Parameter-Efficient Finetuning for Robust Continual Multilingual Learning

Kartikeya Badola   Shachi Dave   Partha Talukdar
Google Research India
{kbadola, shachi, partha}@google.com

Abstract

We introduce and study the problem of **Continual Multilingual Learning** (CML) where a previously trained multilingual model is periodically updated using new data arriving in stages. If the new data is present only in a subset of languages, we find that the resulting model shows improved performance only on the languages included in the latest update (and a few closely related languages) while its performance on all the remaining languages degrade significantly. We address this challenge by proposing **LAFT-URIEL**, a parameter-efficient finetuning strategy which aims to increase the number of languages on which the model improves after an update, while reducing the magnitude of loss in performance for the remaining languages. LAFT-URIEL uses linguistic knowledge to balance overfitting and knowledge sharing across languages, allowing for an additional 25% of task languages to see an improvement in performance after an update, while also reducing the average magnitude of losses on the remaining languages by 78% relative.

1 Introduction

A learning-based NLP model may need to be periodically updated for a variety of reasons, e.g., to incorporate newly available training data, adapt to data shifts, etc. Continual learning (Thrun and Mitchell, 1995; Kirkpatrick et al., 2017) and Online learning (Shalev-Shwartz et al., 2012) are paradigms where a model is sequentially trained on packets of new training data, without having access to old training data. In such settings, the goal is to ensure that the model is able to incorporate incremental knowledge from the new data without forgetting the knowledge obtained from prior training.

As multilingual NLP grows in prominence, the underlying models used for multilingual tasks are increasingly being developed as a single deep neural network trained on data from all supported languages (Devlin et al., 2019; Conneau et al., 2020; Xue et al., 2021). Having a shared multilingual model instead of one model per language allows one to reduce the number of models to train and maintain for the downstream task, improve performance on lower-resource languages due to cross-lingual sharing of knowledge, and improve inference on code-mixed inputs. Just like monolingual models, multilingual NLP models also need to be regularly updated, thereby making them suitable for application of continual learning strategies.

However, continual learning of multilingual models involves additional challenges due to involvement of multiple languages. For instance, during any update, the new training data may cover only a small subset of the languages, which may negatively impact the performance on languages not represented in this new data. This scenario is often true for multilingual models deployed in production settings. In spite of its importance and real-world significance, the problem of **Continual Multilingual Learning (CML)** has not been much explored. We fill this gap in this paper.

In the CML setting, a single-task multilingual model needs to be continually updated with additional training data from a subset of the supported languages arriving in stages, while keeping the model capacity fixed and without relying on any data from previous training steps. Given a shared multilingual model, the goal of updating it on new data for the same task would be to (1) improve the model performance across most, if not all, languages and (2) ensure that none of the languages incur a significant loss in performance. The second scenario may occur if the new training data is highly skewed towards a subset of languages, making it easier for the model to overfit on the language specificities of the new data while forgetting the same for languages not represented in this update. In our study, we find that balancing the two goals is
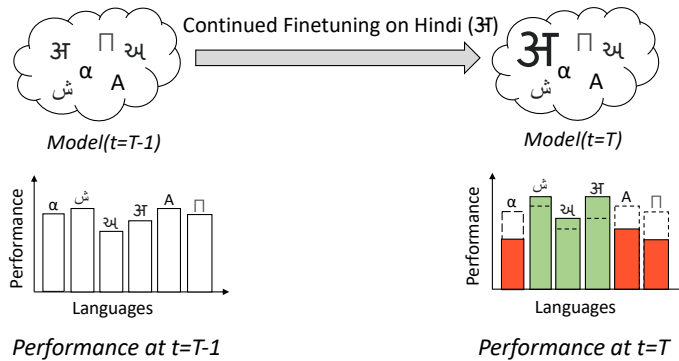
Figure 1: In the Continual Multilingual Learning (CML) setup, a previously trained multilingual model at time $T-1$ is further finetuned on new data (for the same task) coming from a subset of seen languages (only Hindi in this example), resulting in the updated model at time T. We observe that the additional training results in improved performance on the new data languages (Hindi) and a few closely related languages (due to positive transfer), while negatively affecting the remaining languages. This paper proposes strategies to maximize positive transfer while minimizing negative impact on the remaining languages during CML.

non-trivial and the model incurs significant losses across a subset of languages if it is finetuned on the new data in an unconstrained manner. We study this phenomenon over four tasks and find the same non-ideal behaviour across all experiments for the baseline finetuning strategy. The CML setup is closest in spirit to M'hamdi et al. (2022), where a multilingual model is trained from *scratch* with additional model parameters added during updates. In contrast, CML builds on top of an *existing* multilingual model while keeping model capacity *fixed*.

We start with the intuition that constraining the number of trainable parameters in the network would help control for losses due to language-specific forgetting. We operationalize this through different parameter-efficient finetuning strategies, namely Adapters (Houlsby et al., 2019) and Composable Sparse-finetuning (Ansell et al., 2022). We find that such methods provide a middleground by allowing limited cross-lingual sharing of knowledge while reducing model's tendency to overspecialize on the languages of the new data.

With this initial promise, we develop **LAFT-URIEL**, a novel finetuning strategy which uses Adapters and URIEL language similarity metrics (Littell et al., 2017) to balance the trade-off between encouraging positive cross-lingual transfer and discouraging language-specific forgetting. Our contributions are as follows:

1. We introduce and study Continual Multilingual Learning (CML) where a multilingual model is periodically updated with batches of new data from a subset of the languages

covered. This is an important but unexplored problem of practical significance.

2. In the CML setup, we show that a model may suffer from drastic language-specific losses if the new training data is skewed towards a subset of languages, thus making the resulting model unfit for multilingual downstream applications.

3. We propose LAFT-URIEL, a novel finetuning strategy which uses Adapters and syntactic language similarity to maximize positive transfer during CML, while minimizing negative impact across languages.

We present the CML setup in Figure 1.

2 Problem Setup

We consider a setting where a trained, task-specific multilingual model, which we will call as the *deployed model*, is further finetuned on new finetuning data for the same task, to give us the *updated model*. To ensure the best possible multilingual performance, we will assume that the deployed model has been previously trained on data from all supported languages for the task (say N_L in number).

In an update, the deployed model will be finetuned on new task-specific data, to give us the updated model. In the real world, as one may have no control over how the new data is distributed across languages, we will assume the worst case scenario for our setup (i.e., maximum skew) where the new data is only present in one of the N_L languages. We divide the entire setup into two stages:

1. Inception stage where we setup the first deployed model by training a transformer model (initialized by pre-trained mBERT (Devlin et al., 2019) or XLM-RoBERTa (Conneau et al., 2020) checkpoints) on task data in all N_L languages.

2. Continuation stage where we further finetune the deployed model on the new finetuning data (in one of the N_L languages) to give us the updated model. There can be multiple continuation stages that are sequentially performed one after another.[1] Formally, we define our setup using the following notations. For inception stage:

$$\text{model}(\text{t} = 0^-) \xrightarrow{\{l_1, l_2, ..., l_{N_L}\}} \text{model}(\text{t} = 0) \quad (1)$$

where $\text{model}(\text{t} = 0^-)$ is the untrained model, $\text{model}(\text{t} = 0)$ is the first deployed model and $\text{model}(\text{t} = \text{T})$ is model after T continuation stages. $\xrightarrow{\{l_{i_1}, l_{i_2}, ..., l_{i_k}\}}$ denotes finetuning using data from languages $\{l_{i_1}, l_{i_2}, ..., l_{i_k}\}$. The Tth continuation stage can be written as:

$$\text{model}(\text{t} = \text{T} - 1) \xrightarrow{\{l_i\}} \text{model}(\text{t} = \text{T}) \quad (2)$$

where $i \in \{1, 2, ..., N_L\}$.

To compare different finetuning strategies, we focus on the $\text{t} = 0$ and $\text{t} = 1$ transition and subsequently study the effects of multiple sequential continuation stages on the model using our proposed strategy. For a task in N_L languages, there can be N_L different continuation stages to transition from the deployed model at $\text{t} = 0$ to the updated model at $\text{t} = 1$ (since the continuation stage data can come from any one of the N_L task languages). We consider all such cases. Training data for each finetuning stage is created by partitioning the full training data into equal parts, independently across all languages.

On a fixed test set, we expect the language-wise performance of the deployed model and the updated model to differ due to multiple reasons: (1) additional task-learning using new task data (2) catastrophic forgetting of language-specific knowledge (3) positive or negative cross-lingual transfer. Ideally, we would want the performance delta to be positive or neutral across each language. Hence, the goal is to devise strategies that encourage language-agnostic task learning and positive

Dataset	Task Type	Languages
PAN-X	Token level	en, hi, bn, zh, ta, ja, ar
UDPOS	Token level	en, hi, ja, ta, zh, ar
MTOP Class.	Sentence level	en, de, es, fr, hi, th
MTOP NSP	Seq2seq	en, de, es, fr, hi, th

Table 1: Tasks studied in the CML setup. We use the ISO 639-1 language codes to denote languages and present the mapping in Appendix B.

cross-lingual transfer while inhibiting catastrophic forgetting.

We select four representative tasks from three families: token-level, sentence-level and seq2seq; in order to ensure that our methodology is broadly applicable. These are: PAN-X (aka WikiANN) (Pan et al., 2017) for NER tagging, Universal Dependencies v2.5 for POS tagging (UDPOS) (Nivre et al., 2018), MTOP (Li et al., 2021) for domain classification and semantic parsing (NSP). More details for each task is provided in Table 1 and Appendix B. For PAN-X and UDPOS, the language selection is done based on the pre-trained weights available for Lottery Ticket Sparse Fine-Tuning (§4.1). The resulting set of languages across tasks offer a diverse mix of typologies, language families and geographic location of prominence.

Each experiment is repeated three times by varying the random seed. The seed also varies the examples selected for constructing finetuning sets of different stages.

3 Baseline Finetuning Strategy and Metrics

The baseline finetuning strategy in our setup would be finetuning while keeping all the parameters of the model trainable during a continuation stage. We call this the *Full Finetuning* (**FFT**) baseline.

We anticipate that continued finetuning on new task data (which is skewed towards a particular language) would cause non-uniform changes in language-wise performance on a fixed test set. In particular, we expect performance gains on the language seen during a continuation stage and losses across some subset of the remaining languages.

To compare different finetuning strategies, we measure the percentage change in language-wise performance after the continuation stage[2].

---

[1]The dev and test sets are kept the same across all finetuning stages and covers all languages for the given task.

[2]We use percentage change as opposed to magnitude change since negative change in performance on a language which had poor base performance should be penalized more than the same for a language with high base performance.

For the updated model to be fit for deployment (e.g., in a production setting), it is necessary to ensure that the performance drop on any language is not too high. Also, given the shared multilingual model, an ideal strategy should be able to spread the gains in performance across most if not all languages. To this end, we construct the following metrics to compare different strategies in our setup:

AvgPercentLoss: *Average magnitude of percentage loss after continuation*. Calculated by averaging the absolute percentage change in performance over all languages which suffered a loss in performance. For an ideal model, this should be 0.

NumImprovedLangs: *Average number of languages with a positive change in performance after a continuation stage*. For an ideal model, this should be the count of all supported languages for a given task, N_L.

Since there can be N_L different continuation stages (where the new data is only present in one of N_L languages for the task), we report the average of the above two metrics across all such transitions.

Additional constraints: After a continuation stage, we would at the very least expect that, (1) the sum of gains are higher than the magnitude of the sum of losses and (2) the magnitude of maximum gain are higher than the magnitude of the maximum loss in performance. A finetuning strategy which is unable to obey these constraints can simply be declared as unfit for our setup. We therefore compute an average of $\mathrm{sum(gains)}/\mathrm{abs(sum(losses))}$ (**SumRatio**) and $\mathrm{max(gains)}/\mathrm{abs(max(losses))}$ (**MaxRatio**) across the different continuation stages and check whether the two values are ≥ 1.

4 Parameter-efficient Finetuning for Continual Multilingual Learning

We propose the use of parameter-efficient finetuning methods to build improved finetuning strategies in our setup. The benefits of using these methods would be two-fold. Firstly, such methods should allow one to constrain the changes being made in the model, which should help in controlling the losses due to forgetting. Secondly, these methods can be used to decompose task learning into language-specific and language-agnostic parts. This property can be used to update the model in a language-agnostic fashion which should help in spreading gains across languages.

The following subsection will give an overview of the methods we intend to use to build improved

finetuning strategies for the task.

4.1 Methodologies

Lottery Ticket Sparse Fine-Tuning (LT-SFT) (Ansell et al., 2022): proposes to keep only a subset of parameters trainable during finetuning. This allows one to learn a sparse vector of differences (*update matrix*) with respect to the base model. Update matrices for different sub-tasks can be composed together by simply summing up the diffs.

Using the above compositionality property, one can build a pipeline to decompose multilingual task learning into task-specific and language-specific parts. For the language-specific part, we use the pre-trained sparse update matrices for each language, obtained by finetuning on language-specific data for masked language modelling.

Given finetuning data for a task, the task-specific sparse updates are learnt by first applying the language-specific update matrix for the language of the training example and then performing gradient descent on this sparsely modified model. The learned vector of differences can be assumed to be language-agnostic since the model already has language-specific knowledge from the update matrix applied before the forward pass. For multilingual finetuning (e.g., during inception stage), we follow *multi-source* training where data batches are constructed per language and uniformly sampled across languages throughout finetuning. We use LT-SFT to build a stronger baseline for the task, which we will call ***SFT (sparse finetuning)***.

Adapters (Houlsby et al., 2019): Adapters are trainable modules that are inserted in the layers of a transformer network. During finetuning, usually only the adapter modules are kept trainable and these constitute to about 5-10% of the parameters of the network.

In our work, we use adapters to split the model into language-agnostic and language-specific parts and propose a finetuning strategy called ***LAFT (language-specific adapter finetuning)***.

URIEL vectors (Littell et al., 2017): We propose to use URIEL vectors to estimate whether language-specific learning for a given language would be useful for another language by computing the URIEL syntactic distance between the two languages. The syntactic distance is computed as the cosine similarity between the syntactic vectors of any two languages obtained from the URIEL database.

Prior works such as MAD-G (Ansell et al.,

2021) have used URIEL vectors in conjunction with parameter-efficient finetuning methods for generating adapter modules for unseen languages. The generated adapters sometimes perform slightly worse than vanilla adapters on the seen set of languages depending upon the task. We hence stick with vanilla adapters and propose our novel finetuning strategy, **LAFT-URIEL**, for integrating knowledge from the URIEL vectors.

4.2 Proposed Finetuning Strategies

We build our finetuning strategies using the methodologies described in §4.1 and describe the inception and continuation stage for each case. In each strategy our goal is to a) make minimal changes to the shared parameters of the deployed model and b) ensure that such changes are language-agnostic. This should help in spreading the performance gains while also minimizing losses.

Sparse Finetuning (SFT) For the inception stage we follow the standard *multi-source* training (§4.1) using the pre-trained language-specific sparse update matrices. In this stage, the base model is sparsely trainable and the classifier (or the decoder for the seq2seq task) is fully trainable[3].

During the continuation stage, we sparsely update the entire model (base model and the classifier or decoder) on the new finetuning data (again by first applying language-specific sparse updates before forward pass as described in §4.1). Sparse finetuning ensures that the updated model is minimally different from the deployed model (roughly 5-10% parameters are kept trainable).

During inference, we apply the sparse update matrix of the test language before the forward pass.

Language-specific Adapter Finetuning (LAFT) Here the goal is to split the model into language-specific (adapters) and language-agnostic (base model) parts. For the inception stage, we first take the deployed model from FFT and insert (randomly initialized) adapters in each layer of the network. We train the adapter layers and the classifier or decoder on inception stage data for all languages for the task and then create N_L copies of the trained adapters (one for each language). The ith copy is again finetuned (with the base model frozen) on inception stage data but this time only using the data for the l_i language. This gives N_L language specific adapters, a shared base model and a shared
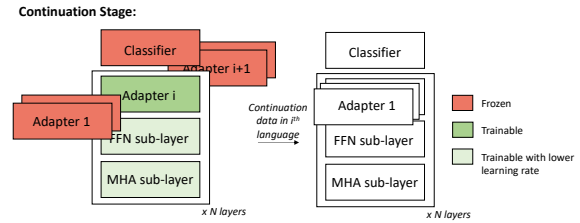

Figure 2: Continued finetuning using our proposed method, LAFT-URIEL. Here we split the model into language-agnostic (base model) and language-specific parts (adapters). The language-agnostic part is trained with a lower learning rate compared to the language-specific part of the network. Lowering of the learning rate is dynamically decided based on composition of the continuation stage data. This helps in sharing performance gains across languages while reducing model's tendency to become overspecialized on the language of the new data. §4.2 for more details.

classifier or decoder (*inception stage* diagram at appendix C). During inference, one can simply swap to the adapter corresponding to the test language.

For the continuation stage, given access to new finetuning data in l_j language, we use the jth adapter during the forward pass and update it using gradient descent. Usually in adapter-based strategies, the shared base model is kept frozen. However, in our setting, we would want to encourage knowledge transfer between languages. At the same time, we would want to make minimum changes to the shared base model to avoid losses due to forgetting. We balance the two goals by keeping the base model trainable but with a much lower learning rate (compared to the adapter layers). Since in the inception stage, the model has associated language-specific learning with the adapters and language-agnostic learning with the base model, this would incentivize the model to not overfit the shared base model on the language regularities of the new data. We find that LAFT shows improved behaviour compared to FFT & SFT.

LAFT using URIEL distances (LAFT-URIEL) We argue that the selection of learning rate (LR) of the base model should be made based on the language-wise composition of the new finetuning data. If we know that the new data is skewed towards a language which is very "different" from the remaining languages, then keeping the LR low would be the desired choice as it is very unlikely that finetuning on this new data would lead to shared gains in performance.

We use URIEL syntactic distance as a measure

---

[3]This gives us a better performing deployed model compared to when the decoder is kept sparsely trainable too.
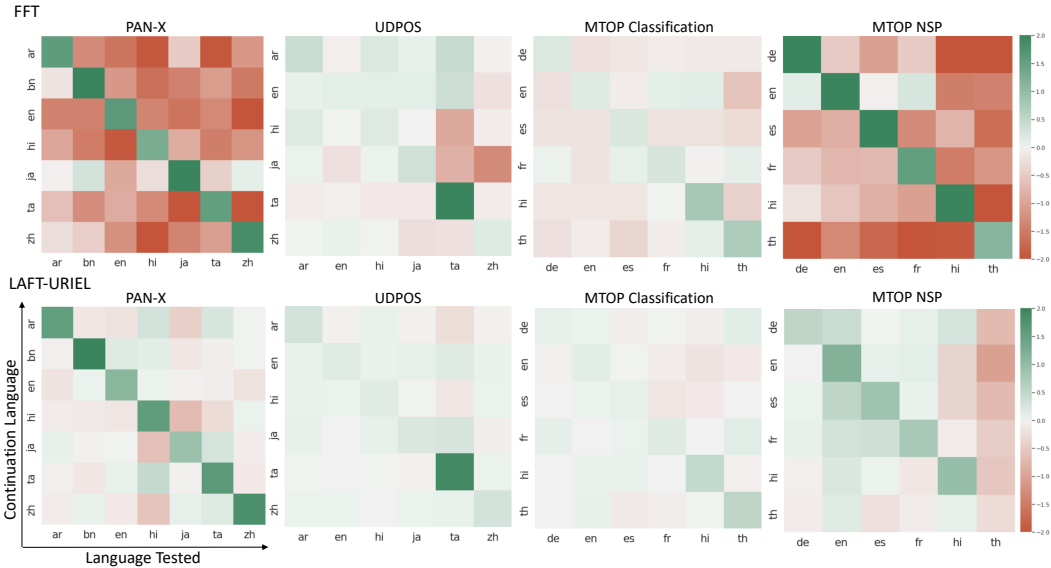
Figure 3: Performance change heatmaps (see §5.1) using FFT (top) and the LAFT-URIEL (bottom) strategies on all four tasks, plotted on the same scale on the right end (+2 to -2%). Here each red cell indicates that there was a loss in performance on the language denoted by the column, after continued finetuning on the language denoted by the row. The colour intensity corresponds to the magnitude of change. Our proposed method, LAFT-URIEL, greatly improves upon the baseline (FFT) by reducing both the number and intensity of red cells in the heatmap.

Lang	Avg syn. distance	LR of base model (in 10^{-5})
en	0.405	5/35
de	0.435	5/50
es	0.415	5/40
fr	0.422	5/45
hi	0.498	5/80
th	0.540	5/100

Table 2: Learning rate (LR) of the base model changing with the language of the continuation stage data for MTOP NSP using LAFT-URIEL strategy (§4.2).

of similarity between different languages. We calculate the LR of the base model by dividing the LR of the adapter layers (kept the same across languages) by a division factor. For continuation stage with data in l_i language, the division factor is computed as a linear function of the average syntactic distance of l_i from $\{l_1, l_2, ..., l_{N_L}\} \setminus \{l_i\}$. We show this calculation for the MTOP NSP task in Table 2. We call this strategy LAFT-URIEL and its continuation stage diagram is represented in Fig 2.

5 Comparison of Finetuning Strategies

In this section, we use the metrics defined in §3 to compare the four finetuning strategies (FFT, SFT, LAFT, LAFT-URIEL) on the four tasks described in §2. It is important to note that the absolute language-wise performances of the deployed models in SFT, LAFT and LAFT-URIEL cases are at

par or slightly greater than the same for FFT (see appendix D). Since our metrics are computed over changes in performance, the above fact ensures that the comparison is fair (or slightly favourable towards FFT).

We ask the following research questions:[4]

1. *How does the behaviour of our proposed strategy differ from that of the naïve baseline (§5.1),*

2. *Do parameter-efficient finetuning methods improve spread of gains after while constraining the losses in our setup? (§5.2 and §5.3),*

3. *How does our proposed strategy perform when there are multiple continuation stages? (§5.4)*

5.1 Behaviour of FFT and LAFT-URIEL

We construct heatmaps to visualize the performance changes observed after a continuation stage (Figure 3). Here each row corresponds to a continuation stage between the $t = 0$ and the $t = 1$ models where the new finetuning data is only present in the language corresponding to the row index. In other words, given the same deployed model, each row corresponds to a different updated model. The column index corresponds to the language used to evaluate the updated model. We present heatmaps for both the FFT (top row) and the LAFT-URIEL (bottom row) strategies in Figure 3. For the FFT

---

[4]results in main paper are using mBERT, see appendix E for results using XLM-RoBERTa
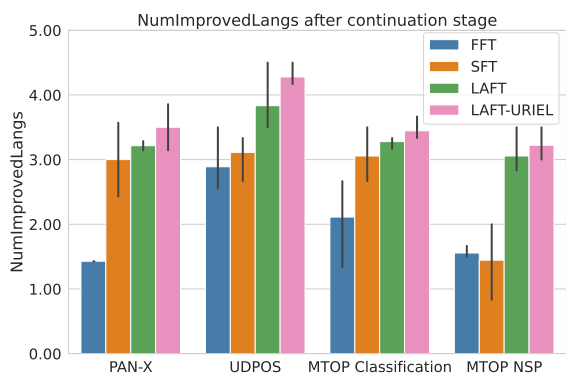
Figure 4: NumImprovedLangs across tasks (std-dev in black). Higher the better. LAFT and LAFT-URIEL, improve spread of gains compared to both FFT and SFT. See §5.2 for more details.
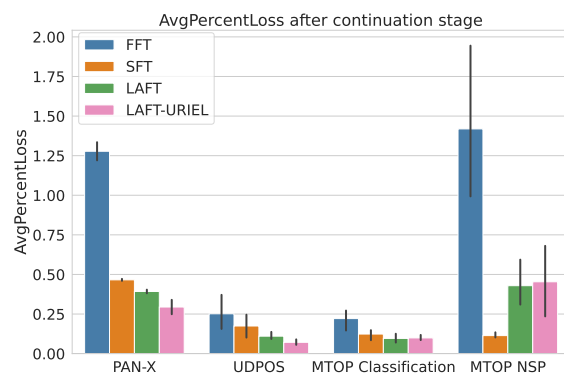

Figure 5: AvgPercentLoss across four tasks (std-dev in black). Lower the better. LAFT-URIEL shows significant improvement over FFT and SFT on 3 tasks. SFT doesn't meet the constraints (Table 3) for MTOP NSP. See §5.3 for more details.

baseline, the diagonal entries are highly positive while many of the off-diagonal entries are negative across all four tasks. This indicates that the model is overfitting on the language specificities of the new finetuning data leading to degraded generalization capabilities across the remaining languages.

For LAFT-URIEL, we observe improved behaviour across all four tasks. The green cells in the LAFT-URIEL are much more evenly spread and higher in number compared to FFT. We also notice that the intensity of the red cells have reduced significantly. This behaviour is much closer to the ideal behaviour than FFT[5]. In the subsequent subsections, we will quantify these observations using the metrics proposed in section 3.

5.2 Measuring Spread of Gains

We plot NumImprovedLangs on all four tasks in Figure 4. We see significant gains over the naïve FFT baseline using SFT across all tasks but UD-POS, indicating that SFT is indeed a stronger baseline for our setup. Both LAFT and LAFT-URIEL improve upon the SFT strategy on this metric. The gains of LAFT-URIEL over LAFT indicates that using URIEL syntactic distance to dynamically compute the learning rate of the base model given the composition of the continuation stage data helps in improving positive transfer across languages. With FFT, an average of only **32.48%** of task languages could improve after the $t = 0$ to $t = 1$ continuation stage. This number increases to **58.08%** using LAFT-URIEL, therefore suggesting that *majority of languages are expected to improve using our proposed strategy when a multilingual model is fur-*

[5]similar improvements using XLM-RoBERTa, appendix E

ther finetuned on new language-specific data for the same task.

5.3 Comparing Losses Incurred

We plot AvgPercentLoss for all the four strategies on all four tasks in Figure 5. We again observe considerable improvement over the FFT baseline using SFT. LAFT and LAFT-URIEL improves upon SFT across all tasks except MTOP NSP. Upon closer analysis, we find out that both the magnitude of gains and losses for SFT in this task are severely constrained, because of which the changes in language-wise performance are close to zero. This is also reflected in Table 3, where the MaxRatio and SumRatio values for SFT are non-ideal for MTOP NSP. We believe that this might be due to the fact that the language-specific update matrices are only available for the encoder of the network because of which the task-language decomposition is hampered. LAFT and LAFT-URIEL satisfy the minimum criteria (value ≥ 1) for all tasks (Table 3). For LAFT-URIEL in the UDPOS task, there are continuation stages where none of the languages incur a loss in performance because of which the average of the two ratios come out to be ∞. This is a significant improvement in behaviour compared to both SFT and FFT. *LAFT-URIEL reduces the magnitude of losses incurred by around 78% relative on an average compared to FFT.*

5.4 Multiple Continuation Stages

We also evaluate LAFT-URIEL on multiple continuation stages, sequentially performed one after another. We consider two trajectories for sequential finetuning: high-resource to low-resource (H2L)

Strategy	PAN-X		UDPOS		MTOP Classification		MTOP NSP	
	MaxRatio	SumRatio	MaxRatio	SumRatio	MaxRatio	SumRatio	MaxRatio	SumRatio
FFT	$1.23_{\pm 0.25}$	$0.50_{\pm 0.11}$	$6.85_{\pm 2.26}$	$5.10_{\pm 3.23}$	$1.19_{\pm 1.02}$	$0.82_{\pm 0.8}$	$1.16_{\pm 0.48}$	$0.48_{\pm 0.27}$
SFT	$1.26_{\pm 0.19}$	$0.76_{\pm 0.58}$	$4.95_{\pm 2.31}$	$2.58_{\pm 0.95}$	$2.73_{\pm 0.72}$	$2.62_{\pm 1.12}$	$0.54_{\pm 0.13}$	$0.36_{\pm 0.16}$
LAFT	$4.62_{\pm 0.53}$	$2.55_{\pm 0.51}$	$11.72_{\pm 3.17}$	$13.25_{\pm 4.23}$	$9.65_{\pm 1.45}$	$17.64_{\pm 3.56}$	$1.26_{\pm 0.31}$	$1.70_{\pm 0.26}$
LAFT-URIEL	$6.30_{\pm 0.68}$	$4.10_{\pm 0.76}$	∞	∞	$12.04_{\pm 3.31}$	$20.82_{\pm 4.54}$	$1.18_{\pm 0.22}$	$1.48_{\pm 0.34}$

Table 3: MaxRatio and SumRatio (§3) for all four strategies across all four tasks. Higher the better. Values highlighted in red are unfavorable and indicates that the updated model is worse than the previous version on an overall aggregated metric. LAFT-URIEL shows close to ideal behaviour on all tasks.

Strategy	PAN-X		UDPOS		MTOP Classification		MTOP NSP	
	%Loss	+veLangs	%Loss	+veLangs	%Loss	+veLangs	%Loss	+veLangs
FFT (L2H)	1.737	1	0.839	1	0.378	3	5.459	5
LAFT-URIEL (L2H)	0.505	4	0.182	2	0.210	4	0.778	2
FFT (H2L)	1.406	1	0.598	1	0.283	3	2.334	2
LAFT-URIEL (H2L)	0.613	4	0.123	4	0.192	4	0.773	2

Table 4: Magnitude of AvgPercentLoss (denoted by %Loss) and NumImprovedLangs (denoted by +veLangs) calculated on the *worst-case continuation stage* for FFT and LAFT-URIEL after multiple continuation stages. H2L and L2H represent two trajectories we consider (§5.4). LAFT-URIEL ensures that the losses are constrained even after multiple continuation stages.

and low-resource to high-resource (L2H) languages (inspired by M'hamdi et al. (2022)) based on number of examples in the training data for a given task[6] and report the metrics on the *worst-case continuation stage* in a trajectory. Given a trajectory of the form $\xrightarrow{\{l_{i_1}\}} \xrightarrow{\{l_{i_2}\}} \dots \xrightarrow{\{l_{i_{N_L}}\}}$, we define the *worst-case continuation stage*, t_w, as follows:

$$\text{model}(t = t_w - 1) \xrightarrow{\{l_{t_w}\}} \text{model}(t = t_w) \quad (3)$$

$$where \; t_w = \underset{t}{\arg\max} \; \text{AvgPercentLoss}(t) \quad (4)$$

i.e the continuation stage where the AvgPercentLoss was maximum in the trajectory. We report the metrics on the worst-case continuation stage in the Table 4. We observe that our proposed strategy consistently reports a value $\leq 1\%$ for worst-case AvgPercentLoss across all tasks. Also, there are > 1 languages with improved performance, even after the worst-case continuation stage. This is a strong result which indicates that our finetuning strategy is able to control losses and spread gains even after multiple continuation stages.

We refer the readers to the Appendix for further experiments which aim to understand (1)

how the size of adapter layers affect the performance of the LAFT strategy (§F), (2) the effect of continued finetuning on closely related languages (§G), and (3) the variance in cross-lingual transfer across seeds, tasks and encoders (§H)

6   Related Works

Continual Learning: A large body of work in the continual learning literature is focused on the *task incremental* setting (De Lange et al., 2021) where the goal is to sequentially introduce new tasks to the network. Elastic weight consolidation (Kirkpatrick et al., 2017) is one of the most widely used algorithms for this setting, however, it assumes that the old training data is available for computing the regularization term. Chen et al. (2020) proposes the RecAdam optimizer which further approximates the computation of the Fisher information matrix so that there is no need for having access to the old training data. The resulting optimizer imposes a quadratic penalty on the difference of the current values and the old values of the parameters of the network. A similar penalty is also already incorporated in the SFT strategy (L_1 norm of the difference in this case). Recent works in studying multilingual modelling from a continual learning perspective include works of M'hamdi et al. (2022); Yang et al. (2022) which study incrementally adding task data

---

[6]We believe that these ordered trajectories would be more challenging compared to a random trajectory since it is easier for the model to overfit/forget the low resource languages

in *unseen languages* and Berard (2021); Garcia et al. (2021) on extending the language capacity of MT models; both very different from our setup.

Parameter-efficient finetuning: Parameter-efficient finetuning methods such as adapters have shown promise in multi-task continual learning setups (Ke et al., 2021) as well as zero-shot cross-lingual transfer (Pfeiffer et al., 2020b; Ansell et al., 2021). Recent works (Ponti et al., 2022) utilize such methods to decompose task learning into underlying skill learning and allocation.

7 Conclusion

In this paper we introduce and study the problem of **Continual Multilingual Learning (CML)** where a multilingual model is continually updated using new data from a subset of the languages at a time. We observe that unconstrained updates to the model can lead to drastic losses for a subset of the languages, especially those not covered during an update. We propose LAFT-URIEL, a parameter-efficient finetuning stragegy which uses linguistic information to effectively balance overfitting and knowledge sharing across different languages, resulting in 25% increase in the proportion of task languages whose performances improve during an update while achieving 78% relative decrease in average magnitude of losses on the remaining languages.

8 Limitations and Future Work

Since this is one of the first studies on understanding the effects of continued finetuning of multilingual models, the focus of this paper was to lay the groundwork by establishing the experimental setting on a set of representative NLP tasks and languages. The resulting set of languages chosen in our setup for evaluation (en, hi, bn, zh, ta, ja, ar, de, es, fr, th), although diverse, are still relatively higher resource. Extending the analysis to languages which were severely underrepresented (or even absent) during the pretraining of the underlying model may provide interesting insights and would be an important future work to pursue.

Acknowledgements

References

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. 2022. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland. Association for Computational Linguistics.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alexandre Berard. 2021. Continual learning in multilingual nmt via language-specific embeddings. *arXiv preprint arXiv:2110.10478*.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xavier Garcia, Noah Constant, Ankur P Parikh, and Orhan Firat. 2021. Towards continual learning for multilingual machine translation via vocabulary substitution. *arXiv preprint arXiv:2103.06799*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Meryem M'hamdi, Xiang Ren, and Jonathan May. 2022. Cross-lingual lifelong learning. *arXiv preprint arXiv:2205.11152*.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, et al. 2018. Universal dependencies 2.2.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Edoardo M. Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2022. Combining modular skills in multitask learning.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Shai Shalev-Shwartz et al. 2012. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194.

Sebastian Thrun and Tom M Mitchell. 1995. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural*

language processing: system demonstrations, pages 38–45.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Mu Yang, Shaojin Ding, Tianlong Chen, Tong Wang, and Zhangyang Wang. 2022. Towards lifelong learning of multilingual text-to-speech synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8022–8026. IEEE.

A Experimental Settings

All of our experiments are performed on four NVIDIA A100-SXM4-40GB GPUs. Our implementation uses PyTorch (Paszke et al., 2019), the Transformers library (Wolf et al., 2020), AdapterHub (Pfeiffer et al., 2020a) and Composable-SFT (Ansell et al., 2022). We use bert-base-cased and xlm-roberta-base checkpoints to initialize our models. For the seq2seq task, both the encoder and decoder are initialized by the above multilingual checkpoints, as suggested by Rothe et al. (2020). The new cross-attention terms in the decoder are initialized from scratch.

We present the hyperparameters selected for each finetuning strategy in Table 5. We use the AdamW optimizer (Loshchilov and Hutter, 2019; Kingma and Ba, 2015) with weight decay of 1e-5 for each pipeline and perform a search across 3 learning rate values (2, 5 and 8 $\times 10^{-5}$) for each strategy and finetuning stage and select the best performing model using the dev set. For SFT continuation, we experiment with different percentages of the number of trainable parameters in the network and report the best configuration. We also find that freezing the layer norm parameters while sparsely finetuning the entire model (both base model and classifier/decoder) for SFT leads to improvement in behaviour for our task.

We use a batch size of 64 for PAN-X and UPDOS, 128 for MTOP Classification and 96 for MTOP semantic parsing.

	Strategy	PAN-X	UDPOS	MTOP Class.	MTOP NSP
FFT	*Inception*	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 8e-5, num epochs: 40
	Continuation	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 20
SFT	*Inception*	lr: 2e-5, ft epochs: 3, st epochs: 10, base trainable params: 14155776	lr: 2e-5, ft epochs: 3, st epochs: 10, base trainable params: 14155776	lr: 2e-5, ft epochs: 3, st epochs: 10, base trainable params: 14155776	lr: 8e-5, ft epochs: 10, st epochs: 40, base trainable params: 88926720
	Continuation	lr: 2e-5, ft epochs: 3, st epochs: 10, base trainable params: 7077888	lr: 2e-5, ft epochs: 3, st epochs: 10, trainable params: 7077888	lr: 2e-5, ft epochs: 3, st epochs: 10, base trainable params: 7077888	lr: 2e-5, ft epochs: 10, st epochs: 20, base trainable params: 14155776
LAFT	*Inception (shared adapters)*	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 8e-5, num epochs: 40
	Inception (lang-specific adapters)	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 10	lr: 2e-5, num epochs: 20
	Continuation	lr: 2e-5, num epochs: 10, div factor: 10	lr: 2e-5, num epochs: 10, div factor: 10	lr: 2e-5, num epochs: 10, div factor: 10	lr: 5e-5, num epochs: 20, div factor: 50

Table 5: Best hyperparameters for each strategy, stage and task. LR denotes learning rate of the entire model for FFT/SFT and for the adapter layers in LAFT. Div factor denotes the division factor used to calculate the learning rate of the base model relative to those of adapter layers for the LAFT strategies. For SFT, FT epochs denote the number of pilot training epochs used to select the top-k parameters which will be kept trainable in the subsequent sparse finetuning epochs (denoted by ST).

B Dataset Details

We provide the language-codes to language mapping in Table 6. We also present the training data distribution across different languages and the evaluation metric used for the four tasks we consider in our study in Table 7.

To evaluate performance on token level tasks (PAN-X and UPDOS), we use seqeval toolkit (Nakayama, 2018). We obtain these two datasets from the XTREME benchmark (Hu et al., 2020).

For UDPOS, we consider the following POS tags: ['ADJ', 'ADP', 'ADV', 'AUX', 'CCONJ', 'DET',

ISO 639-1 Language Code	Language
en	English
hi	Hindi
bn	Bengali
zh	Chinese
ta	Tamil
ja	Japanese
ar	Arabic
de	German
es	Spanish
fr	French
th	Thai

Table 6: Language-code mapping for all the languages covered in our study.

Dataset	Evaluation Metric	Ratio of Training Examples	Total
PAN-X	Macro-F1	1 (en) : 0.25 (hi) : 0.5 (bn) : 1 (zh) : 0.75 (ta) : 1 (ja) : 1 (ar)	110k
UDPOS	Macro-F1	1 (en) : 0.62 (hi) : 0.33 (ja) : 0.019 (ta) : 0.89 (zh) : 0.28 (ar)	67k
MTOP Class.	Macro-F1	1 (en) : 0.85 (de) : 0.69 (es) : 0.75 (fr) : 0.72 (hi) : 0.68 (th)	74k
MTOP NSP	Exact Match Acc. (EMA)	1 (en) : 0.85 (de) : 0.69 (es) : 0.75 (fr) : 0.72 (hi) : 0.68 (th)	74k

Table 7: Languages covered for the tasks we consider in our setup.

'INTJ', 'NOUN', 'NUM', 'PART', 'PRON', 'PROPN', 'PUNCT', 'SCONJ', 'SYM', 'VERB', 'X'
]
For PAN-X we consider the following NER tags: ['O', 'B-PER', 'I-PER', 'B-ORG', 'I-ORG', 'B-LOC', 'I-LOC']
MTOP Domain classification is a 11-way sentence classification task. The dataset has 117 intents and 78 slots for the semantic parsing task.
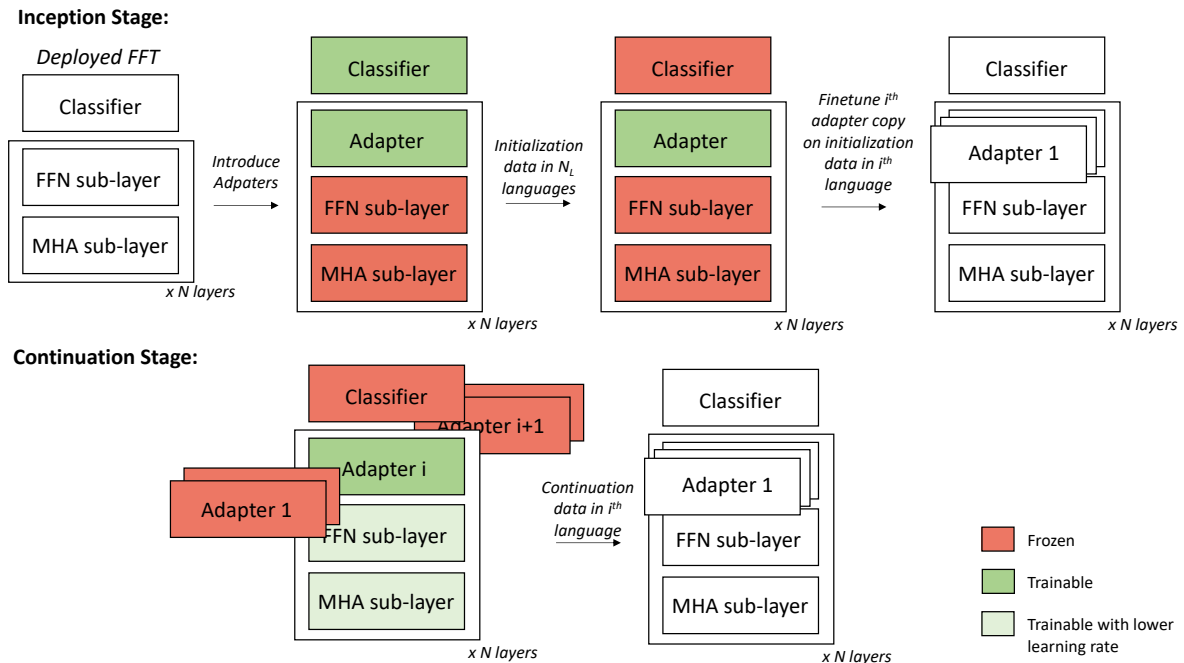
C Diagrammatic View of the LAFT Strategy

Figure 6: Full diagrammatic representation of the LAFT strategy

Dataset	FFT	SFT	LAFT
PAN-X	81.00	82.50	81.65
UDPOS	88.67	88.97	89.15
MTOP Classification	96.71	97.06	97.08
MTOP NSP	59.41	60.61	61.16

Table 8: Macro average performance (avg taken across languages) after the inception stage for each strategy. LAFT and LAFT-URIEL share the same model after inception. Training data for the inception stage is 50% training data for the task in each language.

D Model Performance Comparison after Inception Stage

In Table 8, we report the model performance after inception stage for each strategy (macro average across languages). Variance across different strategies is low. LAFT most often produces the best performing model after inception. Since our metrics are defined on changes in performance, the above fact ensures that our analysis is fair (or slightly favourable) towards the baseline since the first deployed model for the LAFT strategy has less scope of improvement after continuation.

E Heatmaps for XLM-RoBERTa

We compare performance change heatmaps for FFT and LAFT-URIEL across all four tasks in Figure 7. We notice the same improved behaviour as observed in §5.1 using the mBERT initialization. This suggests that gains observed using our proposed strategy are consistent across different model initializations.
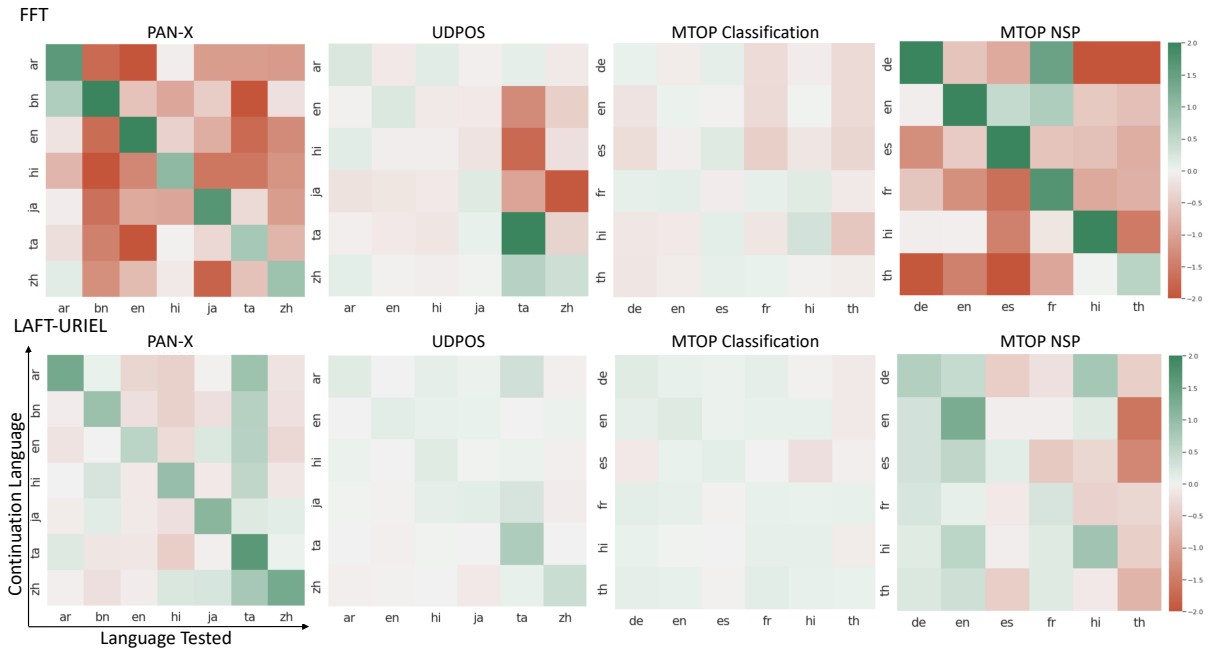

Figure 7: Performance change heatmaps for the FFT (top) and the LAFT-URIEL (bottom) using XLM-RoBERTa initialization on all four tasks. LAFT-URIEL again shows improved behaviour with more green cells and reduced intensity of red cells.

F Size of Adapters for the LAFT Strategy

We perform experiments to study how the size of the adapter layers in the LAFT strategy affects the model's ability to (1) share the gains due to continued finetuning across languages and (2) control for language specific losses. For this, we calculate $NumImprovedLangs$ and $AvgPercentLoss$ on the PAN-X dev set for the $t = 0$ to $t = 1$ transition, by varying the size of the adapter layers. The size of an adapter

layer is controlled by the bottleneck dimension, b_{dim} (throughout our experiments, we use $b_{dim} = 48$ for both LAFT and LAFT-URIEL). We present results in table 9.

b_{dim}	AvgPercentLoss	NumImprovedLangs
48	0.40	3.14
24	0.46	3.14
12	0.41	2.85

Table 9: AvgPercentLoss and NumImprovedLangs for the LAFT strategy by varying b_{dim} on PAN-X dev set

As we reduce b_{dim} for LAFT, we see there is either a hit in positive transfer or increase in magnitude of loss. Since the capacity of the language-specific part of the network is decreased, the model might be relying more on the language-agnostic part during continued finetuning, making it more susceptible to overfit and degrade its multilingual capabilities. We therefore use $b_{dim} = 48$ which correspond to roughly 3.5% of the base model parameters for the tagging task. In comparison, the language-specific update matrices used in the SFT strategy corresponds to roughly 4% of the base model parameters, and results in an AvgPercentLoss of 0.47 and NumImprovedLangs of 2.85 on the same dev set used in this analysis.

G Effect of Continued Finetuning on Closely-related Languages

In our setting, one may assume that continued finetuning on new task-specific data in some language should also benefit the languages which are most closely related to it. To study this, we use the URIEL syntactic distance metric to find the two languages closest to a given language L, denoted as L_1 (closest) and L_2 (second-closest). We expect that continued finetuning in language L would benefit L_1 more than L_2 since L_1 is more closely related to L.

To test this, we calculate the percentage of times change in performance in L_1 is greater than that of L_2 by varying L for a given task. We report the numbers in table 10 for LAFT-URIEL and find that they support our hypothesis that the language closest to L is more likely to experience more favorable performance change than the second closest language.

Task	% of times performance change in $L_1 \geq$ change in L_2
PAN-X	85.70
UDPOS	66.66
MTOP Classification	83.33
MTOP NSP	83.33

Table 10: Percentage of times performance change of closest language to L is greater than second closest language to L after continued finetuning in L for different tasks using LAFT-URIEL

H Quantifying Variance in Language-wise Performance Change after Continued Finetuning

Throughout different experiments in our setting, we observe significant variation on the order of languages which are most improved to most degraeded after continued finetuning on new data for a given language. We attempt to quantify this by analyzing how the behaviour of the model changes when we vary (1) random seed keeping task constant (2) encoder keeping task constant (3) task keeping dataset constant.

To do this, we first construct performance change heatmaps after performing the above-listed variations and then find out the order in which language-wise performance is negatively impacted for a given continuation stage (i.e sorting % change in performance across languages). We compare this order with the original order by computing the edit distance between the two. High edit distance would indicate that order of performance change is very sensitive to the factors we are changing in this analysis. We present the following results (evaluated using FFT):

- **Varying random seed for same task**: Average edit distance after changing seed for the four tasks is as follows: PANX: 4 (N_L=7); UDPOS: 3.66 (N_L=6); MTOP Classification: 4.16 (N_L=6); MTOP NSP: 3.55 (N_L=6)

- **Varying task for same dataset (MTOP)**: Average edit distance between rows of heatmaps obtained after MTOP Classification and MTOP NSP is 3.66 (N_L=6)

- **Varying encoder (mBERT or XLM-RoBERTa) for same task**: Average edit distance after changing encoder for the four tasks is as follows: PANX: 4.57 (N_L=7); UDPOS: 3.33 (N_L=6); MTOP Classification: 4.5 (N_L=6); MTOP NSP: 3.83 (N_L=6)

One can infer from the above numbers that behaviour of the model in our setup is sensitive to changes in random seed, model initialization and task at hand. We therefore stress that it is important to present results in our setup averaged across different seeds (as we have done in our work).

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8: Limitations and future work

☒ A2. Did you discuss any potential risks of your work?
Our work is aimed at improving the robustness of multilingual models in a continual learning setting to ensure that gains in performances are equitably distributed across languages. We don't anticipate any additional potential risks of our work apart from general concerns regarding the advancements of multilingual language technologies.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1 summarizes our main claims

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Yes, section 2 and 4

☑ B1. Did you cite the creators of artifacts you used?
Yes, section 2 and 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All software, checkpoint and datasets we use are publicly available for use in research studies.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
All software, checkpoint and datasets are broadly known and used across the NLP community

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
All datasets we used are broadly used across the NLP community for model finetuning and evaluation

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix B

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix B. We use the standard dev and test sets available for the datasets.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ **Did you run computational experiments?**

Appendix A

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Have made it clear that we are reporting the average across runs, variance reported in results (section 5)

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix A and B

D ☒ **Did you use human annotators (e.g., crowdworkers) or research with human participants?**

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.