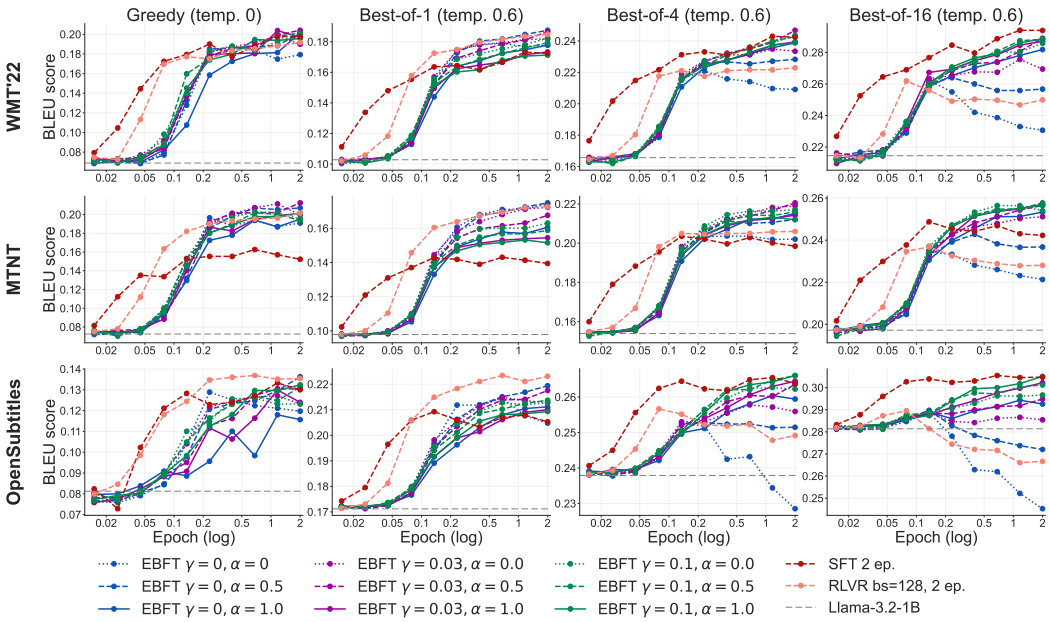

Figure 14: **On translation BLEU scores, EBFT consistently outperforms both SFT and RLVR, while RLVR degrades at best-of-16 later in training—consistent with the distribution sharpening induced by RLVR.** We report BLEU scores for the same α and γ sweep as the previous figure, initialized from base Llama-3.2-1B. Although RLVR is more competitive on BLEU than on COMET, EBFT still achieves higher scores across most benchmarks and decoding strategies. Notably, at best-of-16 (temp. 0.6), RLVR performance drops in the second half of training, suggesting that at large k the model suffers from reduced sample diversity, consistent with the distribution sharpening induced by RLVR [39]. EBFT configurations with $\gamma > 0$ do not exhibit this degradation and maintain stable or improving BLEU throughout training. The legend is shared with the previous figures.

outputs that fail under strict evaluation due to truncation, missing definitions (e.g., referencing `is_prime` without defining it), or non-code formatting/exposition that breaks executability. The figures below highlight these patterns across multiple HumanEval prompts.

I.3 Qualitative Analysis and Examples - Translation

We provide MTNT EN→FR examples from downstream evaluation using generations from the final checkpoint of the 2-epoch runs for EBFT, SFT, and RLVR, along with the base Llama-3.2-1B model. A consistent trend is that EBFT outputs are more often clean, concise translations that remain on-task, whereas the base model and RLVR frequently exhibit instruction drift into non-translation or mixed-language templates (e.g., repeating the English source, emitting "Spanish:"/"Português:" tag lists), suggesting instability with respect to the intended output format. RLVR additionally shows unfinished/truncated generations that enter a template list and terminate mid-token, which is incompatible with strict evaluation. Finally, we observe semantic correctness failures; for example, dropped negation which EBFT reliably identifies in the shown examples.

HumanEval/18: Count overlapping substring occurrences.

EBFT (correct):

```
count = 0
start = 0
while start < len(string):
    pos = string.find(substring, start)
    if pos != -1:
        count += 1
        start = pos + 1
    else:
        break
return count
```

BASE (incorrect; test failure):

```
return string.count(substring)
```

(completion continued with extra helper variants / doctests; omitted)

SFT (incorrect; test failure):

```
count = 0
start = 0
while start < len(string):
    start = string.find(substring, start)
    if start == -1:
        break
    count += 1
    start += len(substring)
return count
```

(completion continued with additional helper functions; omitted)

RLVR (incorrect; test failure):

```
count = 0
i = 0
while i < len(string):
    if string[i:i+len(substring)] == substring:
        count += 1
        i += len(substring)
    else:
        i += 1
return count
```

(completion continued with example-usage code; omitted)

Failure mode: Base uses non-overlapping `count`; SFT/RLVR advance by `len(substring)` (non-overlapping), missing overlaps.

Figure 15: **HumanEval/18: counting overlapping substring occurrences.** EBFT correctly advances by one character after each match and therefore counts overlapping occurrences. By contrast, the base model uses Python's non-overlapping `string.count`, while both SFT and RLVR advance by `len(substring)`, which misses overlap-sensitive cases and causes test failures.

HumanEval/94: Largest prime in list; return sum of its digits.

EBFT (correct):

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

largest_prime = None
for num in lst:
    if is_prime(num):
        if largest_prime is None or num > largest_prime:
            largest_prime = num
return sum(int(d) for d in str(largest_prime))
```

BASE (incorrect; test failure):

```
return sum(int(digit) for digit in str(max([num for num in lst if is_prime(num)])))
```

(completion continued with example usage / extra text; omitted)

SFT (syntax failure):

```
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        r
```

(completion truncated mid-token / non-executable; remainder omitted)

RLVR (incorrect; test failure):

```
largest_prime = None
for num in lst:
    if is_prime(num):
        if largest_prime is None or num > largest_prime:
            largest_prime = num
if largest_prime is None:
    return 0
return sum(int(d) for d in str(largest_prime))
```

(completion continued with surrounding scaffolding; omitted)

Failure mode: Base/RLVR reference `is_prime` without defining it; SFT emits invalid Python (syntax).

Figure 16: **HumanEval/94: largest prime digit sum.** EBFT produces a self-contained executable solution by defining `is_prime` and then computing the digit sum of the largest prime in the list. The base model and RLVR instead rely on an undefined helper function, while SFT truncates mid-function and yields syntactically invalid Python, illustrating a recurring executability gap between EBFT and the competing methods.

HumanEval/105: Filter digits 1–9; sort, reverse; map digits to names.

EBFT (correct):

```
if not arr:
    return []
filtered_arr = [num for num in arr if 1 <= num <= 9]
sorted_arr = sorted(filtered_arr)
reversed_arr = sorted_arr[::-1]
digit_names = {
    1: "One", 2: "Two", 3: "Three", 4: "Four", 5: "Five",
    6: "Six", 7: "Seven", 8: "Eight", 9: "Nine"
}
return [digit_names[num] for num in reversed_arr]
```

BASE (incorrect; test failure):

```
return arr
```

(completion included commented examples / extra text; omitted)

SFT (incorrect; test failure):

```
if not arr:
    return []
if not all(1 <= x <= 9 for x in arr):
    return arr
sorted_arr = sorted(arr)[::-1]
digit_names = ["One","Two","Three","Four","Five","Six","Seven","Eight","Nine"]
return [digit_names[x-1] for x in sorted_arr]
```

(completion continued with auxiliary text; omitted)

RLVR (incorrect; test failure):

```
filtered_arr = [num for num in arr if 1 <= num <= 9]
reversed_arr = sorted(filtered_arr)[::-1]
name_arr = []
for num in reversed_arr:
    name_arr.append(str(num))
return name_arr
```

(completion continued with example usage / prints; omitted)

Failure mode: Base returns input unchanged; SFT returns original array if *any* element is out of range (should filter); RLVR returns digit strings rather than names.

Figure 17: **HumanEval/105: filtering, ordering, and lexical mapping.** EBFT correctly implements the full prompt: it filters to digits 1 to 9, sorts the valid entries, reverses the order, and maps each digit to its English name. The other outputs each satisfy only part of the specification: the base model returns the input unchanged, SFT incorrectly returns the original array whenever any element is out of range instead of filtering, and RLVR outputs numeric strings rather than digit names.

HumanEval/69: Return greatest positive integer with frequency $\geq$ its value.

EBFT (correct):

```
max_value = -1
for num in lst:
    if num > 0 and lst.count(num) >= num:
        if num > max_value:
            max_value = num
return max_value
```

BASE (incorrect; test failure):

```
# (no valid return produced; incomplete solution)
```

(completion was essentially comments / placeholder; omitted)

SFT (incorrect; test failure):

```
from collections import Counter
count = Counter(lst)
for num in count:
    if count[num] >= num:
        return num
return -1
```

(completion continued with ad-hoc test prints; omitted)

RLVR (syntax failure):

```
# core logic resembles EBFT, but output contained extra non-code/formatting
# -> non-executable under strict evaluation
```

Failure mode: Base is incomplete; SFT returns the first qualifying key rather than the *greatest*; RLVR marked non-executable (syntax/formatting).

Figure 18: **HumanEval/69: selecting the greatest frequency-qualified integer.** EBFT returns the maximum positive integer whose frequency is at least its value, matching the full prompt semantics. The base model fails to provide a usable implementation, SFT returns the first qualifying key rather than the greatest one, and RLVR follows the right high-level logic but is not executable under strict evaluation because the output includes formatting artifacts outside valid code.

MTNT: Non-translation (model repeats source language)

(EN→FR). Source: TIL Tigers will take revenge on those who have done wrong to them.
Ref: Aujourd'hui j'ai appris Les tigres prendront leur revanche sur ceux qui leur ont causé du tort.

EBFT (clean/on-task):

```
TIL:Les tigres vont prendre vengeance sur ceux qui ont
fait du mal à eux.
```

BASE (failure):

```
TIL Tigers will take revenge on those who have done
wrong to them.
Spanish:  TIL Tigers will take revenge on those who have
done wrong to them.
Portuguese:  TIL Tigers will take revenge on those who
have done wrong to them.
TIL Tigers will take revenge on those who have done
wrong to them.
```

(continuation omitted)

SFT (failure; stays in English):

```
TIL that the Tigers will take revenge on those who have
done wrong to them.
```

RLVR (failure):

```
TIL Les tigres vont se venger de ceux qui ont fait du
mal à eux.
Português:  TIL Os tigres vão se vingar dos que têm
feito mal a eles.
Português-Celta:  TIL Os tigres vão se vingar dos que
têm feito mal a eles.
...
```

(continuation omitted)

Failure mode: Output is partially or entirely non-translation (keeps English), often followed by multilingual tag repetition.

Figure 19: **MTNT EN→FR: non-translation and multilingual runaway.** EBFT remains on task and produces a direct French translation, whereas the base model and RLVR drift into multilingual template continuations that repeat the source sentence and append language-tagged variants. SFT avoids the runaway list structure but still fails to translate, remaining almost entirely in English.

MTNT: Truncation / unfinished generation (ends mid-template)

(EN→FR). Source: What a way for spring training to end.
Ref: Quelle manière de finir les entraînements de printemps.
EBFT (clean/on-task):

```
Quel bonheur de fin de campagne de pré-saison.
```

BASE (failure):

```
Le match de la saison régulière a été terminé.
Spanish:  El partido de la temporada regular se terminó.
What a way for spring training to end.
The 2019 season is over.  The 2019 season is over.
```

(continuation omitted)
SFT (failure; drift):

```
C'est une fin de saison de baseball incroyable.
```

RLVR (failure; truncates mid-template):

```
Ce fut une fin de printemps merveilleuse.
Português:  Esta foi uma final de verão maravilhosa.
Português-Corrigido por-em-tudo-como-que-
```

(continuation omitted)

Failure mode: Generation enters a template list and then stops mid-token.

Figure 20: **MTNT EN→FR: truncation after template drift.** EBFT produces a complete on-task translation, while the base model drifts into unrelated multilingual continuation and SFT paraphrases too loosely away from the original meaning. RLVR begins a language-tagged template expansion and terminates mid-token, yielding an unfinished output that is incompatible with strict evaluation.

MTNT: Meaning flip via dropped negation ("no more" → "more")

(EN→FR). Source: Then there are no more available rooms.
Ref: Il n'y a plus de chambres disponibles.
EBFT (clean/on-task):

```
Puis il n'y a plus d'espaces disponibles.
```

BASE (failure; runaway):

```
Il n'y a plus d'espaces disponibles.
Spanish:  No hay mas habitaciones disponibles.
Italian:  Non ci sono più stanze disponibili.
...
```

(continuation omitted)
SFT (failure; weaker paraphrase):

```
Puis il n'y a plus d'offres de chambres.
```

RLVR (failure; meaning flip):

```
Puis il y a plus de chambres.
Português:  Portanto, existem mais quartos.
Português-Cogitado:  Portanto, existem mais quartos.
...
```

(continuation omitted)

Failure mode: Dropped negation.

Figure 21: **MTNT EN→FR: semantic failure from dropped negation.** Not all errors are formatting failures: this example shows that negation can be overlooked by the fine-tuned models. EBFT preserves the negative meaning of the source sentence, whereas RLVR drops the negation and changes "no more available rooms" into "more rooms." The base model again exhibits multilingual repetition after an initially plausible translation, and SFT produces a weaker paraphrase that underspecifies the original statement.