CRIM's Speech Recognition system description for OpenASR21 evaluation

Vishwa Gupta
Speech research group
Computer Research Institute of Montreal (CRIM)
Montréal, Canada
vishwa.gupta@crim.ca

Gilles Boulianne
Speech research group
Computer Research Institute of Montreal (CRIM)
Montréal, Canada
gilles.boulianne@crim.ca

***Abstract*—CRIM participated in all the 15 low resource languages and the three languages with case sensitive scoring in the OpenASR21 Challenge for the constrained condition. For acoustic modeling, we developed both hybrid DNN-HMM systems and a conformer based system. We used three different multi-stream acoustic models for decoding: one based on MFCC features alone, second acoustic model based on combined MFCC and conformer embeddings as input, and third one as combined MFCC and VAD embeddings from a DNN-based voice activity detector (VAD) as input. In the final submission, we used two different voice activity detectors for segmenting the development and evaluation audio: one based on a GMM-HMM system, and another one based on a TDNN system.**

For language modeling, we used 4-gram language models followed by lattice rescoring with LSTM-based language models. For language model text, we used the training text from LDC disks when available. We also found significant amount of text over the internet. We used sentence selection with this internet text in order to use it effectively to reduce word error rates (WER). For most languages, we were able to reduce WER with this additional text. Our best results combined six decodes after LSTM LM rescoring: two different voice activity detector based segments, and three different acoustic models (MFCC only, MFCC + conformer embedding, MFCC + VAD embedding).

In the final evaluation, we were ranked second in Tamil, third in Farsi and Javanese, and fourth in seven other languages.

Keywords*—*OpenASR21, low-resource, speech recognition

I. Introduction

The OpenASR21 (Open Automatic Speech Recognition 2021) Challenge set out to assess the state of the art of ASR (Automatic Speech Recognition) technologies under low-resource language constraints. The task consisted of performing ASR on audio datasets in up to 15 different low-resource languages and 3 languages with case sensitive scoring, to produce the recognized written text. Ten languages were carried over from the OpenASR20 challenge [1], and five new languages were added for OpenASR21. A case sensitive scoring was also added for three of these languages: Kazakh, Swahili and Tagalog.

CRIM took part in the constrained condition for all the 15 languages and the 3 languages with case sensitive scoring. In the Constrained Training condition, the only speech data permissible for training is a 10-hour subset of the Build dataset provided for the language being processed, clearly marked for that purpose. Additional text data, either from the provided Build dataset or publicly available resources, is permissible for training in the Constrained Training condition. Any such additional text training data must be specified in sufficient detail in the system description.

For OpenASR20 challenge, two teams achieved very good results [2] [3]. They both used larger training text and larger lexicon from LDC disks for training the language models. These language models gave significant reduction in word error rates for that language. For 13 of the 15 languages in OpenASR21 challenge, CRIM was able to download training text from Linguistic Data Consortium (LDC) builds to augment the language model training text. This additional text had a significant impact on the word error rate (WER) of the development set. We were also able to download significant amount of text for all the languages from the internet. The sources for this text are outlined in the appropriate sections. We had to do sentence selection in order to use the text effectively to reduce the word error rate (WER). The sentence selection reduced the total amount of text available for training significantly. For some languages, even sentence selection did not help reduce WER, so we fell back to LDC text only for language modeling.

II. Dataset and preprocessing

In the constrained condition, the acoustic data available for training an acoustic model is limited to a 10-hour Build dataset provided by NIST for the language being processed, with the corresponding transcripts in UTF-8 encoding. No other acoustic data can be used, either private or public. Training and development lexicons were also provided by NIST. For the 13 languages with LDC disks, we used the expanded lexicon provided in those disks. The lexicons include the words <hes>, <noise>, and <v-noise> to represent the various kinds of noise. Hesitation is actually transcribed, while noise is represented as <sss> and vocal noise is represented as <vns>.

We converted all audio files to 16 bits, 8 kHz sample frequency wave files using NIST sph2pipe or sox.

III. ASR approach

Our system is a hybrid HMM-DNN based on WFSTs (Weighted Finite-State Transducers) and trained with the Kaldi

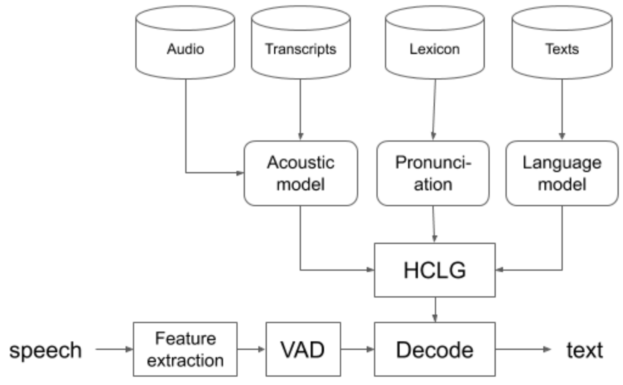

Fig. 1. WFST-based speech recognition.

toolkit [4]. As illustrated in Fig. 1, it is composed of distinct components for acoustic, pronunciation and language modeling, represented as WFSTs and trained separately as described in the following sections[1].

We trained three different hybrid DNN-HMM acoustic models. These hybrid DNN-HMM systems need alignments between the acoustic data and the training transcripts. This initial alignment is generated by a GMM-HMM based system. We trained this GMM-HMM system using the Kaldi recipe in babel egs[2]. This recipe uses 13-dimensional PLP feature parameters, except for Cantonese and Vietnamese. For Cantonese and Vietnamese, we add 3 additional pitch features for a total of 16 features. For generating the segments with noise removed for the development and evaluation sets, we followed the recipe in the babel egs of Kaldi. The resulting segments were aggressive in removing noise segments, as compared to the ones generated by TDNN-based voice activity detector (VAD) described in the following Section.

IV. Voice activity detection (VAD)

To optimize DNN-based speech activity detection, we tried two different TDNN architectures:

1) VAD-TDNN: TDNN as outlined in Chime6 track2 speech activity detection[3] : 40-dimensional MFCC features, 5 TDNN layers and 2 layers of statistics pooling [5], the overall context of the TDNN is about 1 sec, with 0.8 sec of left context and 0.2 sec of right context, with 2 (speech/nonspeech) posteriors, simple Viterbi decoding on an HMM with duration constraints of 0.3 sec for speech and 0.1 sec for silence to decode frames as speech/nonspeech. We added a bottleneck layer with a dimension of 40 just before the last hidden layer to facilitate generation of embeddings. Since every 3rd frame is input to the TDNN, posteriors for three consecutive frames are averaged to generate the desired posteriors for training. This TDNN is trained only from the openASR21 training set separately for each language. For Cantonese and Vietnamese, we added 3 pitch features to better represent the tones in these languages.
2) VAD-TDNN with specAugment layer: We added a specAugment layer after the input layer in the VAD-TDNN described above. This VAD-TDNN gave lower WER than the VAD-TDNN without specAugment layer above, so this VAD was used for final segmentation of development (dev) and evaluation (eval) sets for decoding. This VAD-TDNN gave lower WER than GMM-HMM based VAD for 10 out of 18 languages for the dev set, and 12 out of 18 languages for the eval set. Some example WER differences are shown in the Table I for the dev set. Even though the WER differences are small, the speech segments for the two VAD's are quite different. The GMM-HMM based VAD is much more aggressive (labels more regions as non-speech) than the TDNN-based VAD. For this reason, we used both the VAD's for segmenting the evaluation set. The resulting ctm files were later combined using ROVER [6]. This TDNN-based VAD with specAugment layer was also used to generate 40-dimensional embeddings for training multi-stream TDNN-F acoustic models with combined MFCC + VAD embedding features.

[1]Except for pronunciation probabilities which are not trained here.
[2]https://github.com/kaldi-asr/kaldi/tree/master/egs/babel/s5d
[3]https://chimechallenge.github.io/chime6/track2 software.html

TABLE I. *Comparison of WER for GMM-HMM based VAD versus TDNN-based VAD for segmenting development sets for different languages.*

Language	**GMM-HMM VAD**	**TDNN VAD**
Amharic	39.7%	39.2%
Cantonese	47.9%	48.6%
Farsi	53.8%	53.5%
Georgian	41.2%	41.3%
Kazakh	48.5%	48.2%
Mongolian	50.1%	49.1%

V. Acoustic model

For each language, we trained three different acoustic models based on a multi-stream convolutional neural net (CNN) architecture outlined in [7]. We varied this architecture to get the lowest possible WER for three different features: 40-dimensional MFCC's, 40-dim MFCC + 40-dim conformer embeddings from a conformer model trained for the same language, 40 dim MFCC + 40-dim embeddings from the TDNN-VAD trained in Section IV for the same language. The details of optimisation are outlined below:

A. *Multi-stream acoustic models with 40-dimensional MFCCs*

In the proposed multi-stream CNN architecture in [7], the input features are processed by 5 CNN layers in a single stream. This single stream is then branched out into 3 streams with 17 TDNN-F layers in each stream and with different dilation rate in each stream. The input MFCC features to the single stream are first converted into filter-bank features by IDCT (inverse-discrete cosine transform), followed by SpecAugment and then followed by 5 2D-CNN layers. The output of CNN is fed to 3 different streams with a stack of 17 TDNN-F layers in each stream. We experimented with one, two or three streams, and also with the number of TDNN-F layers in the stream.

We also compared multi-stream architecture with TDNN-F architecture with 17 layers in the Kaldi librispeech egs[4] [8] (lines 1 and 2 in Table II). As can be seen from Table II, the best multi-stream architecture has 2-streams, 12 TDNN-F layers per stream with the dimension of the output layer reduced by half. This architecture was used for training 40-dim MFCC acoustic models for all the 15 languages and the three case-sensitive scoring languages.

TABLE II. *Comparison of WER for Amharic dev set for different TDNN-F and multi-stream architectures. All the architectures use the same language model (3-gram from OpenASR21 build).*

Architecture	**WER**
1. TDNN-F (17 layers)	57.5%
2. TDNN-F (1) with specAugment layer	54.2%
3. one-stream of multi-stream CNN	52.4%
4. reduce output layer dim by half in (3)	52.3%
5. reduce 17 TDNN-F layers to 12 in (4)	52.26%
6. reduce TDNN-F layer dimensions by half in (5)	54.0%
7. multi-stream (6)	52.8%
8. multi-stream (5)	52.17%
9. 2-stream (5)	51.94%

B. Multi-stream acoustic models with 40-dimensional MFCCs plus 40-dimensional conformer embeddings

We experimented with many different acoustic model architectures in order to get the lowest possible WER with conformer embeddings. We tried embeddings from the conformer model (Section V-D) alone, and with 40-dimensional MFCC features. The architecture that worked the best is the 2-stream architecture in line 9 of Table II with a third stream with 40-dim conformer embeddings as input to this stream. The conformer embeddings are produced every 40 msec, so each frame is duplicated 4 times. We varied the number of TDNN-F layers in this third stream for conformer embeddings, and we found that 6 TDNN-F layers worked the best. We used this architecture for all the languages. The various experiments are summarized in the Table III. If we replace MFCC's with conformer embeddings in line 2 of Table II, then the WER goes up from 54.2% to 62.9%. Table III uses a 4-gram language model from LDC build text, so the WERs are significantly lower than in Table II. Note that 45.04% WER in Table III corresponds to 40.4% WER for 40-dim MFCC features without the 3rd stream for conformer embeddings.

TABLE III. *Comparative WER for Amharic dev set for different multi-stream architectures with conformer embeddings input to the third stream. All the architectures use the same language model (4-gram from LDC build).*

Architecture	**WER**
1. one TDNN-F layer in 3rd stream	48.55%
2. 6 TDNN-F layers in 3rd stream	45.04%
3. 12 TDNN-F layers in 3rd stream	45.83%

C. Multi-stream acoustic models with 40-dimensional MFCCs plus 40-dimensional TDNN-VAD embeddings

We also trained acoustic models with 40-dim MFCCs concatenated with 40-dimensional embeddings from TDNN-based voice activity detector (VAD) (VAD-TDNN with specAugment layer in Sec. IV). The VAD embeddings are generated every 10 ms. The idea here is that VAD embeddings will provide a gradual transition from silence frames to voiced frames and may lead to better results, specially for noisy utterances. The same 3-stream architecture was used as for MFCC plus conformer embeddings: the third stream had VAD embeddings as input and there were 6 TDNN-F layers in the third stream. The comparative results between MFCC features and MFCC + VAD features are shown in Table IV for some of the languages. Note that for Farsi, the VAD embedding features reduce WER by 0.4% absolute.

TABLE IV. *Comparison of dev set WER for MFCC features versus MFCC+VAD embeddings features for some of the languages. The same language model (4-gram from LDC build) is used.*

Language	**40-dim MFCCs**	**40-dim MFCCs + VAD embeddings**
Amharic	39.7%	39.85%
Cantonese	47.9%	48.54%
Farsi	53.8%	53.38%
Georgian	41.2%	41.72%
Guarani	42.8%	43.39%
Javanese	54.0%	54.19%

D. Conformer embeddings

To generate these embeddings we used a Conformer model [9], a transformer-based architecture augmented with convolutional input layers that we trained from scratch on each language, with a LF-MMI criterion [10]. We based our implementation on the snowfall k2-fsa[5] version. Features were filterbanks with 80 mel bins. The default model sizes were reduced to 6 encoder layers, 4 attention heads, and the bottleneck and hidden dimensions to the embedding dimension of 40. We performed data augmentation with 5 speed perturbation values $[0.8, 0.9, 1.0, 1.1, 1.2]$ but no other data augmentation such as SpecAugment or noise/music/reverberation. We ran training for 50 epochs for all languages, with 3000 warmup steps. When extracting the embeddings, we averaged the model over the last 5 epochs, except for Cantonese for which the latest epoch was 15.

VI. Language model

For language modeling in the constrained condition, we could use any text publicly available over the internet. For 13 of the 15 languages, we used the LDC IARPA Babel language packs from 2016 to 2020, taking care to exclude transcriptions of any recording that appeared in OpenASR21 build or dev set (we kept transcriptions for 7061 out of 8597 Babel recordings). Only Farsi and Somali did not have a Babel language pack. We used the training text and the expanded lexicon (from the LDC build directory) to enhance our text and lexicon for language modeling. This text together with the larger lexicon had a significant impact on WER for all the languages involved. This text is from conversational speech, probably transcribed by the same group of transcribers with

[4]https://github.com/kaldi-asr/kaldi/egs/librispeech/s5/local/chain/run_tdnn.sh

[5]https://github.com/k2-fsa/snowfall

the same transcribing instructions, so the text is probably quite consistent in transcription and has a significant impact on WER. Table V shows the impact of LDC build on WER of the dev set for some of the languages. For example, for Amharic, the WER goes down from 51.7% to 39.7%, while for Georgian, the WER goes down from 53.4% to 41.2%.

TABLE V. *Comparative dev set WER for language model (LM) from OpenASR21 build versus LM from LDC build. The 2-stream acoustic models use 40-dim MFCC features.*

Language	OpenASR21 build	LDC build
Amharic	51.7%	39.7%
Georgian	53.4%	41.2%
Guarani	52.5%	42.8%
Javanese	58.9%	54.0%
Kazakh	55.6%	48.5%

We also used two other sources of publicly available text, collected for machine translation research. Monolingual NewsCrawls[6] [11] is extracted from online newspapers, and CommonCrawl[7] [12] from web pages. The amount of text in NewsCrawls (NC) and CommonCrawl (CC) for each language is shown in Table VI. We also found additional text which appeared more relevant to our data: 22 million words for Amharic in DKE[8] [13] and 230,000 words in the Hong Kong Cantonese corpus[9] [14].

If we take Amharic as an example, after downloading the text data from NewsCrawls, there was only a small amount of preprocessing to do (punctuation marks have their own special Amharic writing). But as soon as we add any amount of this new text to the acoustic training text, the perplexity on dev set goes up: from 296 to 725 if we use only the smallest of the 3 years of news, using larger texts the perplexity gets even worse.

To reduce domain mismatch between conversational speech and the news sources, we used sentence selection [15]. This method selects a set of sentences from the out-of-domain texts such that it has a distribution as similar as possible to the overall in-domain distribution, rather than just match its peak. We tried sentence selection using the acoustic training text as the in-domain data and the news text as out-of-domain; perplexity on dev set is better but still worse than using just the original acoustic text (315 with our best combination of sentence selection hyper-parameters).

We get slightly smaller degradations of perplexity when adding sentences selected with LDC + acoustic training texts. But still, as soon as we add any amount of NewsCrawls or DKE texts, the perplexity gets worse. We finally selected 233K words of text from NewsCrawls and DKE text for Amharic through sentence selection. This additional text resulted in WER reduction from 38.46% (using LDC text) to 37.8% (using LDC + 233k words) after LSTM LM rescoring of decoded lattices (forward LSTM LM rescoring followed by backward LSTM LM rescoring). The WER reduction is good enough

TABLE VI. *Amount of text (millions of words) in NewCrawls (NC) and CommonCrawl (CC).*

Language	NC	CC
Amharic	8.1	67.4
Cantonese	24.6	0.0
Farsi	61.1	0.0
Georgian	0.0	460.1
Guarani	0.0	1.0
Javanese	0.0	23.4
Kazakh	34.9	471.1
Kurmanji-kurdish	0.0	65.9
Mongolian	0.0	245.9
Pashto	17.9	95.7
Somali	13.2	62.6
Swahili	18.2	272.0
Tagalog	6.1	562.3
Tamil	17.9	580.3
Vietnamese	0.0	875.6

that we used LSTM LM generated from this augmented text for our final submissions. We also tried sentence selection with NC+CC text for Amharic. Using 1.18 million words of text from LDC+NC+CC for training LSTM language models (LM), the WER went up from 38.0% to 38.1%.

The LSTM language models were trained using the recipe in Kaldi swbd egs[10] with reduced dimensions. Basically, we used a 2-layer LSTM LM with both the cell dimension and embedding dimension of 256. When rescoring the lattices generated by forward LSTM LM with the backward LSTM LM, the memory used becomes very large and many lattice rescoring attempts fail. To avoid these failures, we reduced the ngram_order during backward rescoring from 4 to 3. This change merges histories in the lattice if they share the same 3-gram history and this prevents the lattice from exploding exponentially. This change avoided memory overflow, and significantly reduced the compute times without any significant impact on WER. It also allowed us to finish all the computing for decoding the evaluation audio in time.

For other languages, we extracted text from NC and CC with sentence selection. The amount of text selected from each and the total amount used for training LSTM language models appears in Table VII. We rescored the decoded lattices with the LSTM language models. The results are shown in Table VIII. For seven languages, plus for the 3 case sensitive scoring languages, we were able to reduce the word error rates. So for these languages, we used the LSTM LM language models from text obtained with NC+CC text after sentence selection. Note that for all the case sensitive scoring languages, we got significant reduction in WER.

VII. Combining Multiple Decodes using ROVER

In the past, we have found that combining multiple ctm files with ROVER [6] results in a lower WER than combining two lattices and then doing MBR decoding. So we generated 6 different ctm files[11] using two different voice activity detectors (GMM-HMM VAD and TDNN VAD), and three different acoustic models: 40-dimensional MFCC based 2-stream

[6]http://data.statmt.org/news-crawl
[7]http://data.statmt.org/cc-100/
[8]https://wwwiti.cs.uni-magdeburg.de/iti_dke/Datasets
[9]http://compling.hss.ntu.edu.sg/hkcancor

[10]https://github.com/kaldi-asr/kaldi/egs/swbd/s5c/local/rnnlm/run_tdnn_lstm.sh
[11]ctm files contain time synchronous word sequence of the decoded audio

TABLE VII. *Number of words (in thousands) selected from each source, and total used in language model.*

Language	ASR21	LDC	NC	CC	Total
Amharic	80	238	150	83[a]	550
Cantonese	123	844	307	20[b]	1293
Farsi	77	0	124	0	201
Georgian	86	259	0	810	1154
Guarani	85	261	0	115	462
Javanese	85	263	0	317	665
Kazakh	78	235	206	145	665
Kazakh_css	83	252	189	130	654
Kurmanji-kurdish	100	285	0	576	960
Mongolian	109	328	0	612	1048
Pashto	123	826	948	728	2625
Somali	104	0	55	53	212
Swahili	82	243	188	135	648
Swahili_css	84	253	202	150	690
Tagalog	85	597	347	215	1244
Tagalog_css	97	618	353	237	1304
Tamil	93	451	211	104	859
Vietnamese	137	919	0	2510	3566

[a]For Amharic, this is selected from the DKE corpus.
[b]For Cantonese, this is selected from the Hong Kong Cantonese corpus.

TABLE VIII. *Comparison of WER for dev sets for all the languages after rescoring lattices with LSTM LM trained with LDC text only versus LDC+ (NC+CC text after sentence selection).*

Language	LSTM LM LDC text	LSTM LM LDC+NC+CC text
Amharic	38.46%	37.8%
Cantonese	46.47%	45.97%
Georgian	40.27%	40.42%
Farsi	52.96%	52.71%
Guarani	41.84%	41.73%
Javanese	52.97%	53.50%
Kazakh	46.78%	46.89%
Kazakh_css	54.43%	52.22%
Kurdish-kurmanji	64.88%	64.91%
Mongolian	47.88%	47.73%
Pashto	47.14%	46.98%
Somali	58.65%	58.93%
Swahili	36.06%	35.95%
Swahili_css	50.21%	47.55%
tagalog	44.11%	44.61%
tagalog_css	47.71%	46.14%
Tamil	60.67%	60.51%
Vietnamese	47.71%	48.21%

TDNN-F, 40-dimensional MFCC + 40-dimensional conformer based 3-stream acoustic models, and 40-dimensional MFCC + 40-dimensional VAD embedding based 3-stream acoustic models. The 6 decoded ctm files were then combined using ROVER. The results on the evaluation set for all the languages are shown in the Table IX. In this Table, the various columns are explained below:

1) The 2nd column shows the best single decode word error rate (WER). The best decode is either the TDNN-VAD based segments with 2-stream TDNN-F with 40-dim MFCC features, or GMM-HMM based VAD segments with 2-stream TDNN-F acoustic models with 40-dim MFCC features (with a 4-gram LM). Since we could have up to 5 submissions for the eval set for each language, we used first two submissions to find out which one in better.
2) Column 3 shows WER after LSTM LM rescoring (both forward followed by backward LSTM LM rescoring of decoded lattices). The text used for training the LSTM LM language models is shown in the last column. Except for case sensitive scoring of Swahili, we got a significant reduction in WER for other languages. The best reduction in WER was 2.5% for Pashto. This was our 3rd submission.
3) Column 4 shows ROVER of 5 decodes (except for Farsi): 2 decodes with 2-stream TDNN-F with 40-dim MFCC features, with TDNN-VAD segments and with GMM-HMM VAD segments, 2 decodes with 3-stream TDNN-F with 40-dim MFCC + 40-dim VAD TDNN embeddings, with TDNN-VAD segments and with GMM-HMM VAD segments, and 1 decode with 3-stream TDNN-F with 40-dim MFCC + 40-dim conformer embeddings with eval set segmented with GMM-HMM based VAD. This was our 4th submission.
4) Column 5 shows WER after ROVER of 6 decodes: 5 decodes from column 4 plus the decode with 3-stream TDNN-F with 40-dim MFCC + 40-dim conformer embeddings with segments from TDNN-VAD. Except for case sensitive scoring of Kazakh, we got over 1.0% reduction in WER with ROVER. The best WER reduction was 2.2% for Swahili case sensitive scoring. This was our 5th submission.
5) Column 6 is the best WER in the leaderboard for the eval set, while column 7 is CRIM's rank among all participants, and column 8 is the text used for language modeling as described in the language modeling section. CRIM came 2nd in Tamil, 3rd in Farsi and Javanese, and 4th in 7 other languages.
6) For Farsi, the decodes with 2-stream TDNN-F with 40-dim MFCC features failed. Basically, 9 audio files had empty segments, so the scoring server did not accept the submission. However, decodes with 3-stream acoustic models with 40-dim MFCC + 40-dim VAD embeddings, and with 40-dim MFCC + 40-dim conformer embeddings did not have any empty decodes and they were accepted by the scoring server. The best decode was with 40-dim MFCC + 40-dim VAD embeddings with TDNN-VAD segments. The *ROVER of 5* in column 4 actually corresponds to ROVER of 4 ctm files for Farsi.

VIII. Resources needed

The compute server at CRIM is linux based and has 3 sets of 8 machines bought at different times. One set of machines has Intel(R) Core(TM) i9-7980XE CPU @ 2.60GHz, and 128 GBytes of memory. Each of these machines has two GTX 1080 Ti GPUs. The LF-MMI ASR training algorithms used 4 GPUs in parallel and the total elapsed times (by adding up elapsed times for all the processes) are shown in Table X for training with the 40-dimensional MFCC features. The Table also shows the total elapsed times for discriminative training that follows the LF-MMI training. Except for Amharic, the

TABLE IX. *Comparison of WER for eval sets for all the languages after final submission.*

Lang	best decode	LSTM LM	rover 5	rover 6	best WER	rank CRIM	text src
amh	46.1	44.1	43.3	43.1	39.9	4	nc+dke
cant	45.9	44.1	42.8	42.8	37.6	5	nc+cc
farsi	80.4		79.3		68.0	3	nc+cc
georg	46.1	44.6	43.1	42.55	39.2	4	ldc
guar	49.1	47.7	46.3	46.0	42.6	5	nc+cc
java	55.1	53.7	52.3	52.0	48.1	3	ldc
kaz	59.3	57.8	57.4	56.9	50.0	5	ldc
kaz css	61.1	59.0	58.8	58.6	49.8	3	nc+cc
kurm	69.3	67.2	66.0	65.7	61.7	4	ldc
mong	49.8	47.8	46.7	46.0	41.0	7	nc+cc
pashto	51.6	49.1	47.7	47.2	43.2	4	nc+cc
somali	62.4	61.0	59.8	59.2	55.6	5	ASR21
swa	38.3	36.4	35.2	35.0	32.4	4	nc+cc
swa css	51.0	50.7	49.0	48.5	43.5	3	nc+cc
tag	47.4	45.2	43.6	43.2	40.4	4	ldc
tag css	55.8	54.8	53.7	53.2	46.2	3	nc+cc
tamil	67.1	65.0	64.0	63.8	62.3	2	nc+cc
viet	47.6	46.1	44.4	44.0	40.3	4	ldc

TABLE X. *Total elapsed times in hours for LF-MMI training of the 2-stream TDNN-F models with 40-dim MFCCs followed by discriminative training of the trained LF-MMI models.*

Language	LF-MMI	Discriminative
Amharic	15.6	388.9 (4.6+320+0.3+64)
Cantonese		380 (336+ 1+ 143)
Farsi		141.4 (49+0.4+ 92)

LF-MMI training logs were deleted, so we cannot give those training times.

The LF-MMI training for Amharic took 15.6 hours, and the discriminative training on top of LF-MMI training took 388.9 hours: 4.6 hours for discriminative training from generated degs (discriminative egs) files, 320 hours for computing alignments for the training audio from LF-MMI models (we used 60 parallel jobs with a maximum elapsed time of 8 hours for the longest job), 0.3 hours for computing the discriminative egs files, and 64 hours for computing the denominator lattices (20 jobs were run in parallel). Some of the corresponding times are shown for Cantonese and Farsi also. The alignments for Farsi took only 49 hours compared to 320 hours for Amharic and 336 hours for Cantonese. The longest alignment for Farsi took 1.2 hours.

The decoding elapsed times are shown in the Table XI. The actual elapsed wall time is much smaller, as the decoding was run on 20 CPUs. No GPU was used for decoding.

Note that the third column includes the times for both forward and backward rescoring of decoded lattices, and the two rescores take a significant amount of time. The ROVER of 4 and 5 decodes is much longer as it includes LSTM LM rescoring for all 5 or 6 decodes. So if we multiply line 3 times by 5 or 6, we will get the approximate times for ROVER of 5 or 6 decodes. Since the decodes use 20 CPUs in parallel, the elapsed wall clock times are significantly shorter. The decoding times are also affected by the voice activity detector (VAD). For example, the GMM-HMM based VAD is much more aggressive than the TDNN based VAD, so the decode times with GMM-HMM VAD based segments is about half that for TDNN VAD based segments.

TABLE XI. *Total elapsed times in hours for decoding of the eval sets for single decode, LSTM LM decode, rover of 5, and ROVER of 6 as submitted for evaluation.*

Language	Best decode	LSTM LM	rover 5	rover 6
Amharic	4.8	14.5	65.7	76.5
Cantonese	4.8	12.1	75.0	95.9
Farsi	5.3			
Georgian	3.9	11.75	65.7	78.5
Guarani	4.3	11.75	64.1	75.9
Javanese	7.2	33.4	115.6	141.0

The LSTM LM backward rescoring takes around 100 gigabytes of memory. The rest of the decodes use much smaller memory, in the order of a few gigabytes.

Training the Conformer model used for embeddings took 8 hours 20 minutes of elapsed GPU time on average for each language; extraction of the embeddings took a total of 40 minutes for all languages.

IX. CONCLUSION

CRIM participated in all the 15 low resource languages and the three languages with case sensitive scoring in the OpenASR21 Challenge for the constrained condition. For acoustic modeling, we developed both hybrid DNN-HMM systems and a conformer based system. We used three different multi-stream acoustic models for decoding: one based on MFCC features alone, a second acoustic model based on combined MFCC and conformer embeddings as input, and third one as combined MFCC and VAD embeddings from a DNN-based voice activity detector (VAD) as input. The big improvement in acoustic modeling was the use of 2-stream CNN based TDNN-F system with 5 CNN layers followed by two streams of 12 TDNN-F layers instead of the traditional single-stream TDNN-F system with 17 TDNN-F layers. In one experiment this 2-stream architecture reduced the WER from 57.5% to 51.9% for Amharic development set.

In the final submission, we used two different voice activity detectors for segmenting the development and evaluation audio: one based on a GMM-HMM system, and another one based on a TDNN system. The TDNN-based VAD gave lower WER for 12 of the 18 languages, while the GMM-HMM based VAD gave lower WER for the other languages. Also GMM-HMM VAD is much more aggressive than the TDNN-based VAD (shorter segments). The decoded outputs from the two VAD's when combined probably contributed significantly to the reduction in WER.

The biggest reduction in WER probably came from using a much larger language model training text from LDC builds and the larger lexicon in this LDC build. In one experiment, the WER for Amharic dev set went down from 51.7% (with OpenASR21 build) to 39.7% with LDC build.

We found significant amount of text over the internet for each language. However, if we use this text for language modeling without any sentence selection, then the WER goes up. So we had to be very aggressive in sentence selection so that the extracted text actually led to a small reduction in word error rate (WER) for many languages.

Combining multiple decodes using ROVER also led to significant reduction in WER, as much as 2.1% for Vietnamese. The systems being combined were quite diverse: using conformer embeddings in two systems and VAD embeddings in two other systems.

All the above improvements led to CRIM ranking second in Tamil, third in Farsi and Javanese, and fourth in seven other languages for the constrained condition on the eval set leaderboard.

Acknowledgments

The authors would like to thank Ministry of Economy and Innovation (MEI) of the Government of Quebec for the continued support.

References

[1] K. Peterson, A. Tong, and Y. Yu, "OpenASR20: An Open Challenge for Automatic Speech Recognition of Conversational Telephone Speech in Low-Resource Languages," in *Proc. Interspeech*, 2021, pp. 4324–4328.

[2] T. Alumäe and J. Kong, "Combining Hybrid and End-to-end Approaches for the OpenASR20 Challenge," in *Proc. Interspeech*, 2021, pp. 4349–4353.

[3] J. Zhao, Z. Lv, A. Han, G.-B. Wang, G. Shi, J. Kang, J. Yan, P. Hu, S. Huang, and W.-Q. Zhang, "The TNT Team System Descriptions of Cantonese and Mongolian for IARPA OpenASR20," in *Proc. Interspeech*, 2021, pp. 4344–4348.

[4] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[5] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic Modelling from the Signal Domain Using CNNs," in *Proc. Interspeech*, 2016, pp. 3434–3438.

[6] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *Proc. ASRU*, 1997, pp. 347–352.

[7] K. J. Han, J. Pan, V. K. N. Tadala, T. Ma, and D. Povey, "Multistream Cnn for Robust Acoustic Modeling," in *Proc. ICASSP*, 2021, pp. 6873–6877.

[8] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohamadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proc. Interspeech*, 2018, pp. 3743–3747. [Online]. Available: http://www.danielpovey.com/files/2018_interspeech_tdnnf.pdf

[9] A. Gulati, J. Qin, C. C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[10] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016, pp. 2751–2755.

[11] A. Birch, B. Haddow, I. Titov, A. Valerio, M. Barone, R. Bawden, S. Felipe, M. L. Forcada, M. Espl, M. S, J. A. P, W. Aziz, A. Secker, and P. V. D. Kreeft, "Global Under-Resourced Media Translation (GoURMET)," in *Proc. Machine Translation Summit XVII*, vol. 2, 2019, pp. 122–122.

[12] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale," in *Proc. ACL*, 2020, pp. 8440–8451.

[13] A. M. Gezmu, B. E. Seyoum, M. Gasser, and A. Nürnberger, "Contemporary {A}mharic Corpus: Automatically Morpho-Syntactically Tagged {A}mharic Corpus," in *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, 2018, pp. 65–70. [Online]. Available: https://www.aclweb.org/anthology/W18-3809

[14] K.-K. Luke and M. L. Y. Wong, "The Hong Kong Cantonese corpus: design and uses," *Journal of Chinese Linguistics*, vol. 25, no. 2015, pp. 309–330, 2015.

[15] A. Sethy, P. G. Georgiou, B. Ramabhadran, and S. Narayanan, "An iterative relative entropy minimization-based data selection approach for n-gram model adaptation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 13–23, jan 2009.