

Latent Space Communication via K-V Cache Alignment

Lucio M. Dery¹, Zohar Yahav¹, Henry Prior¹, Qixuan Feng¹, Jiajun Shen¹ and Arthur Szlam¹

¹Google DeepMind

Solving increasingly complex problems with large language models (LLMs) necessitates a move beyond individual models and towards multi-model systems that can effectively collaborate. While text has traditionally served as the medium for inter-model communication, a richer and more efficient exchange is possible if models can access each other’s internal states directly. In this paper, we propose learning a shared representation space that aligns the k-v caches of multiple models, creating a high-bandwidth channel for collaboration without altering the underlying pre-trained parameters. We do so by augmenting each model with adapters to translate its state into and out of this shared space. Via a suite of experiments with Gemma-2 models, we demonstrate that this approach not only enables seamless inter-model communication but also improves individual model performance. We also show that the shared space allows for the direct transfer of learned skills, such as soft prompts, between different models. Our work represents a significant step towards a future where models can fluidly share knowledge and capabilities.

Keywords: *model collaboration, k-v cache, module portability, LLM*

1. Introduction

As the abilities of frontier large language models (LLMs) continue to expand, our community is rapidly producing many models with a diverse range of both generalist and domain-specific capabilities (Brown et al., 2020; Chowdhery et al., 2023; Comanici et al., 2025). This growing pool of models presents both an opportunity and a challenge: how can we effectively orchestrate these individual models to solve problems that are beyond the scope of any single one? Such collaboration between models can take two primary forms. The first is multi-model systems, such as LLM agents (Lu et al., 2024; Wang et al., 2024), model cascades (Kolawole et al., 2024; Yue et al., 2023), model frankensteins (Akiba et al., 2025) and mixture-of-expert models (Filippova et al., 2024). The second is skill reuse (Sabry and Belz, 2024), where modules or knowledge from one model – like soft prompts (Lester et al., 2021) or prefix k-v caches (Li and Liang, 2021) – are shared and leveraged by others.

Towards the north-star of seamless inter-model collaboration (Kandpal et al., 2023), we provide evidence showing the feasibility of allowing mod-

els to read and write to each other’s key-value (k-v) cache latent space. A transformer’s (Vaswani et al., 2017) k-v cache is a rich, natural representation of its internal state. A sufficiently high dimensional cache vector space could allow for much higher communication bandwidth between models than token-level text exchange (Hao et al., 2024; Moschella et al., 2022). By accessing each other’s latent space, models could directly read each others’ state and extract relevant knowledge, follow (soft) reasoning chains, and reuse partial computations or skills, leading to a more efficient collaboration on challenging tasks.

Unfortunately, even given the same input text, we cannot expect the latent spaces of two models trained under different conditions to be interchangeable (Ainsworth et al., 2022). This is because any given token passing through a model creates conditional dependencies on the k-v space of the layer(s) before it, causing the discrepancies in latent space between any pair of models with different parameters to compound with depth (Friedman et al., 2023; Nguyen et al., 2020).

Taking inspiration from the machine translation literature (Lu et al., 2018; Richens, 1956),

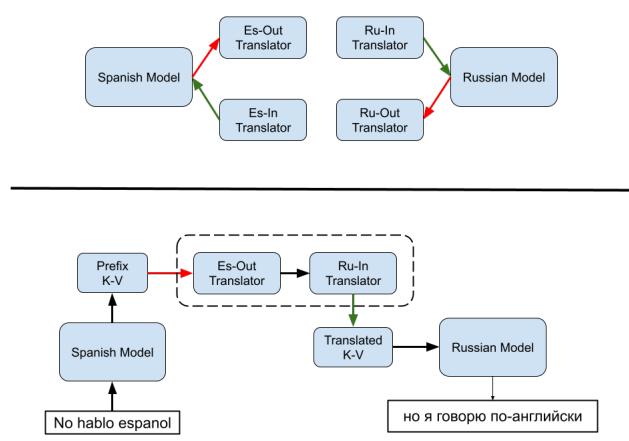


Figure 1 | Overview of an example instantiation of our framework. [Upper] Each model (here Russian and Spanish LMs) is augmented with adapters for translating a k-v cache block into and out of the shared latent space. E.g Es-Out translates a k-v cache out of the Es model’s latent space and into the shared space. [Lower] We can perform mixed language modelling (for context switching settings), by translating generated prefix caches from a source model and continuing to decode from the target model.

given a set of models with possibly disjoint latent spaces, we propose to learn a global, shared k-v cache representation space that any model in the pool can read from and write to. Specifically, we augment each model with two adapters: one for translating its k-v cache block into the shared representation space, and another for translating a given shared space cache block into its model-specific latent space (Figure 1). As designed, this is an implicit (shared) latent space and presents a scalable communication channel with parameter overhead that only grows linearly with the number of models in the pool.

Through a battery of experiments on Gemma-2 (Team-Gemma et al., 2024) open models, we demonstrate that we can teach a set of language models trained under a range of different conditions to read and write to each other’s k-v cache latent space. To achieve this, we provide a scalable architecture and efficient algorithm for learning a global latent space without modifying the individual models’ pre-trained parameters. We

share the following findings:

- Given a set of language models that differ along axes such as size, training data distribution and random initialization, we are able to learn a global shared k-v cache latent representation space.
- Given sufficient data for learning the global latent space, we can improve the individual models’ language modelling performance by passing prefix k-v caches through the shared space before modelling the suffix text.
- A global shared latent space enables learned modules to be portable between models. Specifically, we conduct experiments where soft-prompts, learned on a particular task for one model, can be translated via our global latent space, and used directly by another model to perform the same task. This allows modules learned by individual models to become a shared resource of the whole pool.

2. Methodology

We enhance a pool of models with the ability to read and write to each other’s k-v cache representation spaces. By *read*, we mean a model can translate from another’s k-v cache space into its own, and *write* implies a model translating its k-v cache into another model’s representation space. We do so via a mechanism where individual model latent spaces are mediated by a learned globally shared, but implicit, latent space. In this section, we will detail our framework with architecture specifics and a learning algorithm.

2.1. Setup

Consider a pool of N models indexed as $\{m_1, \dots, m_i, \dots, m_N\}$. We would like to learn a shared latent space, Σ , amongst these models parameterized as follows: for each model, we learn a map $T^{[m_i \rightarrow \Sigma]}$ into the shared space and another map $T^{[\Sigma \rightarrow m_i]}$ out of the space. Thus, assuming that we have the cache for model m_i , $KV^i \in \mathbf{R}^{B \times S \times L_i \times D_i}$, our parameterization instantiates $T^{[m_i \rightarrow \Sigma]} : \mathbf{R}^{B \times S \times L_i \times D_i} \rightarrow \mathbf{R}^{B \times S \times Q}$ and $T^{[\Sigma \rightarrow m_i]} : \mathbf{R}^{B \times S \times Q} \rightarrow \mathbf{R}^{B \times S \times L_i \times D_i}$. Where B , L , S , D are batch, layer, sequence and model dimensions

respectively. For simplicity, we assume that all the models in the pool share the same vocabulary and so for a piece of text, S is the same for all models. Our method does not require this assumption, but it is convenient for conveying a simple setup.

Though each model may have a different number of layers L_i and model dimension D_i , we map all of them to a shared embedding space of fixed size, Q . The shared space therefore has no direct layer-wise demarcation. Instead of forcing a pre-specified assignment per-model, we leave it to the output maps to learn to appropriately reconstruct the layer demarcations. Each model's map into the global shared space mixes layer-wise information ($L_i \times D_i \rightarrow Q$) whilst the mapping out of the shared space performs a model-specific reconstruction of layer-wise information ($Q \rightarrow L_i \times D_i$). We prefer such a design because layers might not have one-to-one correspondence across models of different sizes and even if a pair of models have the same number of layers, it is not guaranteed that they map directly to each other due to the presence of residual connections (Nguyen et al., 2020).

We highlight that each model has an adapter¹ into and an adapter out of the shared space. Since the shared space is global, once a model translates into the shared space, that latent representation can be translated into the local latent space of any of the other models using the corresponding adapter (including the original model itself). This ensures that the number of parameters required to learn a shared space only grows linearly as the number of models. Another advantage of this design is that it is easy to incorporate new models into the pool. Adding a new model, m_k , would simply involve learning the pair $\mathbf{T}^{[m_k \rightarrow \Sigma]}$ and $\mathbf{T}^{[\Sigma \rightarrow m_k]}$ whilst keeping the maps of all the other models (and the shared space itself, Σ) fixed.

2.2. Translator Architecture

Given that the relationship between k-v caches of different models may be highly non-linear, we parameterize the transformations into and out

¹We will use the terms adapter, mapping and translator interchangeably

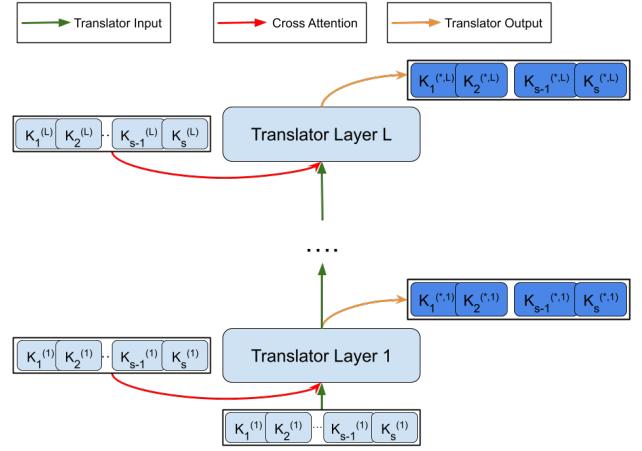


Figure 2 | Cross attention architecture for mapping into and out of the shared latent space. The first layer k-or-v cache serves as seed input to the model. Each layer in the module cross-attends to the corresponding layer in the input k-or-v (not both at the same time) block, meaning that the number of layers in the translator is equal to the number of layers in the corresponding model. We use the superscript * to represent vectors in Σ .

of the shared embedding space with multi-layer transformer models. For any model m_i in our pool, we use the same high level architecture for both $\mathbf{T}^{[m_i \rightarrow \Sigma]}$ and $\mathbf{T}^{[\Sigma \rightarrow m_i]}$.

1. **Input Transformation:** Consider an input cache block (either key or value) KV . For $\mathbf{T}^{[m_i \rightarrow \Sigma]}$, this block will have dimensions $KV^{\text{local}} \in \mathbf{R}^{B \times S \times L_i \times D_i}$ but for $\mathbf{T}^{[\Sigma \rightarrow m_i]}$, which receives a block from the global shared space, $KV^{\text{global}} \in \mathbf{R}^{B \times S \times Q}$. To keep the architecture symmetric we perform a pre-processing step of reshaping $\mathbf{R}^{B \times S \times Q} \rightarrow \mathbf{R}^{B \times S \times L_i \times (Q//L_i)}$ when translating from global to local space with $\mathbf{T}^{[\Sigma \rightarrow m_i]}$. This requires adding a constraint that $\{\forall L \in L_i, Q \bmod L = 0\}$ so the global space is the same dimension for all models. We now apply a simple non-linear transformation consisting of a layer-normalization (Ba et al., 2016), linear transform and a GELU activation (Hendrycks and Gimpel, 2016). The resulting intermediate result has dimensions $\mathbf{R}^{B \times S \times L_i \times D'_i}$ where D'_i is the output dimension of the linear transform.² Note

²Our choice of linear transform preserves L_i but we are

that we differentiates between key and value caches by using different sets of parameters for the linear transform.

2. **Cross Attention:** The main workhorse of the latent space adapter is the cross attention module. Figure 2 is a pictorial representation of the architecture. Using this layer-wise cross-attention architecture allows us to model the hierarchical process that generated the k-v cache block in the first place. For this module, we share parameters between key and value type cache input blocks. At a given layer, we use the output of the previous layer to generate the query vector whilst the key and value vectors are generated by the corresponding k/v cache designated for cross-attention.
3. **Output Transformation:** We concatenate the outputs of the cross-attention layers together (along the last dimension) to obtain $\mathbf{R}^{B \times S \times (L_i \times D'_i)}$ and pass this through a non-linear output transformation (exact same structure as input transformation described above) to output a cache block in $\mathbf{R}^{B \times S \times (L_i \times D_i)}$. This is then reshaped to produce $KV^{\text{local}} \in \mathbf{R}^{B \times S \times L_i \times D_i}$ as the final output.

2.3. Learning Algorithm

We explore two losses to provide signal for learning the global shared latent space.

2.3.1. Reconstruction Loss

We consider a reconstruction loss for converting the k-v cache of one model to another. Let KV_x denote a model's k-v cache corresponding to a piece of text, x .

$$\mathcal{L}_{\text{recon}} = \sum_{m_i, m_j} \left\| \mathbf{T}^{[\Sigma \rightarrow m_j]} (\mathbf{T}^{[m_i \rightarrow \Sigma]} (KV_x^i)) - KV_x^j \right\|^2$$

The equation above shows a translation between models m_i and m_j which passes through the Σ only once. In theory, we can also have chains of translations in and out of the shared space ($\mathbf{T}^{[\Sigma \rightarrow m_j]} \dots \mathbf{T}^{[m_l \rightarrow \Sigma]} \odot \mathbf{T}^{[\Sigma \rightarrow m_l]} \odot \mathbf{T}^{[m_{l-1} \rightarrow \Sigma]} \odot \dots \mathbf{T}^{[\Sigma \rightarrow m_1]}$) linking the two models at the ends of the chain. However, back propagation through this loss would be computationally expensive.

$\mathbf{T}^{[\Sigma \rightarrow m_{l-1}]} \dots \mathbf{T}^{[m_i \rightarrow \Sigma]})$ linking the two models at the ends of the chain. However, back propagation through this loss would be computationally expensive.

2.3.2. Suffix Language Modelling

Even small reconstruction errors can cascade into large prediction errors. Also, an exact reconstruction of the target model's k-v cache means that we are upper bounded by its original language modelling performance. Since we are primarily concerned with language models, we propose a suffix language modelling loss that conditions on a prefix k-v cache translated from another model. Specifically, given a text sequence, x , of length τ , and a source-target model pair (m_i, m_j) , we use our cache-translation on an s -length chunk of generated k-v cache: $KV_{x[:s]}^{i \rightarrow j} = \mathbf{T}^{[\Sigma \rightarrow m_j]} (\mathbf{T}^{[m_i \rightarrow \Sigma]} (KV_{x[:s]}^i))$. We then use this chunk as a prefix for language modelling the remaining $\tau-s$ tokens by the target model:

$$\mathcal{L}_{\text{LM}} = \sum_{m_i, m_j} CE \left(x_{[s:\tau]}, m_j (KV_{x[:s]}^{i \rightarrow j})_{[s \rightarrow \tau]} \right)$$

CE is the cross entropy loss and $m_j(\cdot)_{[s \rightarrow \tau]}$ is the $\tau-s$ length suffix output of model m_j given an input k-v cache prefix. Note that since each model involved in the suffix language modelling loss operates on disjoint sections of the text, we can apply it to models that have different vocabularies. However, the reconstruction loss can not be directly applied to models with disjoint vocabularies unless we a-priori define a mapping between the two vocabularies.

3. Experimental Setting

Pre-trained Models: For our experiments, we pre-train a range of Gemma-2 (Team-Gemma et al., 2024) models of sizes between 100M-400M parameters (not counting the token embeddings). For all our experiments, we keep the pre-trained models frozen and only learn the adapters into and out of the latent space. We use the multilingual C4 dataset (Raffel et al., 2019) with a maximum sequence length of 512 tokens per example sequence. Unless otherwise specified, we

free to expand or contract $L_i \rightarrow \tilde{L}_i$. Using $\tilde{L}_i < L_i$ can reduce the latency of running the adapter but at the cost of expressiveness.

focus on the English, Russian and Spanish splits of this dataset for all our experiments.

Adapter architecture Details: For the adapters into the shared space, we use the Gemma-2 architecture but modify it to allow for cross-attention. We fix the number of heads to 32 and head dimension to 64 for all experiments. Two hyper-parameters control the size of the adapter models (since it has the same number of layers as the base model): `embed_dim_factor` ($= \frac{D'_i}{D_i}$) and `translation_dim_factor` (by what factor the fully connected layers expand the residual stream dimensionality). For all experiments except when we are ablating the adapter model size, we set `embed_dim_factor` = 2 and `translation_dim_factor` = 1. This generally results in each translator being $\sim \frac{1}{4}$ the base model size.

Training the translator: We train for 50k steps using learning rate warmup for 2.5k (5%) steps and cosine decay learning rate schedule (Loshchilov and Hutter, 2016). The initial learning rate value is set to $1e^{-6}$ before warmup and it is eventually decayed to 0.0 at the end of the training. We use the AdamW optimizer (Loshchilov and Hutter, 2017) with the default hyper-parameters in Optax (Hessel et al., 2020) except for the learning rate which we sweep over in the set $[1e^{-3}, 1e^{-4}, 1e^{-5}]$. All gradient norms were clipped to 1.0 before being applied. During training, we sweep batch sizes in the range [256, 512, 1024]. Unless otherwise specified, our experiments use the first 128 tokens as prefix and the remaining as suffix for generation with the target model.

4. Models can be taught to communicate via a global k-v latent space

Using our framework, we design a series of experiments to show that it is possible to teach a set of models (that share the same token vocabulary) to cooperate by leveraging a global shared k-v latent space. We consider two main kinds of model sets: models that are fine-tuned or adapted from the same origin and a set of models that have entirely disjoint training trajectories. For all our experiments in this section, unless otherwise specified, we use only the suffix language modelling loss.

4.1. Same trajectory, different checkpoints

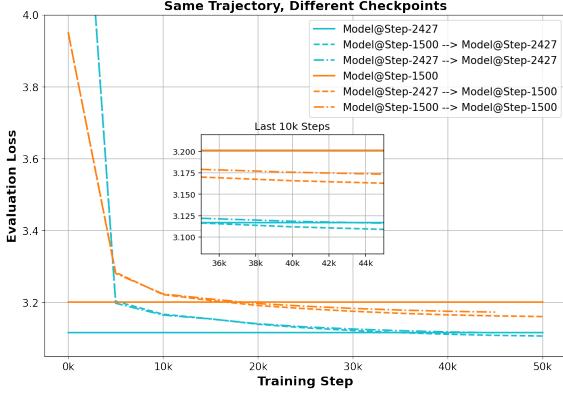
We train a 200M model on a mixed language distribution of 50% English, and 25% each of Spanish and Russian till Chinchilla optimal (Hoffmann et al., 2022). We take 2 checkpoints – the last checkpoint and a checkpoint in the middle of training. We consider the former model as a “stronger” checkpoint and primarily seek to improve the language modelling performance on the earlier checkpoint. Since the training trajectory of the earlier checkpoint is entirely contained in the last, we posit that their k-v latent spaces should be highly related, and translating between these models should be reasonably easy.

Figure 3a shows the results of this experiment. We see an improvement beyond the base model performance when the prefix k-v cache is the output of the global shared space. We posit that the latent space sharpens the relevant k-v cache features in a way that improves the end model’s task performance. We observe this improvement even in the case where we do a cyclic translation of the model’s prefix-kv cache into the global space and back into its local latent space. Figure 3a also suggests that it takes less data to learn a translation from the stronger model’s k-v cache space to the weaker model’s, as indicated by the training step at which using the global k-v space matches or improves upon the base model performance.

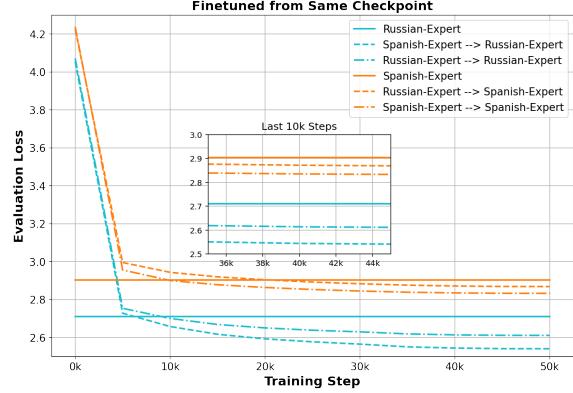
4.2. Same origin model, different fine-tuning distributions

We produce two expert models, one in Russian (trained on 100% Russian data) and another in Spanish (trained on 100% Spanish data) that are branched from the same parent model: the last checkpoint model from Section 4.1 which was trained on the specified mixed language distribution.

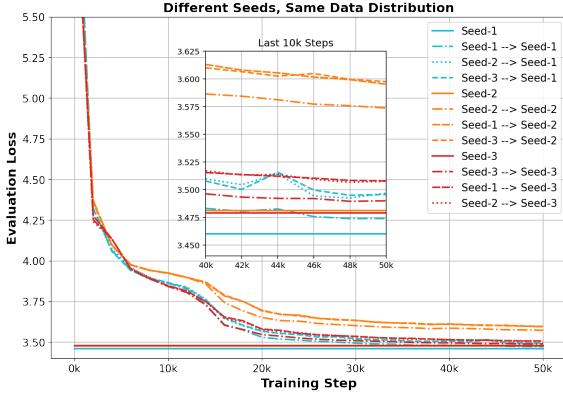
We created a Russian-Spanish parallel dataset for this experiment using the English C4 dataset as the pivot source. Using publicly available Gemini API call, we translated a subset of the English text into Russian and Spanish. By pairing the corresponding translations from each language, we formed a direct Russian-Spanish parallel corpus with 10M pairs for the training set and an 800K



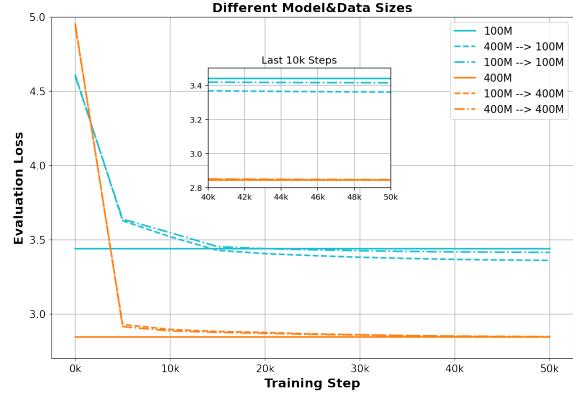
(a) We can improve the performance (on language modelling text suffixes) of two models that are different checkpoints of the same training trajectory by translating prefixes through the global latent-space. All curves are comparable to each other since the eval sets are the same.



(b) We take the same checkpoint and fine-tune it on different data distributions to produce expert models in Russian and Spanish. We are able to learn a global latent-space between these two models. Only curves of the same color are comparable since each language has a different eval set.



(c) Our method works with pools of models that have different random initialization. We see that all our paths come close to or surpass the base models performances.



(d) The 400M model has 4 \times more layers and is trained on more data than the 100M model. Our framework allows this pair of models to collaborate on the language modelling task despite these differences.

Figure 3 | We can learn a shared latent space between different models with different degrees of overlap in their training trajectories. Evaluation loss is negative log likelihood on held-out text.

pairs for the evaluation set.

To illustrate the setup, suppose we are using Russian as the target language and Spanish as the source language. We pass a prefix of Spanish text to the Spanish language expert to obtain a prefix cache, and then generate the Russian suffix text with the Russian language expert after translating the (prefix) cache through the global latent space into the Russian expert’s latent space. We do very weak alignment of the parallel text – we simply skip the first prefix-length-tokens in the target language and continue decoding.

Though this setting represents a more extreme latent space divergence than we have in Section 4.1, our approach is similarly capable of learning a good shared latent space as Figure 3b indicates. Again, when trained on enough data, cyclicly translating a prefix k-v cache through the latent space can actually outperform decoding using the model’s untouched k-v cache.

4.3. Same data distribution but different random initialization

For the above sections, the training trajectories of the model pool have had non-zero overlap. Next, we investigate pools that share no trajectory overlap, but are trained on the same data distribution. We use 3 100M models trained on the same mixed language distribution as in Section 4.1 but initialize each model with different random seeds.

We use a prefix length of 64 tokens and sequence length of 128 for these experiments since 3 models presents a higher memory overhead. As in previous sections, even though these models do not share training trajectories, we are still able to teach them to leverage each other’s k-v caches (Figure 3c).

4.4. Different Model Sizes

Figure 3d shows that we are able to translate the k-v caches of models of different sizes. Specifically, though the 400M model has 16 layers whilst the 100M model has 4 layers, our adapters are able to learn an appropriate mapping. Similar to the earlier experiments in this section, we are able to improve the performance of the weaker 100M model by using prefix k-v caches translated from the stronger 400M model. Note that the two models have disjoint training trajectories not only because of different random initialization, but also due to different amounts of training data (since they are both trained Chinchilla-optimally and thus the 400M is trained with more data than the 100M model).

5. The global k-v latent space is extensible

A desirable property of the global latent space is that it is easily extensible when new models are added to the pool – without necessarily having to retrain the adapters. In this section, we demonstrate that our framework possesses this property. We use the experiment in Section 4.3 as a scaffold and introduce a 4th model (Seed-4) trained with a different seed. We learn the pairs ($\mathbf{T}^{[\text{Seed-4} \rightarrow \Sigma]}$, $\mathbf{T}^{[\Sigma \rightarrow \text{Seed-4}]}$) whilst keeping the global latent-space adapters of Seed-{1,

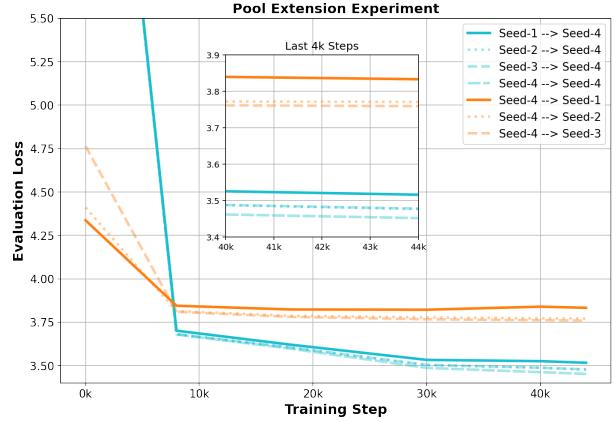


Figure 4 | We extend a pool of 3 models with different seeds-{1, 2, 3} with a fourth model. **Solid lines mark translation paths that were not trained on.** Without explicitly training on these translation paths, we are still able to zero-shot translate between the k-v caches of the two models – seeds-{1, 4} – with only mild performance degradation compared to fully trained paths.

2, 3} fixed.

Our results are shown in Figure 4. Note that during training, we only use the paths involving Seed-{2, 3} to learn the latent-space adapters for Seed-4. Though we never train on the paths (Seed-4 → Seed-1) and (Seed-1 → Seed-4), we are able to zero-shot translate between the two models, achieving comparable performance relative to paths that were explicitly trained on. This demonstrates that the original latent space learned using Seed-{1, 2, 3} is indeed global – cache blocks from different models translated into the shared latent-space are reasonably indistinguishable from each other, thus allowing arbitrary source-target translation paths once a new model learns adapters into the space. Generalization to unseen paths would not be possible if the shared latent-space was path-wise segmented or delineated by model.

Our results in this section also demonstrate that as the size of the pool grows, we may not need all the existing models in the pool to learn the translator for incoming models. This reduces the memory and compute overhead of extending the pool to new models.

6. A global k-v latent space enables module portability

An implication of having a shared k-v latent-space is that learned task specific modules that manifest as k-v caches (like prefix-tuning [Li and Liang \(2021\)](#); and soft-prompts [Lester et al. \(2021\)](#)) can become a shared resource across models. A module trained for one model to perform a specific task can be translated into the k-v cache space of any of the other models in the pool, making those task specific skills accessible to all models without per-model training.

Such module portability is desirable for several reasons. First, the computational cost of learning a skill is amortized across all the models in the pool. We do not have to train a module for each model since we learn the module once and then translate it into the k-v cache space of the remaining models when needed. The saved compute can be redirected to expand the pool of skills instead. Next, module portability enhances data privacy. Only one model needs to be directly exposed to the task data but the resulting module can be used by all other models. Stronger privacy guarantees can be achieved if the original module is trained via differential privacy ([Abadi et al., 2016](#), [Dwork, 2006](#)).

6.1. Setup

We investigate module portability by running meta-learning experiments on the prompt recovery task. The prompt recovery task is defined as follows: given a list of k sequences of text, $\{x_1 \dots x_k\}$, corresponding to completions from a prompted LLM: $x_i \sim m(\cdot | C)$, we wish to recover the context prompt, C by learning a soft prompt ([Lester et al., 2021](#)), $P \in \mathbb{R}^{p \times D}$ such that $m(\cdot | [P]) \approx m(\cdot | C)$. P should generalize to a held out set of example completions as measured by likelihood under the LLM. Since we learn P without seeing C , we are learning to recover the original prompt, but as soft-tokens, instead of text. The prompt completion task is friendly to investigating module portability since we can easily create many task instances, $(C_j, \{x_1 \dots x_k\}_j)$, for training the shared latent-space.

We create a seed set of 65 human written prompts ³ which we use to prompt Gemini-2.5 Flash to generate ~ 6000 similar prompts. We specify that the generated prompts should be diverse and span multiple domains. For each of these 6k prompts, we use an instruction tuned ([Wei et al., 2021](#)) Gemma-2 model to generate 300-800 possible completions of length 30-50 words of which 50 completions are used for evaluations. We use 5.8K prompt completion tasks as meta-train and 200 tasks as the meta-test set.

We use the meta-train task to train (or fine-tune) the latent-space adapters to be able to translate the k-v cache of the learned soft-prompt from the source to the target model. Given a translated cache, we evaluate its loss on the validation set on the target model (i.e zero-shot) and this loss is used as learning signal for updating the adapters into the shared global latent space. At test time, we freeze all the translation and base-model parameters, learn a soft-prompt for each of the 200 meta-eval completion task on a source model and evaluate the performance after zero-shot translating it into the latent-space of the target model.

6.2. Results

[Figure 5](#) shows that we can learn to zero-shot port modules that have been trained on one model to be used by another model. Our approach performs significantly better than the lower bound of using a random (untrained) soft-prompt per-task on the target model. With only a few thousand meta-train tasks, our approaches the upper bound performance of learning the soft-prompt directly for each target model. We posit that with more data, we can match and possibly exceed this baseline performance since our experiments featured in [Figure 3](#) show that the shared global-space can learn to generalize beyond the base model performance.

Note from [Figure 5](#) that it is beneficial to pre-train the global latent space before fine-tuning on the meta-train set. We get the best performance by pre-training on the suffix language modelling task without the reconstruction loss.

³See Appendix B for examples

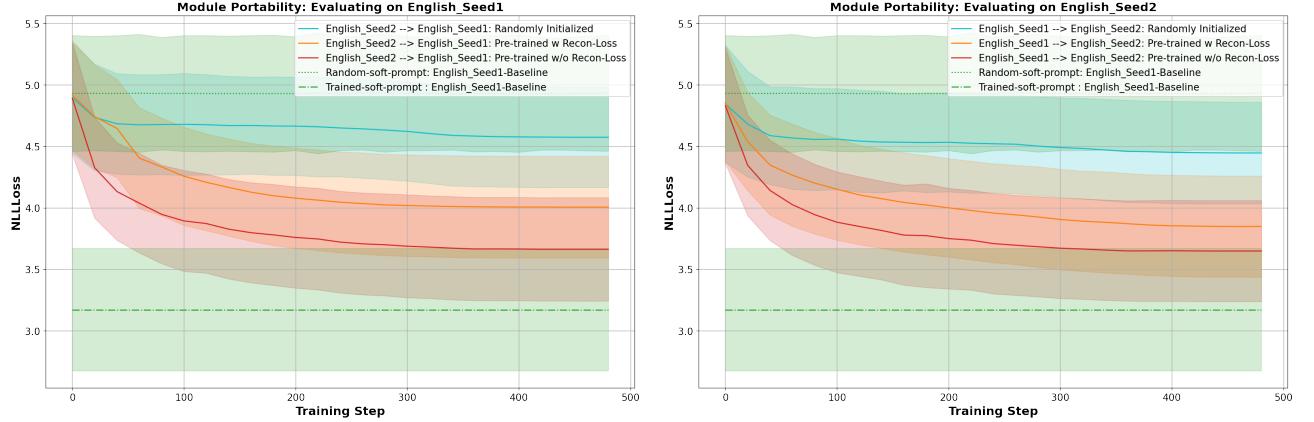


Figure 5 | We adapt the shared global space between two models trained on different seeds to allow for zero-shot module portability. Shaded areas correspond to standard deviation over the 200 meta-eval task. We compare using pre-trained latent space adapters versus randomly initialized ones (cyan).

	Eval Loss	Identity	Linear		Our Translator (Best HP)			
			269M	806M	238M	323M	476M	645M
Seed 1	3.089	4.941	3.090	3.092	3.076	3.068	3.061	3.056
Seed 2	3.117	3.967	3.110	3.111	3.110	3.105	3.091	3.076

Table 1 | We experiment with 2 200M models with different random initialization. Losses reported are language modelling loss when the target model is applied to suffix text. Translator parameter counts in are the total counts across all adapters.

7. Ablations

7.1. Performance improves with shared space adapter model size

We ablate the use of the multi-layer cross-attention architecture introduced in Section 2.2. We compare to an identity and linear mapping as baselines. For the linear mapping, we have separate maps for k- and v-caches respectively which are applied to the layer-wise concatenated k-v caches.

The results in Table 1 show that using an identity mapping (no learned parameters) yields poor performance even though the two models differ only in their random initialization. This mapping has the further disadvantage that it can only be used when the models’ k-v cache dimensions match exactly. It is interesting to see that that we match or exceed the base model performance using a simple linear map. This is in line with recent work like Lähner and Moeller (2024); Moschella et al. (2022) that suggest that the geometries of

latent spaces of deep neural networks trained on the same data distribution tend to be approximately equal modulo linear transformations. A caveat of the results in Table 1 is that that the linear maps significantly expand the size of the k-v cache dimension to create the shared latent-space (upward projections of 8x and 24x respectively for the 269M and 806M models).

Our translator outperforms the simple linear map with fewer parameters. Unlike the linear map, our cross-attention architecture exhibits desirable scaling behavior: consistent improvement in performance as we increase the size of the latent-space adapters.

7.2. We do not need to see all paths in a single batch during training

For the learning objectives proposed in Section 2.3, we sum over multiple paths through the latent space to compute the loss. The number of possible paths is combinatorial in the number of models in the pool. Thus, it would be computa-

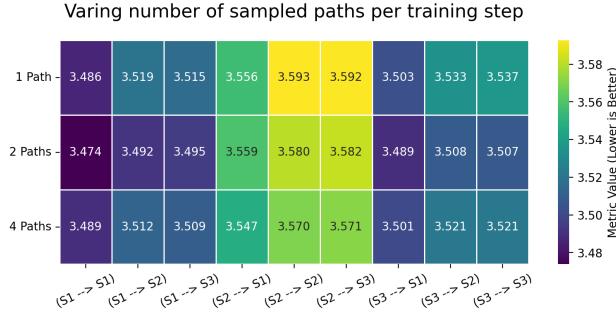


Figure 6 | We can learn to align latent-spaces without using all possible paths per-step during training. More paths per-step is not unilaterally better. For some translation directions, having 2 paths results in superior performance over 4.

tionally inadvisable, to consider all possible paths during each step when training the shared latent space.

For all our experiments with 2 models, we use all the possible 4 paths during training. However, instantiating 9 paths for training the latent space for 3 models would be both slow and memory intensive. We use the setup from Section 4.3 and experiment with the impact of using significantly fewer paths than possible for training. Figure 6 shows the results of our experiments – each point is the minimum over a set of hyper-parameter sweeps.

The average base performance of the 3 seed models is 3.460 NLL (negative log-likelihood) on the evaluation set. Even with 1 translation path per step, we can learn a map that brings us close to this baseline when we average over all 9 possible paths (3.53 NLL). When translating from Seed-1 , 3 to other models, we get the best performance using 2 paths per batch whilst 4 paths work best for translating from Seed-2 to other models. We posit that some amount of stochasticity is important to achieve good generalization of the learned latent space.

7.3. Reconstruction loss is unnecessary in the presence of the suffix language modelling loss

Many works in the space of latent space alignment use the reconstruction loss as a primary

	English-Seed1 → English-Seed2	English-Seed2 → English-Seed1
Recon Weight = 0.0	3.406	3.411
Recon Weight = 0.5	3.531	3.501
Recon Weight = 1.0	3.626	3.534

Table 2 | At fixed data, using the reconstruction loss did not improve learning efficiency

objective (Jha et al., 2025; Lähner and Moeller, 2024). We however find that the presence of the target task objective (in our case, language modelling loss), obviates the need for the reconstruction loss. Thus, for all our primary results, we excluded the reconstruction loss unless otherwise explicitly stated.

8. Related Work

Many areas of machine learning are concerned with solving complex problems by enabling effective communication between groups of models. Recent works in using LLMs as agents feature component models communicating with each at a high level of abstraction via natural language or structured data formats like JSON or query language (Wang et al., 2024, 2023). In the fields of multi-agent reinforcement learning (Gronauer and Diepold, 2022) and emergent communication (Lazaridou and Baroni, 2020), the messages passed between models usually take the form of low dimensional vectors that connect the output of one model to the input of another (Chaabouni et al., 2022; Foerster et al., 2016; Lo et al., 2023; Wu et al., 2025). Our framework provides higher bandwidth communication channels and facilitates much stronger model collaboration by allowing models to directly access each others’ internal state in the form of the k-v cache latent spaces.

Our framework also has implications for coarse-grained mixture of expert architectures like Di-PaCo (Douillard et al., 2024) and Small-Talk (Filippova et al., 2024) where text generation proceeds by periodically switching between a set of expert models. These architectures tend to incur high inference latency due to the need to re-compute the prefix k-v caches whenever the router switches between models. By learning to translate between the k-v cache spaces of the mod-

els in the ensemble, we can significantly reduce the cost associated with naive recomputation of the prefix cache per model.

Our research builds upon the foundational work in latent space alignment, which seeks to bridge the semantic gap between disparate, pre-trained models (Jha et al., 2025; Maiorca et al., 2023). Lähner and Moeller (2024) demonstrate that direct linear transformations can effectively map one model’s latent space to another’s, a finding that makes the prospect of inter-model communication computationally feasible. Similarly, work on relative representations by Moschella et al. (2022) provides a powerful method for zero-shot composition, allowing modules from different models to be combined without fine-tuning by creating an invariant representational format. The techniques in this area have largely been applied to vision models and simpler architectures like auto-encoders with the aim of statically “stitching” together components from different models. Our work proposes a different motivation for alignment – enabling dynamic, stateful collaboration between models.

9. Conclusion

In this paper, we presented a scalable and effective method for enabling direct latent space communication between large language models by learning a shared k-v cache representation space. This approach not only bridges the gap between incompatible models trained under various conditions but also unlocks new benefits, including improved performance and the ability to transfer learned skills between models in a zero-shot manner.

This work opens up many exciting avenues for future research. An immediate next step is to scale our framework to a larger, more heterogeneous pool of models, including those from architectural families vastly different from those explored in this work. While this work focused on language modeling as a proof of concept, future work should also explore more complex downstream tasks like agentic research (Lu et al., 2024) and software engineering (Wang et al., 2024). Ultimately, this line of inquiry moves us closer to a

paradigm of seamless inter-model collaboration, where the collective capabilities of diverse agents can be harnessed to solve problems of unprecedented complexity.

References

- M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.
- S. K. Ainsworth, J. Hayase, and S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.
- T. Akiba, M. Shing, Y. Tang, Q. Sun, and D. Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, pages 1–10, 2025.
- J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- R. Chaabouni, F. Strub, F. Altché, E. Tarassov, C. Tallec, E. Davoodi, K. W. Mathewson, O. Tielemans, A. Lazaridou, and B. Piot. Emergent communication at scale. In *International conference on learning representations*, 2022.
- A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- G. Comanici, E. Bieber, M. Schaeckermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini

- 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- A. Douillard, Q. Feng, A. A. Rusu, A. Kuncoro, Y. Donchev, R. Chhaparia, I. Gog, M. Ranzato, J. Shen, and A. Szlam. Dipaco: Distributed path composition. *arXiv preprint arXiv:2403.10616*, 2024.
- C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.
- A. Filippova, A. Katharopoulos, D. Grangier, and R. Collobert. No need to talk: Asynchronous mixture of language models. *arXiv preprint arXiv:2410.03529*, 2024.
- J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.
- D. Friedman, A. K. Lampinen, L. Dixon, D. Chen, and A. Ghandeharioun. Comparing representational and functional similarity in small transformer language models. In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023.
- S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.
- S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, and Y. Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.
- D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: composable gradient transformation and optimisation. *JAX*, <http://github.com/deepmind/optax> (last access: 4 July 2023), version 0.0, 1, 2020.
- J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.
- R. Jha, C. Zhang, V. Shmatikov, and J. X. Morris. Harnessing the universal geometry of embeddings. *arXiv preprint arXiv:2505.12540*, 2025.
- N. Kandpal, B. Lester, M. Muqeeth, A. Mascarenhas, M. Evans, V. Baskaran, T. Huang, H. Liu, and C. Raffel. Git-theta: A git extension for collaborative development of machine learning models. In *International Conference on Machine Learning*, pages 15708–15719. PMLR, 2023.
- S. Kolawole, D. Dennis, A. Talwalkar, and V. Smith. Revisiting cascaded ensembles for efficient inference. *arXiv preprint arXiv:2407.02348*, 2024.
- Z. Lähner and M. Moeller. On the direct alignment of latent spaces. In M. Fumero, E. Rodolá, C. Domine, F. Locatello, K. Dziugaite, and C. Mathilde, editors, *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*, volume 243 of *Proceedings of Machine Learning Research*, pages 158–169. PMLR, 15 Dec 2024. URL <https://proceedings.mlr.press/v243/lahner24a.html>.
- A. Lazaridou and M. Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.
- B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- Y. L. Lo, C. S. de Witt, S. Sokota, J. N. Foerster, and S. Whiteson. Cheap talk discovery and utilization in multi-agent reinforcement learning. *arXiv preprint arXiv:2303.10733*, 2023.
- I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

- I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.
- Y. Lu, P. Keung, F. Ladakh, V. Bhardwaj, S. Zhang, and J. Sun. A neural interlingua for multilingual machine translation. *arXiv preprint arXiv:1804.08198*, 2018.
- V. Maiorca, L. Moschella, A. Norelli, M. Fumero, F. Locatello, and E. Rodolà. Latent space translation via semantic alignment. *Advances in Neural Information Processing Systems*, 36:55394–55414, 2023.
- L. Moschella, V. Maiorca, M. Fumero, A. Norelli, F. Locatello, and E. Rodolà. Relative representations enable zero-shot latent space communication. *arXiv preprint arXiv:2209.15430*, 2022.
- T. Nguyen, M. Raghu, and S. Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.
- R. H. Richens. General program for mechanical translation between any two languages via an algebraic interlingua. *Mechanical Translation*, 3(2):37, 1956.
- M. Sabry and A. Belz. Assessing the portability of parameter matrices trained by parameter-efficient finetuning methods. *arXiv preprint arXiv:2401.14228*, 2024.
- Team-Gemma, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- X. Wang, B. Li, Y. Song, F. F. Xu, X. Tang, M. Zhuge, J. Pan, Y. Song, B. Li, J. Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.
- Z. Wang, G. Zhang, K. Yang, N. Shi, W. Zhou, S. Hao, G. Xiong, Y. Li, M. Y. Sim, X. Chen, et al. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*, 2023.
- J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Fine-tuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- S. Wu, Y. Wang, and Q. Yao. Dense communication between language models. *arXiv preprint arXiv:2505.12741*, 2025.
- M. Yue, J. Zhao, M. Zhang, L. Du, and Z. Yao. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*, 2023.

A. Model Details

Table 3 | Overview of model parameters and design choices.

Parameters	100M	200M	400M
d_{model}	960	960	960
Layers	4	8	16
Pre-norm	yes	yes	yes
Post-norm	yes	yes	yes
Non-linearity	GeGLU	GeGLU	GeGLU
Feedforward dim	15360	15360	15360
Head type	GQA	GQA	GQA
Num heads	4	4	4
Num KV heads	1	1	1
Tied embedding	yes	yes	yes

B. Prompt Examples

Here are some example prompts we used for the prompt completion task.

- tell me a story about how the duck and the rabbit got angry with each other, but then later realized it was all a misunderstanding.
- can you teach me how to solve systems of linear equations? give me lots of examples.
- describe for me the history of the three kingdoms period in Chinese history.
- describe an excercise routine for dogs that could be used as a soft-drink ad.
- write a description of a disaster movie in the form of a cake recipe.
- tell me knock knock jokes, but make sure they are not funny.
- start by telling a childrens story about a dog and a cat but it seemlessly becomes a spy thriller.
- list all the prime ministers of the UK but encode them with a ceaser cipher.
- give me a treatment for an Eat Pray Love sequal Eat2Pray2Love2.
- write the code for red black trees first in python, then c# then in a very obscure language.
- write a musical about Demis Hassabis.
- pick one of the three stooges from the three stooges and write a poem about them.
- roll a 20 sided die and tell me the result.
- generate me a dungeons and dragons character for me, but cheat and make the stats slightly too good.
- write an essay about Robert Moses but make it kind of mediocre and have at least 2 mistakes in it.
- assume $1+1=3$ and write a program to solve the quadratic equation $x^2-3x+2=0$.

- you are a pteridactl who is trying to convince a friend to buy a car.

C. Overflow Experimental Results

	English-Seed1 → English-Seed2	English-Seed2 → English-Seed1
Random Soft-Prompt	4.932 _{0.471}	4.932 _{0.471}
Randomly Initialized	4.847 _{0.474}	4.897 _{0.448}
Pre-trained w Recon-Loss	4.852 _{0.473}	4.917 _{0.445}
Pre-trained w/o Recon-Loss	4.832 _{0.469}	4.890 _{0.453}

Table 4 | Zero-shot performance on module portability task **before meta-training**. Meta-training is key to adapting our framework for module portability.