Scaling Armenian ASR: A Semi-Automated Approach to Building High-Performance Speech Recognition Systems for Low-Resource Languages

Varuzhan Baghdasaryan
National Polytechnic University of Armenia
Yerevan, Armenia
e-mail: varuzh2014@gmail.com

Robert Hakobyan
National Polytechnic University of Armenia
Yerevan, Armenia
e-mail: robhakobyan@gmail.com

***Abstract*— This paper presents an approach to building automatic speech recognition (ASR) for Armenian, a low-resource language. We created a semi-automated pipeline that collected ~8,000 hours of speech from YouTube and other public sources, validating ~5,350 hours. Transcriptions were generated with a base ASR model and refined with a statistical language model trained on 17M text lines, followed by manual filtering.**

We trained two architectures adapted from NVIDIA NeMo: a lightweight Streaming Citrinet for low-resource settings and a higher-capacity FastConformer-CTC. Audio preprocessing included noise cancellation and VAD-based chunking for long recordings and implicit language detection. A multilingual pipeline with CNN+RNN-based language ID further enabled transcription across six languages, including Armenian.

Our system achieved a best WER of 8.56%, contributing both improved ASR performance and new resources for Armenian NLP, with a framework applicable to other low-resource languages.

Keywords*— *Armenian ASR, speech recognition system, speech-to-text, NVIDIA NeMo, Citrinet, FastConformer, Armenian speech corpus, language identification, voice activity detection, multilingual speech processing, low-resource languages.

I. Introduction

Armenian automatic speech recognition (ASR) has received little global attention due to its low-resource status and lack of large-scale datasets. Our earlier work addressed this gap by creating the first systematic foundation for Armenian ASR through dataset development, model benchmarking, and post-processing solutions [1–5].

We introduced the ArmSpeech corpus (15.7 h) from diverse Armenian sources, later expanded with the Speech Corpus of Armenian Q&A Dialogues and Google's FLEURS, reaching 34.84 h [4]. Using this dataset, we compared DeepSpeech, QuartzNet, and Citrinet, showing Citrinet's superiority with 19.41% WER without a language model [4]. Motivated by these results, we optimized Citrinet, reducing WER to 13.4% with a 23-block configuration and achieving 18.2% with a faster 16-block model [4,5].

To support large-scale evaluation and crowdsourced data acquisition, we launched armspeech.com, which added 21.53 h of speech, expanding the combined dataset to 79.37 h [5]. For downstream usability, we developed an Armenian-specific punctuation and capitalization system based on DistilBERT, trained on 11.17M sentences, handling unique orthographic rules absent in multilingual models [5].

The current work builds on this foundation, scaling data collection to 8,000 h with a semi-automated pipeline and extending functionality with multilingual transcription and implicit language detection. These advances move beyond monolingual Armenian ASR toward a comprehensive multilingual solution.

II. Datasets

We developed a semi-automated pipeline for large-scale Armenian speech data collection and processing, designed to handle thousands of hours of audio while ensuring high quality through automated transcription and manual validation (Figure 1). The system collected open Armenian videos from YouTube, standardized audio to 16-bit mono WAV at 16 kHz, and generated initial transcriptions with our best-performing Citrinet model (13.4% WER [5]) using GPU-based batch inference. Predictions were refined with a statistical language model trained on 17M text lines [3–5], applying beam search (beam width = 512) for improved accuracy.

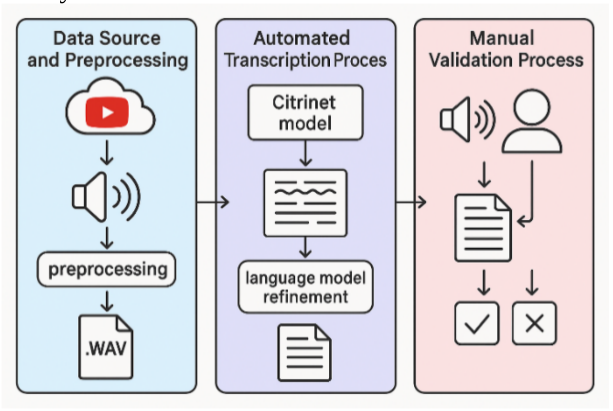

Data Source and Preprocessing
preprocessing
.WAV
Automated Transcription Proces
Citrinet model
language model refinement
Manual Validation Process

Figure 1. Semi-automated pipeline for large-scale Armenian speech data collection, transcription, and validation

Validation combined efficiency and quality control: human annotators reviewed audio–transcription pairs, marking them as valid or invalid, with minor errors corrected directly. Five volunteers validated thousands of hours of data in a few months. Approximately half the dataset was also transcribed with Google's Chirp model [6], which favored grammatically correct outputs, normalizing dialectal forms to standard Armenian.

Following the established ArmSpeech corpus format [1, 2], the dataset was stored using CSV file format for efficient data management and compatibility with existing ASR frameworks. The dataset is divided into subsets following a standard 70-20-10% split ratio for training, validation, and testing respectively. The dataset is organized into three main CSV files: train.csv, which contains information for the training set and represents 70% of the total data; validation.csv, used for model evaluation and comprising 20% of the data; and test.csv, which holds the test set for final model assessment and accounts for the remaining 10%. Each CSV file maintains a consistent structure with the following columns:

- wav_filename: audio clip name (relative path)
- wav_filesize: sample size in bytes
- transcript: punctuated and capitalized transcription
- raw_transcript: normalized transcription without punctuation
- language: language identifier (e.g., "HY" for Armenian, "EN" for English)

This format remains compatible with Common Voice [7] and existing ASR frameworks while extending functionality with dual transcription formats and multilingual labels.

The pipeline successfully processed and validated 5,350 of 8,000 hours, marking the largest Armenian speech corpus to date. While the full dataset is not publicly released due to commercial considerations, trained models are available for research use. These resources advance Armenian ASR and provide a scalable framework for other low-resource languages.

III. Methods

In this study, we employ two state-of-the-art neural architectures for Armenian ASR—Streaming Citrinet [8–10] and FastConformer-CTC [11–13]—both adapted from NVIDIA's NeMo toolkit (Figure 2). These were chosen for their strong multilingual performance and adaptability to low-resource settings.

Streaming Citrinet serves as our primary real-time model. The Streaming Citrinet-1024 variant (~140M parameters) extends QuartzNet with subword encoding and Squeeze-and-Excitation modules, using CTC decoding for efficient inference on 16 kHz mono audio. Subword encoding is particularly beneficial for Armenian's morphological richness, enabling better handling of unseen words. In our earlier work [5], we modified Citrinet with structural and hyperparameter optimizations to improve accuracy and speed. Reducing Jasper blocks from 23 to 16 lowered WER slightly (18.2%) but improved inference speed by 26%, making the model more suitable for latency-sensitive applications.

FastConformer, in contrast, serves as our high-accuracy model. By combining Transformer attention with convolutional layers, it achieves superior performance on long-form audio such as news broadcasts, making it ideal for accuracy-critical scenarios.

FastConformer models (both CTC and RNNT variants) incorporate an optimized Conformer encoder, introducing:

- 8× depthwise-separable convolutional subsampling with 256 channels
- Reduced convolutional kernel size of 9 within the Conformer blocks

Additionally, the FastConformer model uses a CosineAnnealing learning-rate scheduler for training, replacing the Noam scheduler typically used in Conformer training.

These changes make the encoder approximately 2.4× faster than the original, with negligible quality degradation. Further reducing subsampling channels to 128 boosts the speedup to 2.7×, though at the cost of some accuracy. With local attention mechanisms, FastConformer supports inference on long-form audio—over one hour with 256 channels and over two hours with 128 channels.

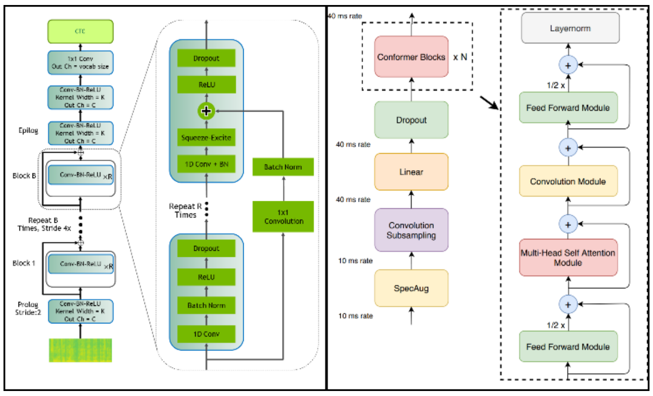


Figure 2. Model architectures: Citrinet and Conformer (the backbone of FastConformer)

The FastConformer-CTC model ($\approx$115M parameters) was adopted as the second baseline, chosen over the heavier Hybrid (Transducer+CTC) variant due to its alignment-free optimization, faster computation, and suitability for the comparatively small dataset [11 – 13]. This choice ensured efficient training, resource economy, and reduced overfitting risk.

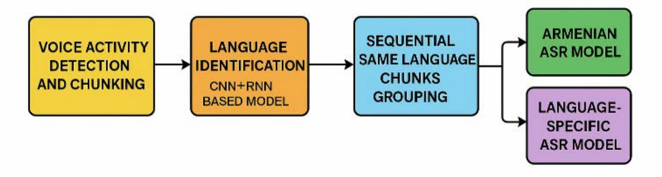
To improve input quality, we implemented a GPU-based noise cancellation pipeline [14], which reduced background noise, music, and other artifacts common in broadcast media while maintaining real-time processing speeds.

A key innovation of our system is a multilingual processing framework that automatically detects language boundaries and routes audio segments to the appropriate monolingual ASR model (Figure 3). Beyond Armenian, the framework supports English, German, French, Russian, Italian, and Persian. While pre-trained NeMo models [15] cover these languages, our design integrates implicit language identification (LID) with user-selectable options, balancing GPU efficiency on shared servers.

The pipeline begins with voice activity detection (VAD) [16], segmenting audio into short chunks ($\approx$ 1 – 3s, adjustable). Segments are then classified by a lightweight CNN – RNN LID model [17], which efficiently captures local spectral-temporal cues and sequential dependencies, providing fast and accurate predictions. Confidence scores enable threshold-based handling of ambiguous cases.

Consecutive chunks with the same label are merged (up to 15 minutes) for context-rich transcription.

Final segments are routed to language-specific ASR models, with Armenian speech processed by Streaming Citrinet or FastConformer, while unknown-language chunks are handled by multilingual Whisper-small/medium models [18].


VOICE ACTIVITY DETECTION AND CHUNKING
LANGUAGE IDENTIFICATION
CNN+RNN BASED MODEL
SEQUENTIAL SAME LANGUAGE CHUNKS GROUPING
ARMENIAN ASR MODEL
LANGUAGE-SPECIFIC ASR MODEL
Figure 3. Processing Flow for Multilingual Speech Recognition

To maximize throughput, language identification and transcription run on the GPU in adjustable batches (default: 2). Larger batches boost throughput but increase memory demand and latency. A GPU queuing system monitors memory, queuing new processes if usage falls below 15%, preventing overload and enabling concurrent ASR/LID instances.

This multilingual framework automatically processes mixed-language audio while preserving language-specific optimizations. Its modular design supports easy integration of new models and parameter tuning for different applications.

IV. Training and Results

Model-specific configurations were largely consistent across architectures. Convolutional filter sizes were retained, transitioning from non-streaming Citrinet-256 to streaming Citrinet-1024. A uniform dropout of 0.05 and learning rate of 0.001 were applied, with training for 150 epochs, data shuffling, early stopping (patience = 10), and a batch size of 16. Streaming Citrinet used Novograd with weight decay 0.001 and 15% warm-up, while FastConformer-CTC used AdamW with 0.0001 decay. Data augmentation included SpecAugment and SpecCutout for Citrinet, and SpecAugment for FastConformer-CTC [19,20].

Transfer learning from English pre-trained NeMo weights [15] benefited both the Streaming Citrinet and FastConformer-CTC models by improving convergence and simplifying their adaptation to Armenian. However, fully freezing the encoder impaired performance on Armenian, so we instead fine-tuned the encoder along with batch-normalization and Squeeze-and-Excitation layers to adapt effectively to the target language. Tokenization experiments comparing WordPiece [21] and BPE [22] showed BPE performed better for Armenian, handling morphological variation and OOV words more effectively. We adopted Google's SentencePiece BPE [23] with a 1,024-subword vocabulary trained on ~17M Armenian text lines, ensuring full token coverage and alignment with the English decoder for efficient transfer learning.

The final ASR dataset contains ~5,500 hours of 16-bit mono WAV at 16 kHz, split 70/20/10% for training, validation, and testing. Experiments ran on a Vast.ai instance with four NVIDIA A100 SXM4 GPUs.

The lightweight CNN–RNN language identification model was trained on ~250 hours of 1–5 s audio clips across six languages, achieving 92.46% accuracy. Segments with confidence below 70% were marked as 'unknown' to handle uncertain classifications.

Both models were evaluated on the same dataset, with and without a statistical language model (LM) (Table 1). In baseline (acoustic-only) evaluation, FastConformer-CTC outperformed Streaming Citrinet, achieving a WER of 10.39% vs. 15.61%. With LM integration, Citrinet improved to 12.44%, while FastConformer reached 8.56%.

Model	**Size (MB)**	**WER**	**WER with LM**
Citrinet Streaming	461	15.61%	12.44%
FastConformer CTC	442	10.39%	8.56%

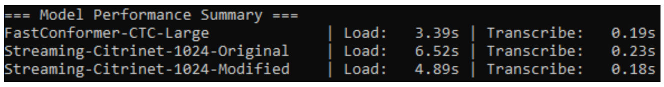
Table 1. Results

The 5.22% gap highlights FastConformer's advantage in modeling Armenian's morphologically rich structure, due to its mix of convolutional context modeling and transformer attention. Citrinet, optimized for real-time streaming, sacrifices bidirectional context for low latency, which limits accuracy.

Both models benefited from LM rescoring, though gains were larger for Citrinet (–3.17 points) than FastConformer (–1.83 points). This suggests Citrinet produces more recoverable linguistic errors, while FastConformer's stronger acoustic modeling leaves less room for correction.

From a deployment perspective, Citrinet Streaming is better suited for real-time, resource-sensitive applications such as live captioning or on-device speech recognition, while FastConformer CTC is preferable for accuracy-critical tasks like offline transcription or subtitling. Despite similar sizes (~450 MB), FastConformer consistently yields higher-quality transcripts, making it the stronger choice when accuracy outweighs latency.

Building on the results summary, we extended our evaluation with runtime measurements to assess deployment feasibility (Figure 4). Tests were run on Ubuntu 20.04 LTS (Intel i7-9700 @ 3.0 GHz, NVIDIA GTX 1660, 6 GB VRAM) using a 60-s audio clip, without an external statistical language model.


```
=== Model Performance Summary ===
FastConformer-CTC-Large             | Load:   3.39s | Transcribe:   0.19s
Streaming-Citrinet-1024-Original    | Load:   6.52s | Transcribe:   0.23s
Streaming-Citrinet-1024-Modified    | Load:   4.89s | Transcribe:   0.18s
```
Figure 4. Processing Flow for Multilingual Speech Recognition

These measurements quantify model cold-start (load time) and inference latency (transcription time), both critical for deployment. FastConformer-CTC-Large achieved the fastest load (3.39 s) and strong inference speed (0.19 s). The original Streaming-Citrinet-1024 was slower—6.52 s to load (≈92% longer) and 0.23 s to transcribe (≈ 21% slower). Our modifications improved Citrinet substantially: load time fell to 4.89 s (≈25% faster than original) and transcription latency to 0.18 s—≈5% faster than FastConformer and 22% faster than original Citrinet.

Deployment trade-offs are clear: FastConformer's minimal load time and high throughput favor server or batch use. Original Citrinet's heavy initialization burden hindered short-lived or serverless scenarios. The modified Citrinet, however, reduces cold-start cost while retaining streaming benefits, making it suitable for live-streaming and edge applications requiring steady low-latency outputs.

Note: absolute timings vary with hardware, memory, I/O, software, model compilation, and system load.

V. Conclusion

This study introduced a practical, semi-automated pipeline for scaling Armenian ASR that combines large-scale web collection, multi-stage automatic transcription, lightweight human validation, and targeted architectural adaptation. From roughly 8,000 hours of collected audio (5,350 hours validated) and a 17-million-line text corpus, we trained and evaluated two production-grade models—Streaming Citrinet (streaming, low-latency) and FastConformer-CTC (high-accuracy)—and a compact CNN+RNN LID (92.46% accuracy) to enable multilingual routing and processing.

Empirically, FastConformer-CTC is the accuracy leader (WER 10.39% acoustic-only; 8.56% with an external LM), while the modified Streaming Citrinet offers a useful latency/throughput tradeoff (15.61% → 12.44% with LM) and meaningful runtime improvements after optimization. Practical measurements (model sizes ≈ 442 – 461 MB, FastConformer cold-start ≈3.4 s and 0.19 s per 60 s clip; modified Citrinet cold-start ≈4.9 s and 0.18 s per clip) illustrate how architectural choice maps to deployment regimes: choose FastConformer for batch/offline, Citrinet for live/edge scenarios where latency and streaming outputs matter.

Beyond improved WERs, our work demonstrates the value of transfer learning, BPE tokenization, aggressive audio preprocessing (VAD and GPU noise reduction), and selective human validation for low-resource languages. The trained models offer an immediate resource for Armenian NLP applications, while the semi-automated methodology remains broadly applicable to other low-resource languages.

Limitations include partial dataset release for commercial reasons, dialect normalization effects introduced by mixed-source transcriptions, and remaining scope to improve LID robustness and on-device/quantized deployments. Future work will focus on enlarging validated corpora, refining LID and dialect handling, releasing more resources to the community, and exploring model compression and semi-supervised learning to further lower resource barriers.

In summary, this study closes a practical gap for Armenian speech technology: it supplies a replicable, scalable pipeline and competitive models that materially advance ASR quality for a low-resource language and provide a clear roadmap for similar efforts elsewhere.

References

[1] V. H. Baghdasaryan. "ArmSpeech: Armenian spoken language corpus", *International Journal of Scientific Advances (IJSCIA)*, vol. 3, no. 3, pp. 454–459, 2022.

[2] V. H. Baghdasaryan. "Extended ArmSpeech: Armenian Spoken Language Corpus", *International Journal of Scientific Advances (IJSCIA)*, vol. 3, no. 4, pp. 573–576, 2022.

[3] V. H. Baghdasaryan. "Armenian Speech Recognition System: Acoustic and Language Models", *International Journal of Scientific Advances (IJSCIA)*, vol. 3, no. 5, pp. 719–724, 2022.

[4] V. Baghdasaryan. "Exploring Armenian Speech Recognition: A Comparative Analysis of ASR Models—Assessing DeepSpeech, Nvidia NeMo QuartzNet, and Citrinet on Varied Armenian Speech Corpora", *Proceedings of International Conference CSIT*, pp. 25–30, 2023.

[5] V. H. Baghdasaryan. "Enhancing Armenian Automatic Speech Recognition Performance: A Comprehensive Strategy for Speed, Accuracy, and Linguistic Refinement", *International Journal of Scientific Advances (IJSCIA)*, vol. 5, no. 2, pp. 281–288, 2024.

[6] Google Cloud. "Chirp 2: Enhanced multilingual accuracy (Cloud Speech-to-Text V2 documentation)", Google Cloud. Retrieved August 11, 2025. [Online]. Available: https://cloud.google.com/speech-to-text/v2/docs/chirp_2-model. (Accessed 2025-08-11)

[7] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, ... G. Weber. "Common voice: A massively-multilingual speech corpus". *arXiv preprint arXiv:1912.06670*, 2019.

[8] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, B. Ginsburg. "Citrinet: Closing the Gap between Non-Autoregressive and Autoregressive End-to-End Models for Automatic Speech Recognition". *arXiv preprint arXiv:2104.01721*, 2021.

[9] NVIDIA. "Automatic speech recognition (ASR) models — NeMo documentation: Citrinet". NVIDIA Developer. Retrieved August 3, 2025. [Online]. Available: https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html#citrinet. (Accessed 2025-08-03)

[10] NVIDIA. "nvidia/stt_en_citrinet_1024_gamma_0_25 [Model]". Hugging Face. Retrieved August 18, 2025. [Online]. Available: https://huggingface.co/nvidia/stt_en_citrinet_1024_gamma_0_25. (Accessed 2025-08-18)

[11] A. Gulati, J. Qin, C. C. Chiu, N. Parmar, Y. Zhang, J. Yu, ... R. Pang. "Conformer: Convolution-augmented transformer for speech recognition". *arXiv preprint arXiv:2005.08100*, 2020.

[12] NVIDIA. "Automatic speech recognition (ASR) models — NeMo documentation: Fast Conformer". NVIDIA Developer. Retrieved August 3, 2025. [Online]. Available: https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html#fast-conformer. (Accessed 2025-08-03)

[13] NVIDIA. "nvidia/stt_en_fastconformer_ctc_large [Model]". Hugging Face. Retrieved August 18, 2025. [Online]. Available: https://huggingface.co/nvidia/stt_en_fastconformer_ctc_large. (Accessed 2025-08-18)

[14] H. Schröter, T. Rosenkranz, A. N. Escalante-B, A. Maier. "Deepfilternet: Perceptually motivated real-time speech enhancement". *arXiv preprint arXiv:2305.08227*, 2023.

[15] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, ... J. M. Cohen. "Nemo: a toolkit for building ai applications using neural modules". *arXiv preprint arXiv:1909.09577*, 2019.

[16] J. Wiseman (host). "py-webrtcvad: Python interface to the WebRTC Voice Activity Detector [Source code]", GitHub. [Online]. Available: https://github.com/wiseman/py-webrtcvad.

[17] C. Bartz, T. Herold, H. Yang, C. Meinel. "Language identification using deep convolutional recurrent neural networks", *International Conference on Neural Information Processing*, Cham: Springer International Publishing, pp. 880–889, October 2017.

[18] R. C. Necula, P. C. Craciun. "Running Automatic Speech Recognition (ASR) Model in the Context of Real Time Data Streaming Architecture", Proceedings of the *International Conference on Business Excellence*, Sciendo, vol. 19, no. 1, pp. 1282–1293, 2025.

[19] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, E. D. Cubuk, Q. V. Le. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". *arXiv preprint arXiv:1904.08779*, 2019.

[20] T. DeVries, G. W. Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout". *arXiv preprint arXiv:1708.04552*, 2017.

[21] G. Grefenstette. "Tokenization", Syntactic Wordclass Tagging, pp. 117–133. *Dordrecht: Springer Netherlands*, 1999.

[22] R. Sennrich, B. Haddow, A. Birch. "Neural machine translation of rare words with subword units". *arXiv preprint arXiv:1508.07909*, 2015.

[23] T. Kudo, J. Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing". *arXiv preprint arXiv:1808.06226*, 2018.