Combatting Language Forgetting in Low-Resourced Settings

Emmanuel Rassou EMMANUEL_RASSOU@COLLEGE.HARVARD.EDU

Abstract

Neural machine translation becomes a continual learning challenge as language evolves over time. While Transformer-based models excel at capturing linguistic patterns from large corpora, they require continual updates to adapt to new language use without losing previously acquired knowledge. In this work, we introduce Latent Replay Buffers to the NLP domain for the first time by implementing and fine-tuning our *Latent Replay Transformer*. We conduct initial experiments on low-resource languages and compact models such as Small-100, which are well-suited for deployment in memory- and data-constrained environments. Our findings reveal an intriguing trade-off in the selection of latent activations to store for effective replay. We release our code to support both the Continual Learning and NLP for Low-Resourced Languages communities.

Keywords: Continual Learning, Transformers, Low-Resourced Languages

1. Introduction

Local languages shift their domain throughout time as frequent neologisms and colloquialisms constantly evolve everyday speech. This is especially true for language-diverse regions where pidgins form dynamically from this multilingual setting. This shift slowly leaks into social media posts, but the pipeline into the web-scraped datasets ultimately remains too slow to be reliable. In particular, the delay in capturing these changes in the datasets commonly used to train NLP models can lead to models that are outdated or less effective in understanding current language use, as seen in studies of shifts in English such as Loureiro et al. (2022).

This natural phenomenon motivates online updates of the model during their deployment at the edge, such as continuously scraping the web for more training examples or storing real-time data in speech translation or transcription applications. This unconventional continuity of training is especially important for low-resourced areas where most of the data available is unstructured and sparse. However, this new setting also opens up a challenge in the form of catastrophic forgetting of language understanding. It is well-known that neural networks trained on multilingual translation have language-agnostic parameters used to represent the concepts. These foundational representation layers are at risk of being overwritten as new training examples update the weights from back propagation, as has been previously studied for Neural Networks Hadsell et al. (2020).

We are motivated by this problem of language forgetting, and the most effective solution so far within the continual learning space is the use of Replay Buffers. While replay of past experiences for more stable learning dynamics was originally created in the context of reinforcement learning Liu and Zou (2017), we investigate its implications in the context of attention-based transformers. Transformers have taken over many application areas with their innovative sparse selectivity mechanism to deal with long contexts of text ever since their conception in 2017 Vaswani et al. (2023). Within the realm of Neural Machine Translation,

Encoder-Decoder structures have been used to learn a common representation of multiple languages.

While stacks of transformer blocks in the form of pipelined encoders and decoders have been required to learn the intricacies of linguistic relationships, the value in learning small but powerful models that can help bridge the gap between high- and low-resourced language translation. We explore this novel intersection between continual learning and low-resourced machine translation.

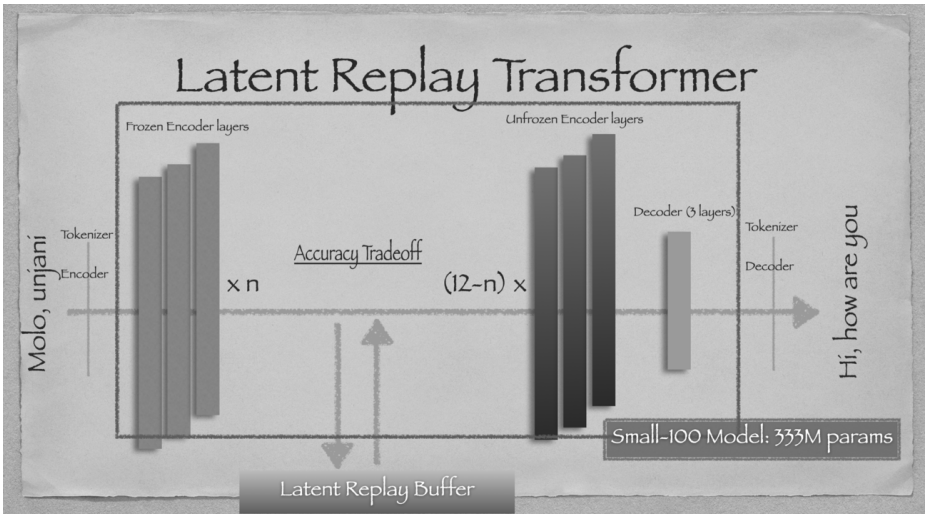


Figure 1: Core end-to-end design of our new model architecture: the Latent Replay Transformer

2. Related Work

The first key area in continual learning is work on replay buffers, where training examples from previous tasks are periodically retrained. Gradient Episodic Memory(GEM) splits up the replay memory among the tasks that have been exposed. Lopez-Paz and Ranzato (2022). They rely on constrained optimization to ensure the loss of previous tasks does not reduce. This explicit prevention of catastrophic forgetting, however, does complicate the training procedure computationally.

There are also more biologically inspired works that center around memory in the brain. Elastic Weight Consolidation (EWC) considers a regularization term added to the loss function to ensure important weights from the previous task are not changed too much Kirkpatrick et al. (2017). Figure 2 gives a good visualization of how gradient descent can be tweaked to optimize for two tasks learned at different times. CRUMB is another neuro-inspired method that consolidates old tasks by reconstructing old feature maps with generic memory blocks stored Talbot et al. (2025). This method optimizes memory by not storing the raw feature maps.

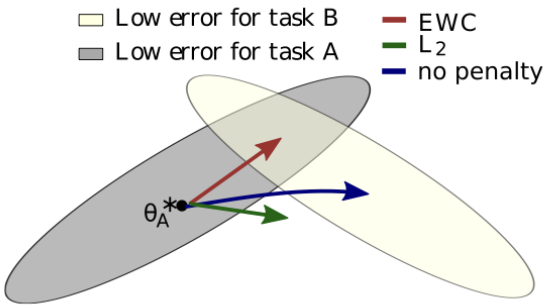
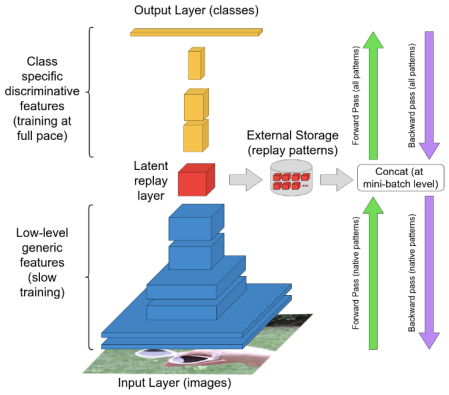
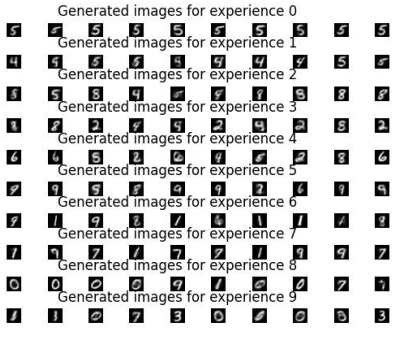


Figure 2: Diagram taken from Kirkpatrick et al. (2017). "Elastic weight consolidation (EWC) ensures task A is remembered whilst training on task B. Training trajectories are illustrated in a schematic parameter space, with parameter regions leading to good performance on task A (gray) and on task B (cream). After learning the first task, the parameters are at θ^*_A. If we take gradient steps according to task B alone (blue arrow), we will minimize the loss of task B but destroy what we have learnt for task A. On the other hand, if we constrain each weight with the same coefficient (green arrow) the restriction imposed is too severe and we can only remember task A at the expense of not learning task B. EWC, conversely, finds a solution for task B without incurring a significant loss on task A (red arrow) by explicitly computing how important weights are for task A."



(*a*) Diagram from Pellegrini et al. (2020), first introduced in computer vision with CNN.



(*b*) Synthetic replay data generated with Generative Latent Replay using code from Armstrong et al. (2022).

Figure 3: Related continual learning techniques.

Latent Replay Buffer is another promising technique for reducing memory Pellegrini et al. (2020). Storing the raw activations becomes a concern for natural language that has high-dimensional vector embeddings for every token. For both convolutional and attention-based models, the earlier layers are known as representation layers. These are layers that learn high-level, coarse features that can provide useful computation for most tasks of relevance. The later layers provide more fine-grained control and have their weights differ across tasks. As shown in Figure 3(*a*), the latent replay buffer only stores the intermediate activations from some hidden layer in the network and thus also only has to be retrained on the layers from that point. One challenge encountered by this approach is that when the latent vectors are retrieved from the replay buffer, they may be stale since the inner representation of activations has changed during the time elapsed. To mitigate this latent replay drift, the authors propose to reduce the learning rate of the earlier layers and, in extreme cases, freeze them completely if they are already trained sufficiently. The authors have generously open-sourced their implementation in Pytorch on GitHub: https://github.com/vlomonaco/ar1-pytorch. AR1* and AR1* free is the name of their latent replay buffer implementation with and without EWC, respectively. Experiments showed that EWC is not orthogonal to latent replay in continual learning ability, and so AR1* free performs similarly to AR1*.

Generative Latent Replay is another exciting continual learning technique Armstrong et al. (2022). This technique departs from storing traces of examples of previous tasks. Instead, a diffusion model is trained in parallel to generate synthetic data from the learned distribution of examples of previous tasks. The paper implements previous techniques as baselines in their open-source repo: `https://github.com/iacobo/generative-latent-replay`. We ran the method locally on the MNIST dataset to achieve good generative results, as displayed in Figure 3(*b*). We leave the application of this method to the multilingual setting as future work.

3. Methods

3.1. Latent Replay Transformer

Our core contribution is adapting the latent replay buffer from Pellegrini et al. (2020) to work in the NLP domain and for Transformers as well. We base our discussion around the Small Language Model(SLM), Small-100 model Mohammadshahi et al. (2022), which is a distilled version of the original M2M-100 model that was trained to learn simultaneous representations of 100 different languages Fan et al. (2020). This encoder-decoder contains 12 Encoders, 3 Decoders and 333 Million parameters across all layers. The distilled model performs comparably on low-resourced languages but slightly underperforms on medium- and high-resourced languages. The base model choice was specifically tailored for hardware-constrained environments such as those presented by Tiny Machine Learning applications Lin et al. (2023).

Our Latent Replay Transformer as shown in Figure 1, chooses one of the 12 Encoders to be the latent layer as the exit point for storing the future replayed activations. While a Decoder could theoretically be used, the bulk of the representation learning is given by the Encoder half of the model. The choice of the latent layer for the replay buffer is an important design decision. In the original paper Pellegrini et al. (2020), they apply this technique to

a Convolutional Neural Network(CNN) which has a downsizing effect on the dimension of the representation in order to learn more high-level features in the image, given lower-level features from the previous layer. This motivates storing a more compact representation of the original training example in the form of the latent activations of some intermediate layer. The intuition of feature representation still applies in the case of Transformers, however, because an identical transformer block is repeated over and over again, the space savings do not carry over.

Even though Transformers have this drawback, there are still benefits to include a latent replay buffer over simply a replay buffer. (i) There is still a computational saving, since the rehearsed examples only have to do a forward pass starting at the latent layer. (ii) Since the layers before the latent layer have their learning slowed down to prevent stale activations from being stored, the latent replay buffer preserves the core representation of the concepts from changing too much and compromising the general understanding of language beyond the current translation task. This training slowdown could be applied in the vanilla replay buffer setup, however, the choice of where to make the split is more natural and apparent in the case of a latent replay buffer.

3.2. Avalanche Implementation

To aid our Pytorch implementation we leverage the open-source library *Avalanche* created by the continual learning community Continual.AI Carta et al. (2023). While most of the code has been applied in the context of computer vision with images as input, we will adapt the demo code from this library to transformers with natural language. Because most continual learning techniques rely on adjustments made to the training loop, Avalanche establishes *strategies* as a useful primitive. Many common continual learning techniques have already been implemented in the library, such as Replay, EWC, and GEM. Latent Replay has to be reimplemented, and we borrow the custom strategy implemented in the Generative Latent Replay codebase Armstrong et al. (2022).

Although the adaptation seems straightforward, we had to implement additional pieces to the training pipeline to align it with the NLP domain. We establish a collate function for our training batches. Each source and target sentence from the dataset must be tokenized with the model's own tokenizer. In the case of Small-100, its tokenizer is dependent on the language code for that particular training example. Additionally, since each sentence is not the same length within the same batch, one must use padding tokens and an encoder and decoder attention mask for the tokenized source and target sentence, respectively. This is a common practice when training Transformers to ensure that the prediction of padding tokens is not part of the objective. We employ CrossEntropy loss as our Avalanche criterion.

3.3. Practical Considerations

Given our tight resource constraints and lack of funding, we did not have the GPU compute power to pretain different strategies for comparison. We instead downloaded all the weights except the last layer from Hugging Face Mohammadshahi et al. (2022). By reinitializing the last layer only, we resorted to this hybrid fine-tuning solution. This also saved evaluation time and allowed us to employ a smaller dataset such as Goyal et al. (2021). This also allows us to measure significant changes in the training dynamics over time. See Figure 6 for our training

of five different translation tasks without any replay buffer. The different streams for the translation tasks will be described in the Evaluation section. Our code and model checkpoints are available here https://github.com/emmanuel2406/LatentReplayTransformer.git.

4. Evaluation

4.1. Dataset

Since we do not require a huge training dataset, we use the Flores-200 dataset, which is an extension of the popular machine translation dataset Flores-101 Goyal et al. (2021). It contains 2,000 parallel sentences from 200 different languages, including low-resourced languages. Half will be used for fine-tuning, with the remaining half for evaluation.

4.2. Task Description

With our fine-tuning setup we define a task to be the translation of all 1000 sentences of a foreign language to English. We measure the BLEU (Bilingual Evaluation Understudy) score during evaluation Papineni et al. (2002). The scores measures the proportion of matching n-grams between the attempted translation and the reference target sentence. We use the score implementation from the NLTK library Bird et al. (2009). For the purposes of these experiments, we just use greedy decoding where the token with the maximum logit is selected to optimize for exact matches, however, for deployment in conversational settings, a temperature-based decoding should ideally be used, and as such, a better evaluation metric can be adopted to enable matches based on semantic meaning such as COMET Rei et al. (2022).

Within one experiment we give t tasks in series. After every task finishes we evaluate the current model checkpoint on the languages of all tasks up to that point. I.e. task i will be reevaluated at task iteration $i, i+1, .., t$. This maps out a trace of the strategies ability to retain performance even after weights are updated from other tasks. In settings of multilingual translation with continuous tuning of the parameters, we hypothesize our Latent Replay Transformer can limit degradation of language forgetting in these settings.

4.3. Replay Memory Size

We first do a preliminary experiment on the buffer size of our latent replay buffer size (R). In the computer vision domain, Pellegrini et al. (2020) found a buffer size of 1500 was well-adjusted to their training setup. Our setup uses a much smaller dataset, and as such should expect small buffer sizes to be a better fit. We adopt the same buffer update rule to randomly replace h latent activations in replay memory with new activations from the latest iteration. $h = \frac{R}{i}$ for iteration i. By dividing by i, we gradually decay the number of examples injected in the buffer to maintain a balanced contribution to the buffer from each training batch.

We set $t = 2$, with first French $\implies$ English, and then Italian $\implies$ English. We consider replay buffer sizes of $R \in \{50, 100, 250, 500\}$ and measure its training dynamics across the two tasks.

As seen in Figure 4, training is a lot more stable for $R = 50, 100$, whereas for $R = 250, 500$, there is a phase transition where the loss spikes during the first few iterations. This shows

an imbalance between new training examples and rehearsed latent activations being injected when the buffer size is too large. In order to retain our goal of combatting language forgetting we still set R to be as large as possible and so $100 < R < 250$ should suffice.

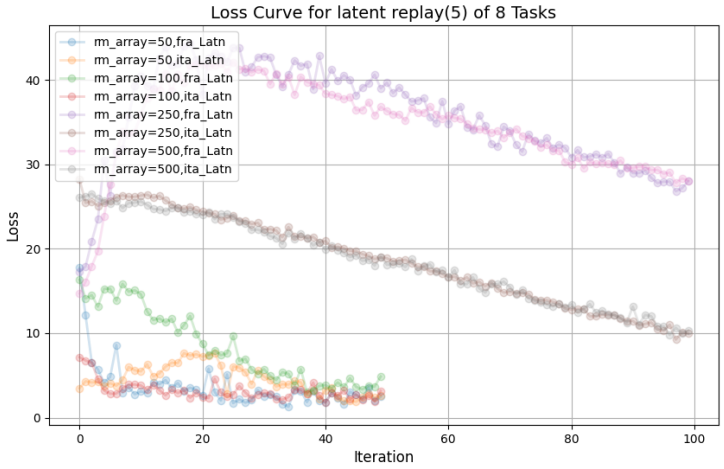


Figure 4: Ablation study on the optimal Replay Buffer Size. Insights show that the replay buffer size R should be tuned based on the dataset used.

In online settings, we lack an offline dataset to tune the replay buffer size. While our experimental results do not directly translate to real-world deployments, where language usage is uncontrolled, they highlight a key insight: *bigger is not always better*. This principle underscores the subtle trade-off between recalling past examples and adapting to new ones, which becomes critical in dynamic, multilingual environments.

4.4. Latent Layer Number

We do an ablation study on the effect of the latent activations we store. Let $1 \leq l \leq 12$ denote the latent layer number. Recall that l represents which encoder's outputs are stored in the latent replay buffer. The training configuration is shown in Table 1 in the Appendix. For $R = 200$, we test and compare evaluation scores for $l \in \{1, 3, 7, 11\}$. We consider 5 different tasks training the translation from the low-resourced languages of Afrikaans, Xhosa, Zulu, Tswana and Swahili in this order. We trace the BLEU score of the first language Afrikaans across the different tasks as shown in Figure 5. An interesting tradeoff arises where the highest BLEU score occurs not at the two extremes but rather in the middle at latent layers 3 and 7.

When the latent activations are too late as is the case with $l = 11$, the weights are more prone to staleness, as there are more language-specific layers before it change their parameters. If the latent activations are too early as is the case with $l = 1$, we see that the earlier representation layers have their parameters compromised, which explains the unstable training for the first few tasks. Additionally, latent layers $l = 7$ and $l = 11$ experience less forgetting compared to the early latent layers, particularly for the drop between task 4 and 5. Thus, storing latent activations from $l = 7$ achieves a good balance of both preventing

forgetting and activation staleness. We also showcase the evolution of scores for Xhosa and Zulu in Figure 7.

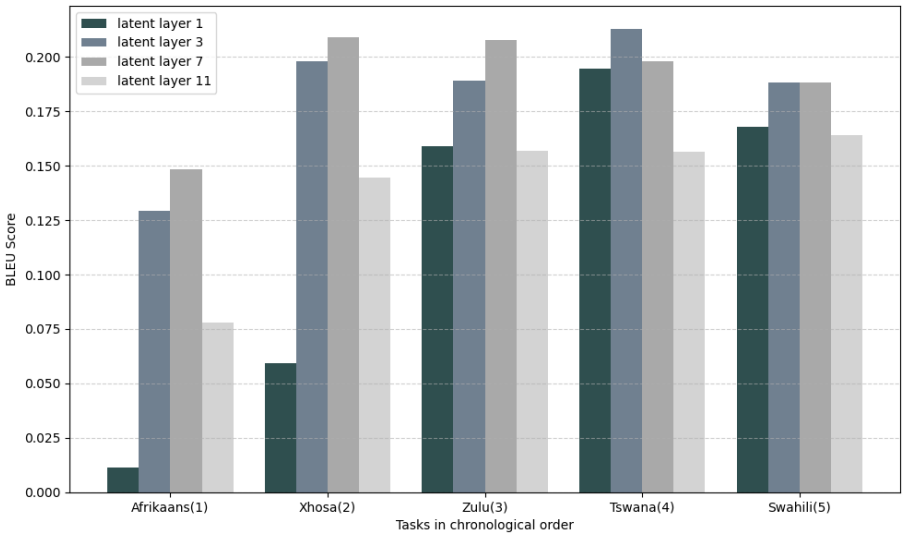


Figure 5: BLEU score of Afrikaans to English across 5 different tasks where different layer activations are stored.

4.5. Training Instability Challenge

For $l = 1$, there were failed attempts where the training was too unstable, which led to a divergent loss. This required us to clip the gradient norm to have max norm 1.0. Although this challenge was partly due to the fine-tuning setup and choice of learning rate, such a challenge is consistent with the lower scores found after training $l = 1$.

5. Future Directions

A key limitation of this work is the constrained fine-tuning environment. In future work, we aim to explore the pre-training stage, where the effects of forgetting may be more pronounced and revealing. Despite this limitation, our introduction of a latent replay buffer for continual learning demonstrates practical value for deployed systems. While our current analysis focused on performance degradation within individual languages, a cross-lingual comparison of forgetting susceptibility would provide deeper insights into language-specific vulnerabilities. Finally, deploying our new architecture on edge devices presents a promising avenue, particularly due to the computational savings of and compactness of the Small Language Model used.

6. Conclusion

As language evolves, neural machine translation systems must adapt without sacrificing prior knowledge. In this work, we introduce Latent Replay Transformers as a novel architecture to

mitigate catastrophic forgetting in natural language settings, particularly for low-resource language translation. We demonstrate that storing and replaying latent representations can strike a meaningful balance between learning new information and retaining prior knowledge. Through targeted experiments on compact models like Small-100, we show that this approach holds promise for real-world deployment, especially in memory-constrained or edge-device environments. Our findings contribute to a growing effort to bridge the gap between continual learning and multilingual NLP, and we hope they serve as a foundation for further exploration in cross-lingual forgetting patterns and low-resource language adaptation.

References

J. Armstrong, A. Thakur, and D. Clifton. Generative latent replay for continual learning. https://github.com/iacobo/generative-latent-replay/blob/main/Generative_Latent_Replay.pdf?raw, 2022.

Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media, Inc., Sebastopol, CA, 2009. ISBN 9780596516499.

Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A pytorch library for deep continual learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023. URL http://jmlr.org/papers/v24/23-0130.html.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond english-centric multilingual machine translation, 2020. URL https://arxiv.org/abs/2010.11125.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzman, and Angela Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation, 2021. URL https://arxiv.org/abs/2106.03193.

Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020. doi: 10.1016/j.tics.2020.09.004.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, and Song Han. Tiny machine learning: Progress and futures [feature]. *IEEE Circuits and Systems Magazine*, 23(3):8–34, 2023. ISSN 1558-0830. doi: 10.1109/mcas.2023.3302182. URL http://dx.doi.org/10.1109/MCAS.2023.3302182.

Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning, 2017. URL https://arxiv.org/abs/1710.06574.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

Daniel Loureiro, Aminette D'Souza, Areej Nasser Muhajab, Isabella A. White, Gabriel Wong, Luis Espinosa Anke, Leonardo Neves, Francesco Barbieri, and Jose Camacho-Collados. Tempowic: An evaluation benchmark for detecting meaning shift in social media, 2022. URL https://arxiv.org/abs/2209.07216.

Alireza Mohammadshahi, Vassilina Nikoulina, Alexandre Berard, Caroline Brun, James Henderson, and Laurent Besacier. Small-100: Introducing shallow multilingual machine translation model for low-resource languages, 2022. URL https://arxiv.org/abs/2210.11621.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209, 2020. doi: 10.1109/IROS45743.2020.9341460.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.52.

Morgan B. Talbot, Rushikesh Zawar, Rohil Badkundri, Mengmi Zhang, and Gabriel Kreiman. Tuned compositional feature replays for efficient stream learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2):3300–3314, February 2025. ISSN 2162-2388. doi: 10.1109/tnnls.2023.3344085. URL http://dx.doi.org/10.1109/TNNLS.2023.3344085.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

7. Appendix

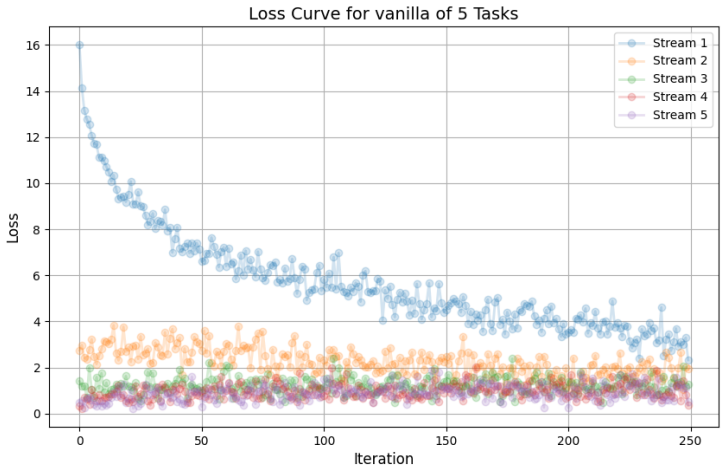


Figure 6: Testing learning dynamics for fine-tuning setup without any replay buffer across 5 tasks/streams. Because the last layer is reinitialized, training improves loss drastically during stream 1 until it starts to plateau once the last layer is at a good enough level.

Hyperparameter	**Value**
Learning Rate	5×10^{-5}
Train Epochs	3
Replay Memory Size(R)	200
Max Sequence Length	128
Training Minibatch Size	1
Evaluation Minibatch Size	4
Warmup Steps	4000
Momentum	900
Weight Decay	0.0005

Table 1: Latent Layer training configuration

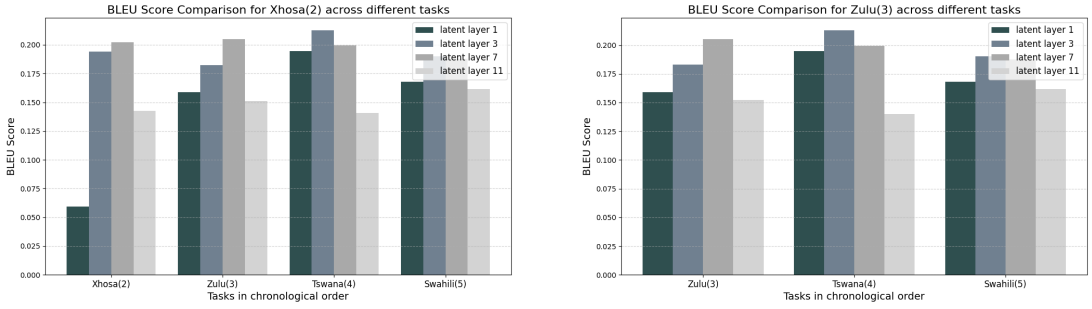


Figure 7: Ablation study of Latent Layer Number for Xhosa(left) and Zulu(right)