Efficient Rehearsal for Continual Learning in ASR via Singular Value Tuning

Steven Vander Eeckt, Hugo Van hamme *Senior, IEEE*

***Abstract*—Continual Learning (CL) in Automatic Speech Recognition (ASR) suffers from catastrophic forgetting when adapting to new tasks, domains, or speakers. A common strategy to mitigate this is to store a subset of past data in memory for rehearsal. However, rehearsal-based methods face key limitations: storing data is often costly, infeasible with pre-trained models, or restricted by privacy regulations. Running existing rehearsal-based methods with smaller memory sizes to alleviate these issues usually leads to degraded performance.**

We propose a rehearsal-based CL method that remains effective even with minimal memory. It operates in two stages: first, fine-tuning on the new task; second, applying Singular Value Decomposition (SVD) to the changes in linear layers and, in a parameter-efficient manner, retraining only gating vectors on the singular values, which control to extent to which updates from the first stage are accepted, using rehearsal. We extensively test and analyze our method on two monolingual and two multilingual benchmarks. Our method reduces forgetting and outperforms state-of-the-art CL approaches for ASR, even when limited to a single utterance per previous task.

***Index Terms*—continual learning, automatic speech recognition, rehearsal-based continual learning, singular value decomposition**

I. Introduction

Automatic Speech Recognition (ASR) systems are increasingly deployed in real-world, dynamic environments where they must continuously adapt to new speakers, domains, accents or conditions. In such settings, continual learning (CL) becomes essential as it allows models to incrementally learn new tasks without access to previously seen data. However, a major challenge in CL is *catastrophic forgetting*—the tendency of neural networks to lose performance on previously learned tasks when updated with new data [1].

A prominent strategy to mitigate forgetting is *rehearsal*, where a small subset of past data is retained and interleaved with new task data during training [2], [3]. *Rehearsal-based methods* have proven effective in ASR [4], [5], but come with limitations: storing real user data may be infeasible due to privacy or regulatory constraints, and managing large memories can be computationally prohibitive. These issues motivate the development of *memory-efficient* rehearsal methods that can operate with very limited memory.

In this work, we introduce a rehearsal-based continual learning method for ASR that is both effective and highly memory-efficient. Our approach involves two stages. First, we fine-tune the model on the new task. Then, we perform a structured update using Singular Value Decomposition (SVD) of the linear layer weight changes. By introducing a learnable gating vector on the singular values, we selectively retain only those updates that balance learning the new task with preserving performance on previous ones. Importantly, only a small number of parameters are updated during this stage, making our method well-suited to low-memory settings.

We extensively evaluate our method across four experiments: two monolingual setups involving microphone and accent shifts, one multilingual setup with task-specific model components, and one multilingual setup based on a Whisper-style [6] foundation model. The results show that our method is highly effective even when using an extremely small memory—down to a single utterance per task—and consistently outperforms state-of-the-art baselines, including both non-rehearsal methods and rehearsal-based approaches that rely on significantly larger memory budgets. Our method achieves a strong balance between retaining past knowledge and adapting to new tasks. In addition, we conduct an in-depth analysis of the learned gating vectors, which reveals that the model tends to either fully suppress or fully accept individual rank-one updates. Finally, an ablation study confirms the importance of our method's key components toward its overall performance.

S. Vander Eeckt and H. Van hamme are with Department Electrical Engineering ESAT-PSI, KU Leuven, B-3001 Leuven, Belgium

II. Continual Learning for ASR

A. Model and Notation

The ASR model used in this work is an encoder-decoder architecture trained in a hybrid fashion [7]. It takes as input an utterance $\boldsymbol{X} \in \mathbb{R}^{l \times d_i}$, where l is the number of acoustic frames and d_i is the dimensionality of each frame. The output is a sequence $\hat{\boldsymbol{y}}$ of $\tilde{w}$ predicted word pieces.

The model parameters are denoted by $\boldsymbol{\theta} \in \mathbb{R}^N$. The training objective combines two losses: a CTC loss with weight c, and a decoder-based cross-entropy loss with weight $1-c$. The model is trained on paired data $(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{y}$ is the ground truth sequence of w word pieces corresponding to input $\boldsymbol{X}$. Eq. 1 denotes the loss of the model:

$$\mathcal{L}_{\text{ce}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) = c\mathcal{L}_{\text{ctc}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) + (1-c)\mathcal{L}_{\text{dec}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) \quad (1)$$

With $\mathcal{L}_{\text{ctc}}(\cdot)$ and $\mathcal{L}_{\text{dec}}(\cdot)$ the CTC loss and decoder-based cross-entropy loss, respectively.

We denote the output distributions (after applying the softmax) of the two branches as follows: $f^{\text{ctc}}(\boldsymbol{X}; \boldsymbol{\theta}) \in \mathbb{R}^{l \times C}$ is the CTC output; $f^{\text{dec}}(\boldsymbol{X}; \boldsymbol{\theta}) \in \mathbb{R}^{w \times C}$ is the decoder output, computed during training and conditioned on previous ground-truth tokens, with C the output size and size of the vocabulary.

B. Problem Formulation

We consider a continual learning (CL) setting for ASR where a model must learn a sequence of T tasks, each

becoming available one at a time. Tasks may vary in language, speaker, accent, recording environment, etc. For each task $t \in \{1, \dots, T\}$, we are given a training set $\mathcal{D}_t^{\text{train}}$ and a validation set $\mathcal{D}_t^{\text{val}}$, both consisting of paired utterances $(\boldsymbol{X}, \boldsymbol{y})$. Once task t becomes available, access to the training and validation sets of the previous task $t-1$ is lost—that is, $\mathcal{D}_{t-1}^{\text{train}}$ and $\mathcal{D}_{t-1}^{\text{val}}$ are no longer accessible.

Given a model with parameters $\boldsymbol{\theta}^{t-1} \in \mathbb{R}^N$ performing well on tasks $1, \dots, t-1$, the goal of CL is to obtain updated parameters $\boldsymbol{\theta}^t$ (using $\mathcal{D}_t^{\text{train}}$ and $\mathcal{D}_t^{\text{val}}$) that satisfy two criteria [8]:

(i) *Retain previously learned knowledge*: Maintain performance on all previous tasks. Ideally, for each $j \in \{1, \dots, t-1\}$,

$$\sum_{(\boldsymbol{X},\boldsymbol{y}) \in \mathcal{D}_j^{\text{train}}} \mathcal{L}_{\text{ce}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}^t) \leq \sum_{(\boldsymbol{X},\boldsymbol{y}) \in \mathcal{D}_j^{\text{train}}} \mathcal{L}_{\text{ce}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}^{t-1}), \tag{2}$$

(ii) *Adapt effectively to new data*: Perform well on the new task t. That is, as a proxy:

$$\boldsymbol{\theta}^t = \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{X},\boldsymbol{y}) \in \mathcal{D}_t^{\text{train}}} \mathcal{L}_{\text{ce}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}). \tag{3}$$

To meet these objectives, CL methods typically modify the optimization procedure, adjust the architecture, or incorporate a small memory buffer $\mathcal{M}$ containing $|\mathcal{M}|$ samples from previous tasks. Rehearsal-based methods use $\mathcal{M}$ to replay past samples during training on task t, mitigating forgetting by approximating access to $\mathcal{D}_1^{\text{train}}, \dots, \mathcal{D}_{t-1}^{\text{train}}$. However, storing a large memory may be impractical—due to privacy constraints, storage limitations, or restrictions when using pretrained models. As such, designing methods that remain effective even with a highly limited memory is crucial, especially as naively fine-tuning $\boldsymbol{\theta}^{t-1}$ on task t without such strategies generally results in catastrophic forgetting.

C. Related Work

Continual Learning. Following [9], CL methods can be categorized into three groups. *Regularization-based methods*, e.g. [10]–[13], alleviate forgetting through auxiliary loss terms that penalize changes to important parameters. *Architectural-based methods*, e.g., [14], [15], modify the model architecture to allocate task-specific capacity. Finally, *Rehearsal-based methods* store a small memory of training examples from past tasks and interleave them with current task data to mitigate forgetting. Despite their simplicity, rehearsal-based strategies remain among the most effective approaches in continual learning [2], [3], [16]–[18]. [19] shows that optimal continual learning is NP-hard and would require perfect memorization of past tasks, making approximate recall strategies such as experience replay particularly promising [20]. Their theoretical analysis supports the empirical success and widespread use of rehearsal-based methods [21] over regularization-based ones. The effectiveness of rehearsal-based methods has also been attributed to their ability to remain within low-loss regions of previously learned tasks [21].

Continual Learning for ASR. In recent years, CL has gained increasing attention within the ASR community. [4], [5] show that rehearsal-based methods significantly outperform regularization-based methods, highlighting the importance of a memory of past samples. To omit this memory requirement, [22] considers task-specific adapters [23], while [24] combines expert models, and [25] introduces a disentangled architecture with task-specific parameters—examples of architectural-based strategies. Regularization-based methods for ASR were proposed by [26], which introduces weight averaging, illustrating its efficacy for CL in ASR; [27], which proposes updating only the encoder for CL; and [28], which suggests to update only a subset of encoders during each epoch. [29]–[31] focus on CL for multilingual Whisper-based [6] ASR, with the latter two adopting low-rank adaptation (LoRA) [32] within architectural-based methods. [33]–[36] explore online CL for ASR, where the model is exposed to a stream of non-i.i.d. data rather than a sequence of tasks with known boundaries. Online CL is not considered in this paper.

SVD-Based Learning. SVD-based learning has been explored within the context of parameter-efficient fine-tuning of pretrained models. [37], [38] consider LoRA [32] decomposed via SVD. [39] further reduces the number of trainable parameters by fine-tuning only scale vectors. [40] considers the adaptation presented in its CUR decomposition [41], training only the matrix $\boldsymbol{U}$. [42], [43] consider singular-value fine-tuning. [44] performs SVD on the product of pretrained weights and task-specific covariance matrices, orienting the decomposition toward task context and using it to initialize LoRA. Note that in these works, the focus is on adapting a pretrained model to a new task, not on preventing forgetting as in our work.

III. Singular Value-based Rehearsal

We propose a memory-efficient rehearsal-based continual learning method (which we name Singular Value-based Rehearsal [SVR]) for ASR, consisting of two stages. Given a model trained on the previous $t-1$ tasks with parameters $\boldsymbol{\theta}^{t-1}$, the objective is to train $\boldsymbol{\theta}$ on task t, using $\boldsymbol{\theta}^{t-1}$, $\mathcal{D}_t^{\text{train}}$ and $\mathcal{M}$, in order to obtain $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}$ which works well on all seen tasks $1, \dots, t$.

A. Stage 1: Fine-Tuning to learn the new task t

We first perform conventional fine-tuning on the new task t using $\mathcal{D}_t^{\text{train}}$, starting from model parameters $\boldsymbol{\theta}^{t-1}$. This results in an updated set of parameters $\tilde{\boldsymbol{\theta}}^t$. Let $\Delta\tilde{\boldsymbol{\theta}}_t = \tilde{\boldsymbol{\theta}}^t - \boldsymbol{\theta}^{t-1}$ denote the parameter changes induced by this adaptation.

Although effective for the current task, $\Delta\tilde{\boldsymbol{\theta}}_t$ often degrades performance on previously learned tasks, resulting in catastrophic forgetting. To address this, we introduce a second stage that filters out components of $\Delta\tilde{\boldsymbol{\theta}}_t$ that are detrimental to tasks $1, \dots, t-1$, yielding a model that maintains performance on both past and current tasks.

Note that the memory $\mathcal{M}$ is not used during this stage.

B. Stage 2a: Preparing Singular Value-Based Rehearsal

Consider a linear layer in the ASR model with parameters $\boldsymbol{\theta}$, represented by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_o \times d_i}$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^{d_o}$. Given an input $\boldsymbol{h} \in \mathbb{R}^{d_i}$, the output of the layer is

computed as $(\boldsymbol{W}\boldsymbol{h}+\boldsymbol{b}) \in \mathbb{R}^{d_o}$. We hypothesize that the weight matrix $\boldsymbol{W}$ plays the primary role in both learning new tasks and forgetting old ones, whereas the bias $\boldsymbol{b}$ contributes only marginally. Therefore, our focus is on achieving an effective balance between retaining knowledge of previous tasks and adapting to new ones by carefully managing updates to $\boldsymbol{W}$.

Let $\tilde{\boldsymbol{W}}_t$ denote the linear layer of the model with parameters $\tilde{\boldsymbol{\theta}}^t$ after fine-tuning on task t (Stage 1 from Sec. III-A), and let $\boldsymbol{W}_{t-1}$ denote the corresponding layer in the model with parameters $\boldsymbol{\theta}_{t-1}$ prior to adaptation. We define the weight update as $\Delta\boldsymbol{W}_t = \tilde{\boldsymbol{W}}_t - \boldsymbol{W}_{t-1}$. Writing this update $\Delta\boldsymbol{W}_t$, which is applied to $\boldsymbol{W}_{t-1}$ to adapt to the new task, in terms of its Singular Value Decomposition (SVD), we have:

$$\Delta\boldsymbol{W}_t = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{k} s_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{4}$$

where $k \leq \min(d_o, d_i)$. Here, $\boldsymbol{U} \in \mathbb{R}^{d_o \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{d_i \times k}$ contain the left and right singular vectors, respectively, with $\boldsymbol{u}_i \in \mathbb{R}^{d_o}$ and $\boldsymbol{v}_i \in \mathbb{R}^{d_i}$ denoting the i-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$. The diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}$ holds the nonnegative singular values $\boldsymbol{s}_i$ on its diagonal. Each rank-one component $s_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ represents a principal direction of change induced by the update $\Delta\boldsymbol{W}_t$.

The update $\Delta\boldsymbol{W}_t$ can thus be expressed as a sum of k rank-one components: $s_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ for $i = 1, \ldots, k$. Importantly, not all of these components contribute equally to learning the new task or to forgetting previously learned tasks. Ideally, we would retain those updates that significantly improve performance on the new task while causing minimal interference with earlier tasks, and discard those that contribute little to learning but induce substantial forgetting. For updates that are beneficial for the new task but also harmful to prior tasks, a compromise may be achieved through averaging [26]. To implement such selective control over the influence of each component, we introduce a weighting vector $\boldsymbol{\alpha} \in \mathbb{R}^k$ and define a modified update $\Delta\hat{\boldsymbol{W}}_t$ as:

$$\Delta\hat{\boldsymbol{W}}_t = \boldsymbol{U}\text{diag}(\sigma(\boldsymbol{\alpha}) \odot \boldsymbol{s})\boldsymbol{V}^T = \sum_{i=1}^{k} \sigma(\alpha_i) s_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{s} \in \mathbb{R}^k$ is the diagonal of $\boldsymbol{\Sigma}$, $\text{diag}(\boldsymbol{x})$ turns $\boldsymbol{x} \in \mathbb{R}^d$ into a $d \times d$ diagonal matrix, and $\odot$ denotes element-wise multiplication. The updated linear layer in the current model $\boldsymbol{\theta}$ is computed as:

$$\boldsymbol{W} \leftarrow \boldsymbol{W}_{t-1} + \Delta\hat{\boldsymbol{W}}_t = \boldsymbol{W}_{t-1} + \boldsymbol{U}\text{diag}\left(\sigma(\boldsymbol{\alpha}) \odot \boldsymbol{s}\right)\boldsymbol{V}^T \tag{6}$$

Here, $\boldsymbol{\alpha} \in \mathbb{R}^k$ is a trainable vector, and each $\sigma(\alpha_i) \in [0, 1]$ controls the contribution of the i-th rank-one update. During training, only $\boldsymbol{\alpha}$ is updated, while $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ (thus $\boldsymbol{s}$), $\boldsymbol{V}$, and $\boldsymbol{W}_{t-1}$ remain fixed.

To achieve a balance between retaining performance on previous tasks and adapting to the new task, $\boldsymbol{\alpha}$ is optimized using rehearsal: it is trained jointly on the memory of old samples and the new task data. Through this *Stage 2 training*, the model learns to selectively scale the rank-one updates proposed by Stage 1. If a particular update negatively impacts prior tasks, the model can suppress it by driving $\sigma(\alpha_i) \approx 0$. Conversely, if an update is beneficial and causes minimal forgetting, it may be preserved with $\sigma(\alpha_i) \approx 1.0$. In cases where an update benefits the new task but harms the old ones, the model may seek a trade-off by setting $\sigma(\alpha_i) \approx 0.5$.

The proposed SVD-based rehearsal strategy can be applied to all linear layers in the model, which collectively account for the vast majority of the model's parameters. All other parameters, including those in bias vectors, and convolutional and embedding layers, are kept frozen throughout this stage. However, motivated by the effectiveness of weight averaging [26], we initialize these other model parameters (to which our SVD-based rehearsal is *not* applied) as the average of their values in $\boldsymbol{\theta}^{t-1}$ and $\tilde{\boldsymbol{\theta}}^t$. For instance, if $\boldsymbol{p}$ is such a model parameter in the current model, and $\tilde{\boldsymbol{p}}_t$ and $\boldsymbol{p}_{t-1}$ are the corresponding model parameters from $\tilde{\boldsymbol{\theta}}^t$ and $\boldsymbol{\theta}^{t-1}$, respectively, we set $\boldsymbol{p}$ as follows:

$$\boldsymbol{p} \leftarrow \frac{\tilde{\boldsymbol{p}}_t + \boldsymbol{p}_{t-1}}{2} \tag{7}$$

$\boldsymbol{p}$ then remains frozen. As the number of trainable parameters is thus limited to the size of all $\boldsymbol{\alpha}$'s, our approach offers substantial efficiency: for each linear layer, only $k \leq \min(d_o, d_i)$ parameters in $\sigma(\boldsymbol{\alpha})$ are trained, as opposed to the full $d_o \times d_i$ weight matrix. We hypothesize that this reduction in parameter count allows Stage 2 to be trained effectively using a minimal memory buffer, minimizing the risk of overfitting on the memory while mitigating forgetting of old tasks.

C. Stage 2b: Singular Value-Based Rehearsal

To train $\sigma(\boldsymbol{\alpha})$ for each linear layer to which our SVD-based rehearsal is applied—jointly on the memory of old tasks and the new task—we minimize the following loss:

$$\begin{aligned}\mathcal{L}_{\text{st2}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_M, \boldsymbol{y}_M; \boldsymbol{\theta}) &= \mathcal{L}_{\text{ce}}(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) \\ &+ \frac{t-1}{2}\left(\mathcal{L}_{\text{ce}}(\boldsymbol{X}_M, \boldsymbol{y}_M; \boldsymbol{\theta}) + \mathcal{L}_{\text{kd}}(\boldsymbol{X}_M; \boldsymbol{\theta})\right)\end{aligned} \tag{8}$$

with $(\boldsymbol{X}, \boldsymbol{y}) \sim D_t^{\text{train}}$, $(\boldsymbol{X}_M, \boldsymbol{y}_M) \sim \mathcal{M}$, while $\mathcal{L}_{\text{kd}}$ is the knowledge distillation [4], [45] loss given by Eq. 9:

$$\begin{aligned}\mathcal{L}_{\text{kd}}(\boldsymbol{X}; \boldsymbol{\theta}) = c \sum_{i=1}^{l} \sum_{j=1}^{C} f_{i,j}^{\text{ctc}}(\boldsymbol{X}; \boldsymbol{\theta}^{t-1}) \log f_{i,j}^{\text{ctc}}(\boldsymbol{X}; \boldsymbol{\theta}) \\ + (1-c) \sum_{i=1}^{w} \sum_{j=1}^{C} f_{i,j}^{\text{dec}}(\boldsymbol{X}; \boldsymbol{\theta}^{t-1}) \log f_{i,j}^{\text{dec}}(\boldsymbol{X}; \boldsymbol{\theta})\end{aligned} \tag{9}$$

Note that $(\boldsymbol{X}, \boldsymbol{y})$ and $(\boldsymbol{X}_M, \boldsymbol{y}_M)$ here represent mini-batches of size b. The loss is thus computed jointly on a mini-batch from the new task's training set and a mini-batch from the memory buffer. For the mini-batch $(\boldsymbol{X}_M, \boldsymbol{y}_M)$ from memory, similar to [18], we include both a cross-entropy loss and a knowledge distillation loss to encourage the current model to preserve the behavior of the previous model ($\boldsymbol{\theta}^{t-1}$). The memory loss is weighted by $(t-1)/2$: the division by 2 accounts for the presence of two loss components (cross-entropy and distillation), while the multiplication by $(t-1)$ reflects the fact that the memory represents all $t-1$ previous tasks. Intuitively, if one were to maintain a separate memory buffer and corresponding loss term for each prior task, this would result in $t-1$ such terms.

At the start of training, we initialize $\boldsymbol{\alpha}$ such that $\sigma(\boldsymbol{\alpha}) \approx 0$, ensuring that initially $\boldsymbol{W} \approx \boldsymbol{W}_{t-1}$. This initialization preserves the model's performance on the previously learned tasks $1, \ldots, t-1$. It allows our method to stay within the low-loss region of old tasks (from which it does not necessarily start when $\sigma(\boldsymbol{\alpha}) > 0$) during the trajectory of training, which is generally the case for rehearsal-based methods and partly explains their effectiveness [21].

D. Summary

Algorithm 1 summarizes our approach. Line 2 represents Stage 1 from Sec. III-A, in which we fine-tune $\boldsymbol{\theta}^{t-1}$ on $\mathcal{D}_t^{\text{train}}$, obtaining $\tilde{\boldsymbol{\theta}}^t$. For Stage 2, we first apply, as explained in III-B, SVD to and introduce $\boldsymbol{\alpha}$ for each linear layer $\boldsymbol{W}$ of $\boldsymbol{\theta}$ (Lines 4-11). The remaining model parameters are averaged and frozen (Lines 13-17). For each mini-batch of the new task's training set $\mathcal{D}_t^{\text{train}}$ and mini-batch sampled from the memory $\mathcal{M}$ (Line 20), the loss from Eq. 8 (Sec. III-C) is computed in Line 21. Note that Stage 2 training for loop (Lines 19-23) can in practice be repeated for e_{st2} epochs. Before the model $\boldsymbol{\theta}^t$ is returned (Line 26), the memory $\mathcal{M}$ is updated by adding utterances from $\mathcal{D}_t^{\text{train}}$ (Line 25).

Algorithm 1 **S**ingular **V**alue-Based **R**ehearsal (SVR)

Require: Model $\boldsymbol{\theta}^{t-1}$, new task data $\mathcal{D}_t^{\text{train}}$, memory $\mathcal{M}$
Ensure: Updated model $\boldsymbol{\theta}^t$
1: **# Stage 1: Fine-tune on new task**
2: $\tilde{\boldsymbol{\theta}}^t \leftarrow \text{FineTune}(\boldsymbol{\theta}^{t-1}, \mathcal{D}_t^{\text{train}})$
3: **# Stage 2: SVD-based Rehearsal**
4: **for** each linear layer $\boldsymbol{W}$ in $\boldsymbol{\theta}$ **do**
5: # $\tilde{\boldsymbol{W}}_t$ and $\boldsymbol{W}_{t-1}$ are $\boldsymbol{W}$ from $\tilde{\boldsymbol{\theta}}^t$ and $\boldsymbol{\theta}^{t-1}$, resp.
6: $\Delta \boldsymbol{W}_t \leftarrow \tilde{\boldsymbol{W}}_t - \boldsymbol{W}_{t-1}$
7: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^T \leftarrow \text{SVD}(\Delta \boldsymbol{W}_t)$
8: # Introduce trainable vector $\boldsymbol{\alpha}$, $\boldsymbol{s}$ is diagonal of $\boldsymbol{\Sigma}$
9: $\boldsymbol{W} \leftarrow \boldsymbol{W}_{t-1} + \boldsymbol{U}\text{diag}(\sigma(\boldsymbol{\alpha}) \odot \boldsymbol{s})\boldsymbol{V}^T$
10: Freeze $\boldsymbol{W}_{t-1}$, $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ (thus $\boldsymbol{s}$), $\boldsymbol{V}^T$. Train only $\boldsymbol{\alpha}$.
11: **end for**
12: # Average all other parameters
13: **for** each other parameter $\boldsymbol{p}$ in $\boldsymbol{\theta}$ **do**
14: # $\tilde{\boldsymbol{p}}_t$ and $\boldsymbol{p}_{t-1}$ are parameter $\boldsymbol{p}$ from $\tilde{\boldsymbol{\theta}}^t$ and $\boldsymbol{\theta}^{t-1}$, resp.
15: $\boldsymbol{p} \leftarrow (\boldsymbol{p}_{t-1} + \tilde{\boldsymbol{p}}_t)/2$
16: Freeze $\boldsymbol{p}$
17: **end for**
18: # Jointly train on new task and memory
19: **for** each mini-batch $(\boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{D}_t^{\text{train}}$ **do**
20: Sample $(\boldsymbol{X}_M, \boldsymbol{y}_M) \sim \mathcal{M}$ from memory
21: Compute $\mathcal{L}_{st2}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_M, \boldsymbol{y}_M; \boldsymbol{\theta})$ using Eq. 8
22: Update $\boldsymbol{\theta}$ (thus all $\boldsymbol{\alpha}$'s).
23: **end for**
24: # Add samples from $\mathcal{D}_t^{\text{train}}$ to $\mathcal{M}$
25: $\mathcal{M} \leftarrow \text{UpdateMemory}(\mathcal{M}, \mathcal{D}_t^{\text{train}})$
26: **return** $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}$

To populate the memory $\mathcal{M}$ with samples from the task t, using UpdateMemory$(\mathcal{M}, \mathcal{D}_t^{\text{train}})$ from Line 25, we sample uniformly from the training set $\mathcal{D}_t^{\text{train}}$. We consider multiple memory sizes $|\mathcal{M}|$, both fixed and increasing. In the fixed setting, the memory size $|\mathcal{M}| = M$ remains constant across tasks, so each task contributes M/t samples; to make room for new samples, we remove entries by uniformly sampling per task. In the increasing setting, the memory grows linearly with the number of tasks, $|\mathcal{M}| = Mt$, allowing each task to store M utterances. In multilingual settings, tasks may belong to different languages. For fairness, we allocate equal total weight to each language when populating the memory: with fixed $|\mathcal{M}|$, samples are divided equally across languages and then across their tasks; with increasing $|\mathcal{M}| = Mt$, we subsample during training so that each language contributes equally to the regularization, regardless of its number of tasks.

We use uniform sampling to populate the memory, as it is generally applicable, since it does not require access to large candidate pools, unlike more selective strategies.

IV. Experiments

Experiments are done in ESPnet2 [46]. For code and more detailed information, we refer to our Github repository [1].

Data. We conduct four experiments:

1) *Experiment 1*. Following [26], we considering the Common Voice [47] dataset and split the Common Voice English dataset into 5 English accents: United States (US), England (ENG), Australia (AUS), India (IND) and Scotland (SCO). The tasks are learned in this order.
2) *Experiment 2*. Starting from a model trained on LibriSpeech-360h (LIB) [48], it is adapted to four Libri-Adapt (LIB-APT) [49] tasks, with a domain shift in terms of microphone (USB [U], Matrix [M]) and accent (United States [US], Indian [IN], British [GB] English). This closely follows the set-up from [36], where it served as an Online CL benchmark.
3) *Experiment 3*. In addition to the two monolingual experiments above, we also consider one multilingual experiment, based on [26]. The tasks are five languages from the Common Voice dataset: English (US), Russian (RU), Dutch (NL), Polish (PL) and Swedish (SV).
4) *Experiment 4*. Finally, we adapt the Open Whisper-style Speech Model (OWSM) [50] to Dutch across two tasks. As pretrained knowledge, i.e., as old tasks, we consider Common Voice (CV) [47], both English (CV/E) and Dutch (CV/N). Performance on these tasks, which were part of the pretraining for OWSM and therefore considered task 1, should not deteriorate as we adapt OWSM to new tasks. These new tasks are obtained from Corpus Gesproken Nederlands (CGN) [51], a dataset which contains approximately 900 hours of spoken Dutch covering a wide range of speech registers not represented in CV, including spontaneous conversations, interviews, debates, lectures, and broadcast material. The CGN data are split into Dutch from the Netherlands (CG/N) and Dutch from Belgium (CG/V), using non-spontaneous speech to define the two tasks. Compared to [4], the tasks contain fewer samples, as the segments are concatenated into longer utterances of 30s.

[1] https://github.com/StevenVdEeckt/efficient-rehearsal-for-cl-in-asr

TABLE I
NUMBER OF UTTERANCES PER TASK FOR EACH EXPERIMENT, LISTED IN THE ORDER IN WHICH THE TASKS ARE PRESENTED TO THE MODEL.

Task	Exp. 1		Exp. 2		Exp. 3		Exp. 4	
1	US	349,561	LIB	103,997	US	349,561	CV/E	—
2	ENG	116,464	GB/M	9,001	RU	82,862	CV/N	—
3	AUS	56,287	US/U	27,536	NL	59,146	CG/N	27,363
4	IND	71,086	IN/U	9,271	PL	92,221	CG/V	20,664
5	SCO	10,744	IN/M	9,271	SV	23,411	—	
Total	**604,142**		**159,076**		**607,201**		**56,659**	

TABLE II
NUMBER OF PARAMETERS (*Params.*) AND PERCENTAGE OF TOTAL MODEL SIZE (*Perc.*) FOR THE SHARED MODEL (SHARED ACROSS TASKS), ALL (TRAINABLE) $\boldsymbol{W}$ MATRICES, AND THEIR CORRESPONDING $\boldsymbol{\alpha}$'S.

Group	Exp. 1–2		Exp. 3		Exp. 4	
	Params.	Perc.	Params.	Perc.	Params.	Perc.
Shared model	46.8M	100.0%	43.0M	92.0%	366.7M	100.0%
All $\boldsymbol{W}$ matrices	42.4M	90.7%	39.9M	85.3%	244.8M	66.8%
All $\boldsymbol{\alpha}$'s	44.2k	<0.1%	43.8k	<0.1%	145.9k	<0.1%

The number of utterances per task and the task order is given for each experiment in Table I.

Model. Taking MFCC features as input, our ASR model for Experiments 1-3 consists of 12 Conformer [52] encoder and 6 Transformer [53] decoder layers with dimension 2048 and 4 attention heads with dimension 256. The weight of CTC is $c = 0.3$ during training and decoding. For the monolingual experiments, *SentencePiece* [54] is used to generate $C = 5000$ output tokens on the first task; while for the multilingual experiment, it generates $C = 5000$ output tokens for each task separately. In the multilingual experiment, therefore, the token embedding in the decoder and the output layers of CTC and decoder are task-specific (thus language-specific) and not shared across tasks. The remaining 92% of the parameters are shared and thus susceptible to forgetting. An overview of the number of (shared) parameters in the model and the portion affected by our method is provided in Table II. As shown, in the monolingual experiments (Exp. 1-2), the entire model is shared and contains 46.7M parameters, of which 90.7% are in the weight matrices of the linear layers. By applying SVD to the adaptation in these matrices, and making only $\boldsymbol{\alpha}$ trainable, the number of trainable parameters is reduced to 44.2k, or less than 0.1% of total parameters. For the multilingual experiment (Exp. 3), only 43.8k parameters are trained during Stage 2 of our method, while 43.0M parameters are shared across tasks. For Experiment 4, we use the pretrained OWSM v3.2 small [50], which consists of eight E-Branchformer [55] encoder and eight Transformer decoder layers, and an output vocabulary of $C = 50{,}000$ tokens, amounting to 366.7M parameters. The model was pretrained on 180k hours of speech covering 151 languages, with output tokens shared between languages. When adapting OWSM to CGN, we freeze the output layers and decoder token embeddings and only update the linear weight matrices, analogous to LoRA but without the low rank component. When naively fine-tuning the model, restricting updates to linear weights led to both better learning and less forgetting, due to less overfitting. Adaptation is thus applied to 244.8M parameters (66.8% of the entire model), corresponding, as shown in Table II, to 145.9k parameters during Stage 2 of our method. CTC is only used during training, with $c = 0.3$.

Training. For Experiments 1-3, the model is trained for 80 epochs on the initial task 1, and for 10 epochs on each subsequent tasks $2, ..., T$. In the multilingual experiment, before training the entire model on a new task (i.e. language), a new task-specific part is initialized and trained for 10 epochs. For Stage 2 training of our method, the number of epochs is set to $e_{\text{st}} = 3$. The optimizer is Adam [56], reset before learning a new task and before each stage. Following [4], the learning rate is set 10 times smaller (0.0001) for tasks $2, ..., T$ compared to task 1 (0.001), since training does not start from scratch but from a reasonably good starting point (i.e. the previous task model parameters). During Stage 2 of our method, as $\boldsymbol{\alpha}$ is learned from scratch, we set the learning rate to 0.01. This value ensures that, when regularization is disabled, SVR can effectively recover performance on the new task, thus allowing $\sigma(\boldsymbol{\alpha})$ to converge. For the language-specific components of the multilingual experiment, the learning rate is also set to 0.01, which worked well to train the language-specific components of RU with the initial US model. The batch size is set to 64 at all times. For Experiment 4, OWSM is fine-tuned for 20 epochs using the Adam optimizer with a learning rate of 5.0e-5 when adapting to a new task, and a learning rate of 1.0e-2 during Stage 2 training of our method. The batch size is 64.

Baselines. We compare our method to state-of-the-art rehearsal-based and regularization-based CL methods for ASR, as well as to the initial model and to two methods setting the worst-case and best-case performance (though violating the principle of CL). Methods marked with † are rehearsal-based, requiring access to a memory of past data, as our method.

1) *Initial model*. Initial model is the model prior to adaptation, i.e. with parameters $\boldsymbol{\theta}^1$. For the multilingual experiment, the task-specific parameters are trained for each task before computing Initial model's performance. The shared part of the model, however, remains fixed and only trained on the first task.
2) *Fine-Tuning*. Fine-Tuning naively adapts the model to the subsequent tasks, without taking any measures to prevent forgetting. It represents the worst-case scenario.
3) *Seperate Model (Sep. Model)*. Sep. Model keeps a separate model for each task, avoiding task interference and forgetting. The model kept per task is the model finetuned on that task with Fine-Tuning. Sep. Model thus performances as well as Fine-Tuning on new tasks but with zero forgetting (as it can still use the old models for old tasks). Although this violates the principle of CL, it represents a best-case scenario.
4) *Experience Replay (ER)* † [17]. ER trains jointly on the current mini-batch from the new task and on a mini-batch sampled from the memory, by computing the cross-entropy loss from Eq. 1 for both.
5) *Knowledge Distillation (KD)* † [4]. Similar to ER, KD trains jointly on the new task's data and the memory $\mathcal{M}$.

The difference with ER is that the KD loss from Eq. 9 is used on the mini-batch from the memory, instead of the cross-entropy loss.

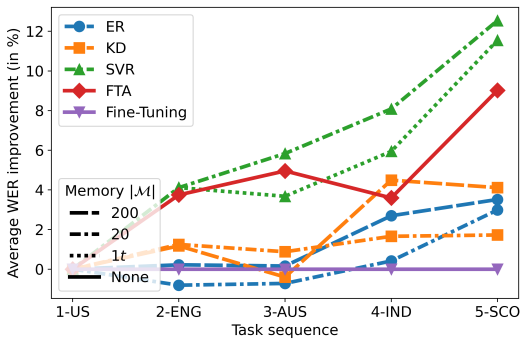
6) *Learning Without Forgetting (LWF)* [13]. LWF uses the KD loss from Eq. 9 on the new task's data to transfer knowledge from the old to the current model. LWF thus does not require access to a memory.
7) *Fine-Tuning with Averaging (FTA)* [26]. If $\tilde{\boldsymbol{\theta}}_t$ are the previous model parameters $\boldsymbol{\theta}_{t-1}$ fine-tuned to task t, then FTA computes and stores for task t the model $\boldsymbol{\theta}_t = (1-\eta)\boldsymbol{\theta}_{t-1} + \eta\tilde{\boldsymbol{\theta}}_t$. Models before and after fine-tuning on task t are thus averaged with a weight η. In [26], $\eta = 0.50$ or $\eta = 1/t$ are tested, with the latter obtaining the best results; we therefore consider the latter.
8) *Update Only Encoders (UOE)* [27]. UOE suggests updating only the encoder, excluding the layer normalization layers, while freezing the rest of the model.
9) *CL by Random Layer-wise Tuning (CLRL-Tuning)* [28]. CLRL-Tuning proposes to only update K randomly selected encoder layers each epoch. We set $K = 1$ (the best setting in [28]) and increase the number of epochs to train a new task from 10 to 50 for CLRL-Tuning.

Hyper-parameters. ER, KD, and LWF require a hyper-parameter λ that controls the weight of the regularization loss. For LWF, we adopt the optimal value $\lambda = 0.1$ from [4]. For ER and KD, we select the best value between $\lambda = 0.1$ (optimal value from [4]) and $\lambda = 1.0$ (optimal value from [35], where the regularization loss on previous tasks is given equal weight to the cross-entropy loss on the current task). The optimal value is determined on the first adaptation of each experiment using the corresponding validation sets. The value of λ is selected independently for each memory size $|\mathcal{M}|$.

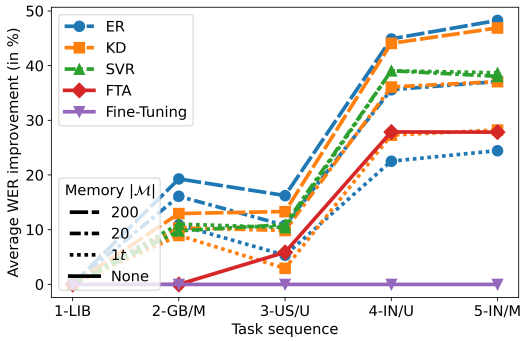
Metrics. We report WER (in %) per task evaluated on the model trained on all tasks. In addition, we also report Average WER, obtained by averaging the WERs of the tasks, to asses the overall performance on the model trained with the given method, and the Backward Transfer (BWT) to asses its forgetting. If $R_{i,j}$ is the WER on task j of the model trained on the first i tasks, then the Average WER is given by Average WER $= \frac{1}{T}\sum_{k=1}^{T} R_{T,k}$, while BWT is given by Eq. 10. Note that negative BWT indicates forgetting, i.e. the average increase of WER of old tasks when learning new tasks. Positive BWT, on the other hand, indicates that the model exploits the learning of new tasks to further improve its performance on old tasks.

$$\text{BWT} = \frac{1}{T-1}\sum_{k=1}^{T-1}(R_{k,k} - R_{T,k}) \tag{10}$$

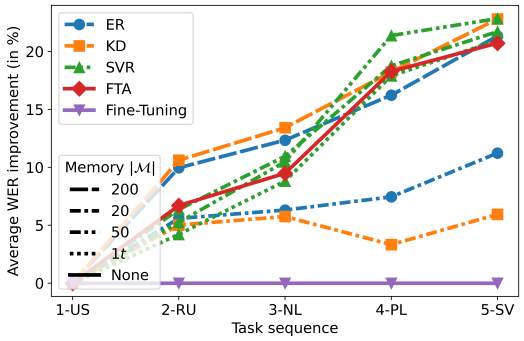
Significance testing. Average WER is the main metric to compare the performance of the CL methods. Therefore, we perform significance testing on Average WER, using Wilcoxon signed-rank test on the number of errors per utterance [57], considering significance levels $\alpha = 0.05$ (*), $\alpha = 0.01$ (**) and $\alpha = 0.001$ (***), or ns for non-significant ($\alpha > 0.05$).



(a) Exp. 1



(b) Exp. 2



(c) Exp. 3

Fig. 1. Average WER improvement (%) over Fine-Tuning, reported per method and after each of the five tasks. The values indicate the relative percentage reduction in Average WER compared to Fine-Tuning at each task.

V. Results

A. Experiment 1

The results for Exp. 1 are shown in Table III and visualized in Figure 1(a). We evaluate SVR with memory sizes $|\mathcal{M}| = 20$ and $|\mathcal{M}| = 1t$, and compare against baselines ER and KD, each tested with memory sizes $|\mathcal{M}| = 200$, $|\mathcal{M}| = 20$ and $|\mathcal{M}| = 1t$. We draw the following observations.

Regarding the performance of baselines, we observe that, due to the diversity in speakers, accents, and lexical content in Common Voice, even rehearsal-based methods such as ER and KD perform relatively poorly. With a memory of $|\mathcal{M}| = 200$ utterances, *KD and ER reduce forgetting by only*

TABLE III

RESULTS OF THE MONOLINGUAL EXP. 1 AND EXP. 2. TASKS ARE LEARNED FROM LEFT TO RIGHT, WITH WERS OBTAINED AFTER LEARNING ALL TASKS. FOR REHEARSAL-BASED METHODS, THE MEMORY SIZE $|\mathcal{M}|$ IS INDICATED. SVR IS OUR PROPOSED METHOD. BEST AVERAGE WER WITH SMALL MEMORY ($|\mathcal{M}| < 200$) IS IN BOLD, SECOND BEST IS UNDERLINED. SIGNIFICANCE TESTING IS APPLIED TO COMPARE SVR AGAINST BASELINES AND TO ASSESS THE EFFECT OF MEMORY SIZE REDUCTION FOR REHEARSAL-BASED METHODS (SVR, ER, KD).

Method	$\lvert\mathcal{M}\rvert$	*Exp. 1*							*Exp. 2*						
		WER↓ per task					Average		WER↓ per task					Average	
		1–US	2–ENG	3–AUS	4–IND	5–SCO	WER↓	BWT↑	1–LIB	2–GB/M	3–US/U	4–IN/U	5–IN/M	WER↓	BWT↑
Initial model		15.4	12.7	13.4	21.4	13.5	15.25	—	6.4	19.4	7.9	18.8	32.9	17.08	—
Fine-Tuning		18.2	12.1	12.4	22.2	10.5	15.07	-3.6	17.0	19.6	16.5	4.1	4.9	12.43	-9.0
Sep. Model		15.4	10.2	8.9	16.0	10.5	12.17	0.0	6.4	4.7	5.9	4.3	4.9	5.23	0.0
FTA		15.9	10.9	10.7	19.3	11.8	13.71	-0.3	7.5	8.6	8.1	8.5	12.2	8.97	-0.1
LWF		17.8	11.7	12.4	21.0	10.1	14.60	-3.2	17.2	18.6	16.6	4.1	4.9	12.27	-8.7
UOE		18.5	12.5	12.8	22.6	10.5	15.36	-3.8	14.3	19.4	17.1	4.4	5.4	12.10	-8.2
CLRL-Tuning		17.9	12.2	12.4	22.3	11.6	15.26	-2.8	11.7	16.4	14.4	5.6	7.1	11.02	-5.5
KD	200	17.9	11.8	12.1	20.0	10.4	14.45[d]	-2.8	7.8	6.9	8.0	4.6	5.6	6.60[d]	-1.1
	20	18.2	12.1	12.2	21.0	10.6	14.81[d,f]	-3.3	9.5	9.9	9.9	4.5	5.3	7.82[d]	-2.8
	$1t$	18.3	12.3	12.2	20.8	20.4	14.81[f]	-3.2	10.8	12.6	11.1	4.7	5.6	8.92[d]	-4.1
ER	200	17.9	12.1	12.0	20.2	10.5	14.54[f]	-2.9	7.6	6.8	7.9	4.5	5.4	6.43[d]	-1.1
	20	17.8	11.8	12.3	20.8	10.5	14.62[e,f]	-3.0	9.4	10.4	9.9	4.3	5.1	7.82[d]	-3.0
	$1t$	18.2	11.8	12.5	21.0	10.5	14.80[e]	-3.2	12.2	13.7	11.8	4.2	5.1	9.39[d]	-5.1
SVR	20	15.9	10.4	10.4	18.3	10.9	**13.18**[a,f]	-0.6	8.3	7.9	8.8	5.8	7.7	7.70[b,f]	-1.1
	$1t$	16.0	10.5	10.5	18.6	11.0	13.33[a,f]	-0.7	8.4	8.2	8.7	5.5	7.3	**7.62**[c,f]	-1.3

[a] Significantly outperforms all baselines, including those with large memory $|\mathcal{M}| = 200$, with level ***.
[b] Significantly outperforms all baselines with small ($|\mathcal{M}| < 200$) or no memory with level ***, except KD with $|\mathcal{M}| = 20$ (*ns*).
[c] Significantly outperforms all baselines with small ($|\mathcal{M}| < 200$) or no memory with level ***, except KD with $|\mathcal{M}| = 20$ with level *.
[d, e] Reducing memory significantly deteriorates performance of given method, with level *** for *d* and * for *e*.
[f] Reducing memory does not significantly deteriorate performance of given method (level is *ns*).

TABLE IV

RESULTS OF MULTILINGUAL EXP. 3 AND OWSM-BASED EXP. 4. TASKS ARE LEARNED FROM LEFT TO RIGHT, WITH WERS OBTAINED AFTER LEARNING ALL TASKS. FOR REHEARSAL-BASED METHODS, THE MEMORY SIZE $|\mathcal{M}|$ IS INDICATED. SVR IS OUR PROPOSED METHOD. BEST AVERAGE WER WITH SMALL MEMORY ($|\mathcal{M}| < 200$) IS IN BOLD, SECOND BEST IS UNDERLINED. SIGNIFICANCE TESTING IS APPLIED TO COMPARE SVR AGAINST BASELINES AND TO ASSESS THE EFFECT OF MEMORY SIZE REDUCTION FOR REHEARSAL-BASED METHODS (SVR, ER, KD).

Method	$\lvert\mathcal{M}\rvert$	*Exp. 3*							*Exp. 4*					
		WER↓ per task					Avg.		WER↓ per task				Avg.	
		1–US	2–RU	3–NL	4–PL	5–SV	WER↓	BWT↑	1–CV/E	2–CV/N	3–CG/N	4–CG/V	WER↓	BWT↑
Initial model		15.4	45.7	27.6	29.8	44.8	32.64	—	13.4	15.4	45.7	37.9	28.08	—
Fine-Tuning		44.6	47.1	18.0	14.6	22.1	29.29	-14.2	24.3	9.6	27.7	13.6	18.81	-2.4
Sep. Model		15.4	26.3	13.6	12.2	22.1	17.90	0.0	13.4	15.4	22.4	13.6	16.18	0.0
FTA		19.9	30.8	18.0	18.5	29.0	23.22	-0.7	14.8	9.5	26.8	18.5	17.42	+2.2
LWF		50.7	62.3	19.3	14.0	20.8	33.39	-19.9	—	—	—	—	—	—
CLRL-Tuning		24.8	43.5	23.3	21.5	31.8	28.99	-4.1	—	—	—	—	—	—
KD	200	27.2	32.2	16.1	14.2	23.4	22.61[e]	-5.3	18.7	9.7	26.0	13.5	16.98[e]	-0.3
	20	44.3	39.8	17.2	14.3	22.2	27.55[e]	-12.0	20.4	10.5	26.0	13.6	17.64[e]	-1.6
ER	200	25.9	34.0	17.5	14.6	23.2	23.04[e]	-5.6	17.4	8.9	27.0	13.4	16.72[e]	-0.2
	20	33.9	39.6	18.2	15.6	22.8	26.00[e]	-9.7	19.5	10.6	26.3	13.3	17.41[e]	-1.6
SVR	50	21.2	31.1	17.1	16.4	27.2	**22.60**[c,f]	-1.2	18.0	8.4	24.4	14.9	**16.42**[d,g]	+0.4
	20	21.9	31.5	17.1	16.5	27.6	22.93[b,f,h]	-1.8	18.7	8.5	24.5	15.0	16.65[a,g,e]	+0.2
	$1t$	23.5	31.5	17.2	16.2	27.5	23.15[b,h]	-1.7	18.9	8.8	25.9	14.8	17.10[a,e]	-1.9

[a, b] Significantly outperforms all baselines with small ($|\mathcal{M}| < 200$) or no memory with level *** (*a*), except FTA with level *ns* (*b*).
[c] Significantly outperforms all baselines with level ***, except FTA with level ** and KD with $|\mathcal{M}| = 200$ with level *ns*.
[d] Significantly outperforms all baselines with level ***, except ER with $|M| = 200$ with level ns.
[e, f, g] Reducing memory significantly deteriorates performance of given method with level *** for *e*, level ** for f and level * for g.
[h] Reducing memory does not significantly deteriorate performance of given method (level is *ns*).

1/5th, and neither approach comes close to the best case performance given by *Sep. Model*. When the memory is reduced to $|\mathcal{M}| = 20$ utterances, ER and KD only marginally improve over Fine-Tuning, offering limited mitigation of forgetting. On the other hand, the *non-rehearsal-based baselines* such as LWF, CLRL-Tuning, and UOE fail to substantially improve upon Fine-Tuning. While LWF yields slight improvements, forgetting remains substantial, and the performance gap to *Sep. Model* persists. The exception is FTA, which nearly eliminates forgetting, but at the cost of limited learning on new tasks—especially as the number of tasks t increases, since the weight of new tasks decreases proportionally $(1/t)$.

Our method, SVR, outperforms FTA, as well as ER and KD—even with a large memory of $|\mathcal{M}| = 200$ utterances, in addition achieving better final performance on four out of five tasks. This holds not only with, for SVR, access to a small memory of $|\mathcal{M}| = 20$ utterances, but even with access to just *one utterance per task* ($|\mathcal{M}| = 1t$). In both cases, *SVR reduces forgetting compared to Fine-Tuning by more than* 80%, while still achieving high performance on new tasks. Unlike FTA, *SVR strikes a much better balance between retaining old knowledge and integrating new information*. Overall, SVR closes the gap between Fine-Tuning (worst case) and Sep. Model (base case) by nearly *two-thirds* using $|\mathcal{M}| = 20$, and by over *60%* using $|\mathcal{M}| = 1t$.

Figure 1(a) shows the relative improvement in Average WER over Fine-Tuning after each task. *SVR with 20 memory samples consistently outperforms all other methods after each stage*. While FTA performs competitively in the early stages, it fails to adapt well as more tasks are introduced, and the gap between FTA and SVR widens over time. Notably, *SVR with just one memory sample per task outperforms KD and ER with $|\mathcal{M}| = 200$ utterances after all tasks*, despite the $200\times$ (when learning 2–ENG) to $50\times$ (when leaning 5–SCO) difference in memory size.

B. Experiment 2

The results for Exp. 2 are reported in Table III and visualized in Figure 1(b). While the overall trends are consistent with those observed in Exp. 1 (Sec. V-A), some important distinctions arise due to the nature of the tasks involved.

Compared to Common Voice, the LibriSpeech and Libri-Adapt tasks consist of much longer utterances with less variation in terms of speakers and lexical content. For example, comparing 2–ENG from Exp. 1 and 2–GB/M from Exp. 2, we find that the former includes 1.8k speakers, 58.4k unique words, and 9.2 words per utterance, while the latter contains 211 speakers, 19.8k unique words, and 34.88 words per utterance. This difference is reflected in the results: in Exp. 2, both ER and KD perform strongly with $|\mathcal{M}| = 200$, and even with $|\mathcal{M}| = 20$, they still yield substantial improvements over Fine-Tuning. Fewer memory samples are thus required to maintain strong performance.

Despite this, SVR remains highly competitive. *With a memory of just $|\mathcal{M}| = 20$, SVR continues to outperform both ER and KD, and even with only one utterance per task ($|\mathcal{M}| = 1t$), it exceeds their performance when they use 20 samples*. In this low-memory regime, ER and KD suffer significant degradation, whereas SVR maintains robust performance—forgetting approximately 60% less than ER and KD when the latter use a larger memory of $|\mathcal{M}| = 20$ utterances. Relative to Fine-Tuning, SVR reduces forgetting by more than 85% with just one utterance per task.

SVR also outperforms all non-rehearsal-based methods. As in Exp. 1, LWF, UOE, and CLRL-Tuning offer only modest gains over Fine-Tuning, and fail to match the effectiveness of memory-based approaches. FTA again nearly eliminates forgetting, but does so at the cost of poor plasticity—its ability to learn new information is significantly constrained.

Figure 1(b) illustrates the relative improvement in Average WER over Fine-Tuning after each task. From the third task onward (3–US/U), SVR with $|\mathcal{M}| = 20$ and $|\mathcal{M}| = 1t$ outperform both KD and ER with $|\mathcal{M}| = 20$. This reflects a general trend: while SVR forgets less, it may also learn new tasks slightly less than ER and KD. This trade-off is visible in the first adaptation (to 2–GB/M), where ER and KD enjoy an initial advantage in Average WER due to their higher plasticity. However, the gap narrows and eventually reverses as more tasks are learned—particularly because SVR better preserves prior knowledge, a factor that becomes increasingly important in later stages of continual learning.

C. Experiment 3

Table IV presents the results of the multilingual experiment (Exp. 3). Several insights can be drawn from these results.

First, the *Initial model*, which combines the initial shared model trained on US-English with adapted language-specific components for each target language, performs poorly on all languages except US. This highlights the need to adapt the shared parameters, which account for approximately 92% of the total model parameters (see Table II), in order to achieve strong multilingual performance.

Next, *SVR demonstrates a clear advantage* over existing rehearsal-based methods, even with minimal memory. With $|\mathcal{M}| = 1t$, SVR substantially outperforms both ER and KD configured with $|\mathcal{M}| = 20$, reducing their forgetting by *between 82% and 85%*. SVR also significantly outperforms non-rehearsal baselines such as LWF and CLRL-Tuning.

Compared to rehearsal-based methods with larger memory ($|\mathcal{M}| = 200$), SVR with $|\mathcal{M}| = 20$ or $|\mathcal{M}| = 1t$ still achieves a *reduction in forgetting of 65% to 70%*. However, in terms of average WER, it merely matches ER and slightly underperforms KD, which benefits from greater flexibility and hence learns new tasks more effectively. Nonetheless, we expect SVR's reduced forgetting to yield larger benefits as more tasks are added. This trend is already visible in Figure 1(c), which plots average WER improvement over Fine-Tuning after each task. After tasks 4 and 5, the performance gap between SVR ($|\mathcal{M}| = 20$ and $|\mathcal{M}| = 1t$) and the large-memory baselines (ER, KD with $|\mathcal{M}| = 200$) has noticeably narrowed, suggesting that *SVR's advantage may increase with longer task sequences*.

Despite these strengths, SVR with small memory ($|\mathcal{M}| = 20$ or $|\mathcal{M}| = 1t$) does not significantly outperform FTA. As

previously discussed in Sections V-A and V-B, FTA achieves strong performance through *minimal forgetting*, but at the cost of limited learning capacity due to the fixed averaging weight of $1/t$. In this multilingual experiment, however, the impact of FTA's rigidity appears mitigated by the presence of *3.8M task-specific parameters* (Table II), which help each new language adapt without substantial updates to the shared model. Consequently, while SVR with $|\mathcal{M}| = 20$ and $|\mathcal{M}| = 1t$ achieve average WERs that are *1.2% and 0.3% better than FTA*, respectively, these differences are not statistically significant.

Increasing SVR's memory size to $|\mathcal{M}| = 50$ yields a notable improvement in this multilingual setting. The average WER is reduced by an additional *1.4%* compared to $|\mathcal{M}| = 20$, reflecting both improved task learning—particularly on the final task (SV)—and *33% less forgetting*. As shown in Figure 1(c), this improvement only becomes apparent from task 4 onward. At earlier stages, when the memory holds 10–20 utterances from prior tasks (e.g., RU and US), SVR with $|\mathcal{M}| = 20$ performs comparably to $|\mathcal{M}| = 50$. However, at task 4, when only 6–7 utterances per previous task remain in memory, and at task 5, with just 5 per task, the limited rehearsal data appears insufficient to maintain SVR's effectiveness.

Crucially, SVR with $|\mathcal{M}| = 50$ now *outperforms all baselines significantly, including FTA and ER with* $|\mathcal{M}| = 200$, except KD with $|\mathcal{M}| = 200$, which it *matches in performance despite requiring four times less memory*.

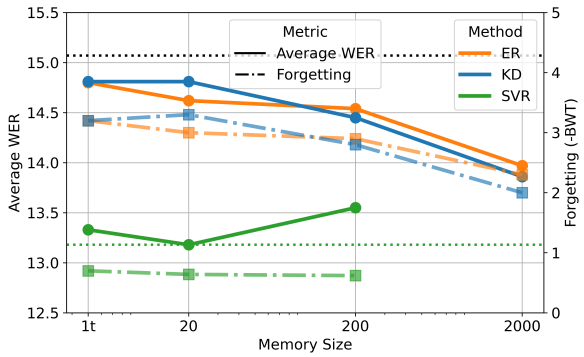
D. Experiment 4

Table IV presents the results for the OWSM-based experiment (Exp. 4). We make the following observations.

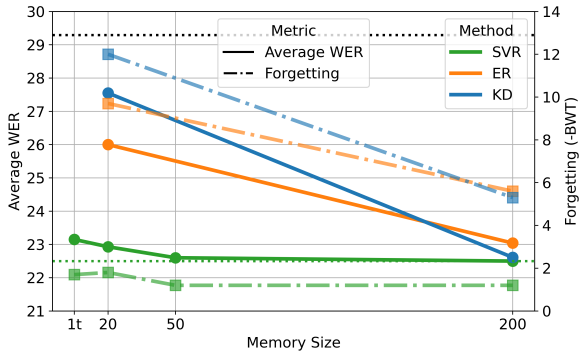
First, although initial task 2–CV/N was part of the OWSM v3.2 small pre–training corpus, its performance improves when learning task 3–CG/N, even under fine-tuning. This indicates *positive backward transfer* [3] from 3–CG/N to 2–CV/N. Since the degradation on 2–CV/N when adapting to 4–CG/V is smaller than this gain for all tested methods, the final performance on 2–CV/N remains higher than initially, thus increasing BWT. For FTA, and SVR with $|\mathcal{M}| = 50$ and $|\mathcal{M}| = 20$, the positive backward transfer on 2–CV/N outweighs cumulative forgetting, yielding a positive BWT value. After four tasks, SVR exploits this effect most strongly, reducing the WER on 2–CV/N to 8.4% (with $|\mathcal{M}| = 50$), *showing that SVR has the greatest ability to leverage positive backward transfer*.

For FTA the BWT is also highly positive, mainly because forgetting on 1–CV/E is limited; however, as in the other experiments, FTA struggles to learn new tasks. SVR forgets more on 1–CV/E than FTA (which could be alleviated by increasing the importance of CV/E in the memory), but overall *finds a better balance between retaining prior knowledge and learning new tasks*, achieving a lower Average WER and outperforming FTA on all Dutch tasks for every memory size.

Compared to ER and KD, SVR forgets substantially less and maintains a better stability–plasticity balance on the three Dutch tasks. With $|\mathcal{M}| = 50$, SVR outperforms KD with $|\mathcal{M}| = 200$ and matches ER with $|\mathcal{M}| = 200$, while SVR with $|\mathcal{M}| = 20$ or $|\mathcal{M}| = 1t$ outperforms KD and ER with



(a) Experiment 1. The x-axis is logarithmically scaled.



(b) Experiment 3

Fig. 2. Comparison of performance of rehearsal-based baselines (KD and ER) and our method (SVR) for different memory sizes $|\mathcal{M}|$. On the left, the y-axis shows the Average WER, where lower is better. On the right, it shows Forgetting, equal to -BWT, such that, here too, lower is better.

their smaller $|\mathcal{M}| = 20$ memory. SVR therefore *achieves competitive or superior performance while relying on four–to–five times fewer stored utterances*, and its performance degrades more gracefully than ER and KD as the memory decreases.

E. Impact of Memory size on Performance

Tables III and IV report results for KD, ER, and our method SVR under varying memory sizes. These results are further summarized in Figure 2(a) for Exp. 1 and Figure 2(b) for the multilingual Exp. 3. In both plots, the average WER (left) and Forgetting (right, equal to -BWT) are shown as a function of the memory size for SVR, ER, and KD. Several trends are apparent from these comparisons.

For Exp. 1 (Figure 2(a), Table III), we observe that the performance of ER and KD improves consistently as the memory size increases from $|\mathcal{M}| = 1t$ to $|\mathcal{M}| = 2000$. At small memory sizes ($|\mathcal{M}| = 1t$ and $|\mathcal{M}| = 20$), their performance is only marginally better than Fine-Tuning, but the gap widens with larger memory. However, even at $|\mathcal{M}| = 2000$, both KD and ER still *underperform SVR with much smaller memory sizes*—namely, $|\mathcal{M}| = 20$ and even $|\mathcal{M}| = 1t$; moreover, *KD's and ER's forgetting is, for all memory sizes, much higher than SVR's*. For ER, each memory increase leads to a statistically significant improvement in performance, except from

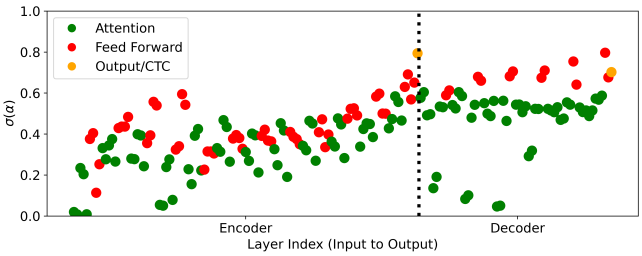
$|\mathcal{M}| = 20$ to $|\mathcal{M}| = 200$. For KD, too, increasing the memory size generally results in significant improvement, except from $|\mathcal{M}| = 1t$ to $|\mathcal{M}| = 20$, which both only marginally improve Fine-Tuning's performance. For SVR, increasing the memory size from $|\mathcal{M}| = 1t$ to $|\mathcal{M}| = 20$ does not result in a significant performance difference. Moreover, increasing the memory size from $|\mathcal{M}| = 20$ to $|\mathcal{M}| = 200$ even results in deterioration in terms of average WER, which can be attributed to reduced flexibility due to a more challenging regularization loss. Up until 3–AUS, SVR with $|\mathcal{M}| = 200$ outperforms SVR with $|\mathcal{M}| = 20$ with an average WER of 11.65 vs. 11.78 and a positive BWT of +0.1 vs. -0.2, i.e. less forgetting. However, from 4–IND onward, SVR with $|\mathcal{M}| = 200$ starts underperforming SVR with $|\mathcal{M}| = 20$. Although its forgetting is still less (BWT of -0.1 vs. -0.3), it learns the new task not as well (18.3 WER on IND vs. 17.5) due to a 16% lower mean $\sigma(\boldsymbol{\alpha})$ (averaged over all linear layers) of 0.32 vs. 0.38 for SVR with $|\mathcal{M}| = 20$. When learning task 5–SCO, the mean $\sigma(\boldsymbol{\alpha})$ is 0.20 for SVR with $|\mathcal{M}| = 200$ compared to 0.28 for SVR with $|\mathcal{M}| = 20$. Reducing the weight of the regularization loss (from Eq. 8) for SVR with $|\mathcal{M}| = 200$ would result in larger mean $\sigma(\boldsymbol{\alpha})$ and facilitate its learning of new tasks, thus improving its performance. However, while the regularization weight from Eq. 8 appears suboptimal for larger memory sizes, it is important to note that our method is designed for and tailored to scenarios with extremely limited memory. In conclusion, we find that our method is highly effective even under minimal memory constraints, and forgets significantly less than ER and KD—even when these baselines use memory sizes as large as $|\mathcal{M}| = 2000$.

For Exp. 3 (Figure 2(b), Table IV), similar trends emerge. Increasing the memory from $|\mathcal{M}| = 20$ to $|\mathcal{M}| = 200$ results in a significant improvement for both ER and KD. Nevertheless, *SVR with a memory of* $|\mathcal{M}| = 50$ *outperforms these baselines* – w.r.t. ER even a memory of $|\mathcal{M}| = 20$ suffices for SVR, despite their use of four to ten times more memory. Moreover, *SVR forgets between* 68% *and* 78% *less* (even with the smallest memory), and thus, as more tasks would be added and forgetting becomes more crucial, SVR's performance is expected to improve further in relation to ER and KD. For SVR, the increase from $|\mathcal{M}| = 1t$ to $|\mathcal{M}| = 20$ is not statistically significant, but the subsequent step to $|\mathcal{M}| = 50$ yields a significant performance gain. As discussed in Sec. V-C and shown in Figure 1(c), this improvement becomes apparent only from task 4 onward. For the earlier tasks, the smaller memory ($|\mathcal{M}| = 20$) remains sufficient to sustain SVR's effectiveness. Increasing $|\mathcal{M}|$ from $|\mathcal{M}| = 50$ to $|\mathcal{M}| = 200$ does not yield significant improvement.

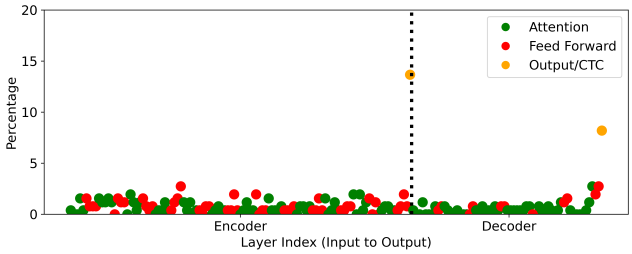
In Exp. 2 (Table III), memory reductions negatively impact performance for ER and KD. Specifically, reducing the memory from $|\mathcal{M}| = 200$ to $|\mathcal{M}| = 20$, and further to $|\mathcal{M}| = 1t$, leads to consistent and significant degradation. In contrast, *SVR maintains stable performance when reducing memory from* $|\mathcal{M}| = 20$ *to* $|\mathcal{M}| = 1t$, and even in the smallest-memory setting ($|\mathcal{M}| = 1t$), it significantly outperforms KD and ER configured with $|\mathcal{M}| = 20$.

Finally, in Exp. 4 (Table IV), the performance of SVR remains stable when reducing the memory size, although the degradation is significant. In the larger OWSM-based model with multilingual setting, SVR clearly benefits from a larger memory, yet—unlike ER and KD—its performance does not collapse when the memory is reduced.

In summary, across all three experiments, SVR exhibits strong robustness to reduced memory size and consistently delivers superior or comparable performance to competing methods, while forgetting much less; even when operating under substantially stricter memory constraints.



(a) Mean $\sigma(\boldsymbol{\alpha})$ per layer



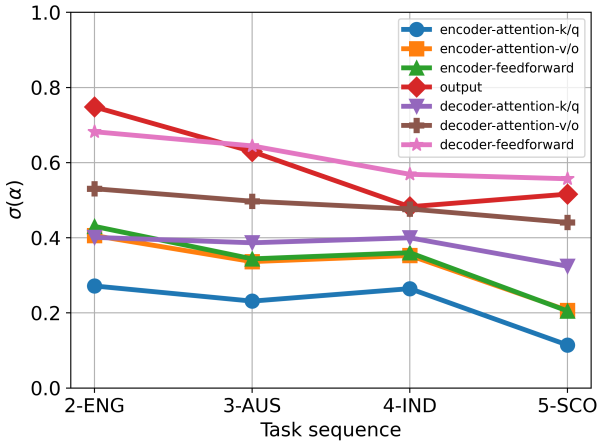
(b) Percentage of $\sigma(\boldsymbol{\alpha})$ for which $0.05 < \sigma(\alpha_i) < 0.95$

Fig. 3. Plot of $\sigma(\boldsymbol{\alpha})$ across layers (to which our method is applied) for SVR [$|\mathcal{M}| = 1$] after the first adaptation (1–US $\rightarrow$ 2–ENG) of Exp. 1. The layers are sorted from input (left) to output (right), with the dotted line indicating the end of encoder and start of decoder. Layers are colored based on layer type. Fig. 3(a) gives the mean $\sigma(\boldsymbol{\alpha})$ value per layer, while Fig. 3(b) gives the percentage of $\sigma(\boldsymbol{\alpha})$ which differ significantly (by more than 0.05) from 0 or 1, i.e. the percentage of $\sigma(\boldsymbol{\alpha})$ for which $0.05 < \sigma(\alpha_i) < 0.95$.

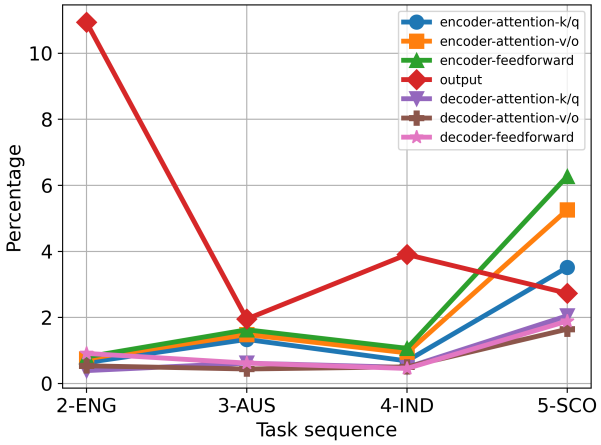
F. Analysis of $\sigma(\boldsymbol{\alpha})$

To gain deeper insight into the behavior of our method, we analyze the learned values of $\sigma(\boldsymbol{\alpha})$ across layers and tasks. Since $\sigma(\boldsymbol{\alpha})$ controls the extent to which each singular value-based rank-one update is applied, examining its distribution helps illuminate how the model balances retaining old knowledge and integrating new information. This analysis not only enhances understanding of our method's internal mechanisms but also reveals which parts of the model are more affected by inter-task conflicts and adaptation needs.

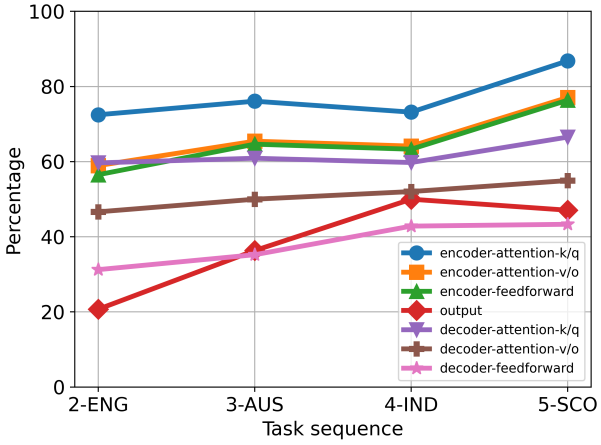
Figure 3 presents plots of $\sigma(\boldsymbol{\alpha})$ after the first adaptation step in Exp. 1 (i.e., from task 1–US to task 2–ENG), using SVR with $|\mathcal{M}| = 1t$. Specifically, Figure 3(a) shows the mean value of $\sigma(\boldsymbol{\alpha})$ for each layer to which our method is applied, while Figure 3(b) displays the percentage of parameters for which $0.05 < \sigma(\alpha_i) < 0.95$, indicating the extent of intermediate, non-binary gating behavior. The layers are ordered from input (left) to output (right), with the dotted line marking the boundary between the encoder and decoder components.



(a) Mean $\sigma(\boldsymbol{\alpha})$ per layer



(b) Percentage of $\sigma(\boldsymbol{\alpha})$ for which $0.05 < \sigma(\alpha_i) < 0.95$



(c) Percentage of $\sigma(\boldsymbol{\alpha})$ for which $\sigma(\alpha_i) < 0.05$

Fig. 4. Mean and percentage of intermediate and small values of $\sigma(\boldsymbol{\alpha})$ across layer groups after learning each task for SVR [$|\mathcal{M}|$=1] on Exp. 1. The layer groups are: (1) encoder key-query (k/q) attention; (2) encoder value-output (v/o) attention; (3) encoder feedforward; (4) output layers (CTC and decoder output); (5) decoder key-query (k/q) attention; (6) decoder value-output (v/o) attention; and (7) decoder feedforward. Fig. 4(a) shows the mean $\sigma(\boldsymbol{\alpha})$ per group after learning each task. Fig. 4(b) shows the percentage of parameters in $\sigma(\boldsymbol{\alpha})$ for which $0.05 < \sigma(\alpha_i) < 0.95$. Fig. 4(c) shows the percentage for which $\sigma(\alpha_i) < 0.05$.

From Figure 3(a), we observe that the CTC output layer exhibits the highest mean $\sigma(\boldsymbol{\alpha})$ values, approaching 0.80. This suggests that rank-one updates in this layer are largely retained during adaptation, contributing significantly to learning the new task without severely impacting performance on the old task. A clear trend is also visible across the network: deeper encoder and decoder layers generally show higher $\sigma(\boldsymbol{\alpha})$ values, while earlier layers tend to suppress more updates. Within the decoder, feedforward submodules exhibit higher $\sigma(\boldsymbol{\alpha})$ values than their attention-related counterparts, indicating that the feedforward layers play a more prominent role in accommodating the new task.

Turning to Figure 3(b), we find that most $\sigma(\boldsymbol{\alpha})$ values are concentrated near 0 or 1, with relatively few falling in the intermediate range. This binary-like behavior suggests a strong preference for either fully applying or completely discarding rank-one updates. The proportion of intermediate values ($0.05 < \sigma(\alpha_i) < 0.95$) is generally low, with output layers showing the highest ratio, around 13% maximum. For all other layers, this proportion remains below 3%. These findings contrast with weight averaging [26], which uniformly scales updates (e.g., by 0.5 or $1/t$); in our method, the model instead learns to make more decisive gating choices, favoring either full integration or full rejection of the learned updates.

Figure 4 extends this analysis by showing how $\sigma(\boldsymbol{\alpha})$ evolves across tasks. In this figure, layers are grouped into seven functional categories: (1) encoder key-query attention; (2) encoder value-output attention; (3) encoder feedforward; (4) output layers (CTC and decoder output); (5) decoder key-query attention; (6) decoder value-output attention; (7) decoder feedforward layers. We split the attention block into key-query and value-output, hypothesizing that the value and output weight matrices in the attention function analogously to feedforward layers, and therefore expect their $\sigma(\boldsymbol{\alpha})$ behavior to resemble that of the feedforward layers rather than that of the query and key weight matrices.

Figure 4(a) shows the mean value of $\sigma(\boldsymbol{\alpha})$ per group after each task in Exp. 1. The highest mean values are consistently found in the output, decoder feedforward and decoder value-output attention layers, suggesting that updates in these layers are most beneficial for adapting to new tasks while preserving performance on past ones. Conversely, the encoder shows the lowest mean values, indicating either that updates in these layers conflict more with previously learned knowledge or that they are less important for learning new tasks. Across tasks, the mean $\sigma(\boldsymbol{\alpha})$ values decrease slightly, but remain relatively high compared to weight averaging, which would hyperbolically decrease adaptation weights as $1/t$.

Figure 4(b) tracks the ratio of $\sigma(\boldsymbol{\alpha})$ values in the intermediate range, that is, values of $\sigma(\boldsymbol{\alpha})$ for which $0.05 < \sigma(\alpha_i) < 0.95$. These remain consistently low across all layer groups and tasks, in most cases below 4%, except for the output layers after the first adaptation ($\approx 11\%$), and the encoder feedforward and value-output attention layers at the last adaptation ($\approx 6\%$). This supports the earlier observation that the model tends to make sharp gating decisions rather than compromise between preserving and discarding updates.

Finally, Figure 4(c) illustrates the proportion of $\sigma(\boldsymbol{\alpha})$ values

below 0.05, effectively indicating how many updates are nearly fully suppressed. This ratio increases slightly over time for all layer groups. Notably, encoder layers and decoder key-value attention layers show suppression rates above 50%, reinforcing the notion that these layers are more susceptible to inter-task interference or less useful for new-task adaptation. In contrast, the output, decoder value-output attention and decoder feedforward layers initially suppress fewer updates (well below 40%), though this ratio gradually increases as more tasks are learned.

In summary, we observe that the model typically exhibits a binary behavior in its use of rank-one updates—either fully suppressing or fully accepting them. Suppression is most pronounced in the encoder and decoder key-query attention layers, while acceptance is highest in the output layers, decoder value-output attention and decoder feedforward layers. This selective behavior appears to stem from two factors. First, high suppression rates may be observed in layers where task interference is substantial—here, rank-one updates that could benefit the new task are nonetheless suppressed due to their detrimental impact on performance for previous tasks. Second, suppression may also occur in more task-agnostic layers, where adaptation offers only marginal gains for the new task while risking (slight) degradation for the old ones. In both cases, the model attempts to find a balance between plasticity (adaptation to the new task) and stability (not forgetting of old tasks) by filtering out updates that harm old tasks more than they improve the performance on the new task.

G. Ablation Study

Table V presents an ablation study of our method, focusing on Exp. 1 and Exp. 2, providing the following insights.

1) Stage 2 loss: When the weight of the regularization term in the stage 2 loss (see Eq. 8) is not scaled with $t-1$, *forgetting increases* and overall performance deteriorates, highlighting the importance of scaling regularization with the number of old tasks. If, in addition, the KD loss is omitted and only the cross-entropy loss is used in Eq. 8, *forgetting further increases* and performance degrades even more; replacing the cross-entropy loss with KD alone does not improve performance either, emphasizing the benefit of combining ER (through the cross-entropy loss) with KD in our formulation.

2) Combining ER and KD in standard rehearsal: Starting from ER with $|\mathcal{M}| = 200$, we observe that neither replacing the cross-entropy loss with knowledge distillation (as in KD), nor combining cross-entropy and distillation losses (as in [18]), closes the performance gap with SVR.

3) Remaining parameters: The treatment of the remaining 9.3% model parameters (from Exp. 1-3)—those not part of the $\boldsymbol{W}$ matrices of linear layers—also has a significant effect. If these parameters are not averaged during the parameter merging step (Lines 13–17 in Algorithm 1), but instead the old values are retained (i.e., replacing Line 15 with $\boldsymbol{p} \leftarrow \boldsymbol{p}_{t-1}$), forgetting is more than halved. However, this comes at the cost of reduced plasticity, ultimately leading to worse average performance. Conversely, using the new parameters (i.e., $\boldsymbol{p} \leftarrow \tilde{\boldsymbol{p}}_t$) increases the ability to learn the current task but *more than doubles forgetting*, resulting in even worse overall performance. These observations suggest that even the remaining 9.3% of parameters significantly affect SVR's balance between learning and forgetting, and extending SVR to include other layer types could yield further improvements.

TABLE V
ABLATION STUDY OF OUR METHOD. ROWS MARKED WITH "+" INDICATE CUMULATIVE CHANGES TO THE CONFIGURATION LISTED DIRECTLY ABOVE, WHILE ROWS MARKED WITH "→" REPRESENT ALTERNATIVE VARIANTS OF THE REFERENCE METHOD.

Model	Average	
	WER↓	BWT↑
Exp. 1. Full adaptation 1–US → ⋯ → 5–SCO		
SVR [$\lvert\mathcal{M}\rvert = 20$]	13.18	-0.6
+ Memory loss not scaled with $t-1$	13.54[a]	-1.4
→ No $\mathcal{L}_{\text{kd}}$ on memory, only $\mathcal{L}_{\text{ce}}$ (Eq. 8)	13.89[b]	-2.0
→ No $\mathcal{L}_{\text{ce}}$ on memory, only $\mathcal{L}_{\text{kd}}$ (Eq. 8)	13.83[c]	-1.5
ER [$\lvert\mathcal{M}\rvert = 200$]	14.54	-2.9
→ Use KD instead of ER	14.45[d]	-2.8
→ Combine KD and ER	14.47[d]	-2.7
SVR [$\lvert\mathcal{M}\rvert = 1t$]	13.33	-0.7
→ Rest of model not averaged, old params used	13.85[b]	-0.3
→ Rest of model not averaged, new params used	13.84[b]	-1.5
Exp. 2. First adaptation 1–LIB → 2–GB/M		
SVR [$\lvert\mathcal{M}\rvert = 1$]	6.46	-0.8
→ $\sigma(\alpha_i)$ not restricted to $[0, 1]$	6.62[c]	-1.8
Exp. 2. Fourth adaptation 1–LIB → ⋯ → 4–IN/U		
FTA: fixed averaging weight $\eta = 1/t$	8.77	-0.9
+ Learn η using ER with $\lvert\mathcal{M}\rvert = 1t$	8.75[g]	-3.9
+ Learn η using our loss (Eq. 8) instead of ER	8.61[f]	-1.3
+ Learn scalar $\sigma(\alpha)$ per linear layer instead of η	7.88[e]	-1.4
+ SVR: learn $\boldsymbol{\alpha} \in \mathbb{R}^k$ per linear layer	7.41[e]	-1.3

[a] Significant deterioration (level ***) w.r.t. the method above.
[b, c, d] Deterioration w.r.t. the reference method is significant with level *** for *b* and level * for *c*, and non-significant (*ns*) for *d*.
[e, f, g] Improvement w.r.t the method above is significant with level *** for *e* and with level ** for *f*, and non-significant (*ns*) for *g*.

4) Restricting gating vectors.: Removing the constraint that $\sigma(\boldsymbol{\alpha})$ lies within $[0, 1]$ gives the model more freedom to learn new tasks and improves short-term learning. However, it *more than doubles forgetting*, due to overfitting on the memory. This confirms that constraining $\sigma(\boldsymbol{\alpha})$ is essential for maintaining generalization within the small memory constraint.

5) From FTA to SVR: Compared to FTA, SVR learns a gating vector $\sigma(\boldsymbol{\alpha})$ per linear layer, corresponding to one gating scalar per rank-one parameter update, rather than fixing a single model-wise weight $\eta = 1/t$. By incrementally adding the components of SVR [$\lvert\mathcal{M}\rvert = 1t$] to FTA, we analyze the contribution of each improvement. We evaluate this on the fourth adaptation of Exp. 2, which—as indicated by the large performance gap between Fine-Tuning and the CL methods in Fig. 1(b)—is particularly challenging due to (a) the high zero-shot WER on 4–IN/U and (b) the latter's high interference with previous tasks. Results show that learning η instead of fixing it significantly improves performance, though only when using our loss from Eq. 8. In this case, learning η closes 11% of the gap between FTA and SVR. Replacing the model-wise averaging weight η with a (linear) layer-wise scalar $\sigma(\alpha)$

provides additional flexibility—allowing some linear layers to accept updates while others suppress them—closing an additional 54% of the performance gap between FTA and SVR. Finally, learning a gating vector $\sigma(\boldsymbol{\alpha})$ per linear layer rather than a single scalar allows selective control over individual update directions within each layer, yielding a further 35% reduction of the gap and matching SVR's full performance.

VI. Discussion

Our results demonstrate that *Singular Value-based Rehearsal (SVR) offers a strong and versatile solution for continual learning in ASR, particularly under tight memory constraints*. Across all three experiments—two monolingual and one multilingual—*SVR consistently achieved the best performance* among all tested methods. While it required a slightly larger memory size in the multilingual setting to match its monolingual performance, it remained significantly more memory-efficient than the rehearsal-based baselines.

One of *SVR's core strengths lies in its ability to balance stability and plasticity*: it forgets substantially less than ER and KD while learning new tasks more effectively than methods such as FTA. his performance is enabled by a structured and interpretable adaptation mechanism: SVR learns for each rank-one parameter update whether to fully suppress or accept it, rather than partially weighting it, as is done in weight averaging [26]. An analysis of this binary behavior gives insights into where adaptation is most beneficial to the new task or most harmful to the old tasks.

Despite its strengths, there are avenues for improving SVR.

In the multilingual setting, a larger memory was required to reach the same level of performance relative to the baselines as in the monolingual setups. This suggests that additional regularization on $\sigma(\boldsymbol{\alpha})$ values may help better constrain the model's flexibility when few examples per task are available.

Another limitation is that SVR is currently applied only to the weight matrices of linear layers—which covers the majority of model parameters, but not all. Our ablation study (Sec. V-G) showed that the remaining 9.3% of parameters (from Exp. 1-3) can significantly affect forgetting and overall performance. Extending SVR to these components could further improve results, particularly in more heterogeneous or modern architectures. Alternatively, for these components a single scalar $\sigma(\alpha_{\text{avg}}) \in [0, 1]$ could be trained to learn the weight of the averaging automatically and replace Line 15 in Algorithm 1 with $\boldsymbol{p} \leftarrow (1 - \sigma(\alpha_{\text{avg}}))\boldsymbol{p}_{t-1} + \sigma(\alpha_{\text{avg}})\tilde{\boldsymbol{p}}_t$.

In addition, while SVR effectively exploits the available memory, it currently uses uniform sampling for memory population. A central challenge for any rehearsal-based continual learning method lies in the dual task of selecting and exploiting memory samples effectively. Given strict memory constraints, such methods must not only leverage the limited samples to their fullest potential during training – in which SVR succeeds – but also ensure that the most representative or informative examples are selected for storage. We currently rely on uniform sampling to populate the memory, without any mechanism to prioritize sample selection. Future work could therefore improve our approach by investigating smarter sampling, which may further enhance performance.

Moreover, combining our method with parameter-efficient fine-tuning (PEFT) methods such as LoRA [30], [32] and other PEFT-techniques [44], [58] could improve both scalability and performance. Similarly, integrating complementary continual learning strategies—such as regularization- approaches—into the initial stage of our method may further reduce forgetting, especially in more adversarial task sequences.

Although our method was developed and evaluated in the context of end-to-end ASR, it is inherently model-agnostic and not specific to the speech domain. Applying it to other areas such as natural language processing or computer vision—where continual learning challenges also arise—could help demonstrate its broader utility and robustness.

VII. Conclusion

In this work, we introduced Singular Value-based Rehearsal (SVR), a memory-efficient rehearsal method for continual learning (CL) in ASR. SVR is designed to operate effectively even under extreme memory constraints—a common limitation in real-world applications where storage, privacy, or model accessibility pose significant challenges.

By decomposing the linear weight updates from task-specific fine-tuning into their singular value components, and training a gating vector to modulate these updates based on a small rehearsal set, SVR offers a principled mechanism for balancing retention and adaptation. Our extensive experiments across monolingual and multilingual ASR settings show that SVR consistently outperforms existing rehearsal-based and regularization-based methods, achieving state-of-the-art performance even under minimal memory constraints.

Extensive ablation studies further underscore the importance of each component in SVR—ranging from the use of knowledge distillation, the averaging of non-linear layer weights, to the constrained gating on singular values. Moreover, the method's interpretability reveals a distinctive binary gating behavior, selectively retaining only updates that are beneficial for the new task without sacrificing performance on past ones.

In summary, SVR delivers a strong trade-off between plasticity and stability in continual ASR learning. It is scalable, interpretable, and operates effectively under strict memory limitations. While we have focused on ASR, the method is broadly applicable to other domains where continual learning and efficient adaptation are required. Future work may expand SVR to other parameter types, integrate smarter memory selection strategies, or explore its potential within large-scale models and non-speech modalities.

Acknowledgments

Research supported by Research Foundation Flanders (FWO) under grant S004923N of the SBO programme.

References

[1] M. McCloskey *et al.*, "Catastrophic interference in connectionist networks: The sequential learning problem," vol. 24 of *Psychology of Learning and Motivation*, pp. 109–165, 1989.

[2] S.-A. Rebuffi *et al.*, "icarl: Incremental classifier and representation learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017.

[3] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6470–6479, 2017.

[4] S. Vander Eeckt and H. Van hamme, "Continual learning for monolingual end-to-end automatic speech recognition," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022.

[5] H.-J. Chang *et al.*, "Towards Lifelong Learning of End-to-End ASR," in *Proc. Interspeech 2021*, pp. 2551–2555, 2021.

[6] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," 2022.

[7] S. Karita *et al.*, "Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration," in *Proc. Interspeech 2019*.

[8] M. Biesialska *et al.*, "Continual lifelong learning in natural language processing: A survey," in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6523–6541, Dec. 2020.

[9] M. Delange *et al.*, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2021.

[10] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[11] R. Aljundi *et al.*, "Memory aware synapses: Learning what (not) to forget," in *Computer Vision – ECCV 2018*, pp. 144–161, Springer International Publishing, 2018.

[12] S. Lee *et al.*, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems 30: Annual Conference NeurIPS 2017*, pp. 4652–4662, 2017.

[13] Z. Li and D. Hoiem, "Learning without forgetting," in *Computer Vision – ECCV 2016*, pp. 614–629, 2016.

[14] J. Serra *et al.*, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. 35th International Conference on Machine Learning*, vol. 80, pp. 4548–4557, 10–15 Jul 2018.

[15] J. Schwarz *et al.*, "Progress & compress: A scalable framework for continual learning," in *Proceedings of the 35th International Conference on Machine Learning, 2018*, vol. 80, pp. 4535–4544, PMLR, 2018.

[16] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *ICLR*, 2019.

[17] D. Rolnick *et al.*, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[18] P. Buzzega *et al.*, "Dark experience for general continual learning: a strong, simple baseline," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 15920–15930, 2020.

[19] J. Knoblauch, H. Husain, and T. Diethe, "Optimal continual learning has perfect memory and is np-hard," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[20] L. Wang *et al.*, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.

[21] E. Verwimp, D. Lange, *et al.*, "Rehearsal revealed: The limits and merits of revisiting samples in continual learning," in *2021 IEEE/CVF International Conference on Computer Vision*, pp. 9365–9374, 2021.

[22] S. Vander Eeckt and H. Van hamme, "Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition," in *ICASSP 2023*, 2023.

[23] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

[24] M. Sustek, S. Sadhu, and H. Hermansky, "Dealing with Unknowns in Continual Learning for End-to-end Automatic Speech Recognition," in *Proc. Interspeech 2022*, pp. 1046–1050, 2022.

[25] A. Diwan, C.-F. Yeh, W.-N. Hsu, P. Tomasello, E. Choi, D. Harwath, and A. Mohamed, "Continual learning for on-device speech recognition using disentangled conformers," in *ICASSP 2023*, pp. 1–5, 2023.

[26] S. Vander Eeckt and H. Van hamme, "Weight averaging: A simple yet effective method to overcome catastrophic forgetting in automatic speech recognition," in *ICASSP 2023*, 2023.

[27] Y. Takashima *et al.*, "Updating only encoders prevents catastrophic forgetting of end-to-end ASR models," in *Proc. Interspeech 2022*, pp. 2218–2222, ISCA, 2022.

[28] Z. Wang, F. Hou, and R. Wang, "Clrl-tuning: A novel continual learning approach for automatic speech recognition," in *Interspeech 2023*, pp. 1279–1283, 2023.

[29] C. Y. Kwok, J. Q. Yip, and E. S. Chng, "Continual learning optimizations for auto-regressive decoder of multilingual asr systems," in *Interspeech 2024*, pp. 1225–1229, 2024.

[30] T. Xu *et al.*, "Towards rehearsal-free multilingual asr: A lora-based case study on whisper," in *Interspeech 2024*, pp. 2534–2538, 2024.

[31] C. Y. Kwok *et al.*, "A two-stage lora strategy for expanding language capabilities in multilingual asr models," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 33, pp. 2576–2590, 2025.

[32] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[33] G. Chennupati *et al.*, "Ilasr: Privacy-preserving incremental learning for automatic speech recognition at production scale," in *Proc. ACM SIGKDD Conf. KDD*, p. 2780–2788, 2022.

[34] M. Yang *et al.*, "Online continual learning of end-to-end speech recognition models," in *Proc. Interspeech 2022*, 2022.

[35] S. Vander Eeckt and H. Van hamme, "Rehearsal-free online continual learning for automatic speech recognition," in *Proc. Interspeech*, 2023.

[36] S. Vander Eeckt and H. Van hamme, "Unsupervised online continual learning for automatic speech recognition," in *Interspeech 2024*, pp. 2845–2849, 2024.

[37] Q. Zhang *et al.*, "Adaptive budget allocation for parameter-efficient fine-tuning," in *11th International Conference on Learning Representations*, 2023.

[38] F. Zhang *et al.*, "Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning," 2023.

[39] D. J. Kopiczko *et al.*, "VeRA: Vector-based random matrix adaptation," in *12th International Conference on Learning Representations*, 2024.

[40] M. Fawi, "CURLoRA: Leveraging CUR Matrix Decomposition for Stable LLM Continual Fine-Tuning and Catastrophic Forgetting Mitigation," July 2024.

[41] M. W. Mahoney and P. Drineas, "Cur matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 3, pp. 697–702, 2009.

[42] Y. Sun *et al.*, "Singular value fine-tuning: Few-shot segmentation requires few-parameters fine-tuning," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 37484–37496, 2022.

[43] V. Lingam *et al.*, "Svft: Parameter-efficient fine-tuning with singular vectors," in *Advances in Neural Information Processing Systems*, vol. 37, pp. 41425–41446, 2024.

[44] Y. Yang *et al.*, "CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning," in *Proc. NeurIPS 2024*, 2024.

[45] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NeurIPS*, vol. abs/1503.02531, 2014.

[46] S. Watanabe *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, pp. 2207–2211, 2018.

[47] R. Ardila *et al.*, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 2020.

[48] V. Panayotov *et al.*, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[49] A. Mathur *et al.*, "Libri-adapt: a new speech dataset for unsupervised domain adaptation," *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7439–7443, 2020.

[50] Y. Peng *et al.*, "Owsm v3.1: Better and faster open whisper-style speech models based on e-branchformer," in *Interspeech*, pp. 352–356, 2024.

[51] N. Oostdijk, "The spoken dutch corpus: Overview and first evaluation," *Proceedings of LREC-2000, Athens*, vol. 2, 01 2000.

[52] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[53] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[54] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, 2018.

[55] K. Kim *et al.*, "E-branchformer: Branchformer with enhanced merging for speech recognition," in *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 84–91, 2023.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[57] H. Strik, C. Cucchiarini, and J. M. Kessens, "Comparing the recognition performance of csrs: in search of an adequate metric and statistical significance test," in *INTERSPEECH*, 2000.

[58] P. Wang, S. Watanabe, and H. Van hamme, "Ssvd: Structured svd for parameter-efficient fine-tuning and benchmarking under domain shift in asr," in *Proceedings of the IEEE ASRU Workshop*, 2025.