TokenVerse++: Towards Flexible Multitask Learning with Dynamic Task Activation

Shashi Kumar[1,2,♣], Srikanth Madikeri[3], Esaú Villatoro-Tello[1], Sergio Burdisso[1], Pradeep Rangappa[1], Andrés Carofilis[1], Petr Motlicek[1,4], Karthik Pandia[5], Shankar Venkatesan[5], Kadri Hacioğlu[5] and Andreas Stolcke[5]

[1]*Idiap Research Institute, Switzerland*; [2]*EPFL, Switzerland*; [3]*University of Zurich, Switzerland*;
[4]*Brno University of Technology, Czech Republic*; [5]*Uniphore, U.S.A. & India*
[♣]Corresponding author: shashi.kumar@idiap.ch

***Abstract*—Token-based multitasking frameworks like TokenVerse require all training utterances to have labels for all tasks, hindering their ability to leverage partially annotated datasets and scale effectively. We propose TokenVerse++, which introduces learnable vectors in the acoustic embedding space of the XLSR-Transducer ASR model for dynamic task activation. This core mechanism enables training with utterances labeled for only a subset of tasks, a key advantage over TokenVerse. We demonstrate this by successfully integrating a dataset with partial labels, specifically for ASR and an additional task, language identification, improving overall performance. TokenVerse++ achieves results on par with or exceeding TokenVerse across multiple tasks, establishing it as a more practical multitask alternative without sacrificing ASR performance.**

***Index Terms*—multitask training, speech recognition, speaker change detection, named entity recognition, language identification, XLSR-Transducer.**

I. Introduction

Multitask learning enhances automatic speech recognition (ASR) by enabling multiple tasks in a single inference step, improving efficiency and functionality. Recent token-based approaches [1]–[8] retain conventional ASR architectures while avoiding multi-step pipelines that require separate NLP models for tasks such as named entity recognition [9] or dialogue modeling [10], [11]. Unlike large language models [12], [13] and speechLLMs [14]–[16] that risk hallucinations, token-based multitask models intersperse task-specific tokens in the hypothesis, preserving the integrity of ASR outputs while augmenting them with additional annotations.

Recently, `TokenVerse` [1] successfully unified four tasks within the XLSR-Transducer [17] ASR model. However, its fundamental requirement for exhaustive labeling of all tasks for every training utterance creates a significant bottleneck. This limits its ability to scale by leveraging vast, existing speech corpora that are often only partially annotated, for instance, with ASR transcripts but without named entities or speaker change labels. This makes the addition of new tasks prohibitively expensive, requiring re-annotation of all data. Although PromptASR [18] partially addresses aspects of task flexibility, it can introduce unnecessary complexity for managing diverse data sources. LLM-inspired methods like prefix-tuning [19] and prompt-tuning [20] are not directly applicable to ASR models.

Here, we build on the `TokenVerse` framework and introduce `TokenVerse++`. It overcomes the task annotation bottleneck through dynamic task activation during training, achieved by learning task-dependent acoustic embeddings within the XLSR-Transducer. It can utilize datasets annotated for only a subset of tasks, ensuring applicability across varied datasets. Experimental results show that `TokenVerse++` matches or outperforms the original `TokenVerse` while offering greater flexibility, demonstrating its potential as an efficient solution for token-based multitask ASR.

This work was supported by the Idiap & Uniphore collaboration project. Part of the work was also supported by EU Horizon 2020 project ELOQUENCE (grant number 101070558).

II. From TokenVerse to TokenVerse++

A. TokenVerse: Foundations and Limitations

Token-based multitasking has shown success in `TokenVerse` [1], even when scaled to four tasks: ASR (the primary task), speaker change detection (SCD), named entity recognition (NER), and endpointing, as illustrated in Figure 1(a). However, training the XLSR-Transducer [17] and its transducer loss function [21], [22] as the base ASR model requires labels for all tasks for every training utterance. This represents a significant limitation, as scaling the number of tasks becomes increasingly difficult, since obtaining gold annotations for multiple tasks for all audio data is challenging. Additionally, `TokenVerse` cannot leverage the large number of existing datasets for tasks like ASR, further limiting its applicability.

B. Proposed Approach: TokenVerse++

The aim of `TokenVerse++` is to learn a task-aware acoustic embedding space within the `TokenVerse` model (see Figure 1). This is achieved by introducing a mechanism to modulate the original acoustic embeddings using a set of learnable vectors, whose composition based on active tasks is detailed in Section II-C. Let $\mathbf{X} \in \mathbb{R}^{T \times d}$ denote the acoustic embeddings from the base ASR model, where T is the number of time steps and d the embedding dimension. We define a general function $f : \mathbb{R}^{T \times d} \times \mathbb{R}^{d} \rightarrow \mathbb{R}^{T \times d}$ that modifies these

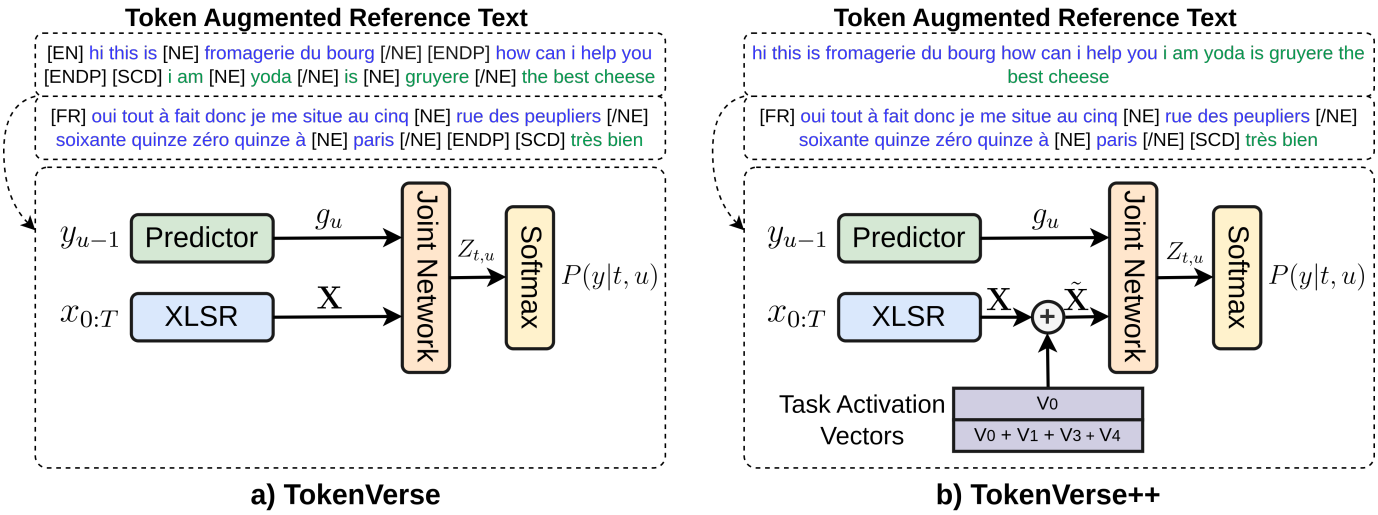

Fig. 1. Comparison of (a) the original TokenVerse model and (b) the proposed TokenVerse++ framework. TokenVerse (a) requires all training utterances to be fully annotated for all tasks. TokenVerse++ (b) introduces dynamic task activation by adding a learnable task-specific vector $\mathbf{v}$ (e.g., formed by summing vectors for active tasks like ASR ($\mathbf{v_0}$) and auxiliary tasks ($\mathbf{v_1}$: SCD, $\mathbf{v_3}$: NER, $\mathbf{v_4}$: LID)) to the acoustic embeddings from the XLSR encoder. This enables flexible training with partially annotated utterances, allowing different task combinations per input.

embeddings using a dynamically formed, learnable vector $\mathbf{v} \in \mathbb{R}^d$ representing the active task(s):

$$\tilde{\mathbf{X}} = f(\mathbf{X}, \mathbf{v}). \tag{1}$$

In our approach, we hypothesize that a simple element-wise addition is sufficient to effectively condition the embeddings:

$$\tilde{\mathbf{X}} = \mathbf{X} + \mathbf{v}. \tag{2}$$

This formulation allows for a straightforward incorporation of task-specific information. We posit that this learned vector $\mathbf{v}$ acts as a task-specific bias or translation in the acoustic embedding space. By shifting the original embeddings $\mathbf{X}$, the model can project them into a region of the embedding space that is more conducive to the currently active set of tasks. Thus, the addition of $\mathbf{v}$ effectively creates a task-conditioned view $\tilde{\mathbf{X}}$ of the acoustic features, enabling the subsequent layers of the XLSR-Transducer to better differentiate and specialize for the activated tasks.

Specifically, for the XLSR-Transducer ASR model used in `TokenVerse`, we investigate three configurations for incorporating this task activation into the acoustic embedding space: (i) after the feature encoder, (ii) after the full XLSR [23] encoder, and (iii) a combination of both.

C. Design Considerations in `TokenVerse++`

Assume there are K tasks apart from a designated always-active, primary task (ASR in our setup). Auxiliary tasks can be activated independently alongside this primary task. Let $\mathcal{C}$ represent the set of all possible combinations of auxiliary tasks. This set also includes the scenario where no auxiliary tasks are active, meaning only the primary task is engaged.

We first propose an approach where each unique task combination $\mathbf{c}_i \in \mathcal{C}$ is associated with its own distinct learnable vector. Here, $\mathbf{c}_i$ denotes the set of active tasks for a given utterance. To activate a specific subset of tasks $\mathbf{c}_i$ during training or inference, its corresponding vector is utilized. For training, a combination $\mathbf{c}_i$ is selected for each utterance, uniformly sampled from all valid combinations $\mathcal{C}$, or determined by the available labels for an utterance. Observe that the number of combinations ($|\mathcal{C}|$), and consequently the number of learnable vectors, grows exponentially with K, which introduces a large number of trainable parameters as K becomes larger.

Building on this, we propose a more scalable solution. In this second approach, each individual task k, including the primary task, is associated with a single learnable vector $\mathbf{v}_k \in \mathbb{R}^d$, where d is the dimension of the acoustic embedding, rather than having a separate learnable vector for each combination in the set $\mathcal{C}$. The tasks to be activated are still selected from $\mathcal{C}$ with equal probability. When a specific task combination $\mathbf{c}_i$ needs to be activated, the final activation vector $\mathbf{v}$ is obtained by summing the vectors corresponding to all tasks present in $\mathbf{c}_i$, giving

$$\tilde{\mathbf{X}} = \mathbf{X} + \sum_{k \in \mathbf{c}_i} \mathbf{v}_k \tag{3}$$

where each $\mathbf{v}_k \in \mathbb{R}^d$ is the learnable vector for task k; these vectors are initialized randomly.[1] With this method, the number of learnable vectors scales linearly with the total number of tasks. During training, the selection of $\mathbf{c}_i$ for an utterance typically corresponds to the set of tasks for which labels are available for that particular utterance.

We explore both methods to better understand the trade-off between representational capacity and scalability. The second approach, using a single vector per task, significantly improves scalability. However, it may reduce task representation

[1]We also tried initializing via approximate determinantal point process (DPP) [24] sampling on a high-dimensional hypersphere, which showed no improvement over random initialization.

granularity due to the many-to-one mapping resulting from the summation. In contrast, the first approach, utilizing a unique vector per combination, may offer finer-grained task representations but at the cost of a potentially large number of trainable parameters. This trade-off is analyzed in Section IV.

During training, as illustrated in Figure 1(b), a task combination $\mathbf{c}_i$ is determined for each utterance, typically based on its available annotations. This system allows different utterances within a batch to activate different task sets. These utterances may originate from multiple datasets with varying annotations. This flexibility is a key advantage of `TokenVerse++`, enabling the model to leverage partially labeled data effectively.

During inference, any combination of tasks $\mathbf{c}_i$ can be activated by selecting or constructing the appropriate task activation vector $\mathbf{v}$. The model was trained to handle various task combinations at test time, allowing it to work in different application scenarios.

III. Experimental Setup

Our experimental design serves two primary objectives: first, to establish a direct performance comparison between `TokenVerse++` and the original `TokenVerse` framework; and second, to show the enhanced flexibility of `TokenVerse++` through the incorporation of partially annotated datasets and an additional task.

A. Baseline Comparison with TokenVerse

To ensure a fair and direct comparison with prior work [1], our initial experiments replicate the setup of the original `TokenVerse`. This involves focusing on the same four tasks: ASR as the primary task, alongside speaker change detection (SCD), endpointing, and named entity recognition (NER). For this baseline comparison, we use the English portion of the DefinedAI[2] dataset, consistent with [1], which comprises contact center conversations between agents and customers and contains complete annotations for all four tasks. The underlying ASR model remains the XLSR-Transducer [17], as in `TokenVerse`. This comparative setup allows for a clear assessment of our proposed dynamic task activation mechanism's impact when all tasks are notionally active during inference, mirroring the `TokenVerse` evaluation conditions.

B. Leveraging Partially Annotated Datasets

To evaluate the core capabilities of `TokenVerse++`, we designed a multilingual, five-task experimental scenario, expanding upon the baseline by introducing language identification (LID) as a fifth task. For LID, language-specific tokens (e.g., `[EN]`, `[FR]`) are prepended to the reference transcripts, following the multitask data preparation from [1].

The training data for this expanded five-task scenario combines fully and partially annotated corpora to test the ability to leverage diverse data sources. We utilize the multilingual DefinedAI dataset, which provides comprehensive annotations for all five tasks across German (de), Spanish (es), French (fr), and English (en) languages. Crucially, to evaluate whether `TokenVerse++` is able to utilize datasets with incomplete task annotations, we integrate a randomly chosen 100-hour subset of the CommonVoice (CV) corpus [25] for each of these four languages. The size of this specific subset was chosen to provide enough partially labeled data to demonstrate the benefits of our approach, while also ensuring manageable computational demands for training and experimentation across multiple languages and tasks. Within our five-task framework, the CV dataset provides labels *only* for ASR and LID, thereby lacking annotations for SCD, endpointing, and NER. As described in Section II, `TokenVerse++` processes such mixed-annotation data by activating only those tasks for which labels are available for any given utterance. In contrast, the original `TokenVerse` would be unable to utilize these CV utterances due to its strict requirement for complete annotations across all defined tasks.

[2]https://www.defined.ai/

TABLE I: Dataset statistics with token metadata for each subset of the multilingual multitask DefinedAI datasets. Note that the token count for LID equals the number of utterances. In **Named Entities (NEs) metadata**, #NE indicates the total count of named entity occurrences, #uniq represents the number of unique entities among them, and %Common shows the percentage of named entities shared with the training set for each language.

Datasets metadata			**Token metadata [%]**			**NEs metadata**		
subset	#utt/word	dur [h]	`[SCD]`	`[NE]`	`[ENDP]`	#NE	#uniq	%Common
English (en) dataset								
train	10k/359k	40	1.9	3.6	2.1	6.5k	2350	–
test	1.1k/42k	4.5	1.9	3.4	2.0	727	378	52.6
German (de) dataset								
train	15k/487k	62.8	2.6	2.3	2.7	5.6k	2808	–
test	1.9k/59k	7.8	2.5	2.3	2.6	679	454	38.3
Spanish (es) dataset								
train	18k/622k	73.3	2.7	2.2	2.9	7.0k	2369	–
test	2.1k/73k	8.7	2.8	2.2	3.0	796	439	52.8
French (fr) dataset								
train	15k/586k	63.6	2.1	1.8	2.2	5.2k	1594	–
test	1.8k/68k	7.4	2.1	1.7	2.2	570	290	49.0

C. Evaluation Metrics

Performance across the five tasks is measured using established metrics. For ASR, we report word error rate (WER). For SCD and endpointing, we use the text-based F1 score. NER performance is evaluated using the *exact-match* F1 score, consistent with [1]. Finally, LID is assessed by accuracy.

D. Training and Inference Details

All `TokenVerse++` models and baselines are trained using hyperparameters identical to those in the original `TokenVerse` for consistency. Specifically, the XLSR-Transducer model [17] is trained with the pruned RNN-T loss [21] for 50 epochs, with an initial learning rate of 1.25×10^{-3}. The best-performing model epoch is selected based on the average WER on the dev sets across all languages. Results are reported on the respective test sets. During inference,

decoding is performed using the "modified_beam_search" algorithm with a beam size of 4.

E. Dataset Details

Table I provides detailed statistics for the multilingual multitask DefinedAI datasets, covering English, German, Spanish, and French, along with token metadata for each subset. The token count for LID matches the number of utterances, as LID is performed at the utterance level. Notably, the percentage of unique named entities in test sets unseen during training ranges from 47.2% to 61.7%, making the datasets especially valuable for evaluating the generalization capabilities of `TokenVerse++`. The diversity in label availability, along with the presence of unseen named entities across tasks and languages, offers a comprehensive and robust testbed for multitask learning in a multilingual context.

IV. Results and Discussions

We first evaluate `TokenVerse++` against the `TokenVerse` baseline using fully annotated data. Subsequently, we analyze its core capability of leveraging partially annotated datasets in a multilingual, multitask setting. Finally, we discuss the behavior of the dynamic task activation mechanism during inference and the overall implications of our approach.

A. Baseline Comparison with `TokenVerse`

Our initial investigation assesses `TokenVerse++` under conditions identical to those of the original `TokenVerse` evaluation, employing the English DefinedAI dataset with annotations for ASR, SCD, endpointing, and NER. To ensure a direct comparison, all tasks were activated during inference, mirroring `TokenVerse`'s standard inference mode. As noted before, during training of `TokenVerse++`, random combinations of tasks will be activated. Results are presented in Table II.

The **per-combination task vector strategy** (Section II-C), when task activation is introduced after the feature encoder, shows an advantage. This configuration yields 4% relative improvement in ASR WER over `TokenVerse` while maintaining similar performance in NER and endpointing. A trade-off is observed for SCD performance. Conversely, intervening *after the full XLSR encoder* with this strategy is detrimental, leading to a general decline in performance across all tasks. This suggests that modifying embeddings at this later stage, with fewer subsequent learnable parameters in the encoder, offers insufficient representational power for effective multitask adaptation. Activating tasks at *both* the feature encoder and full XLSR encoder simultaneously does not yield further benefits over the feature encoder-level intervention alone.

The **more scalable per-task vector summation approach**, where tasks each have independent learnable vectors that are summed to form the final task activation vector (Eq. 3), the results closely match those of `TokenVerse` across most tasks. However, a slight degradation is generally observed compared to the per-combination vector strategy. This may be due to the summation operator, which is a many-to-one mapping that may introduce ambiguity for task activation. Notably, this approach shows some improvement in ASR, SCD, and endpointing when tasks are activated after the full XLSR encoder.

These findings have architectural implications. The outcomes achieved with feature-encoder-level intervention, especially for the per-combination strategy, underscore the benefit of early modulation of acoustic embeddings. This allows the deep XLSR encoder to process task-conditioned inputs, thereby facilitating more effective hierarchical feature learning for diverse tasks. A trade-off emerges: the per-combination strategy achieves the best observed performance at the expense of an exponential increase in task-specific parameters (2^K vectors), whereas the per-task summation strategy offers linear scalability (K vectors) with a slight compromise in performance for certain configurations. Given that `TokenVerse` itself established a strong benchmark by outperforming pipeline methodologies [1], these results establish `TokenVerse++` as an effective and flexible successor.

B. Leveraging Partially Annotated Datasets

A key aspect of `TokenVerse++` is its ability to integrate datasets with incomplete task annotations. This capability was explored through a multilingual, five-task scenario (ASR, SCD, endpointing, NER, and LID), as detailed in Section III-B. The LID task serves here as a practical test case of how `TokenVerse++` accommodates tasks for which label availability varies across datasets. We employed the `TokenVerse++` configuration using per-combination vectors with activation after the feature encoder. Results are presented in Table III.

a) Performance with Fully Annotated Multilingual Data (DefinedAI-only): Initially, training was conducted solely on the multilingual DefinedAI dataset, which provides annotations for all five tasks. In this setting, `TokenVerse++` outperforms `TokenVerse` across the majority of tasks and languages. For instance, ASR WER sees improvements across all evaluated languages, as do metrics for SCD and endpointing. NER performance also shows positive gains. LID accuracy is high for both models. This establishes a five-task baseline before introducing partially labeled data, confirming that `TokenVerse++` can handle multiple tasks robustly.

b) Enhancements via Partially Labeled Dataset Integration (DefinedAI + CommonVoice): Next, we utilize the CommonVoice (CV) dataset during training in addition to the DefinedAI data. The CV data contributes labels *solely* for ASR and LID, thereby representing a corpus of partially annotated speech. As hypothesized, this integration yields further improvements, particularly for ASR. ASR error rates are consistently reduced across all languages, with significant gains observed for German. LID accuracy remains high. For tasks where CV offered no annotations (SCD, endpointing, NER), performance is largely maintained or shows marginal improvements in several language-task pairs. While isolated, minor degradations in these auxiliary tasks are observed for

TABLE II: Performance comparison of TokenVerse++ (with all tasks activated at inference) and TokenVerse on the DefinedAI English test set. TokenVerse++ results are reported when task activation is applied at different embedding spaces within the XLSR encoder (see Section II-B). NER F1 scores are calculated using the *exact-match* approach from [1]. The best TokenVerse++ configuration improves ASR performance, maintains comparable performance in NER and endpointing, but shows a decline in SCD.

Model	ASR (WER ↓)	SCD (F1 ↑)	Endpoint (F1 ↑)	NER (F1 ↑)
TokenVerse [1]	14.7	**90.3**	89.9	57.6
TokenVerse++ (Each Task Combination Has a Learnable Vector)				
After Feature Encoder	**14.1**	88.2	89.7	**57.7**
After Full XLSR Encoder	14.7	86.3	87.6	55.4
After Both	14.3	88.3	89.8	57.4
TokenVerse++ (Each Task Has a Learnable Vector)				
After Feature Encoder	14.5	89.4	89.4	55.4
After Full XLSR Encoder	14.4	89.3	**90.0**	53.1
After Both	14.7	88.7	89.4	56.0

TABLE III: Performance comparison on multilingual DefinedAI eval sets for TokenVerse, and the best-performing TokenVerse++ (with all tasks activated at inference) across four languages. The comparison includes TokenVerse++ evaluated with only the multilingual DefinedAI dataset and after the addition of CommonVoice (CV) in training, which provides labels solely for ASR and LID tasks. Notably, TokenVerse would not allow the inclusion of utterances (from the CV dataset in this case) in training that lack labels for all tasks. In contrast, TokenVerse++ enables such inclusion, improving performance on such tasks.

Model	Lang	ASR (WER ↓)	SCD (F1 ↑)	Endpoint (F1 ↑)	NER (F1 ↑)	LID (acc ↑)
TokenVerse	en	15.2	90	89.7	53.7	99.8
	de	21.3	80.8	80.1	25.5	**99.8**
	es	24.4	66.6	66.4	42.4	**100.0**
	fr	20.8	70.5	70.2	45.3	**99.9**
TokenVerse++	en	13.9	91.1	**91.2**	54.1	99.8
	de	19.9	**82.6**	**82.1**	**25.8**	99.5
	es	22.7	**66.9**	**67.5**	**42.9**	99.6
	fr	18.8	73.6	73.7	47.8	**99.9**
TokenVerse++ (with CV)	en	**13.3**	**91.3**	90.4	**56.0**	**100.0**
	de	**18.8**	80.9	80.4	25.1	99.4
	es	**22.2**	62.7	62.1	39.3	99.5
	fr	**18.1**	**74.0**	**74.1**	**48.6**	**99.9**

specific languages upon adding the CV data. The improvements in ASR performance affirm the utility of leveraging large, partially-annotated ASR corpora. This capability, to enhance performance on tasks by incorporating readily available, albeit incompletely labeled, datasets, is a fundamental advantage of TokenVerse++, unachievable within the constraints of the original TokenVerse framework. The inclusion of LID and its successful training with partial support from CV further underscores this adaptability.

C. Analyzing Dynamic Task Activation Performance

The adaptability of the dynamic task activation mechanism is further explored by evaluating TokenVerse++ with varying subsets of active tasks during inference, on the English DefinedAI dataset. These results, detailed in Table IV, present inter-task dynamics and differential impacts of the per-combination versus per-task summed activation vector strategies.

a) Per-Combination Task Vector Strategy: This strategy, using unique vectors per task combination, shows that ASR performance is sensitive to co-activated tasks. ASR WER generally improves when an auxiliary task (SCD, endpointing, or particularly NER) is co-activated, compared to ASR-only activation. The combination of ASR+Endpointing+NER yields the best ASR and NER performance in these experiments, suggesting strong synergy where the specialized vector benefits all three. Conversely, auxiliary tasks like SCD and endpointing achieve their individual peak F1-scores in smaller, more focused combinations (e.g., ASR with only SCD or only endpointing), indicating that their optimal conditioning might be achieved without the influence of all other tasks.

b) Per-Task Vector Summation Strategy: Summing individual task vectors results in more stable ASR performance across different task combinations, often near its level when co-activated with NER or all tasks. This suggests a more generalized ASR conditioning. Notably, auxiliary tasks like

TABLE IV: Performance of TokenVerse++ for two configurations: (i) each task combination has a learnable vector, and (ii) each task has a learnable vector. Results are reported for the best performing activation positions from Table II, with different subsets of tasks activated at inference while keeping the primary task (ASR) always active. The **Inactive Tasks Token (ITT)** column indicates the count of tokens predicted for tasks not activated, showcasing the model's effectiveness in minimizing irrelevant predictions.

Activated Tasks	ASR (WER ↓)	SCD (F1 ↑)	Endpoint (F1 ↑)	NER (F1 ↑)	ITT (↓)
Each Task Combination Has a Learnable Vector (Activation After Feature Encoder)					
Single Task					
ASR	14.3	–	–	–	0
ASR + SCD	14.2	**89.5**	–	–	0
ASR + Endpoint	14.2	–	**90.1**	–	0
ASR + NER	14.1	–	–	57.5	0
Two Tasks					
ASR + SCD + Endpoint	14.2	87.7	89.4	–	0
ASR + SCD + NER	14.2	**89.5**	–	57.7	1
ASR + Endpoint + NER	**13.9**	–	90.0	**58.3**	1
All Tasks					
ASR + SCD + Endpoint + NER	14.1	88.2	89.7	57.7	–
Each Task Has a Learnable Vector (Activation After Feature Encoder)					
Single Task					
ASR	14.7	–	–	–	0
ASR + SCD	14.7	87.8	–	–	0
ASR + Endpoint	14.7	–	88.6	–	0
ASR + NER	**14.5**	–	–	55.3	0
Two Tasks					
ASR + SCD + Endpoint	14.6	88.9	88.9	–	0
ASR + SCD + NER	**14.5**	88.1	–	**55.5**	1
ASR + Endpoint + NER	**14.5**	–	88.6	55.1	0
All Tasks					
ASR + SCD + Endpoint + NER	**14.5**	**89.4**	**89.4**	55.4	–

SCD and endpointing tend to reach their highest F1-scores when *all* tasks are active, contrasting with the per-combination strategy. This implies that the cumulative effect of summing multiple task vectors can be beneficial for these specific auxiliary tasks in a broadly active setting.

c) Comparative Insights: The **per-combination strategy** excels at optimizing for specific task subsets, potentially capturing specialized synergies (e.g., ASR+Endpointing+NER). Its tailored vectors can lead to peak performance for these targeted combinations. The **per-task summation strategy** offers more distributed task influence. While it may not achieve the same peak ASR synergy, its additive nature can be advantageous for certain auxiliary tasks when many tasks are active, providing a robust, general conditioning.

In essence, the choice of strategy for the task activation vector dictates how inter-task dependencies are manifested and exploited. The per-combination approach allows for fine-tuned optimization for specific task groupings, while the per-task summation provides a more general, scalable mechanism with different optimal activation conditions for auxiliary tasks.

D. Impact

TokenVerse++ demonstrates that ASR models can be trained effectively to perform multiple speech labeling tasks, utilizing both existing ASR datasets and additional datasets annotated for other tasks. This approach improves ASR performance compared to standalone ASR models and TokenVerse, offering a more practical solution. Since TokenVerse++ can predict ASR-type hypotheses while incorporating additional tasks in a single inference step, it is an attractive replacement for ASR-only models, particularly within the transducer architecture.

V. Conclusion

We have introduced TokenVerse++, a token-based multitask framework that overcomes the data inflexibility of prior models. By incorporating learnable vectors for dynamic task activation within the XLSR-Transducer, TokenVerse++ successfully leverages existing ASR datasets alongside additional task-specific datasets, even those with only partial annotations. Our experiments demonstrated that TokenVerse++ achieves results on par with, or exceeding, the original TokenVerse across multiple tasks, notably improving ASR performance. Crucially, by integrating partially labeled data, specifically, the CommonVoice corpus for ASR and language identification as an additional task, we further enhanced ASR performance, showcasing the system's ability to effectively utilize diverse data sources. The flexible task activation scheme in training underscores the potential of our approach for unified modeling across diverse speech and NLP tasks, offering a more practical and adaptable solution for multitask learning.

REFERENCES

[1] S. Kumar, S. Madikeri, J. P. Zuluaga Gomez, I. Thorbecke, E. Villatorotello, S. Burdisso, P. Motlicek, K. P. D. S, and A. Ganapathiraju, "TokenVerse: Towards Unifying Speech and NLP Tasks via Transducer-based ASR," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 20 988–20 995. [Online]. Available: https://aclanthology.org/2024.emnlp-main.1167/

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.

[3] H. Yadav, S. Ghosh, Y. Yu, and R. R. Shah, "End-to-end named entity recognition from english speech," *arXiv preprint arXiv:2005.11184*, 2020.

[4] Y. Wu, S. Maiti, Y. Peng, W. Zhang, C. Li, Y. Wang, X. Wang, S. Watanabe, and R. Song, "SpeechComposer: Unifying Multiple Speech Tasks with Prompt Composition," *arXiv preprint arXiv:2401.18045*, 2024.

[5] K.-W. Chang, Y.-K. Wang, H. Shen, I.-t. Kang, W.-C. Tseng, S.-W. Li, and H.-y. Lee, "Speechprompt v2: Prompt tuning for speech classification tasks," *arXiv preprint arXiv:2303.00733*, 2023.

[6] L. E. Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," *arXiv preprint arXiv:1907.05337*, 2019.

[7] W. Xia, H. Lu, Q. Wang, A. Tripathi, Y. Huang, I. L. Moreno, and H. Sak, "Turn-to-diarize: Online speaker diarization constrained by transformer transducer speaker turn detection," in *ICASSP*. IEEE, 2022, pp. 8077–8081.

[8] S. Kumar, S. Madikeri, N. Iuliia, E. VILLATORO-TELLO, P. Motlicek, K. P. D. S, S. P. Dubagunta, and A. Ganapathiraju, "Multitask Speech Recognition and Speaker Change Detection for Unknown Number of Speakers," in *Proceedings of the 49th IEEE International Conference on Acoustics, Speech, & Signal Processing (ICASSP) 2024*, 2024.

[9] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 1, pp. 50–70, 2020.

[10] Y. Zou, L. Zhao, Y. Kang, J. Lin, M. Peng, Z. Jiang, C. Sun, Q. Zhang, X. Huang, and X. Liu, "Topic-oriented spoken dialogue summarization for customer service with saliency-aware topic modeling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14 665–14 673.

[11] Y. Xu, H. Zhao, and Z. Zhang, "Topic-aware multi-turn dialogue modeling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14 176–14 184.

[12] W. R. Huang, C. Allauzen, T. Chen, K. Gupta, K. Hu, J. Qin, Y. Zhang, Y. Wang, S.-Y. Chang, and T. N. Sainath, "Multilingual and Fully Non-Autoregressive ASR with Large Language Model Fusion: A Comprehensive Study," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 13 306–13 310.

[13] C.-H. H. Yang, Y. Gu, Y.-C. Liu, S. Ghosh, I. Bulyko, and A. Stolcke, "Generative Speech Recognition Error Correction With Large Language Models and Task-Activating Prompting," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–8.

[14] Z. Ma, G. Yang, Y. Yang, Z. Gao, J. Wang, Z. Du, F. Yu, Q. Chen, S. Zheng, S. Zhang *et al.*, "An Embarrassingly Simple Approach for LLM with Strong ASR Capacity," *arXiv preprint arXiv:2402.08846*, 2024.

[15] N. Das, S. Dingliwal, S. Ronanki, R. Paturi, D. Huang, P. Mathur, J. Yuan, D. Bekal, X. Niu, S. M. Jayanthi *et al.*, "SpeechVerse: A Large-scale Generalizable Audio Language Model," *arXiv preprint arXiv:2405.08295*, 2024.

[16] S. Kumar, I. Thorbecke, S. Burdisso, E. Villatoro-Tello, M. KE, K. Hacioğlu, P. Rangappa, P. Motlicek, A. Ganapathiraju, and A. Stolcke, "Performance Evaluation of SLAM-ASR: The Good, the Bad, the Ugly, and the Way Forward," in *2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. IEEE, 2025, pp. 1–5.

[17] S. Kumar, S. Madikeri, J. Zuluaga-Gomez, E. Villatoro-Tello, I. Thorbecke, P. Motlicek, M. KE, and A. Ganapathiraju, "XLSR-Transducer: Streaming ASR for Self-Supervised Pretrained Models," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[18] X. Yang, W. Kang, Z. Yao, Y. Yang, L. Guo, F. Kuang, L. Lin, and D. Povey, "PromptASR for Contextualized ASR with Controllable Style," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 10 536–10 540.

[19] X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353/

[20] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT Understands, Too," 2023. [Online]. Available: https://arxiv.org/abs/2103.10385

[21] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned RNN-T for fast, memory-efficient ASR training," *arXiv preprint arXiv:2206.13236*, 2022.

[22] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[23] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[24] G. Gautier, R. Bardenet, and M. Valko, "On two ways to use determinantal point processes for monte carlo integration," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. Tyers, and G. Weber, "Common Voice: A Massively-Multilingual Speech Corpus," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 4218–4222.