Live Streaming Speech Recognition Using Deep Bidirectional LSTM Acoustic Models and Interpolated Language Models

Javier Jorge, Adrià Giménez, Joan Albert Silvestre-Cerdà, Jorge Civera, Albert Sanchis, and Alfons Juan

***Abstract*—Although Long-Short Term Memory (LSTM) networks and deep Transformers are now extensively used in offline ASR, it is unclear how best offline systems can be adapted to work with them under the streaming setup. After gaining considerable experience on this regard in recent years, in this paper we show how an optimized, low-latency streaming decoder can be built in which bidirectional LSTM acoustic models, together with general interpolated language models, can be nicely integrated with minimal perfomance degradation. In brief, our streaming decoder consists of a one-pass, real-time search engine relying on a limited-duration window sliding over time and a number of ad hoc acoustic and language model pruning techniques. Extensive empirical assessment is provided on truly streaming tasks derived from the well-known LibriSpeech and TED talks datasets, as well as from TV shows on a main Spanish broadcasting station.**

***Index Terms*—Automatic speech recognition, streaming, decoding, acoustic modeling, language modeling, neural networks.**

I. Introduction

LIVE video streaming services over the Internet have increased dramatically in recent years because of higher user demand and bandwidth speeds. This has resulted in a growing need by live video streaming platforms to provide high-quality automatic speech transcriptions. However, the application of state-of-the-art neural-based Automatic Speech Recognition (ASR) models to video streaming is a highly complex and challenging task due to real-time and low-latency decoding constraints.

Manuscript received January 12, 2021; revised June 23, 2021; accepted November 23, 2021. Date of publication December 10, 2021; date of current version January 8, 2022. This work was supported in part by European Union's Horizon 2020 Research and Innovation Programme under Grant 761758 (X5gon), and 952215 (TAILOR) and Erasmus+ Education Program under Grant Agreement 20-226-093604-SCH, in part by MCIN/AEI/10.13039/501100011033 ERDF A way of making Europe under Grant RTI2018-094879-B-I00, and in part by Generalitat Valenciana's Research Project Classroom Activity Recognition under Grant PROMETEO/2019/111. Funding for open access charge: CRUE-Universitat Politècnica de València. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lei Xie. *(Corresponding author: Javier Jorge.)*

The authors are with the Valencian Research Institute for Artificial Intelligence, Machine Learning and Language Processing Group, E-46022 Valencia, Spain (e-mail: jjorge@dsic.upv.es; agimenez@dsic.upv.es; juasilce@vrain.upv.es; jcivera@dsic.upv.es; josanna@dsic.upv.es; ajuan@dsic.upv.es).

Digital Object Identifier 10.1109/TASLP.2021.3133216

At this time, state-of-the-art ASR systems are based on the hybrid Hidden Markov Model (HMM) and neural network approach [1]. In particular, deep Bidirectional Long-Short Term Memory (BLSTM) networks have proven to be a powerful architecture for acoustic modeling in a wide range of ASR tasks [2]–[4]. In the same way, Transformer-based architectures have recently reached very promising results for language modeling [5], though LSTM recurrent neural networks (LSTM-RNN) are still broadly used [6]. It goes without saying that end-to-end systems are attracting great attention, and this includes a number of proposals for operation under low-latency streaming decoding [7]–[9]. However, despite their simplicity and promising prospects, it is still unclear whether or not they will soon surpass state-of-the-art hybrid systems combining independent models trained from vast amounts of data.

Two main challenges need to be addressed so as to properly adapt hybrid ASR systems to the streaming setup. The first one is due to the fact that BLSTM acoustic models can no longer be applied in their full extent, over the whole input signal. Instead, they need to be time-limited within a window sliding over time in which only a small fraction of non-decoded signal (right context) can be captured for the system to respond quickly after the incoming audio stream. This adaptation of BLSTM acoustic models to deal properly with the incoming audio stream also implies to dynamically carry out acoustic mean normalization as opposed to full normalization over the whole signal. An additional issue to be considered is to adapt well-known pruning techniques as acoustic look-ahead [10], [11] to work with BLSTM acoustic models to speed up the decoding process. The second one is that Transformer and LSTM-RNN language models (LMs) cannot likewise be applied as they use to be, by rescoring n-best hypotheses or lattices in a two-pass decoding approach [12]–[14]. In all these cases, efficient techniques are required for on-the-fly scoring under a real-time one-pass decoding scheme.

The use of BLSTM acoustic models under streaming conditions has been explored in several recent works. In [15], a finite sliding window was applied to approximate the acoustic posterior probability of the center frame. This approach was improved in [16] by using a more accurate weighting scheme of overlapping windows. Under this approach, BLSTM-based models outperformed deep neural networks (DNNs) under the streaming setup, also showing that a right context of limited

duration suffices to reach a performance similar to that of the offline setup. In contrast to using a sliding window over the incoming signal, a different approach consists in splitting it into overlapping chunks with appended (past and future) contextual observations. This approach was followed in [3], where the so-called Context-Sensitive-Chunk (CSC) method was proposed to speed up BLSTM training for low-latency decoding by just adding some delay in between consecutive chunks. This method can be accelerated by simply avoiding computations on the left context, as done with the Latency-Controlled BLSTMs proposed in [17], which in turn can be further improved as shown in [18]. However, in all these previous works, empirical evaluations were not performed under genuine streaming conditions, that is, dealing with the speech signal as an incoming audio stream and, therefore, on-the-fly mean normalization was not considered at all. Moreover, basic n-grams LMs were used in experiments which greatly helped to improve the responsiveness of the system in the evaluation of system latencies as were reported only in the case of the CSC approach.

Regarding the use of neural LMs, to our knowledge, the direct use of this technique during decoding was first explored in [19], where the authors proposed the use of a Variance Regularization term together with caching strategies for fast decoding. Despite using feed-forward neural LMs in decoding, empirical results showed significant relative improvements both in speed and accuracy. Other relevant contributions addressing one-pass decoding with neural LMs have focused on heuristics to reduce the number of queries to the model and catching network states [20], alternative one-pass decoding strategies such as on-the-fly rescoring [21], improving CPU-GPU communications [22] and, more recently, combining Gated Recurrent Units with more efficient objective functions, such as Noise Contrastive Estimation [23]. Certainly different from these contributions, other authors have explored the idea of converting neural LMs, either recurrent or not, into n-gram models that can thus be smoothly integrated into a conventional decoder [24], [25]. It is worth noting, however, that these approaches were only focused on language modeling and not on the low-latency streaming decoding problem.

This work takes as a starting point a novel architecture for real-time one-pass decoding with LSTM-RNN LMs proposed in [26]. In it, one-pass decoding was accelerated by estimating look-ahead scores using precomputed static look-ahead tables. Moreover, LSTM-RNN LM probabilities were efficiently computed using Variance Regularization and lazy evaluation. Later on, in [27], this architecture for real-time one-pass decoding was extended to include BLSTM acoustic models within a time sliding window, also used as a window for time-constrained, on-the-fly acoustic feature normalization. Not surprisingly, empirical assessment of this extended architecture under strict streaming conditions proved it was really effective, indeed keeping the pace with non-streaming (offline) systems. The most recent refinement in connection to this research line has consisted in replacing streaming-adapted LSTM-RNN LMs with Transformer LMs [28]. In doing so, empirical results on the well-known LibriSpeech [29] and TED-LIUM [30] tasks have shown that this refinement leads to top, state-of-the-art recognition rates and latencies under streaming conditions. In short, it has been shown that hybrid one-pass ASR systems built in this way can work under both, offline and streaming conditions with no significant differences in quality.

This work is intended to provide a complete, detailed reference of the main contributions made along the research line described above, also including a number of new additional enhancements and a new and extensive empirical evaluation under streaming conditions. In particular, the following important novel algorithmic enhancements are provided:

- The sliding window framework proposed in [16] is revisited to include and adapt necessary concepts for proper streaming decoding.
- Also for proper streaming decoding, novel methods for acoustic feature normalization are explored.
- Along with these two streaming-oriented enhancements, and to improve the general performance of the decoder, new pruning techniques for fast decoding are also considered. More precisely, a new approach for the acoustic look-ahead is provided, together with a more efficient pruning to speed up the use of interpolated neural LMs.

Only after including these enhancements, a fully-fledged streaming ASR system can be effectively deployed into production. For empirical evaluation, apart from the conventional LibriSpeech and TED-LIUM tasks considered in previous work, two genuine streaming tasks also posed for streaming benchmarking: a video-based version of TED-LIUM with unsegmented talks, and a set of full-length videos from a Spanish TV broadcaster.

The paper is organized describing separately the two main components which generally speaking deserve special attention in the deployment of a streaming decoder. In particular, the use of deep BLSTM acoustic models for streaming is described in Section II. On the other hand, the efficient pruning technique for fast one-pass decoding using interpolated neural LMs is presented in Section III. All these components are empirically assessed in Section IV with emphasis on the key adaptation parameters required for finding an appropriate (task-dependent) trade-off between accuracy and latency. Finally, the main conclusions drawn from on this research line are summarized in Section V.

II. Deep Bidirectional LSTM Acoustic Models for Streaming

In this section, all the issues concerning the use of deep BLSTM acoustic models for streaming are described. Firstly, the sliding window framework proposed in [16] is revisited in Section II-A to include and adapt necessary concepts for proper streaming decoding using BLSTM acoustic models. Secondly, a novel approximation for computing acoustic look-ahead scores is proposed in Section II-B. Lastly, several acoustic mean normalization methods for streaming are proposed in Section II-C.

A. Streaming Decoding Using BLSTM Acoustic Models

Let $\mathbf{x}_1^\infty$ be an unbounded sequence of frames computed from the incoming audio stream, which is being processed by application of a sliding window of w frames shifting one frame to

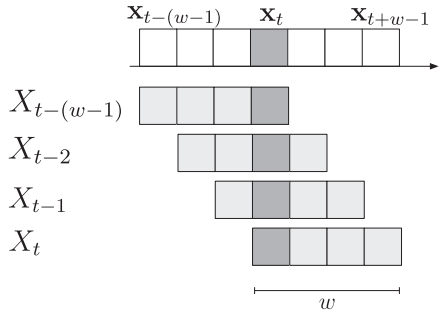


Fig. 1. Frame sequence at the top and just below a sliding window of $w = 4$ frames at all steps embracing frame t, $\mathbf{x}_t$.

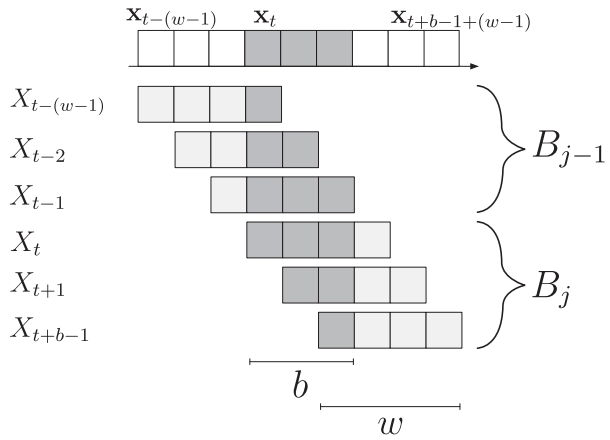


Fig. 2. Computing the acoustic scores for $b = 3$ consecutive frames starting at t, $\mathbf{x}_t^{t+b-1}$, within a sliding window of size $w = 4$ during two consecutive b-step batches, B_{j-1} and B_j.

the right at each step (though a step size of more than one frame can be also used if convenient). Thus, frames t to $t + w - 1$ are covered by the sliding window at step t, $X_t = \mathbf{x}_t^{t+w-1}$. This is illustrated in Fig. 1, where the sliding window is also depicted at all the $w - 1$ preceding steps embracing frame t. $X_{t-(w-1)}^{t-1}$.

For each acoustic state a, we assume that a BLSTM acoustic model is available to compute the posterior probability of the n-th frame within the sliding window at step t, $p_n(a \mid X_t)$. As frame t falls into position $w - i + 1$ of the sliding window at step $t - (w - i)$, $1 \leq i \leq w$, it gets w different posteriors from which its acoustic score is computed by just (weighted) averaging:

$$q(\mathbf{x}_t, a) = \frac{1}{w} \sum_{i=1}^{w} p_{w-i+1}(a \mid X_{t-(w-i)}) \tag{1}$$

For this score to be efficiently computed, we assume that the recurrent state of each BLSTM does not depend on its previous states, and hence the posterior it provides for frame t only depends on (its position in) the window context considered. This assumption enables fast computation of posteriors, not only by running independent BLSTM queries in parallel, but also by avoiding repeated posterior computations during consecutive step batches. Fig. 2 shows how this looks like in a simple example where the acoustic scores for $b = 3$ consecutive frames starting at t, $\mathbf{x}_t^{t+b-1}$, are computed within a sliding window of size $w = 4$ during two consecutive b-step batches, B_{j-1} and B_j.

As shown in Fig. 2, scoring the frame subsequence $\mathbf{x}_t^{t+b-1}$ can be efficiently done by keeping all precomputed posteriors for it and just running b independent BLSTM parallel queries in B_j to get the posteriors still required. In the example given, for each position i ($i = 1, 2, 3$), we have $w - i$ posteriors available and i posteriors to be computed. In general, for any $b \geq 1$, a carousel may be used at each position i, $1 \leq i \leq \min(b, w - 1)$, to keep track of $w - i$ precomputed posteriors and fill in the remaining i from the b parallel BLSTM calls in batch B_j. Clearly, if $b \geq w$, no precomputed posteriors are involved from position w to b.

It is obvious that the window size w is a key adaptation parameter for streaming as it controls the duration of the acoustic context, both in the *past* ($\mathbf{x}_{t-(w-1)}^{t-1}$) and the *future* ($\mathbf{x}_{t+1}^{t+w-1}$), that is used when scoring the *current* frame $\mathbf{x}_t$. Needless to say, we are assuming that the most relevant acoustic context at each frame occurs within a time window of a handful tenths of second. Also, as we need the complete future context to be available for (exact) scoring, time windows longer than that may prevent the system to respond after a reasonable latency of, say, one second. This is of course a topic to be explored empirically. In this regard, note that we are limiting ourselves to symmetrical time windows of fixed duration (w) and exact scoring, as defined in (1). However, if convenient, more general schemes for acoustic context management and scoring can be also devised, such as asymmetrical time windows of variable duration and approximate scoring.

As w, the batch size b is also a key adaptation parameter for streaming though, in contrast to w, its effect is only computational. In principle, we may want b as large as possible, for maximum parallelism, but also small enough for the additional future context (of $b - 1$ frames) it requires not to become the dominant factor in the observed system latency. This is easily understood from Figs. 1 and 2. In Fig. 1, we have $b = 1$ and thus, apart from the future $w - 1$ frames required to complete the sliding window at t, X_t, no additional frame is needed to score $\mathbf{x}_t$. Instead, in Fig. 2, we have $b = 3$, and hence 2 additional frames are needed before running a 3-step batch of parallel BLSTM calls. There are also other hardware-dependent factors such as (GPU) memory bandwidth that may add up significantly to the observed latency as the batch size increases. Therefore, as with w, this is best studied empirically.

B. Acoustic Model Look-Ahead

The acoustic look-ahead refers to the best acoustic score (emission and transition probabilities) that can be reached from a given frame $\mathbf{x}_t$ and an acoustic state a. More precisely, following a similar notation to [10], the exact acoustic look-ahead $l(t, a)$ is defined as

$$l(t, a) = \max_{a_0^L : a_0 = a} \sum_{\tau=1}^{L} q(\mathbf{x}_{t+\tau}, a_\tau) + q(a_\tau, a_{\tau-1}) \,, \tag{2}$$

being L the number of remaining frames until the end of the speech signal, and $q(\mathbf{x}_t, a_t) = \log \frac{p(a_t|\mathbf{x}_t)}{p(a_t)}$ and $q(a_t, a_{t-1}) =$

$\log p(a_t \mid a_{t-1})$ the emission and transition probabilities, respectively.

During decoding, for a given partial hypothesis a_1^t, an upper bound of the acoustic score $\hat{q}(a_1^t)$ is computed by adding the acoustic look-ahead score as

$$\hat{q}(a_1^t) = q(a_1^t) + l(t, a_t) \tag{3}$$

where

$$q(a_1^t) = \sum_{\tau=1}^{t} q(\mathbf{x}_\tau, a_\tau) + q(a_\tau, a_{\tau-1}) \tag{4}$$

It is worth noting that $\hat{q}(a_1^t)$ is an optimistic estimation of the acoustic score, since it is not only considering the score until instant t, but also the score of the best future path that could be reached from that instant according to the acoustic model. This estimated score leads to a more guided beam search, which in turn, could lead to speed up the decoding process.

Therefore, the challenge resides in efficiently computing the acoustic look-ahead score $l(t, a)$ and how to speed up the search of the best future path (see (2)). In fact, let us to refer previous works in which exact look-ahead scores were approximated by limiting the future context to a few frames and/or simplifying the emission models used for look-ahead calculation [10], [11]. A major issue when acoustic look-ahead was applied over Gaussian HMMs lied in the cost of estimating the emission scores. However, in current hybrid systems based on neural networks this is not a problem anymore since, for each frame, the neural network estimates the scores for all HMM states, and usually this is performed in batch mode using GPUs. In current state-of-the-art hybrid system based on triphonemes the number of different states that must be considered during the search of an acoustic alignment (without considering the LM) may vary between half a million and several millions, depending on the total number of HMMs and the vocabulary size. This makes unfeasible to directly apply exact acoustic look-ahead estimation.

As alternative to circumvent all these drawbacks, we propose to approximate the search space by a bigram model using HMM states as tokens. In the bigram model we only consider those transitions that are allowed in the original search space. Additionally, all transitions scores are set to zero, i.e., we only focus on emission scores. Using this approach the acoustic look-ahead can be estimated at low cost using dynamic programming, and without the need of limiting the number of future frames or simplifying emission models. More precisely, for a given frame $\mathbf{x}_t$ and state a, the proposed acoustic look-ahead approximation $\hat{l}(t, a)$ is estimated as

$$\hat{l}(t,a) = \begin{cases} 0 & t = T \\ \max\limits_{a' \in A} \hat{q}(a, a') + q(\mathbf{x}_{t+1}, a') + \hat{l}(t+1, a') & t < T \end{cases} \tag{5}$$

where T is the total number of frames, A is the set of HMM acoustic states, and $\hat{q}(a, a')$ is a function that returns 0 if (a, a') is a non-zero probability bigram transition or $-\infty$ otherwise. During decoding, the look-ahead based acoustic score for a hypothesis can be incrementally updated as

$$\hat{q}(a_1^t) = \hat{q}(a_1^{t-1}) - l(t-1, a_{t-1}) + \\ + q(a_t, a_{t-1}) + q(\mathbf{x}_t, a_t) + l(t, a_t). \tag{6}$$

In an offline setup the proposed acoustic look-ahead scores could be precomputed before decoding without limitation on the number of future frames. However, the streaming setup requires an on-the-fly estimation of acoustic look-ahead scores. More precisely, as described before and was illustrated in Fig. 2, every b frames the BLSTM is queried with $b + w - 1$ frames, and outputs the emission score for the first b frames. In this setup, every time the BLSTM is queried the acoustic look-ahead scores are estimated for the first b frames of the query. In order to take full advantage of the available data, when querying the BLSTM we also retrieve partially averaged emission scores for the frames of the future context, therefore, the acoustic look-ahead score for the first frame of the batch will be estimated considering $b + w - 2$ future frames, while in the last frame of the batch only $w - 1$ frames will be considered. It is worth noting, that in every BLSTM query a computational overhead of w is introduced when compared with an offline scenario. Consequently, the larger the batch size, the smaller the computational overhead and the future context limitation. However, a large batch size also means higher latencies. Therefore, this is a trade-off that must be taken into account when applying the proposed technique for streaming as will be appropriately evaluated in Section IV.

C. Acoustic Feature Normalization for Streaming

Under the offline scenario, Full Sequence Normalization (FSN) is usually performed beforehand applying mean normalization to the whole speech utterance. However, since FSN is not feasible under streaming conditions, we propose different alternatives to carry out on-the-fly sequence normalization.

The first alternative, called Dynamic Threshold Normalization (DTN), consists on the initialization of the mean by considering an initial delay of n_{norm} frames. Afterwards, the mean is dynamically updated for every new frame. In previous works, we proved that two seconds of initial delay should be enough to achieve similar performance to FSN [27], [28]. Although, two seconds of delay could be reasonable in a continuous streaming setup, it could be not so suitable for short utterances such as voice commands.

To overcome this limitation and taking advantage of the sliding window technique introduced in Section II-A, we propose in this work a novel normalization scheme called Weighted Moving Average (WMA), in which mean normalization is performed using the frames of the sliding window. In this way, WMA is applied over a batch B_j of frames as

$$\hat{B}_j = B_j - \hat{\mu}_j \tag{7}$$

where

$$\hat{\mu}_j = \frac{\mathbf{f}_{j-1} + \sum_{t=1}^{b+w} B_{j,t}}{n_{j-1} + b + w} \tag{8}$$

being $\mathbf{f}_{j-1}$ the accumulated values of previous frames until batch B_{j-1}, $B_{j,t}$ the t-th frame in batch B_j, n_{j-1} the number of

frames until batch B_{j-1}, and b and w the batch and window sizes, respectively.

The accumulated values $\mathbf{f}_j$ and n_j are updated by weighting the contribution of previous batches using a parameter α as

$$\mathbf{f}_j = \alpha \cdot \mathbf{f}_{j-1} + \sum_{t=1}^{b} B_{j,t} \tag{9}$$

$$n_j = \alpha \cdot n_{j-1} + b \tag{10}$$

Unlike FSN, WMA dynamically adapts the normalization of the speech signal to local changes, and differently from DTN, without introducing an initial delay. Therefore, it can be used without affecting the global latency even from the beginning of the utterance. Although an initial mean could be precomputed from the training set, WMA has been evaluated in this work starting from scratch on each sample and using only the information that comes from the audio stream as expected in streaming conditions. Similarly to the acoustic look-ahead, the batch size has also an impact regarding the amount of frames used to compute the mean. This impact will be evaluated during the experimental section.

III. Efficient One-Pass Decoding Using Interpolated Neural LMs

The direct use of neural LMs in one-pass decoding takes full advantage of its ability to deal with histories of unlimited length in contrast to n-gram LMs [14]. This makes History Conditioned Search (HCS) decoders perfectly suited for its use with neural LMs, as HCS technique group hypotheses by its history allowing potentially an unlimited representation of continuous contexts [31]. However, in practice, the integration of neural LMs in one-pass HCS decoders for streaming presents some relevant difficulties which need to be solved. For instance, the very efficient computation of LM look-ahead scores and word neural LM probabilities or the use of specific LM pruning parameters to reduce the search space. In the following, all these decoding issues are discussed and how they have been efficiently addressed in this proposal.

A. Language Model Look-Ahead

There are many techniques to deal with look-ahead LM scores as the use of cache strategies, perfect hashing and precomputation of scores [32], computing look-ahead scores bottom-up from back-off LM [33] or leveraging the LM sparseness to partially compute look-ahead tables [34]. Some of these approaches use a lower order n-gram to obtain look-ahead scores, as higher n-gram orders are not feasible due to memory and computation requirements. In the case of [34], 3-grams and 4-grams are also evaluated, but in this case look-ahead scores are dynamically computed.

In HCS-based decoders, LM look-ahead scores are dynamically computed every time a word-end node is reached during decoding. In order to do this efficiently for streaming, we propose to compute beforehand all these look-ahead scores in static look-ahead tables. With this purpose, a heavily pruned version of the n-gram model can be used to represent this model in a compact structure that can be used during decoding efficiently. This is a critical part of the search when it comes to speed, as this score is queried many times during search to fill the search network structure. Therefore, reducing the n-gram model to this static structure and following a cascade structure of look-ahead tables similar to that proposed in [34], we apply an efficient technique to compute the look-ahead scores during decoding. It is worth noting that this pruned n-gram does not constrain the search space in any way, as this is only used to compute the look-ahead scores. This allows the decoder to consider very long word contexts in search hypotheses leveraging the benefits of using neural LMs.

B. Neural LM Integration

During decoding, when word-end nodes are reached, look-ahead table scores are replaced with those computed from the real LM (i.e. n-grams or neural LMs). In the case of neural LM probabilities, this is an important drawback since it involves computational issues, reducing the speed of the decoder as they are usually more complex models than count-based ones. For this reason, neural LMs are typically applied in a second step of recognition using n-best or lattice rescoring. To alleviate this drawback we propose to apply the Variance Regularization technique [19] reducing the complexity of the computation of the output layer where the softmax function is computed. This technique involves a regularization term during training that aims to reduce the variance of the denominator of the softmax adjusting it to a constant. This constant is kept and then used during decoding when computing neural LM probabilities, instead of computing the denominator. As opposed to LSTM-RNN LMs which store the previous context in an internal vector, Transformer LMs need to compute all the previous history when a new word comes in for the attention to work properly. To deal with long audio streams (possibly hours of continuous speech) we should limit the history to the n previous words, where n is a parameter provided in decoding. It is important to remark that the Transformer LM training enforced no history restriction, indeed there could be a training-decoding mismatch regarding history length that may harm the performance, as sentences of different lengths are devoted to training and the history length is not adjusted beforehand.

C. LM Pruning Parameters

In order to further speed up the decoding process, specific LM pruning parameters had to be incorporated to the one-pass decoder, to reduce the search space or the number of queries in the computation of neural LM probabilities [26]. One of these parameters is the Language Model History Recombination (LMHR) which defines the number of words to be considered before performing hypothesis recombination during decoding. LMHR parameter is needed to control the length of histories since, in HCS decoders, hypotheses are grouped according to their history, meaning that without enforcing any back-off recombination previous histories of active hypotheses tend to grow without any limitation. However, this effect that could be considered a feature turned to be a problem when long histories

TABLE I
BASIC STATISTICS OF DEV AND TESTS SETS IN THE EVALUATION TASKS: DURATION IN HOURS, NUMBER OF SAMPLES (SEGMENTS OR VIDEOS), AVERAGE DURATION OF SAMPLES IN SECONDS PLUS-MINUS STANDARD DEVIATION ($d_\mu \pm \sigma$), AND RUNNING WORDS (RW) IN THOUSANDS (K)

Task	Set	Hours	#Samples	$d_\mu \pm \sigma$	RW(K)
LS [29]	dev-other	5.3	2864	6.4 ± 4.3	51
	test-other	5.1	2939	6.5 ± 4.4	52
TDs [30]	dev-legacy	1.6	507	11.3 ± 5.6	18
	test-legacy	2.6	1155	8.1 ± 4.3	28
TDv	dev	1.7	8	771 ± 401	18
	test	3.1	11	1004 ± 404	28
RTVE [35]	dev1-dev	11.9	10	4267 ±1549	120
	test	39.3	59	2395 ±1673	377

TABLE II
STATISTICS OF SPANISH TEXT RESOURCES USED FOR LANGUAGE MODELING. S=SENTENCES, RW=RUNNING WORDS, V=VOCABULARY. UNITS ARE IN THOUSANDS (K)

Corpus	S(K)	RW(K)	V(K)
Internal: TV, entertainment	4799	59235	307
Internal: education	87	1526	35
Internal: politics	1361	35170	126
Opensubtitles [41]	212635	1146861	1576
UFAL [42]	92873	910728	2179
Wikipedia [43]	32686	586068	3373
UN [44]	11196	343594	381
News Crawl [45]	7532	198545	648
eldiario.es [46]	1665	47542	247
El Periódico [47]	2677	46637	291
Common Crawl [48]	1719	41792	486
News Commentary [45]	207	5448	83
TOTAL	369434	3423146	5785

are considered, as active hypotheses cluster over similar contexts that differ only in words far from the current frame. This parameter aims to reduce the uncertainty of having these long and very similar hypotheses making pruning less effective. This is achieved by combining them when they share a given number of words, and that is indeed what this parameter defines, the number of previous words evaluated to combine active hypotheses. The second pruning parameter, named Language Model Histogram Pruning (LMHP), limits the number of hypotheses that will query the neural LM after reaching new word-end nodes during decoding. This is particularly effective in reducing the costly neural LMs computation, as active hypotheses are pruned before performing any computation. Unlike global histogram pruning applied to thousands of hypotheses after each decoding step, LMHP affects tens or hundreds of hypotheses.

D. LM Interpolation

It is worth stressing that the proposed one-pass HCS-based decoder enables the use of linearly interpolated count-based and/or neural LMs which to our knowledge is unprecedented in streaming ASR.

IV. EXPERIMENTS

A. Evaluation Datasets

The proposed ASR system for streaming was evaluated on LibriSpeech (LS) and TED-LIUM release 2 speech corpus. In the case of the TED-LIUM corpus, we defined a new evaluation task referred to as TDv, in which complete video talks were transcribed without any previous segmentation in order to simulate a streaming scenario. This is, to the best of our knowledge, the first time that the TED-LIUM corpus is considered at the talk level and it could be useful to assess streaming ASR systems in future works. In order to do that, we used the complete audio track for each talk along with the STM files provided in the dataset to evaluate the WER. The conventional segment-based TED-LIUM task is referred to as TDs in this work. Additionally, we used the RTVE2018 dataset which comprises a collection of complete TV shows drawn from diverse genres and broadcasted by the public Spanish national television from 2015 to 2018 [35]. Table I summarizes the basic statistics for the dev and test sets of the tasks mentioned above. In the case of RTVE2018, an internal partition of the provided dev1 set (dev1-dev) was created for development purposes, reserving the test set for evaluation.

B. Training Setup

In order to build the English and Spanish hybrid ASR systems, a context-dependent feed-forward DNN-HMM with three left-to-right states using MFCC 16 plus first and second derivatives (48-dim) was initially trained with our own transLectures-UPV ASR toolkit (TLK) [36]. Then, a BLSTM-HMM acoustic model was trained following the procedure described in [4] using filter bank 85-dimensional features and the previous DNN-HMM alignments. The architecture of the BLSTM model has eight bidirectional hidden layers with 512 LSTM cells per layer and direction trained using both, TLK and TensorFlow [37]. Following [4], we performed chunking during training by considering a context to perform back propagation through time to a window size of 50 frames. Additionally, SpecAugmentation was applied by means of time and frequency distortions [38]. Finally, a final step of sequence discriminative training was performed using our in-house implementation of lattice-based MMI to adjust the transition scores and the weights of the softmax layer [39].

English acoustic models were trained on 961 and 207 hours of training speech corpus for LibriSpeech and TED-LIUM release 2, respectively. After applying a phonetic decision tree [40], 8.3 K and 10.8 K tied-states (or senones) were obtained for LibriSpeech and TED-LIUM, respectively. On the other hand, Spanish acoustic models were trained using the 208 hours provided in the RTVE2018 dataset plus about 3.7 k hours of internal resources. The Spanish ASR system comprises 10 K tied-states.

Regarding the LM training, we used the approximately 800 M words of text provided for LibriSpeech to train neural LMs, as the ngram model is provided with the corpus (*fglarge*), whereas for TED-LIUM we trained the LMs with the six provided subsets plus the TED-LIUM training audio transcriptions with up to 230 M running words. Vocabularies were restricted to 200 K and 153 K words for LibriSpeech and TED-LIUM, respectively. In the case of the Spanish system, text resources were obtained from internal sources and other public repositories shown in Table II. The vocabulary size was over 254 K words and a 1-gigaword random subset of the LM data was selected to train the Spanish

TABLE III
PERPLEXITY (PPL) AND WEIGHT (W) FIGURES ON DEVELOPMENT SETS, CONSIDERING SINGLE MODELS AND TWO-WAY AND THREE-WAY INTERPOLATION OF N-GRAM (N), LSTM-RNN LM (L) AND TRANSFORMER LM (T). INTERPOLATION WEIGHTS WERE OPTIMIZED BY MINIMIZING PPLS OF THE INTERPOLATED MODELS

	LS		TDs		TDv		RTVE	
	PPL	W(%)	PPL	W(%)	PPL	W(%)	PPL	W(%)
N	140.6		117.5		126.1		179.5	
L	72.5		84.0		94.2		98.4	
T	54.2		71.0		64.0		63.3	
L+N	71.6	91+ 9	71.4	73+27	77.1	70+30	93.2	85+15
T+N	53.9	96+ 4	60.5	74+26	55.6	78+22	61.6	94+ 6
T+L	53.5	86+13	64.1	62+38	61.0	78+22	60.7	87+13
T+L+N	53.4	86+12+ 2	58.1	56+25+18	54.8	70+12+18	59.5	85+10+ 5

neural LMs. To train neural LMs when the vocabulary is defined in advance, we decided to obtain the vocabulary as the intersection between the provided vocabulary and that of the training data. In this way, the model avoids having null-word probabilities for words that are in the vocabulary but not in the training set. We take this into account when computing perplexities by renormalizing the unknown-word score accordingly.

As LMs, we used n-grams, LSTM-RNN LMs and Transformer LM (TLM), combining them through a linear interpolation. Count-based models were trained using SRILM [49]. Apart from the 4-gram model provided for LibriSpeech, we trained a 4-gram Kneser-Ney smoothed LM for TED-LIUM using the same data as [30]. To compute the static look-ahead tables, a pruned version of these n-gram models was computed for each task. We obtained OOV ratios of less than 0.6% in all tasks.

The CUED-RNNLM toolkit [50] was used to train LSTM-RNN LMs with Noise Contrastive Estimation (NCE) criterion [51], and the normalization constant learned from training was used during decoding [52]. Based on the lowest perplexity on the dev sets, we selected as final models those with 256-unit embedding layer and two hidden LSTM-RNN layer of 2048 units.

The training of TLMs was carried out using our own customized version of the FairSeq toolkit [53] using a 24-layer network with 768 units per layer, 4096-unit FFN, 12 attention heads, and an embedding of 768 dimensions. These models were trained until convergence with batches limited to 512 tokens, 512 sentences, and 512 words per sentence. Parameters of these models were updated every 32 batches. During inference, Variance Regularization was also applied to speed up the computation of the TLM score.

Table III shows the perplexity of LMs on the development sets for all tasks. When comparing single LM performance, neural models outperformed count-based models on every task, with enough margin, almost halving the perplexity for LS and RTVE in the case of the LSTM-RNN. These results were further improved with TLM, reducing the perplexity in approximately 25% for LS, 15% for TDs, 32% for TDv, and 35% for RTVE, with respect to the LSTM-RNN LM. Model interpolation had diverse impact depending on the combination, but in general using all three models provided the best perplexity for each task. Consistently with the single performance, the TLM obtains the highest weights in the different LM combinations for LS and RTVE (∼85-95%), while LSTM-RNN and ngram models still have an important weight for TED-LIUM tasks, ranging from 22% to 38% when are combined with TLM. When considering the three-way interpolation, again LS and RTVE perplexities show a similar behavior to that of the two-way interpolation with high weights for the TLM, while TLM reduces its weight in favor of LSTM-RNN and ngram in TED-LIUM tasks. TLM history limitation was optimized for best perplexity in each case using the same history size when TLM was interpolated with other LMs.

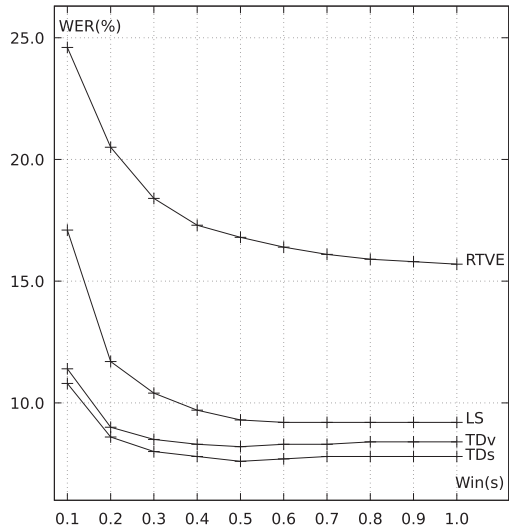


Fig. 3. WER vs. window size in seconds for all tasks.

C. Experiments on Acoustic Modeling for Streaming

The use of BLSTM acoustic models under streaming conditions was evaluated in the following way. First, we studied the effect of the window size presented in Section II-A in the performance of the decoder considering that Full Sequence Normalization (FSN) is performed beforehand. In this way, the optimal window size was fixed for each task in order to be used in the following experiments. Then, the impact of the acoustic look-ahead was gauged to prove its pruning effectiveness, and the different methods for acoustic feature normalization proposed in Section II-C were also assessed. In all these experiments, only count-based LMs were used in order to isolate the effect of the proposed acoustic-related techniques on the decoding.

Fig. 3 shows Word Error Rate (WER) as a function of the window size (w) in seconds from 0.1 (or 10 frames) to 1 s (or 100 frames) for each task. It is worth noting that in this experiment the acoustic models were the same and only the window size was varied during decoding. In LibriSpeech and the TED-LIUM tasks, more context means better performance up to the point at which the windows size is equal to the chunk size used during acoustic training. Beyond this point there is no improvement by increasing the future context leading to more fixed latency without any decrease in WER. Differently from these tasks, RTVE obtains slight improvements after increasing the window size up to about 1 s of future context. This different behavior can be explained because of RTVE is composed of real

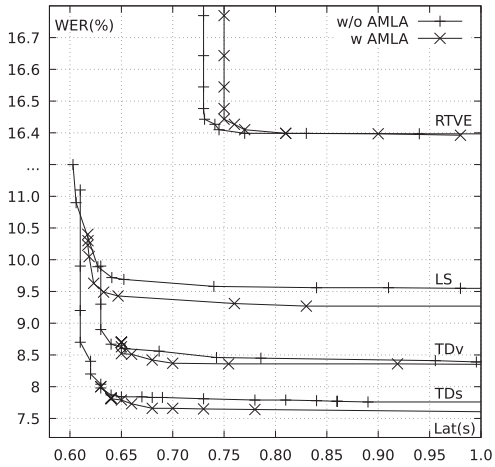


Fig. 4. WER vs. latency in seconds with or without AMLA enabled for each task.

TV shows with heterogeneous conditions and, therefore, further improvements can be expected considering larger contexts to better deal with changing audio conditions. Based on these empirical results, the optimal window sizes were fixed to 0.5 seconds for LibriSpeech and TED-LIUM tasks and, considering that 0.6 for RTVE pays off in WER, we selected this value to include similar fixed delays between all tasks.

In the following experiments, the trade-off between WER and latency is evaluated as it is a critical factor in streaming systems. In all these experiments, latency is measured as the time elapsed between the instant at which an acoustic frame is generated and it is fully processed by the decoder. The final latency for a sample (segment or video) is estimated as the average of the latencies at frame level. These measurements were run on an Intel i7-3820 CPU @ 3.60 GHz, with 64 GB of RAM and a RTX 2080 Ti GPU card. For simplicity, the time required to transform raw audio into filter bank was not included in our measures since this time is negligible and complicates the procedure used to estimate latencies.

Fig. 4 shows WER as a function of latency with or without Acoustic Model Look-Ahead (AMLA) enabled, using $b = 20$. As observed, AMLA is effective to decrease the search effort by exploring more promising paths and, consequently, better performance can be achieved at the same level of latency. This is mainly observed for LibriSpeech and TED-LIUM tasks, but not in RTVE. In RTVE, the number of active hypotheses is less than in LibriSpeech or TED-LIUM, meaning that the reduction in the number of active hypotheses does not compensate for the computational overhead of AMLA.

As stated in Section II, the batch size directly impacts on both, latency and WER when AMLA is enabled, as the set of windows in the batch will be used to compute not only the acoustic scores, but also the acoustic look-ahead with partial acoustic scores. Regarding this impact, we explored batch sizes of 20 and 40 with AMLA enabled, observing that more context to compute AMLA scores lead to similar accuracy for segment-based tasks, such as LibriSpeech and the conventional TED-LIUM, but slightly better WERs in video-based tasks, such as TED-LIUM videos

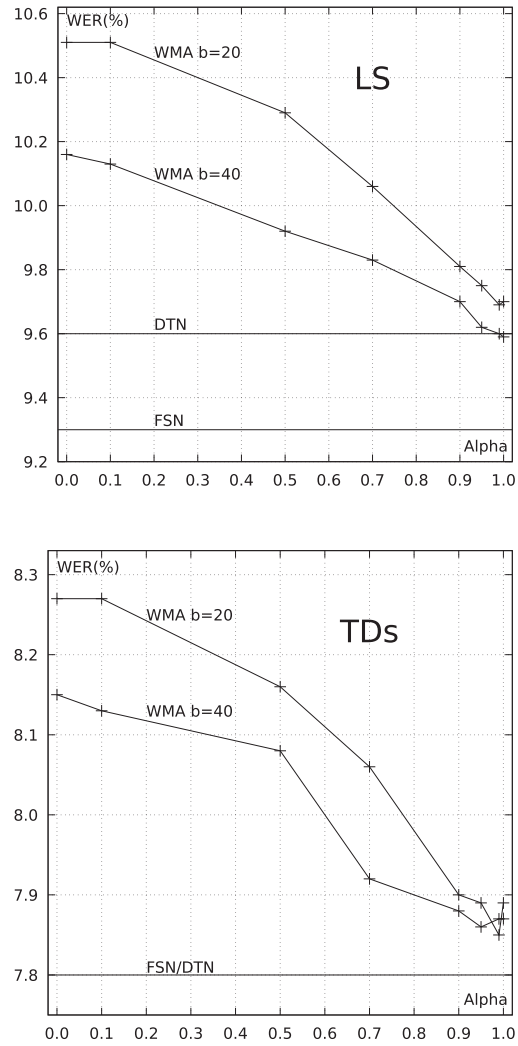


Fig. 5. FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for segment-based tasks.

and RTVE. This is explained by the limited context of short segments of the former tasks compared to the latter tasks. In the case of RTVE, as very similar performance is achieved with or without AMLA enabled, the effect of batch size seems to be negligible. Finally, as expected, higher latencies are obtained with larger batch sizes in all tasks, as longer delays are introduced to gather enough frames to complete the batch. According to these results, in the remaining experiments, AMLA was enabled for LibriSpeech and both TED-LIUM tasks, but not for RTVE.

Figs. 5 and 6 depict for segment-based and video-based tasks, respectively, the effect in WER for the acoustic feature normalization schemes described in Section II-C. In the case of the WMA scheme, WER is also shown as a function of the batch size (b) and the parameter α used to weight the importance of frames in previous batches. As observed, FSN and DTN provided similar WER in all tasks with the exception of LibriSpeech where higher improvements were achieved when FSN is applied.

Nevertheless, WMA clearly outperforms FSN and DTN when decoding long sequences, as shown in Fig. 6 for TED-LIUM

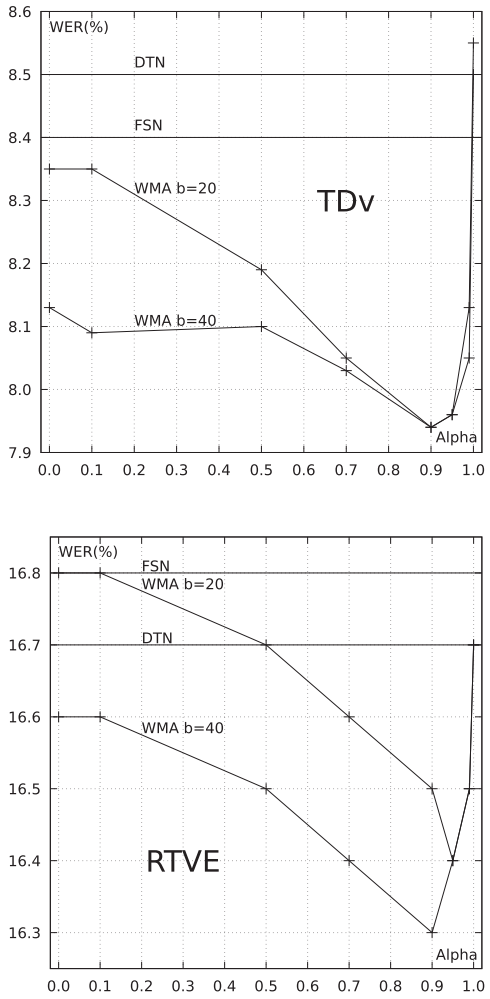


Fig. 6. FSN, DTN and WMA normalization (with different α values) schemes evaluated on WER for video-based tasks.

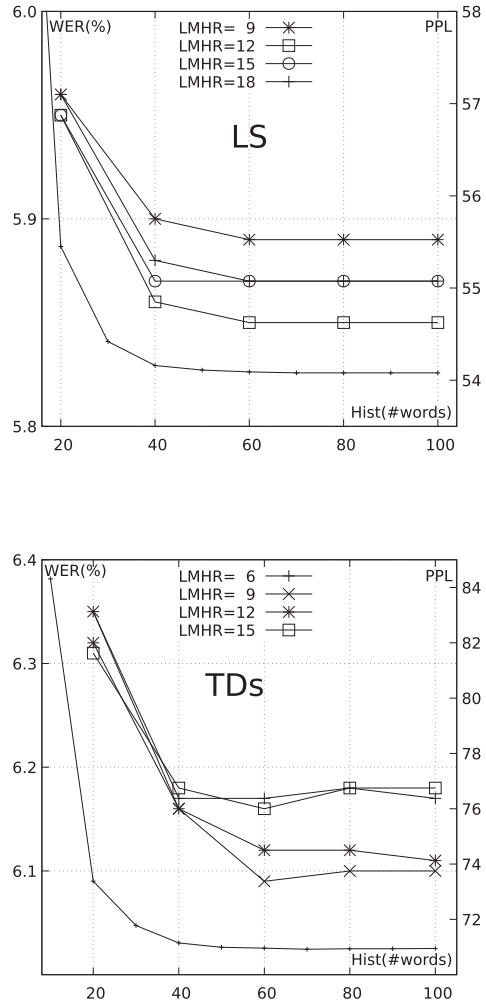


Fig. 7. WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for segment-based tasks. Solid curves represent WER, while the dashed curve is PPL.

videos and RTVE tasks. The capacity of WMA to partially forget the previous context and adapt to new acoustic conditions seems to improve the performance of the recognition as more acoustic variations are likely to appear in long sequences. This is not the case of segment-based tasks shown in Fig. 5, in which WMA did not outperform FSN, since sequences are shorter and acoustic conditions more stable (i.e. usually one sentence with a single speaker). When looking into the parameter α of WMA, it is observed that values close to 1.0 (equally weighting the previous and current batches) benefit segment-based tasks, as this is close to consider the complete sequence for normalization. However, values of α close to 0.9 provide better results in video-based tasks.

Additionally, we assessed the impact of the batch size in the normalization context. In this regard for normalization, unlike AMLA, using a batch size of 40 instead of 20 provided consistently better results along all tasks. Despite of this, taking the WER-latency trade-off into consideration, a batch size of 20 was selected in the following experiments to keep the latency as low as possible.

D. Experiments on Language Modeling for Streaming

As shown in Table III, the TLM provided the best performance measured in terms of PPL for all tasks. This is in line with previous results reported in [5]. For this reason, the following experiments are focused on performing a comprehensive evaluation of the TLM behavior in streaming conditions.

As introduced in Section III, the previous history of word sequences should be limited in order to keep the performance of the streaming decoder. This enforces a limitation in the number of words to consider when computing the LM probabilities that matches the history limitation of the TLM. In addition, the LMHR parameter controls the LM previous context to decide whether hypothesis recombination is performed. Both parameters are interrelated in streaming decoding as shown in the following experiments.

Figs. 7 and 8 plot WER as solid curves (left y-axis) and PPL as a dashed curve (right y-axis), as a function of the TLM history limitation in number of words using different LMHR values for all tasks. To better understand the values of the

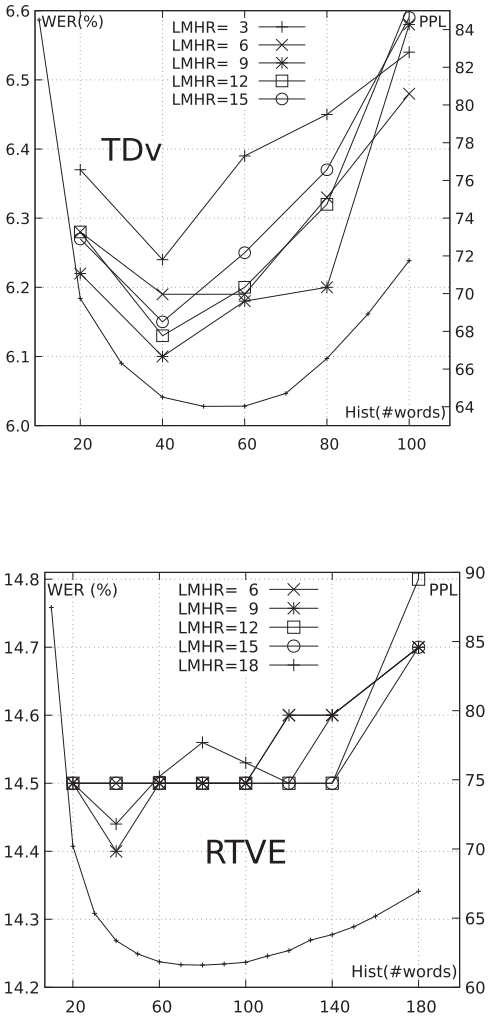


Fig. 8. WER (left y-axis) and PPL (right y-axis) as a function of TLM history limitation and different LMHR values for video-based tasks. Solid curves represent WER, while the dashed curve is PPL.

LMHR parameter, for instance, a LMHR of 3 would indicate that hypothesis recombination is performed at 4-gram level.

Similarly to previous experiments related to acoustic modeling, segment-based tasks in Fig. 7 show a different trend compared to video-based tasks in Fig. 8. Fig. 7 shows that LMHR curves behaved similarly when increasing the TLM history limitation. This limit reached the best operating point in 60 words for LibriSpeech and TED-LIUM matching up with the lowest perplexity in both cases. The best performance was achieved with LMHR values of 12 and 9 for LibriSpeech and TED-LIUM, respectively, while higher values provided slightly higher WERs. This would indicate that for these segment-based tasks, longer histories have very similar contexts that make pruning less effective.

In the case of the video-based tasks, Fig. 8 shows that PPL and WER figures increase beyond a TLM history of about 40 words. It is worth noting that in both TED-LIUM tasks the trained TLM was the same, that is, it was trained from full sentences not complete videos. This would explain why the PPL increased on video-based tasks when more than 60 words were considered for the TLM history. This fact was also reflected in WER, since the best results were consistently achieved using a LMHR value of 9 for both TED-LIUM tasks. The aforementioned optimum operating point of 60 words became on 40 words in video-based tasks as the speech input now is not as structured as in the segment-based tasks, and the beginning and end of sentences can be mixed during decoding, a situation that was not considered when training the TLM.

Regarding the RTVE task, a more stable behavior was observed in performance using a broader range of the TLM history. In this case, the performance only degraded when very high values of TLM history of about 140 words were considered. This might be explained by the fact that this LM was trained with a huge variety of text resources with very different sentence lengths and contexts. However, the performance degradation as a result of longer histories highlights the need of training specific models that take into account intra and inter sentence contexts considering complete videos or documents. Finally, the LM seemed not to play a crucial role in the RTVE task as can be interpreted by the very similar performance achieved by different LMHR values with a slight improvement when using a LMHR value of 9 and a TLM history of 40 words.

As introduced in Section III-C, the LMHP parameter limits the number of active hypotheses that can query the neural LM to obtain its probability score reducing in this way the computational cost. However, the LMHP has a direct impact on WER as it limits the number of hypotheses to be considered during the rescoring step in decoding.

Figs. 9 and 10 depict WER as a function of latency for segment-based and video-based tasks, respectively. The values of the parameter LMHP represent the number of active hypotheses, being LMHP=Inf an unlimited number of active hypotheses. As shown, the use of the LMHP pruning technique achieved an overall reduction in the system latency as can be observed by the left-shifting of the LMHP curves in almost all the LMHP values with respect to an unlimited number of active hypotheses. On the other hand, higher LMHP values translate into better WERs.

Nonetheless, the trade-off between WER and latency is very task dependent. In LibriSpeech, a LMHP value of 20 allows low latencies (about 0.7 seconds) but this limits the best WER to about 6.0%. However, allowing more LM queries (e.g. 40) leads to latencies about 0.9 seconds and WERs of about 5.8%. This behavior is different for TED-LIUM in both versions, where a LMHP value of 20 means an important increase in WER and not so much improvement in latency. This would mean that more queries in this case seems to help the decoder during the search to find best paths. Beyond this LMHP value, competitive WERs were obtained for low latencies getting better results when coming closer to latencies of about one second. In the RTVE task, a LMHP value of 20 provided a very good operating point when latency is closed to 0.8 seconds. Again, in this task the LM did not provide much information so limiting the number of queries only helped the performance of the system in terms of speed. This is specially helpful in this kind of long-recognition tasks where using conservative parameters allows us to discard a high number of active hypotheses during the search speeding

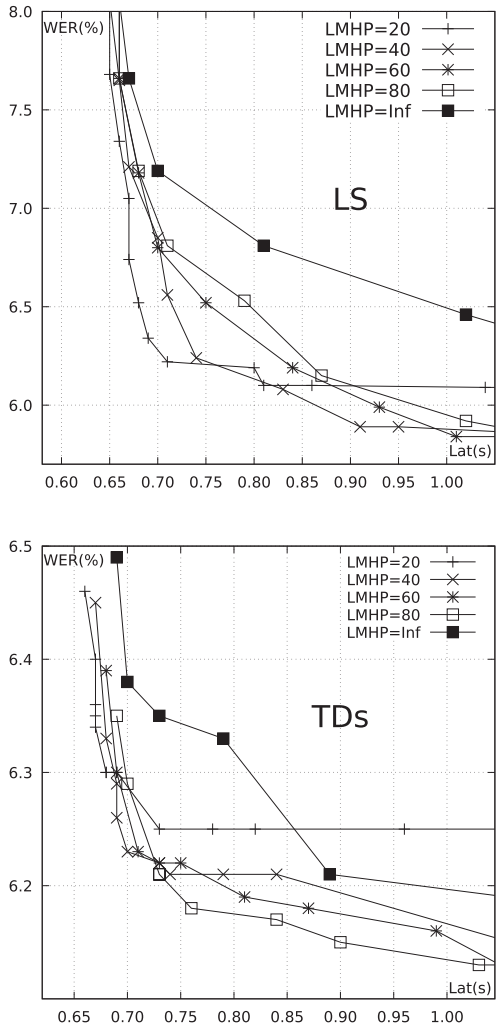


Fig. 9. WER vs. latency in seconds varying LMHP values for segment-based tasks.

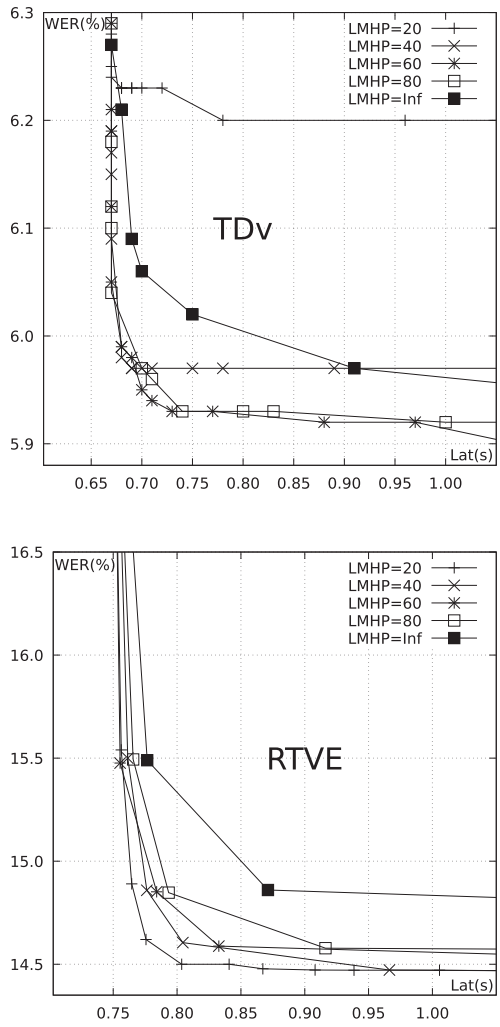


Fig. 10. WER vs. latency in seconds varying LMHP values for video-based tasks.

up the decoding. These results show how the streaming decoder can be adapted very easily to our needs just adjusting the LMHP parameter in order to obtain the desired trade-off between WER and latency.

As described in Section III-D, a relevant feature of the proposed streaming decoder is its capability to interpolate on-the-fly count-based and neural LMs. Extensive experiments were carried out with the aim of evaluating the performance of different LM interpolations combining n-grams, LSTM-RNN LMs and TLMs.

Figs. 11 and 12 show WER as a function of latency applying different combination of LMs for all tasks. Similarly to previous experiments, different behaviors depending on the task can be observed in these figures. In LibriSpeech, the single use of the TLM provided the best result when considering the trade-off between WER and latency. This could be the expected behavior based on the interpolation weights reported in Table III for the TLM (86-96%). The slight improvements in PPL achieved by the interpolation seemed not to have an influence on WER within the considered range of latencies. In the case of TED-LIUM, the weights were distributed in a different way, since n-gram and LSTM-RNN LMs weights were from about 22% to 38% when combined one-on-one with TLM and 30-40% when all the LMs were combined. In this case, while the combination of TLM with n-gram or with LSTM-RNN provided similar performance, the combination of the three LMs consistently provided the best WERs for all the considered range of latencies and for both, segment-based and video-based tasks. In RTVE, no significant differences were found in performance across different LM combinations. As in LibriSpeech, the TLM weight in the interpolation was between 85-94% for RTVE and this seemed to be the reason why the LM interpolation did not have a significant effect on WER.

As a final experiment, the performance of the streaming decoder was evaluated on the test set of all tasks using the hyperparameters optimized on the development sets in the previous experiments. Hyperparameters were optimized aiming at minimizing WER while the average latency was close to 1 s. Table IV reports WER and latency figures on the test sets comparing them with the best WER reported in previous works. In the case of CSC [3] and LC [17], the setup recommended by the authors was

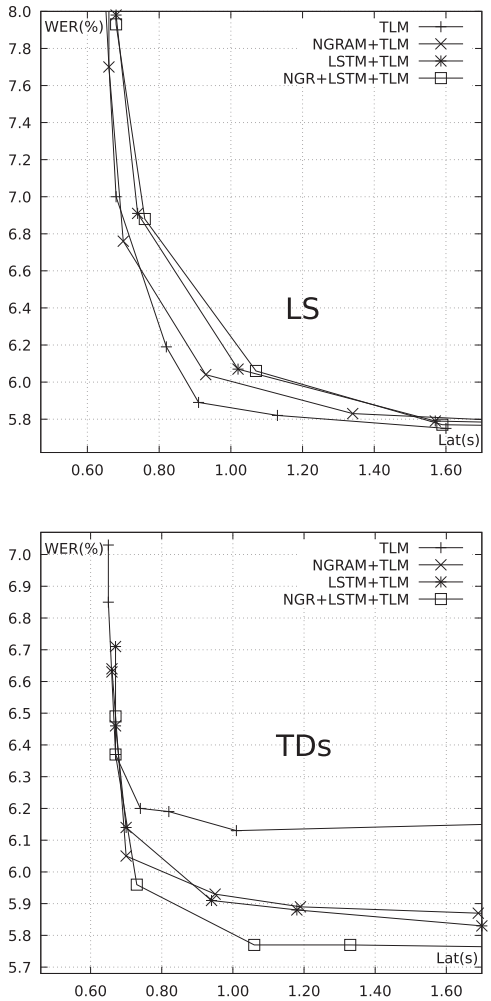


Fig. 11. WER vs. latency in seconds considering different interpolation schemes with TLM for each segmented task.

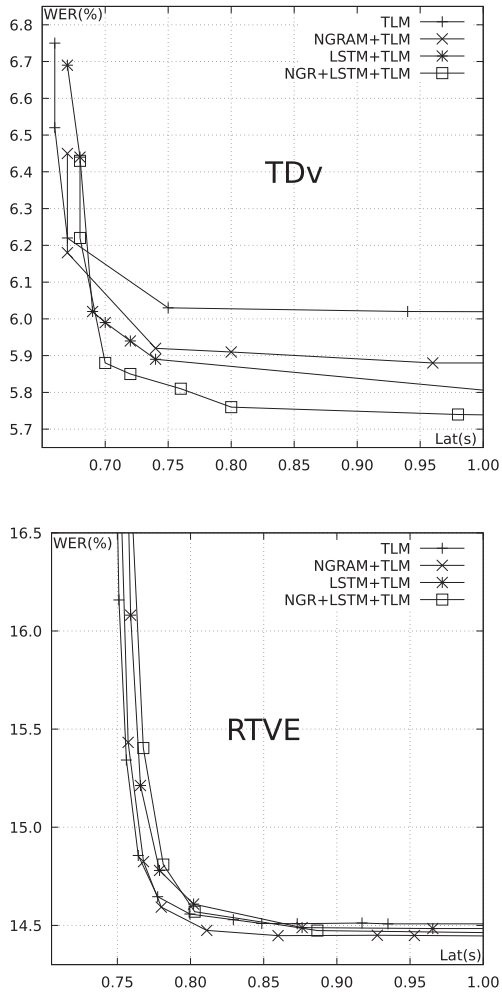


Fig. 12. WER vs. latency in seconds considering different interpolation schemes with TLM for each video task.

TABLE IV
WER AND LATENCY IN SECONDS ON THE TEST SETS USING THE OPTIMIZED STREAMING SYSTEMS FOR ALL THE EVALUATION TASKS COMPARED TO PREVIOUS WORKS

Task	Set	WER	Latency
LibriSpeech			
This work	test-other	6.3	0.93±0.62
CSC		6.8	1.37±0.63
LC		6.7	0.98±0.35
Moritz et al. [7]		7.3–7.6	1.47–2.19
Moritz et al. [7] (offline)		6.1	-
TDNN-F [54] (offline)		8.9	-
TED-LIUM segments (TDs)			
This work	test-legacy	6.4	0.94±0.45
Zhou et al. [55] (offline)		5.6	-
TDNN-F [56] (offline)		8.7	-
TED-LIUM videos (TDv)			
This work	test-legacy	6.2	0.70±0.08
CSC		7.6	0.91±0.11
LC		7.6	0.72±0.12
RTVE			
This work	test	12.4	0.81±0.09
Jorge et al. [57]		16.4	-

properly adapted to our framework. As observed, our streaming decoder offers competitive WERs even compared with offline decoders, demonstrating its applicability to real-world streaming applications.

Regarding latency figures, segment-based tasks, such as LS and TDs, showed a greater variability. This is explained by the fact that pruning was not so aggressive in these tasks in order to minimize WER, leading to latency peaks in some samples in which the decoder could not catch up before the sample ends. However, pruning was easier to adjust to stabilize latency in video-based tasks, as observed in TDv and RTVE.

V. CONCLUSION AND FUTURE WORK

In this work an improved decoder based on the conventional hybrid ASR approach was proposed by adapting state-of-the-art models to the streaming setup. In particular, deep BLSTM acoustic models were adapted to the streaming conditions by using a sliding window of future context. Other techniques such

as on-the-fly normalization of acoustic features and the improvement of pruning techniques related to acoustic and language models were also addressed.

The proposed decoder was evaluated by carrying out a comprehensive experimentation on well-known academic datasets and real-world challenging tasks. As reported, this decoder presented a very competitive performance being easily adapted to the task by tuning the desired trade-off between WER and latency.

Even so, the streaming setup opens some interesting challenges to be further investigated. For instance, in our experiments the same acoustic models were used independently from the window size employed at decoding time in order to alleviate the computational cost of training acoustic models. In order to address this mismatch between training and decoding conditions, the same window size in both training and decoding is desirable to better capture the nature of the task (i.e., segment-based or video-based). Moreover, according to the requirements of the latency, the window size could be dynamically adjusted in the decoding phase. On the other hand, real streaming tasks involve the recognition of long recordings across sentences, in this sense it would be very interesting to evaluate the performance of TLMs taking into account larger contexts.

REFERENCES

[1] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. New York NY, USA: Springer, 2014.

[2] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5/6, pp. 602–610, 2005.

[3] K. Chen and Q. Huo, "Training deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 7, pp. 1185–1193, Jul. 2016.

[4] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2462–2466.

[5] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language modeling with deep transformers," in *Proc. InterSpeech*, 2019, pp. 3905–3909.

[6] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016, Accessed: Dec. 14 2021. [Online]. Available: https://research.google/pubs/pub45446/.

[7] N. Moritz, T. Hori, and J. Le Roux, "Streaming automatic speech recognition with the transformer model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6074–6078.

[8] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online CTC/attention end-to-end speech recognition architecture," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6084–6088.

[9] T.-S. Nguyen, N.-Q. Pham, S. Stueker, and A. Waibel, "High performance sequence-to-sequence model for streaming speech recognition," in *Proc. InterSpeech*, 2020, *arXiv:2003.10022*.

[10] D. Nolden, "Progress in decoding for large vocabulary continuous speech recognition," Ph.D. dissertation, RWTH Aachen Univ., Aachen, Germany, Apr. 2017.

[11] B. Chen, J.-W. Kuo, and W.-H. Tsai, "Lightly supervised and data-driven approaches to mandarin broadcast news transcription," *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, vol. 1, pp. I- 777, 2004.

[12] X. Chen, X. Liu, A. Ragni, Y. Wang, and M. Gales, "Future word contexts in neural network language models," in *Proc. Autom. Speech Recognit. Understanding*, 2017, pp. 97–103.

[13] A. Ogawa, M. Delcroix, S. Karita, and T. Nakatani, "Rescoring N-best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 6099–6103.

[14] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Proc. InterSpeech*, 2011, pp. 2877–2880.

[15] A. Mohamed *et al.*, "Deep bi-directional recurrent networks over spectral windows," in *Proc. Autom. Speech Recognit. Understanding*, 2015, pp. 78–83.

[16] A. Zeyer, R. Schlüter, and H. Ney, "Towards online-recognition with deep bidirectional LSTM acoustic models," in *Proc. InterSpeech*, 2016, pp. 3424–3428.

[17] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. Glass, "Highway long short-term memory RNNs for distant speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 5755–5759.

[18] S. Xue and Z. Yan, "Improving latency-controlled BLSTM acoustic models for online speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 5340–5344.

[19] Y. Shi, W. Zhang, M. Cai, and J. Liu, "Efficient one-pass decoding with NNLM for speech recognition," *IEEE Signal Process. Lett.*, vol. 21, no. 4, pp. 377–381, Apr. 2014.

[20] Z. Huang, G. Zweig, and B. Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6354–6358.

[21] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6364–6368.

[22] K. Lee, C. Park, I. Kim, N. Kim, and J. Lee, "Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR," in *Proc. InterSpeech*, 2015, pp. 2102–2106.

[23] K. Lee, C. Park, N. Kim, and J. Lee, "Accelerating recurrent neural network language model based online speech recognition system," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 5904–5908.

[24] E. Ar1soy, S. F. Chen, B. Ramabhadran, and A. Sethy, "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 1, pp. 184–192, May 2014.

[25] M. Singh, Y. Oualil, and D. Klakow, "Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition," in *Proc. InterSpeech*, 2017, pp. 2720–2724.

[26] J. Jorge, A. Giménez, J. Iranzo-Sánchez, J. Civera, A. Sanchis, and A. Juan, "Real-time one-pass decoder for speech recognition using LSTM language models," in *Proc. InterSpeech*, 2019, pp. 3820–3824.

[27] J. Jorge *et al.*, "LSTM-based one-pass decoder for low-latency streaming," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7814–7818.

[28] P. Baquero-Arnal *et al.*, "Improved hybrid streaming ASR with transformer language models," in *Proc. InterSpeech*, 2020, pp. 2127–2131.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5206–5210.

[30] A. Rousseau, P. Deléglise, and Y. Esteve, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proc. LREC*, 2014, pp. 3935–3939.

[31] H. Ney and S. Ortmanns, "Progress in dynamic programming search for LVCSR," *Proc. IEEE*, vol. 88, no. 8, pp. 1224–1240, Aug. 2000.

[32] A. Cardenal-López, F. J. Diéguez-Tirado, and C. García-Mateo, "Fast LM look-ahead for large vocabulary continuous speech recognition using perfect hashing," *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.* vol. 1, pp. I-705–I-708, 2002.

[33] L. Chen and K. K. Chin, "Efficient language model look-ahead probabilities generation using lower order LM look-ahead information," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2008, pp. 4925–4928.

[34] D. Nolden, H. Ney, and R. Schlüter, "Exploiting sparseness of backing-off language models for efficient look-ahead in LVCSR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2011, pp. 4684–4687.

[35] E. Lleida *et al.*, "RTVE2018 database description," Cátedra RTVE - Universidad Zaragoza, 2018. [Online]. Available: http://catedrartve.unizar.es/reto2018/RTVE2018DB.pd.

[36] M. del Agua *et al.*, "The translectures-UPV toolkit," in *Proc. Adv. Speech Lang. Technol. Iberian Lang.*, 2014, pp. 269–278.

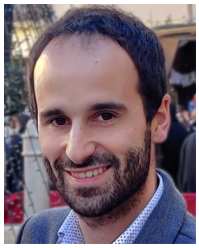
[37] M. Abadi *et al.*, "TensorFlow: a system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

[38] D. S. Park *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. InterSpeech*, 2019, pp. 2613–2617.

[39] A. Giménez, J. Andrés-Ferrer, and A. Juan, "Discriminative bernoulli HMMs for isolated handwritten word recognition," *Pattern Recognit. Lett.*, vol. 35, pp. 157–168, 2014.
[40] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. Workshop Hum. Lang. Technol. Conf.*, 1994, pp. 307–312.
[41] S. Lison, and J. Tiedemann,"OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles," in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, 2016.
[42] UFAL MEDICORP, "UFAL Medical Corpus v.1.0," 2017, Accessed: Dec. 14, 2021. [Online]. Available: https://ufal.mff.cuni.cz/ufal_medical_corpus.
[43] Wikimedia Foundation, "Wikipedia," Accessed: Dec. 14, 2021. [Online]. Available: https://www.wikipedia.org/
[44] C. Callison-Burch *et al.*, "Findings of the 2012 workshop on statistical machine translation," in *Proc. WMT*, 2012, pp. 10–51.
[45] Common Crawl, "News crawl corpus (WMT workshop) 2015," Accessed: Dec. 14, 2021. [Online]. Available: http://www.statmt.org/wmt15/translation-task.html
[46] "Eldiario.es," 2020, Accessed: Dec. 14, 2021. [Online]. Available: https://www.eldiario.es/
[47] "ElPeriodico.com," 2020, Accessed: Dec. 14, 2021. [Online]. Available: https://www.elperiodico.com/
[48] "CommonCrawl 2014," Accessed: Dec. 14, 2021. [Online]. Available: http://commoncrawl.org/
[49] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proc. InterSpeech*, 2002, pp. 901–904.
[50] X. Chen, X. Liu, Y. Qian, M. J. F. Gales, and P. C. Woodland, "CUED-RNNLM - an open-source toolkit for efficient training and evaluation of recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 6000–6004.
[51] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," *Proc. 29th Int. Conf. Mach. Learn.*, vol. 2, 2012, *arXiv:1206.6426*.
[52] X. Chen, X. Liu, Y. Wang, M. J. F. Gales, and P. C. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5401–5405.
[53] M. Ott *et al.*, "Fairseq: A fast, extensible toolkit for sequence modeling," in *Proc. NAACL-HLT*, 2019, pp. 48–53.
[54] K. J. Han, J. Pan, V. K. N. Tadala, T. Ma, and D. Povey, "Multistream CNN for robust acoustic modeling," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6873–6877.
[55] W. Zhou, W. Michel, K. Irie, M. Kitza, R. Schlüter, and H. Ney, "The RWTH ASR system for TED-LIUM release 2: Improving hybrid HMM with specAugment," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7839–7843.
[56] K. J. Han, J. Huang, Y. Tang, X. He, and B. Zhou, "Multi-stride self-attention for speech recognition," in *Proc. InterSpeech*, 2019, pp. 2788–2792.
[57] J. Jorge *et al.*, "MLLP-UPV and RWTH aachen spanish ASR systems for the IberSpeech-RTVE 2018 speech-to-text transcription challenge," in *Proc IberSpeech*, 2018, pp. 257–261.

Javier Jorge received the B.Sc. degree in computer science, in 2014, the M.Sc. degree in artificial intelligence, pattern recognition, and digital imaging, in 2015, and is currently working toward the Ph.D. degree with the Universitat Politècnica de València, Spain, financed by Grant FPU14/03981 from the Spanish Ministry of Education, Culture and Sport. He is a Co-Author of several articles presented at international conferences, and is actively involved in R&D projects (X5gon, Multisubs). His current research interests include streaming speech recognition, specifically decoding with acoustic and language modeling adapted to this environment.

Adrià Giménez received the Ph.D. degree in computer science from Universitat Politècnica de València (UPV), Valencia, Spain, in 2014. He is currently a Postdoctoral Researcher with UPV. He is currently a Member of the UPV's Machine Learning and Language Processing Research Group (MLLP). He has authored or coauthored more than 30 articles in international journals and conferences, and he has been actively involved in three EU research projects and several Spanish research projects. His current research focuses on deep learning for speech recognition.



Joan Albert Silvestre-Cerdà is currently a Ph.D. Assistant Professor of computer science with the Universitat Politècnica de València, Valencia, Spain, - campus d'Alcoi, and a Member of the Machine Learning and Language Processing (MLLP) Research Group, integrated into the Valencian Research Institute on Artificial Intelligence (VRAIN). He has more than ten years of research experience on machine learning and natural language processing applications, mainly in the area of automatic speech recognition, with an special focus on the development of technologies and solutions that can be deployed into real-life production environments. During this time period, he has coauthored more than 20 articles published in international journals and conferences, and has participated in ten publicly-funded EU/Spanish research projects.

Jorge Civera received the Ph.D. degree from Universitat Politècnica de València (UPV), Valencia, Spain, in 2008. He is currently an Associate Professor of computer science with the Universitat Politècnica de València (UPV), and has been a Member of the Machine Learning and Language Processing (MLLP) Research Group since 2014, and also part of the Valencian Research Institute for Artificial INtelligence (VRAIN). He has participated in 30 research projects and has authored or coauthored more than 75 articles in international journals and conferences. He is also an Advisor for three Ph.D. thesis on different MLLP topics. His most recent work include his participation in the EU research projects transLectures, EMMA, and X5gon.

Albert Sanchis received the Ph.D. degree in 2004. He is currently a Ph.D. Associate Professor of computer science with the Universitat Politècnica de València (UPV), Valencia, Spain. He is a Member of the UPV's Machine Learning and Language Processing Research Group (MLLP). He is a Co-Author of more than 60 articles in international journals and conferences. He has participated in six EU research projects and more than ten Spanish research projects. He is also an Advisor for three Ph.D. theses on different MLLP topics. His most recent work includes the participation in the EU projects transLectures, EMMA and X5gon. He is also leading a Spanish government-funded project on fostering open education and parliamentary openness by providing multilingual access to video resources.

Alfons Juan received the Ph.D. degree from Universitat Politècnica de València, Valencia, Spain, in 2000. He is currently a Full Professor of computer science with the Universitat Politècnica de València, where he has been leading a Research Group on Machine Learning and Language Processing (MLLP), since 2014. He has participated in more than 30 research projects and has authored or coauthored more than 150 articles in international journals and conferences. He is also an Advisor for 13 Ph.D. theses on different MLLP topics. His most recent work includes the participation in the EU projects transLectures, EMMA, and X5gon.