Parameter-Efficient Finetuning for Robust Continual Multilingual Learning

Anonymous ACL submission

Abstract

We study the underexplored problem of *Continual Multilingual Learning*, where a multilingual model, already trained on task-specific data from all supported languages, is continually updated using batches of new multilingual training data for the same task. We show that naïvely updating the multilingual model can lead to losses in performance over a subset of languages although the aggregated performance metric shows an improvement. We establish this phenomenon over four tasks belonging to three task families (token-level, sentence-level and seq2seq). We then build upon recent advances in parameter-efficient finetuning to develop novel finetuning strategies that allow us to jointly minimize language-specific forgetting while encouraging positive cross-lingual transfer observed in this setup. Our proposed pipeline, LAFT-URIEL, improves the spread of gains over the supported languages while reducing the magnitude of language-specific losses incurred.

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) is a framework to sequentially train a given model on one or more number of tasks without forgetting the knowledge obtained from prior training. Online learning (Shalev-Shwartz et al., 2012) is a special case of continual learning where the model is incrementally trained on a stream of new training data without having access to the old training data.

Multilingual NLP models are gaining importance in a variety of NLP tasks., e.g., NER tagging (Adelani et al., 2021), Question Answering (Clark et al., 2020), Relation Extraction (Walker et al., 2006; Bhartiya et al., 2022). Such models are increasingly being powered by a single deep neural network trained on data from all supported languages (Devlin et al., 2019; Conneau et al., 2020; Xue et al., 2021). Just like other deep NLP models, multilingual models also need to be regularly updated, thereby making them suitable for application of Continual Learning strategies.

However, Continual Learning of Multilingual models involve additional challenges due to the involvement of multiple languages. For example, during any given update, additional training data from only a small subset of the languages involved may be available. This is often true for multilingual models deployed in production as the new data obtained from the updated usage logs might be skewed towards only a subset of languages. Updating parameters in an unconstrained manner given this limited training data may adversely affect performance on languages not represented in the current training data. 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.

We consider the setting where a single-task multilingual model (trained on all supported languages) needs to be continually updated with additional training data arriving in stages, while keeping the model capacity fixed and without relying on any data from previous training steps. The work most closest to ours is a recent study by M'hamdi et al. (2022). However, they focus on training a multilingual model from *scratch* with data from one language arriving at a time. Moreover, unlike us, their best performing strategy relies on adding new parameters to the network with each new data batch.

Given a shared network for all languages of interest, the goal of updating the model 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 non-trivial and the naïve baseline finetuning strategy incurs significant losses due to forgetting across a subset of languages. We study this phenomenon over four sets of tasks and find the same non-ideal behaviour across all experiments for the baseline strategy.

With the intuition that constraining the number of trainable parameters in the network would help control for losses due to forgetting, we devise finetuning strategies using recent advances in parameter-efficient finetuning methods, namely Adapters (Houlsby et al., 2019) and Composable Sparse-finetuning (Ansell et al., 2022). We find that such methods allow us to control for losses incurred while also increasing the spread of performance gains across languages. We further improve upon the stronger pipelines and devise a novel finetuning strategy using URIEL language similarity metrics (Littell et al., 2017) which optimally balances the trade-off between encouraging positive cross-lingual transfer and discouraging language-specific forgetting in our setting, given the composition of the new data. The key contributions of our paper are:

- We systematically study the continual learning framework from a multilingual perspective where the sequential batches of training data differ in language-wise distribution.
- We show how the naïve strategy (keeping all parameters trainable), leads to language-specific catastrophic forgetting even though the new data is for the same task as before. We construct metrics to quantify this behaviour.
- We propose a LAFT-URIEL, novel finetuning strategy, which uses language-specific adapters and utilizes syntactic similarity between languages to maximize cross-lingual transfer and minimize forgetting based on the new training data distribution.

2 Problem Setup

Our goal is to simulate a setting where an already trained multilingual model for a given NLU task (*deployed model*) is further finetuned on new finetuning data for the same task to give us the *updated model*. In the real-world setting, the deployed model is often trained on all available training data across all supported languages (say N_L number of languages for a given task) to ensure best possible performance. However, one may have no control over how the new finetuning data is distributed across languages. In our setting, we consider the worst-case scenario where the new finetuning data is only present in one of the N_L languages, thus representing the maximum skew.

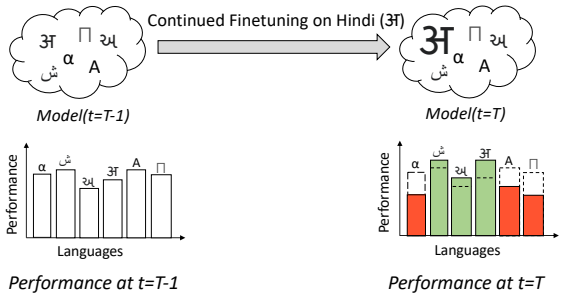

Figure 1: An example of CML at the Tth continuation stage. Model at $t = T - 1$ is further finetuned on new data (for the same task) in Hindi. We find that naïvely finetuning all the parameters of the model leads to degradation in performance on a subset of languages not represented in this finetuning stage.

We divide the entire setup into two stages.

Stage 1: *Inception stage* where we setup the first deployed model by finetuning 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.

Stage 2: *Continuation stage* where we further finetune the deployed model on the new finetuning data to give us the updated model (Figure 1).

The dev set is kept the same across both the inception and the continuation stages and consists of examples from all languages for the given task.

There can be multiple continuation stages sequentially applied one after another i.e. *updated model* from the previous step becomes the *deployed model* for the current step. This enforces a constraint that the architecture of the *deployed model* and the *updated model* have to be exactly the same. Formally, we define the process using the following notations: $\text{model}(\text{t} = 0^-)$: untrained model, $\text{model}(\text{t} = 0)$: first deployed model and $\text{model}(\text{t} = \text{T})$: model after T continuation stages. Hence the inception stage can be written as:

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

where $\xrightarrow{\{l_{i_1}, l_{i_2}, \ldots, l_{i_k}\}}$ denotes finetuning using data

from languages $\{l_{i_1}, l_{i_2}, ..., l_{i_k}\}$. The Tth continuation stage can be written as:

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

where $i \in \{1, 2, ..., N_L\}$. For comparing different finetuning strategies, we focus on the $t = 0$ and $t = 1$ 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 $t = 0$ to the updated model at $t = 1$. We consider all such cases. We divide the complete training data for a given task into two equal parts, independently across all languages[1] and these two parts are used for the inception and continuation stages respectively. The development and testing sets cover all languages and are kept the same across both stages.

On a fixed test set for the given task, we expect the language-wise performance of the deployed model and the updated model to differ due to multiple reasons: 1) additional task-learning due to the new data 2) catastrophic forgetting of language-specific knowledge 3) positive/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 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 analysis 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.

We conduct N_L independent continuation stages, one for each language, to transition from the $t = 0$ model to the $t = 1$ model. 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.

[1]In a real world setting, there is no guarantee that the new data is disjointed from the old data after language normalization. Hence, training data overlap (across finetuning stages) is not an important consideration in our setting.

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 considered in our setting

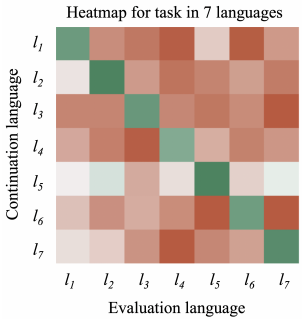

Figure 2: Example heatmap for a task in 7 languages. Each cell measures % change in performance on the language denoted by the column after continued finetuning on the language denoted by the row. +ve changes are highlighted in green, -ve changes in red.

3 Naïve baseline and Metrics

For the naïve baseline finetuning strategy, we consider the case where the entire model is kept trainable during the continuation stage. We call this the *Full Finetuning* (FFT) baseline. The goal is to understand the effects on language-wise performance of the model in our setup using different strategies.

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 used during the continuation stage and losses across some subset of the remaining languages.

To compare different finetuning strategies, we measure the % change in language wise performance after the continuation stage[2]. We construct heatmaps to visualize the performance changes observed after a continuation stage. Figure 2 shows an example of a performance change heatmap. Here each row corresponds to a continuation stage between the $t = T - 1$ and the $t = T$ models where

[2]We use % change as opposed to magnitude change since -ve 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.

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.

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.

Avg % loss: *Average magnitude of percentage loss after continuation*. Equivalent to taking the average magnitude of all the red cells in a given row in the heatmap. For an ideal model, this should be 0.

Number of (#) languages with positive (+ve) delta: *Average number of languages with a +ve change in performance after continuation*. Equivalent to the number of green cells in each row in the heatmap. For an ideal model, this should be the # languages in a given task.

Since each row is a valid continuation stage, we report the average of the above metrics across all rows in the heatmap.

Additional constraints: After a continuation stage, (1) the sum of gains should be higher than the magnitude of the sum of losses (**sum ratio**) and (2) the magnitude of maximum gain should be higher than the magnitude of the maximum loss in performance (**max ratio**). We compute an average of $sum(gains)/abs(sum(losses))$ and $max(gains)/abs(max(losses))$ 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, in the multilingual space, these methods have been 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 subsections will give an overview of the methods we intend to use to build improved finetuning strategies for the task.

4.1 Background

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; Pfeiffer et al., 2020a): Adapters are trainable modules that are inserted in the layers of a transformer network. During finetuning, usually only the adapter modules are kept trainable. This allows one to store trained parameters for multiple tasks using fewer resources (usually 5-10% of the base model).

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 Startegies

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. This gives us a better performing deployed model compared to when the classifier/decoder is kept sparsely trainable too.

During the continuation stage, we sparsely update the entire model (base model and the classifier/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/decoder on inception stage data for all languages for the task and then create N_L copies

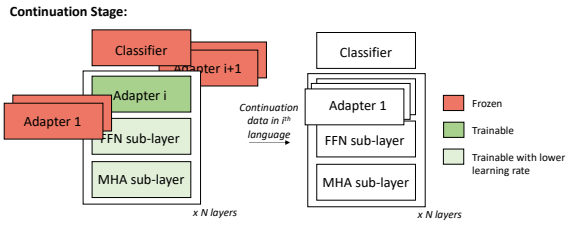

Figure 3: Continuation stage for the LAFT strategy. Here, the base model (shared across different languages) is finetuned at a lower learning rate in order to balance +ve transfer and losses due to forgetting.

of the trained adapters (one for each language). The ith copy is further finetuned (with the base model frozen) on inception stage data for the l_i language. This gives N_L language specific adapters and shared base model and classifier/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 modules). Since from the inception stage, the model has associated language-specific learning with the adapters and language-agnostic learning with the base model, we hope that 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 and SFT. Continuation stage for the LAFT strategy is represented in Figure 3.

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 new data would lead to shared gains in performance.

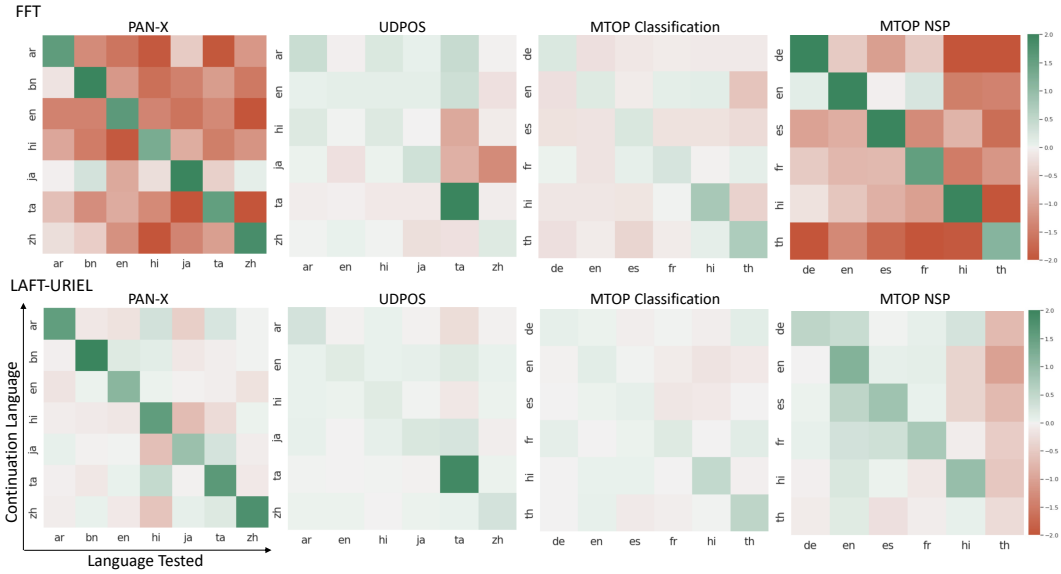

Figure 4: Performance change heatmaps for the FFT (top) and the LAFT-URIEL (bottom) on all four tasks. LAFT-URIEL shows improved behaviour with more green cells and reduced intensity of red cells.

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 for different continuation stages for MTOP semantic parsing using LAFT-URIEL strategy.

We use URIEL syntactic distance as a measure 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 semantic parsing task in Table 2. This modification gives us the LAFT-URIEL strategy.

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). Results in the main paper are using mBERT initialized models. We ask the following research questions:

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 present heatmaps for both the FFT (top row) and the LAFT-URIEL (bottom row) strategies in Figure 4. For the FFT baseline, the diagonal entries have the highest positive magnitude 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[3]. In the subsequent

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

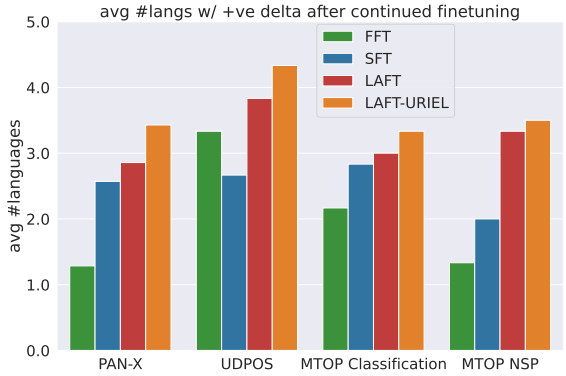

Figure 5: *# languages with +ve delta* across four tasks. Higher the better. LAFT and LAFT-URIEL, improve spread of gains compared to both FFT and SFT.

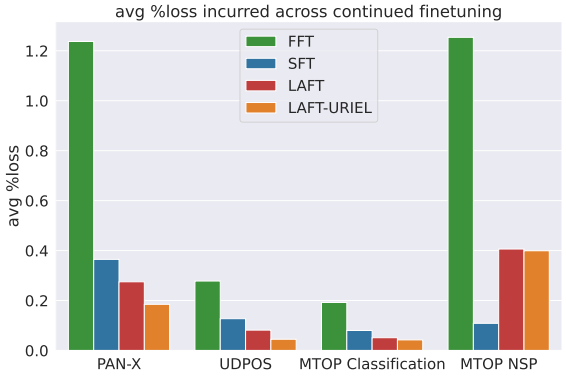

Figure 6: *Avg % loss* across four tasks. Lower the better. LAFT-URIEL shows significant improvement over FFT and SFT on 3 tasks. SFT doesn't meet the minimum constraints (Table 3) for MTOP NSP.

subsections, we will quantify these observations using the metrics proposed in section 3.

5.2 Measuring spread of gains

We plot *# languages with +ve delta* for all the four strategies on all four tasks in Figure 5. We see considerable gains over the naïve FFT baseline using SFT across all tasks but UDPOS, indicating that SFT is indeed a stronger baseline for our setup.

LAFT improves upon SFT and FFT, with LAFT-URIEL improving the gains even further. This indicates that using URIEL syntactic distance to dynamically compute the learning rate of the base model given the composition of the continuation data gives much better results than assigning the same learning rate irrespective of the language of the continuation data. Compared to FFT, our LAFT-URIEL improves the spread of gains by roughly $1.3\times$ to $2.3\times$ depending on the task, making it much more favourable for our setup.

5.3 Comparing losses incurred

We plot *Avg % loss* for all the four strategies on all four tasks in Figure 6. 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 *max ratio* and *sum ratio* 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.

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) 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[4]. Given a trajectory of the form $\xrightarrow{\{l_{i_1}\}} \xrightarrow{\{l_{i_2}\}} \ldots \xrightarrow{\{l_{i_{N_L}}\}}$, we define the worst transition as the continuation stage:

$\text{model}(\text{t} = \text{l} - 1) \xrightarrow{\{l_{i_l}\}} \text{model}(\text{t} = \text{l})$ where the *avg % loss* magnitude is highest across trajectory. We report the metrics on the worst transition in the Table 4. We observe that our proposed strategy consistently reports a bounded worst case value of avg % loss and is $\leq 1\%$ 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.

[4]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

Strategy	PAN-X		UDPOS		MTOP Classification		MTOP NSP	
	max ratio	sum ratio	max ratio	sum ratio	max ratio	sum ratio	max ratio	sum ratio
FFT	1.23	0.50	6.85	5.10	1.19	0.82	1.16	0.48
SFT	1.26	0.76	4.95	2.58	2.73	2.62	0.54	0.36
LAFT	4.62	2.55	11.72	13.25	9.65	17.64	1.26	1.70
LAFT-URIEL	6.30	4.10	∞	∞	12.04	20.82	1.18	1.48

Table 3: *Sum ratio* and *Max ratio* (§3) for all four strategies across all four tasks. Higher the better. Values highlighted in red are unfavorable for an ideal model, blue corresponds to the best results. LAFT-URIEL shows close to ideal behaviour on all tasks.

Strategy	PAN-X		UDPOS		MTOP Classification		MTOP NSP	
	% loss	# +ve langs	% loss	# +ve langs	% loss	# +ve langs	% loss	# +ve langs
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 *Avg % loss* (denoted by % loss) and *# of languages with +ve delta* (denoted by # +ve langs) calculated on the "worst transition" for FFT and LAFT-URIEL after multiple continuation stages. h2l and l2h represent two trajectories we consider (§5.4). LAFT-URIEL is able to ensure that the losses are constrained even after multiple continuation stages.

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 in unseen languages and Berard (2021); Garcia et al. (2021) on extending the language capacity of MT models.

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 and Limitations

In this paper we study continual learning from a multilingual perspective. We design our experiment framework closely following how multilingual models are deployed and updated in the real-world. Through our experiments, we observe that the language-wise performance of a given model is non-uniformly perturbed as continued finetuning causes overfitting on the language-regularities of the new data.

We propose improved finetuning strategies using parameter-efficient finetuning methods which help in significantly reducing the adverse effects of the above phenomenon. Our proposed strategy, LAFT-URIEL, uses linguistic information to optimally balance overfitting and knowledge sharing across different languages. The behaviour of our proposed strategy is much closer to the ideal behaviour one would expect in our problem setup.

We observe significant variance in the behaviour of the model with changes in initialization, random seed and tasks (appendix F). We report the averaged metrics throughout the paper to deal with this issue. Our strategies do not improve upon this variance and we wish to address this in future works.

References

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, et al. 2021. Masakhaner: Named entity recognition for african languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131.

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*.

Abhyuday Bhartiya, Kartikeya Badola, and Mausam . 2022. DiS-ReX: A multilingual dataset for distantly supervised relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 849–863, Dublin, Ireland. Association for Computational Linguistics.

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.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

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.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus.

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 present the training data distribution across different languages and the evaluation metric used for the four tasks we consider in our study in Table 6.

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', 'INTJ', 'NOUN', 'NUM', 'PART', 'PRON', 'PROPN',`

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

`'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

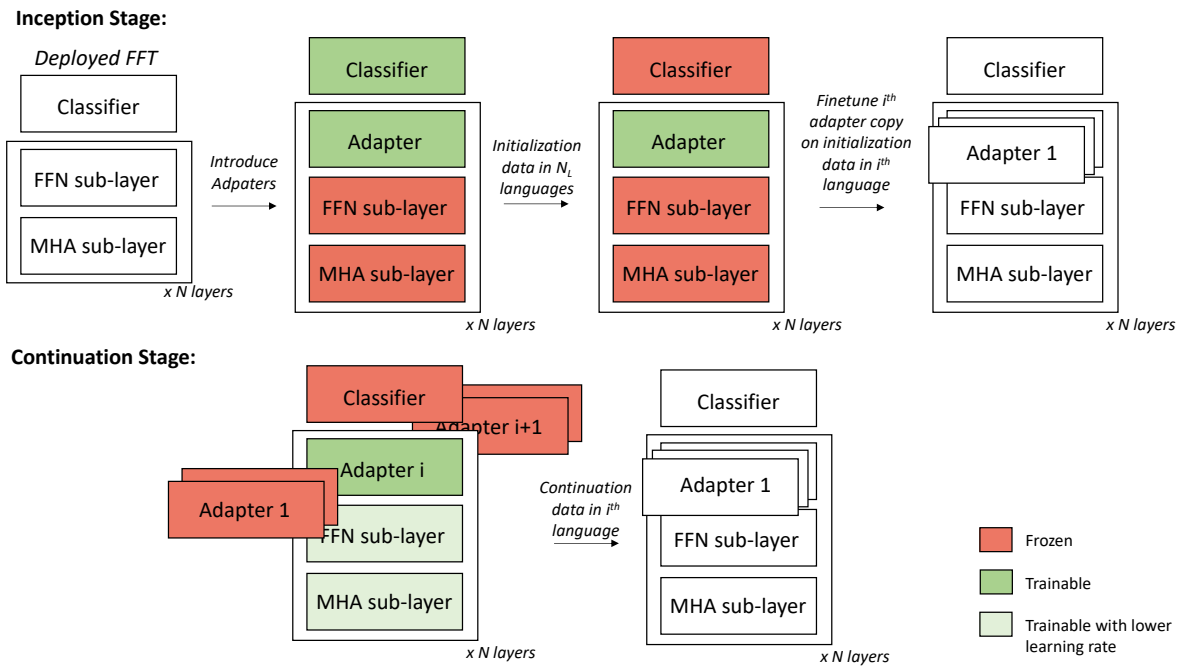

Figure 7: Full diagrammatic representation of the LAFT strategy

D Model performance after inception

In Table 7, 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 8. 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.

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 7: 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.

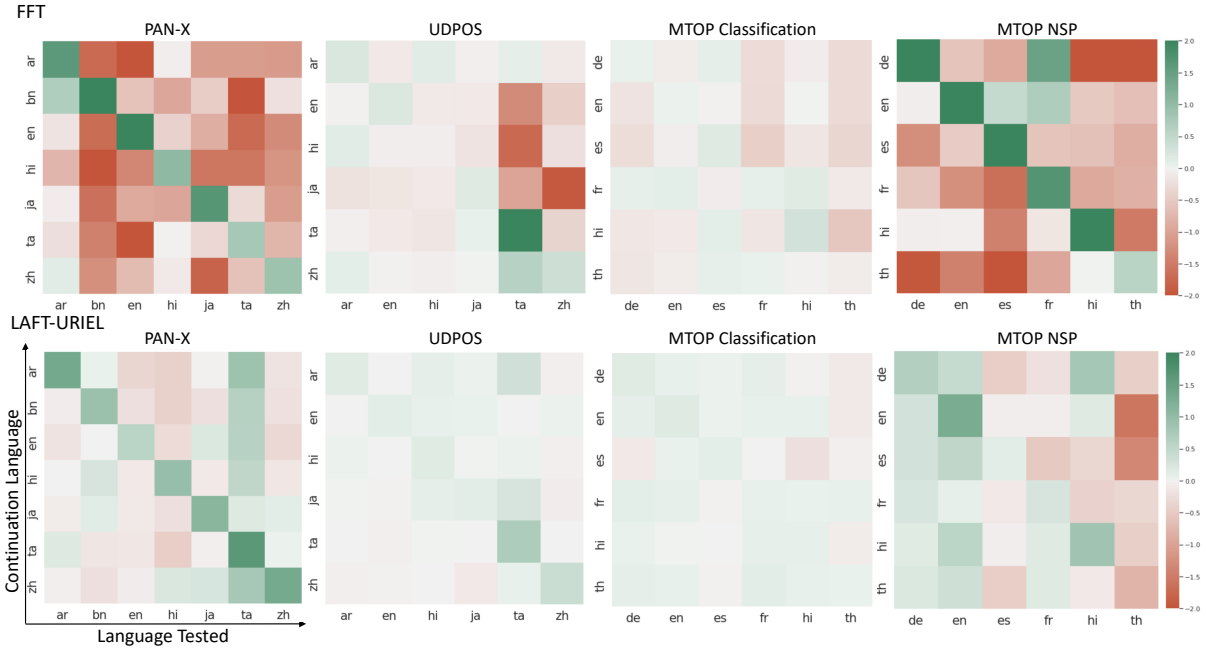

Figure 8: Performance change heatmaps for the FFT (top) and the LAFT-URIEL (bottom) using XLM-RoBERTa initialization on all four tasks. LAFT-URIEL again shows improved behaviour with more green cells and reduced intensity of red cells.

F Quantifying high variance in language-specific losses

Throughout different experiments in our setting, we observe significant variation in most-affected languages after continued finetuning on new data for a given language. We attempt to quantify this high variance 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 chang-

ing 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 presented in our work).