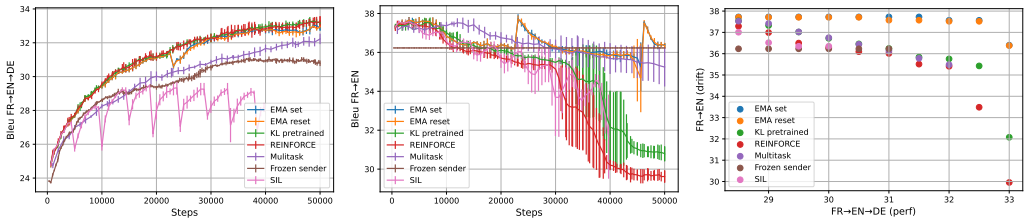


(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 7: All transformer model methods on the Translation Game. We measure (a) Task Performance with FR→EN→DE BLEU and (b) Language Drift with FR→EN BLEU, on the validation set during finetuning. We plot the mean and show error bars for standard deviation over 5 seeds. To compare how methods do on both metrics, we plot (c) the best achieved drift vs task performance across finetuning.

A Translation Game

A.1 Experimental Details

We implement the translation game in the fairseq library [Ott et al., 2019]. All experiments are run with 5 seeds where each run uses a single 16G V100 GPU. All plots show the mean and standard deviation over seeds. We score BLEU using detokenized sacreBLEU [Post, 2018].

A.2 Translation Game with Transformers

Note that Lee et al. [2019] (as well as previous work on the Translation Game) use seq2seq [Sutskever et al., 2014] LSTMs [Hochreiter and Schmidhuber, 1997] with attention [Bahdanau et al., 2015]. For consistency with other experiments and relevance to mainstream research we switch to a Transformer architecture. LSTMs results are similar and results are available in Appendix A.3. Also note that we use REINFORCE as our estimator but Gumbel-Softmax [Jang et al., 2017, Maddison et al., 2017] is also feasible [Lu et al., 2020] and preliminary results have been similar.

We follow Lee et al. [2019] for preprocessing both IWLST and Multi30k. We use the `iwslt-de-en` architecture from the fairseq library [Ott et al., 2019] with their default IWSLT DE-EN hyperparameters and training, specifically training with AdamW [Loshchilov and Hutter, 2022] using default hyperparameters. We pretrain our models on IWSLT following Lee et al. [2019] and early-stop on the tst2013 validation set. Our final validation scores are 43.85 BLEU for FR→EN and 29.4 BLEU for EN→DE.

We report all final validation scores in Table 4 and all curves in Figure 7.

Given that we have a learning reward model (EN→DE), iteratively resetting our FR→EN model to its pretrained weights can be an effective strategy [Rita et al., 2022]. We report this as RESET and find it performs quite well but not as well as Elastic Reset. We did not include these results in the main paper because we focused on methods that could also work with fixed reward models.

Table 4: Translation Game final validation scores

	↑ FR→EN→DE	↑ FR→EN
PRETRAINED	23.8	36.2
FROZEN SENDER	30.8±0.2	**36.3±0.1**
REINFORCE	**33.2±0.3**	29.6±0.3
+ SIL	28.2±0.4	27.3±4.4
+ MULTITASK (S2P)	32.2±0.3	35.2±1.0
+ KL PRETRAINED	**33.2±0.2**	30.8±0.4
+ RESET	32.5±0.1	33.3±0.1
+ RESET TO EMA	32.9±0.1	**36.3±0.1**
+ ELASTIC RESET	33.0±0.1	**36.3±0.1**

A.3 Translation Game with LSTMs

We follow Lee et al. [2019] for preprocessing and pretraining on IWSLT. We compare our pretraining results to both Lee et al. [2019] and Lu et al. [2020] in Table 5

Table 5: BLEU score of IWSLT-pretrained LSTM models on IWSLT 2013 validation set

	FR→EN	EN→DE
LEE ET AL. [2019]	34.1	22.0
LU ET AL. [2020]	32.2	20.2
OURS	38.5	23.2

Next we finetune on Multi30k using the same hyperparameters as Lu et al. [2020]. We plot results in Table 6 and compare to published numbers from previous work. As usual, drift is the negative change in FR→EN BLEU from the pretrained model. Task performance is the positive change in FR→EN→DE BLEU from the pretrained models. Combined is the sum of drift and task performance. We show the pretrained, baseline, and best-performing model from previous work. Note that our results are not directly comparable to previous work because we evaluate using detokenized sacreBLEU [Post, 2018] whereas previous work wrote their own BLEU evaluation code and did not detokenize.

Table 6: BLEU scores and ± standard deviation on the Multi30k Translation Game using IWSLT-pretrained LSTM models.

	METHOD	FR→EN	FR→EN→DE	DRIFT	PERF	COMBINED
LEE ET AL. [2019]	PRETRAINED	27.2	16.3			
	REINFORCE	12.4 ± 0.7	24.5 ± 1.5	-14.8	+8.2	-6.6
	+ LM	23.6 ± 1.1	27.7 ± 0.4	-3.6	+11.4	+7.8
	+ LM + G	24.8 ± 0.4	28.1 ± 0.7	-2.4	+11.8	+9.4
LU ET AL. [2020]	PRETRAINED	29.4	15.7			
	GUMBEL-SOFTMAX	14.5 ± 0.8	27.1 ± 0.1	-14.9	+11.4	-3.5
	SIL	29.4 ± 0.3	28.3 ± 0.2	0	+12.6	+12.6
OURS	PRETRAINED	32.6	18.0			
	FROZEN-SENDER	32.6 ± 0	25.1 ± 0.1	0	+7.1	+7.1
	REINFORCE	30.0 ± 0.2	30.3 ± 0.2	-2.6	+12.3	+9.7
	LM $\lambda = 0.01$	31.5 ± 0.1	30.2 ± 0.2	-1.1	+12.2	+11.1
	MULTITASK $\lambda = 0.1$	32.9 ± 0.2	28.5 ± 0.3	+0.3	+10.5	+10.8
	SIL	27.7 ± 0.7	24.3 ± 0.1	-4.9	6.3	+1.4
	ELASTIC RESET	32.6 ± 0.1	30.0 ± 0.1	0	+12.0	**+12.0**

Our results for the baseline are notably better than Lee et al. [2019], Lu et al. [2020]. The only difference between our code and theirs as far as we can tell is (1) we use an exponential moving average baseline for REINFORCE whereas [Lee et al., 2019] use an Actor-Critic method, (2) Lu et al. [2020] uses 0.1 gradient clipping and we do not use gradient clipping (3) in preprocessing Multi30k, we first tokenize [Koehn et al., 2007] then lowercase whereas previous works did the opposite order.

A more reasonable explanation for the improvement in results is how we choose to evaluate. Previous work simply ran all methods for the same number of updates but this doesn't account for, even implicit, hyperparameter optimization. Previous methods show that the baseline's FR→EN→DE BLEU scores plateau and there is significant language drift in FR→EN without real improvements to the task score. We hypothesize that these extra training episodes only serve to increase the drift without measuring what we actually care about: performance gain for drift. Since the number of updates is arbitrary, we believe that early stopping on a reasonable metric is a better evaluation protocol and choose hyperparameters such that methods plateau at the end.

We also note that our results with the SIL method of Lu et al. [2020] are negative. We do not manage to gain any improvement in performance. We collaborated with the authors of Lu et al. [2020] for many months but, in our setup using the fairseq library [Ott et al., 2019] could not reproduce their results. One of their fundamental results is that a student sender can outperform a teacher sender that it is distilling from. We could not reproduce this and believe it is a difference in the pretrained models.

B IMDB Mock Sentiment

B.1 Experimental Details

We run experiments and implement our method in the RL4LMs library [Ramamurthy et al., 2022]. All experiments are run with 5 seeds where run uses a single 40G A100 GPU. All plots show the mean and standard error over seeds.

For PPO, we use the default hyperparemters provided by Ramamurthy et al. [2022]. For NLPO, we found the defaults had a mistake and after communication with Ramamurthy et al. [2022] we changed the learning rate to 1e-6 and target update iterations to 50. This still did not manage to reproduce the original NLPO test scores from Ramamurthy et al. [2022] but we found that our validation curves matched their provided curves in the appendix. After communications [Ammanabrolu, 2023], we both agreed that we should use our reproducible test scores for NLPO in lieu of the original test scores.

Our best Elastic Reset hyperparameters are the default PPO parameters (`gpt2_ppo.yml`) from RL4LMs [Ramamurthy et al., 2022] with a few modifications: target kl is set from 0.5 to 1.0 and KL coefficient is set much lower to 0.001. We use an EMA decay of 0.995 and reset every 17 epochs. Configs to reproduce the experiments can be found in the RL4LMs folder under `scripts/training/task_configs/imdb_text_continuation`

C StackLlama

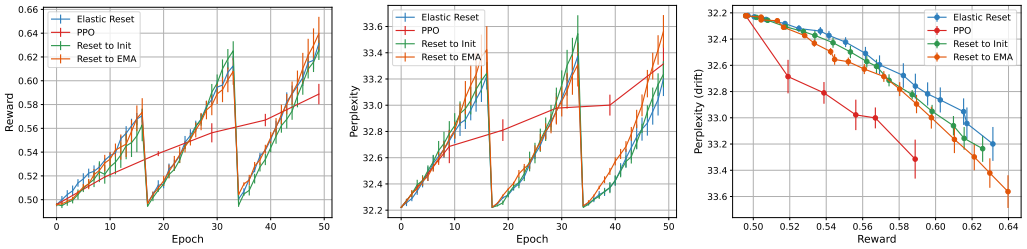
C.1 Experimental Details

We follow the original StackLlama and use the trl library from Huggingface to train the model on the StackExchange dataset. We use the original authors' LoRA adapter weights to create our supervised model LLaMA-7B-SE but train our own reward model as there were issues loading the pretrained reward model. As noted by [Beeching et al., 2023], the reward modelling task is difficult enough that humans struggle with it and our final model achieves 64% accuracy compared to the original 67%. Although we follow the original authors method and use their codebase, we note that our results may be different but valid. There have been many updates and fixes to the `trl` codebase since the authors' original blog post and specifically a possible issue in the code for creating reward models could have affected the original authors' run.

Furthermore, to speed up training, we used a 2x smaller KL coeffient and ran with half the number of GPUs. Specifically, RL training is run on 4x 80G A100 GPUs for 20 hours. We evaluate perplexity of the model on the validation set used in supervised finetuning, 4000 examples from the supervised training dataset. The configs for reproducing our experiments are in the `trl` library folder under `examples/stackllama/scripts/configs`

C.2 HumanEval

Our measure of alignment tax, HumanEval [Chen et al., 2021], is a programming benchmark where each question was hand-written by humans (OpenAI engineers) to be unseen in training data. We believe these questions were unseen in LLaMA training data as well. We prompt the model with the question and it writes the corresponding code. The measure of success is functionally correct code i.e. we actually execute the code LLaMA wrote and see if it gets the right output. Since we decode by sampling, we generate $N = 100$ continuations of the prompt and then follow Chen et al. [2021] to estimate how often our model would get the right answer on the first generated continuation (pass @ 1) and within the first 10 generated continuations (pass @ 10). We evaluate HumanEval using CodeCapybara's evaluation harness [To et al., 2023].



(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

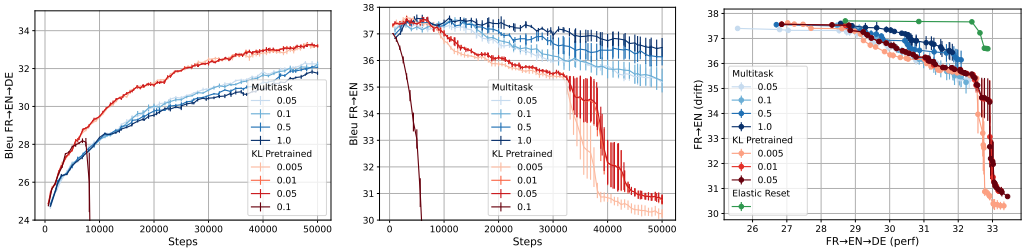
Figure 8: Ablation of Elastic Reset, Reset to EMA, and Reset to Init on IMDB. We plot the mean and show error bars for standard error over 5 seeds.

D Ablations Full Graphs

D.1 Resets

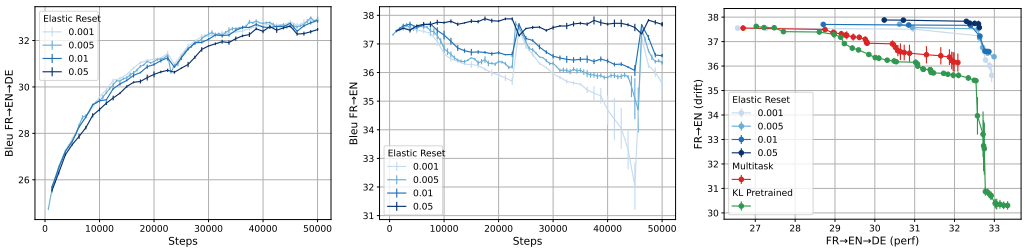
D.2 KL Coefficient

As shown in Figure 5, Elastic Reset performance in relatively unaffected by the presence or absence of KL. In contrast, PPO and REINFORCE performance is known to be sensitive [Lu et al., 2020]. This raises the question of whether different coefficients of the KL loss would lead to different pareto frontiers. We find in Figure 9 that both Multitask and KL with Pretrained methods do have slightly different pareto frontiers. Choosing a



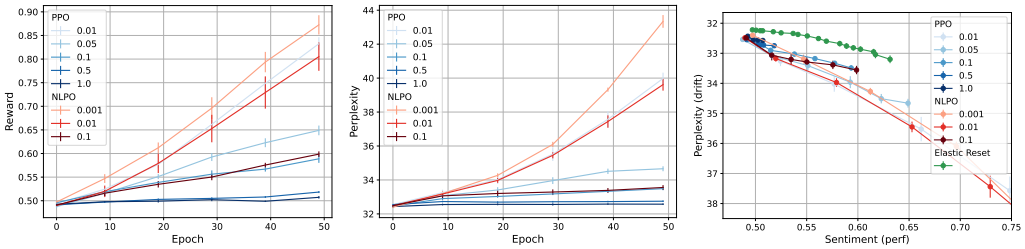
(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 9: Ablation of KL and Multitask coefficients on Translation Game. We exclude KL 0.1 from the pareto graph since it fails. We plot the mean and show error bars for standard error over 5 seeds.



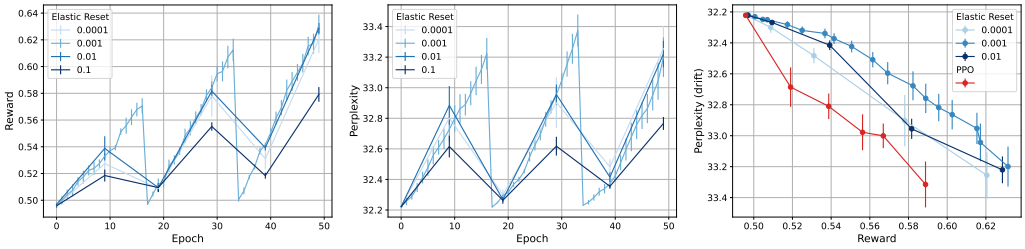
(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 10: Ablation of KL coefficients for Elastic Reset on Translation Game. We plot the mean and show error bars for standard error over 5 seeds.



(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

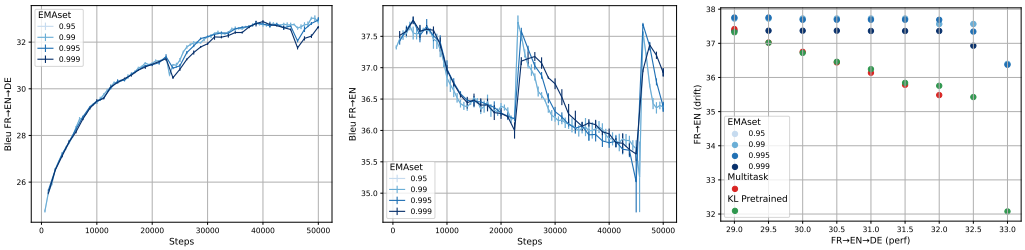
Figure 11: Ablation of PPO and NLPO coefficients on IMDB. We do more ablations of PPO since NLPO is always similar but worse than PPO. We plot the mean and show error bars for standard error over 5 seeds.



(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 12: Ablation of KL coefficients for Elastic Reset on IMDB. We plot the mean and show error bars for standard error over 5 seeds.

D.3 Decay Rate



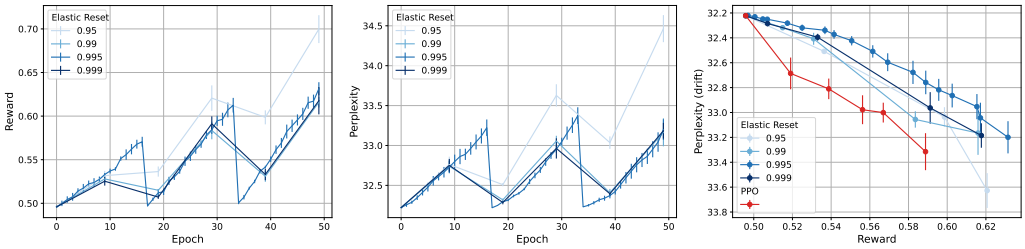
(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 13: Ablation of decay coefficients for Elastic Reset on Translation Game. Original decay is 0.99. We plot the mean and show error bars for standard error over 5 seeds.

D.4 Explaining Resets: Value Function

Drift is a problem of noisy optimization, and Elastic Reset tackles this in two ways.

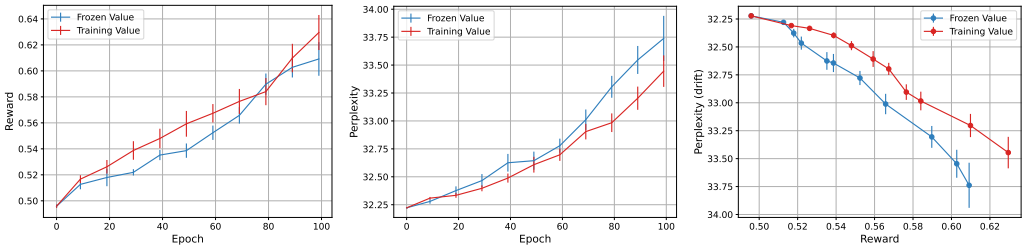
The first is by re-training with an improved value function. Though we begin with a sequence-level reward model, over training we learn a *token-level* value function. We believe that the value function greatly improves over time so early training may have exacerbated drift because it uses the early, sub-optimal value function. With each reset of the policy while maintaining the value function, we re-train with less noise and more direct gradients to alignment. Another way, both the reward model and optimal value function point towards high reward, but not in the same way. The reward model points towards high reward but not necessarily in the most direct way. Using the frozen reward model as our value function leads to equivalent reward but higher drift compared to using an optimal value function.



(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 14: Ablation of decay coefficient for Elastic Reset on IMDB. Original decay is 0.995. We plot the mean and show error bars for standard error over 5 seeds.

To demonstrate this phenomenon, we compared what happens if we don't train our value function at all but use a frozen reward model as our value function. Since our value function is GPT-2 but our reward model is DistilBERT, we first train a GPT-2 reward model similar to DistilBERT. We then compare regular PPO with a frozen value function i.e. just the sequence-level reward model vs a training value function. In Figure 15 we find that the learning value function is clearly better than the frozen one. They both reach the same reward but a more optimal value function leads to less drift for the same reward.

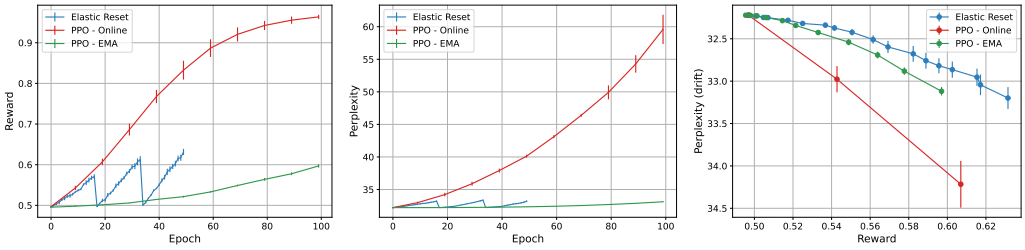


(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 15: Ablation of learning vs frozen value function for PPO on IMDB.

D.5 Explaining Elastic Reset: The Benefit of EMA

The second is by smoothing optimization with an EMA. Work in other fields e.g. DINO in SSL, has leveraged EMA for stability and improved generalization. By resetting to an EMA and resetting our EMA, we further smooth the gradients to alignment. To demonstrate, we run PPO as usual but keep track of its EMA, without resetting to it. We then plot the accuracy of the online and EMA networks over training. To make the effect larger, we use a smaller KL $\beta = 0.01$ and train our PPO for 100 epochs and plot results in Figure 16. We find that just keeping the EMA model is an effective way to improve the task / drift tradeoff. But we also see that it is noticeably slower than our method. PPO - Online needs to reach nearly the maximum possible reward for its EMA to improve only slightly on performance.



(a) Task Performance (b) Language Drift (c) Pareto Frontier Graph

Figure 16: PPO trained on IMDB with a lower KL coefficient for 100 epochs, comparing its online network performance vs EMA model performance. We cut off PPO - online to just three data points in the pareto graph for clarity and visual scale.