Reward-Guided Fine-Tuning of Whisper for Low-Resource Nepali Speech Recognition

Aadarsh Pandit, Yudhin Khanal, Ishan Pandey, Kushal Kunwar, Sunil Regmi
aadarshapandit@gmail.com, yudhinkhanal548@gmail.com, eshanpandey804@gmail.com,
kunwarkushal190@gmail.com, sunil.regmi@ku.edu.np
Department of Artificial Intelligence, Kathmandu University, Dhulikhel, Nepal

Abstract

Fine-tuning speech recognition models on noisy real-world data is tricky. The model has no way of knowing which training samples are reliable and which are not, so it ends up learning from bad examples just as readily as good ones. This is a real problem for Nepali, where most available training data comes from YouTube videos with automatically generated subtitles that are often inaccurate. In this work, we tried a simple fix. Instead of feeding everything to the model, we first asked humans to rate the quality of a sample of transcriptions, trained a small Random Forest classifier on those 2,000 ratings, and used it to filter out the bad samples before each retraining round. The classifier uses four automatically computable features, Word Error Rate (WER), Character Error Rate (CER), length ratio, and length difference, and achieves 81% accuracy on a held-out set. Running two filtering and retraining cycles on a 40,000 clip training subset drawn from a 68.4 hour corpus improves substantially over our own standard fine-tuning baseline of 5.60% WER and 5.10% CER, reaching 4.89% WER and 4.52% CER, which corresponds to an 11–13% relative gain. The approach is much lighter than full Reinforcement Learning from Human Feedback (RLHF) but still uses real human judgment to guide training.

Keywords: Low-Resource ASR, Reward Model, Data Filtering, Whisper, Nepali, Human Feedback

1. Introduction

Automatic Speech Recognition has improved a lot in recent years, largely because of large pretrained models like wav2vec 2.0 (Baevski et al., 2020), HuBERT (Hsu et al., 2021), and Whisper (Radford et al., 2023). These models work surprisingly well across many languages out of the box, which makes them a natural starting point when you want to build ASR for a language that does not have much data. The usual approach is to just take one of these pretrained models and fine-tune it on whatever local data you can find.

Nepali is a morphologically rich South Asian language written in Devanagari, an abugida script used across South Asia for languages including Hindi, Sanskrit, and Marathi, and like many South Asian languages it remains severely underresourced for speech technology despite having tens of millions of speakers. For Nepali, the most accessible source of training data is YouTube, specifically audio clips paired with SRT (SubRip Subtitle) files. The problem is that the subtitle quality varies a lot. Some are accurate, many are not, and when you fine-tune a model on all of it equally, it learns from the bad examples just as much as the good ones and gradually gets worse at the things it was already getting wrong.

Prior work has tried to get around this. Ghimire et al. (2023) used language model scores to pick which pseudo-labeled samples to keep, reaching 6.77% CER on the OpenSLR Nepali test set. But Language Model (LM) scores measure whether the output sounds fluent, not whether it actually matches what was said. Pseudo-labeling methods (Singh et al., 2023; Xu et al., 2020) face the same limitation. They rely entirely on the model's own confidence.

Our approach is different: we ask humans. We built a small annotation tool, collected 2,000 quality ratings on reference-hypothesis pairs, and trained a simple Random Forest classifier to predict those ratings from four error metrics. That classifier then acts as a filter. Before each retraining round, it removes the samples the model is most likely to get wrong. The result is a progressively cleaner training set and steadily improving accuracy, without any of the complexity of full RLHF.

Our main contributions are:

- A browser-based annotation platform for rating Nepali ASR transcription quality.
- A Random Forest reward model using four automatic features, achieving 81% accuracy.
- An iterative filtering pipeline that improves WER from 5.60% to 4.89% and CER from 5.10% to 4.52%, an 11–13% relative gain over our own standard fine-tuning baseline.
- Results on a 68.4-hour real-world Nepali speech corpus covering news, podcasts, and serials.

2. Related Work

2.1. Nepali ASR

Early Nepali ASR work relied on classical architectures. Regmi et al. (2019) built an RNN-CTC system, while Regmi and Bal (2021) pushed further

Proceedings of the Second workshop on Challenges in Processing South Asian Languages (CHiPSAL2026) @ LREC, pages 169–175
16 May 2026.

with a joint CTC-attention model that dropped the need for a pronunciation lexicon, reporting 0.30% CER on the controlled OpenSLR test set. CNN-based approaches from Banjara et al. (2020) and Dhakal et al. (2022) trailed behind at 27.72% and 17.07% CER respectively. More recently, Rijal et al. (2024) showed that fine-tuning OpenAI Whisper on Nepali data yields strong gains, and Ghimire et al. (2023) used language model scores to iteratively select pseudo-labeled samples for MMS-1B (Pratap et al., 2023), reaching 6.77% CER.

2.2. Active Learning and Self-Training

Several groups have explored using the model's own outputs to drive training. Zheng et al. (2023) used HuBERT-based perplexity scores for contrastive data selection, cutting WER by 11%. Singh et al. (2023) iterated on pseudo-labels above a confidence threshold with wav2vec 2.0 XLSR-53. Xu et al. (2020) showed iterative pseudo-labeling helps on LibriSpeech, and Xu et al. (2021) confirmed self-training and pretraining are complementary. All of these approaches share a limitation: the selection signal comes from the model itself, not from external human judgment.

2.3. Reward Models and Human Feedback

RLHF has been highly effective for aligning language models (Ouyang et al., 2022), but applying it directly to encoder-decoder ASR systems is cumbersome. It requires running multiple model copies, managing KL (Kullback-Leibler divergence) constraints, and keeping policy gradient training stable. We take a lighter-weight approach: train a reward classifier from human annotations and use it purely as a data filter, keeping all the benefit of human-grounded quality signals without the Reinforcement Learning (RL) machinery.

3. Methodology

Our pipeline has three stages: fine-tune Whisper Small on the 40,000-clip training pool to get a baseline model, train a reward classifier on human-annotated quality scores, then iteratively filter the training pool and retrain. We use the term *reward model* analogously to its use in RLHF: a function trained on human judgments that assigns a scalar quality score to a model output. Unlike full RLHF, however, we use this score purely as a data filter rather than as a gradient signal. Figure 1 shows the overall flow.

3.1. Baseline Fine-Tuning

We start from OpenAI's Whisper Small (Radford et al., 2023), a 244M-parameter transformer encoder-decoder pretrained on multilingual audio. Audio goes through a convolutional frontend that produces log-mel spectrograms, which the encoder processes into acoustic representations; the decoder then generates text conditioned on those representations and task tokens.

Fine-tuning uses `WhisperForConditionalGeneration` and `WhisperProcessor` from Hugging Face Transformers, with a custom data collator that pads sequences dynamically and masks padding positions from the loss using -100. We train for 3 epochs with batch size 4, gradient accumulation over 2 steps, FP16 precision, and checkpoint saves every 500 steps. This produces FT_{model_0}.

3.2. Reward Model

3.2.1. Annotation Platform

To get human quality judgments, we built a small Flask-based annotation tool. We ran FT_{model_0} on audio from OpenSLR54 (Kjartansson et al., 2018) to generate hypothesis transcriptions, then presented each annotator with the reference text and the model's output side by side. They could play the audio and assign one of three labels: **1 (Good)** for accurate output with minimal errors, **0 (Neutral)** for output that is understandable but noticeably imperfect, and **-1 (Bad)** for output with major errors or hallucinations. We collected exactly 2,000 labeled pairs this way. Annotation was performed by five members of the research team, with each sample assigned to one annotator. To verify consistency, 200 samples (10%) were independently labeled by a second annotator; the two annotators agreed on 83% of these, with disagreements concentrated at the Good/Neutral boundary rather than the Bad category, suggesting the Bad label which drives all filtering decisions is applied reliably.

3.2.2. Feature Engineering

Each (reference, hypothesis) pair is described by four features: WER (word-level edit distance), CER (character-level edit distance, which matters a lot for morphologically rich Devanagari), length_ratio (hypothesis word count divided by reference word count), and length_diff (absolute word count difference). These four numbers can be computed automatically from any reference-hypothesis pair without needing extra linguistic tools.

3.2.3. Reward Model Training

We train a `RandomForestClassifier` on the 2,000 annotated pairs using an 80/20 split. The model predicts the three-class quality label and

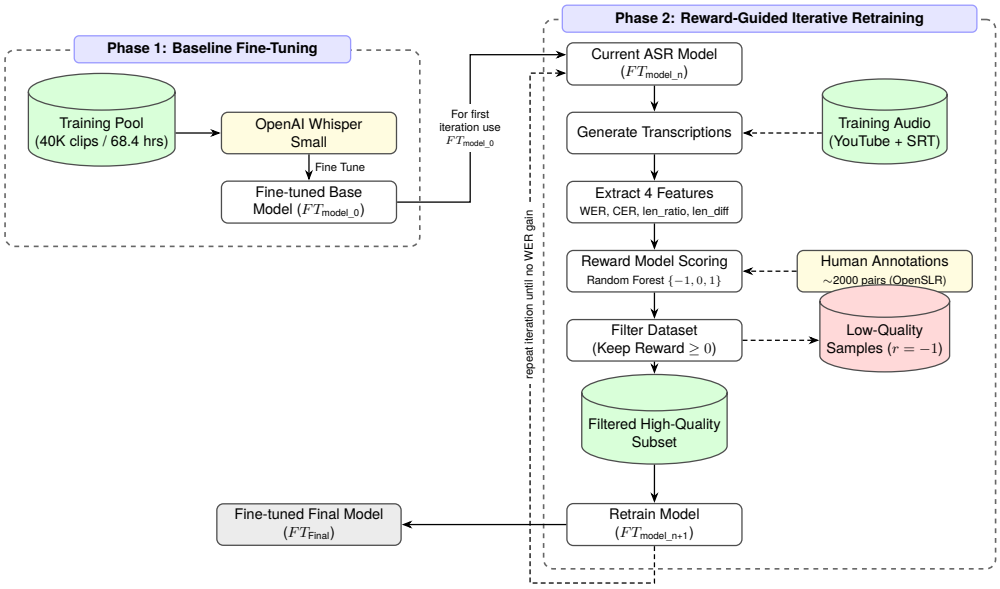

Figure 1: Architecture of the reward-guided fine-tuning pipeline.

reaches **81% accuracy** on the held-out set. A natural question is why a classifier is needed at all rather than a simple WER or CER threshold. The reason is that human quality judgments do not reduce cleanly to a single metric: annotators frequently rated samples as Bad despite low WER (e.g., when the hypothesis was fluent but semantically wrong), and as Good despite moderate WER (e.g., when the reference SRT itself contained a minor typo). A fixed threshold on any single metric cannot capture these interactions; the Random Forest learns them from the annotated data. We also evaluated a Multi-Layer Perceptron neural network as an alternative, but it overfit severely on the 2,000-sample training set. Random Forest was selected because it handles nonlinear feature interactions well, is robust to the outlier WER/CER values that come from very short clips, and achieved the best held-out validation accuracy. We used default scikit-learn hyperparameters (100 trees, `max_features='sqrt'`) without further tuning, which proved sufficient. Feature importance analysis put WER and CER together at about 65% of total importance, with length features covering the rest. The length_diff feature in particular catches hallucinations that error rates miss when the reference is very short. We keep samples with predicted reward $r \geq 0$ (Good or Neutral) and drop only the Bad ones; removing Neutral samples too would shrink the training set too aggressively.

3.3. Reward-Guided Iterative Filtering

Algorithm 1 shows the full procedure. After the baseline fine-tuning step, we run the current model over the 40,000-clip training pool, compute the four features for each sample, and ask the reward model whether to keep it. Samples predicted as Bad ($r = -1$) are dropped; the rest go into the next round of fine-tuning. We repeat until WER stops improving on the held-out test set.

It is important to note the two-stage nature of the feature computation. During reward model *training*, WER and CER are computed against the clean, human-verified OpenSLR54 transcriptions, so the classifier learns from high-quality reference signal. During reward model *inference* on the YouTube training pool, however, the only available reference is the noisy SRT subtitle. This means that in cases where the model produces a correct transcription but the SRT is wrong, the computed WER will be artificially high and the reward model may incorrectly mark the sample as Bad. We acknowledge this as a limitation: the filter is bounded by the quality of the SRT proxy reference. In practice, the consistent WER improvements across iterations suggest that correctly-transcribed samples with bad SRTs are a small minority and do not dominate the filtering signal, but this assumption would benefit from explicit verification in future work.

The key difference from Ghimire et al. (2023) is what the filter actually measures. Their LM score tells you whether the hypothesis sounds like fluent Nepali. Ours tells you whether it matches the reference. That is a more direct proxy for transcription quality, and it does not require maintaining a separate language model.

4. Dataset

Our training corpus consists of Nepali audio collected from YouTube: news programs, podcasts,

Algorithm 1 Reward-Guided Fine-Tuning

Require: Pretrained model M_0, training dataset D, reward model R, test dataset D_{test}
Ensure: Final fine-tuned model M_{final}

$M_{final} \leftarrow \text{FineTune}(M_0,\ D)$
repeat
  $D_{filtered} \leftarrow \{\}$
  for each $(x,\ y)$ **in** D **do**
    $\hat{y} \leftarrow M_{final}.\text{Transcribe}(x)$
    $F \leftarrow \text{features}(\hat{y},\ y)$ ▷ WER, CER, len_ratio, len_diff
    $r \leftarrow R.\text{Predict}(F)$
    if $r \geq 0$ **then**
      $D_{filtered}.\text{Append}((x,\ y))$
    end if
  end for
  $M_{final} \leftarrow \text{FineTune}(M_{final},\ D_{filtered})$
  $\text{WER} \leftarrow \text{ComputeWER}(M_{final},\ D_{test})$
until WER does not improve
return M_{final}

drama serials, and documentaries. We downloaded SRT subtitle files alongside the videos and used ffmpeg to cut the audio at subtitle boundaries, then resampled everything to 16 kHz. Table 1 summarizes the corpus statistics.

Property	Value
Total audio segments	49,216
Total duration	68.4 hours
Dataset size	18.9 GB
Training pool	40,000 clips
Validation/Test set	9,216 clips
Average clip length	∼6 seconds
Sampling rate	16 kHz
Sources	News, podcasts, serials, docs
Annotation samples (OpenSLR)	2,000

Table 1: Training dataset statistics.

4.1. Data Processing

Raw SRT files were parsed to extract timestamp-text pairs, and ffmpeg was used to slice the corresponding audio segments. Clips shorter than 1 second or longer than 30 seconds were discarded to avoid uninformative or over-segmented samples. All audio was converted to mono, resampled to 16 kHz, and saved as WAV files. From the resulting 49,216 clips, we randomly partitioned the data into a training pool of 40,000 clips and a held-out evaluation set of 9,216 clips (split between validation and testing). Text transcriptions were normalized by removing special characters, extra whitespace, and non-Nepali tokens.

For the reward model annotations, we drew from OpenSLR54 (Kjartansson et al., 2018), a crowd-sourced Nepali corpus with 165 hours from 527 speakers. Human annotators labeled 2,000 reference-hypothesis pairs; the distribution came out at approximately 38% Good, 29% Neutral, and 33% Bad, confirming that a meaningful fraction of the model's outputs are noisy enough to hurt retraining if left in.

5. Experiments and Results

5.1. Experimental Setup

All experiments ran on an NVIDIA RTX 4060 GPU paired with an AMD Ryzen 7 7000-series CPU and 16 GB RAM. The reward model was trained with `scikit-learn`, and WER/CER were computed with the `jiwer` library. All models share the same held-out test split.

5.2. Quantitative Results

Table 2 shows performance at each stage. We first fine-tuned Whisper Small on our 40,000-clip training pool for 3 epochs to produce FT_{model_0}, our own baseline, which achieved 5.60% WER and 5.10% CER (row 2). This serves as the direct starting point for all reward-guided iterations and is the model against which percentage improvements are measured. For reference we also include the published Whisper FT result of Rijal et al. (2024) on Nepali (row 3), which used the same architecture on different data. Applying our reward-guided iterative filtering to the training pool consistently improves on our own baseline. Each filtering pass removed approximately 30–35% of that round's available samples: Iteration 1 retained 27,200 clips from the 40,000-clip pool (removing 32%), and Iteration 2 retained 18,500 clips from the 27,200 remaining after Iteration 1 (removing 32%). The final model reaches 4.89% WER and 4.52% CER, an 11–13% relative reduction over the baseline.

Training loss dropped steadily across all stages: from 1.426 to 0.052 during the baseline fine-tuning run, then from 0.052 to 0.031 after Reward-Guided Iteration 1, and from 0.031 to 0.024 after Iteration 2. This progressive reduction reflects a cleaner, less contradictory training signal at each round rather than the task simply becoming easier due to dataset shrinkage.

5.3. Comparison with Prior Work

Table 3 compares our results with previous Nepali ASR systems. The reported numbers are taken directly from the respective papers, and it is important to note that they were evaluated on different test sets; we did not re-run these systems on our dataset.

Training Stage	WER (%)	CER (%)
Base Whisper (no fine-tuning)	8.10	7.40
Our Baseline FT, 40K clips (this work)	5.60	5.10
Whisper FT (Rijal et al., 2024) (reference)	—	5.04
Reward-Guided Iteration 1 (27,200 clips)	5.12	4.71
Reward-Guided Iteration 2 (18,500 clips)	**4.89**	**4.52**

Table 2: WER and CER across training stages. Row 2 is our own baseline fine-tuning of Whisper Small on the 40,000-clip training pool; this is the direct starting point for all reward-guided iterations. Row 3 shows the published Rijal et al. result for architectural reference; WER was not reported in that work (—). All rows from Base Whisper onward are evaluated on the same held-out test split.

Among systems evaluated on diverse real-world Nepali audio, excluding the controlled OpenSLR result of Regmi and Bal (2021), our model achieves the lowest reported CER at 4.52%. We include Rijal et al. (2024) primarily as an architectural reference point, and its result is also presented in Table 2 as a baseline.

The 0.30% CER reported by Regmi and Bal (2021) was obtained on the clean and controlled OpenSLR test split, and is therefore not directly comparable to evaluations on real-world data.

Method	Architecture	CER (%)
Banjara et al. (2020)	CNN-GRU	27.72
Dhakal et al. (2022)	CNN-BiLSTM	17.07
Regmi and Bal (2021)	CTC-Attention	0.30†
Ghimire et al. (2023)	MMS-1B + Active LM	6.77
Rijal et al. (2024)	Whisper FT	5.04
Proposed	**Reward-Guided Whisper**	**4.52**

Table 3: Comparison with prior Nepali ASR systems. †Evaluated on a controlled OpenSLR test split; not directly comparable to our real-world corpus.

5.4. Qualitative Analysis

Table 4 shows a few examples comparing outputs from the baseline fine-tuned model (FT_{model_0}) and the reward-guided final model. The baseline fine-tuning makes repeatable errors: dropped syllables in compound words, wrong morphological endings, incorrect word boundaries (e.g., splitting *Nepalako* into *Nepal ko*). Reward-guided retraining tends to get these right, most likely because the bad training samples responsible for those errors have been filtered out.

Reference	Baseline FT	Reward-Guided
नेपालको राजधानी काठमाडौं हो।	नेपाल को रजधान काटमाण्डु हो।	नेपालको राजधानी काठमाडौं हो।
आज मौसम निकै सफा छ।	आज मसम ननको साफ छ।	आज मौसम निकै सफा छ।
हामी अहिले बैठकमा छौं।	हामी अहिले बैठमा छौ।	हामी अहिले बैठकमा छौं।

Table 4: Sample transcription comparisons.

6. Conclusion

We set out to tackle a straightforward problem: when you fine-tune a speech model on noisy YouTube data, it learns from bad examples as readily as good ones. Our solution was to bring in human judgment, not through the full machinery of RLHF, but through a simple classifier trained on 2,000 annotated transcription pairs. Two rounds of reward-guided filtering and retraining improved WER from 5.60% to 4.89% and CER from 5.10% to 4.52%, an 11–13% relative gain over our own standard fine-tuning baseline, running on a single consumer GPU. Looking ahead, more annotations would improve reward model coverage, neural alternatives to Random Forest could capture richer quality signals, and integrating language model rescoring at decode time could squeeze out further gains. The same pipeline should transfer naturally to other low-resource South Asian languages facing the same noisy subtitle problem.

Ethical Considerations

This work involves human annotation of speech transcription quality. Annotators participated voluntarily and were members of the research team. No personally identifiable information was collected during annotation. The training data was sourced from publicly available YouTube content; we do not redistribute the raw audio. The annotation labels and extracted features used to train the reward model do not contain sensitive information.

Limitations

The reward model is trained on only 2,000 annotated samples, which limits its ability to capture the full diversity of Nepali speech, including variations in accent, speaking style, and domain. The four hand-crafted features (WER, CER,

length_ratio, and length_diff) are useful but do not cover all aspects of transcription quality, especially errors related to meaning rather than surface form.

Another important limitation comes from how features are computed during inference. When applied to the YouTube training pool, WER and CER are calculated against noisy SRT subtitles rather than clean references. As a result, the filter may discard cases where the model produces a correct transcription but the subtitle is incorrect, effectively penalizing the model for fixing errors in the data. Although the consistent improvements across iterations suggest that such cases are relatively uncommon, this remains a fundamental limitation of the approach. Fully addressing it would require access to clean, human-verified references at scale.

Finally, our experiments were conducted on a single GPU setup, and the iterative retraining process can become computationally expensive when scaled to larger datasets or more iterations.

Data and Code Availability

The Nepali ASR audio dataset (49,216 clips, ∼68.4 hours, 18.9 GB) is publicly available on Hugging Face at https://huggingface.co/datasets/Aadarsh17/nepali-asr-dataset-public under a CC BY-NC 4.0 licence. The reward model code and full pipeline are available on GitHub at https://github.com/Aadarshttech/nepali-asr-reward-guided. The OpenSLR54 corpus used for annotation is publicly available at https://www.openslr.org/54/.

Acknowledgments

The authors thank the members of the Department of Artificial Intelligence at Kathmandu University for their support during annotation and evaluation.

Bibliographical References

A. Baevski, H. Zhou, A. Mohamed, and M. Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates Inc.

J. Banjara, K. R. Mishra, J. Rathi, K. Karki, and S. Shakya. 2020. Nepali speech recognition using CNN and sequence models. In *2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, pages 1–5. IEEE.

M. Dhakal, A. Chhetri, A. K. Gupta, P. Lamichhane, S. Pandey, and S. Shakya. 2022. Automatic speech recognition for the Nepali language using CNN, bidirectional LSTM and ResNet. In *2022 International Conference on Inventive Computation Technologies (ICICT)*, pages 515–521. IEEE.

R. R. Ghimire, B. K. Bal, and P. Poudyal. 2023. Active learning approach for fine-tuning pre-trained ASR model for a low-resourced language. In *Proceedings of the 20th International Conference on Natural Language Processing (ICON 2023)*, pages 82–89. NLP Association of India.

W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. 2021. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

O. Kjartansson, S. Sarin, K. Pipatsrisawat, M. Jansche, and L. Ha. 2018. Crowd-sourced speech corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali. In *Proceedings of the 6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*, pages 52–55. ISCA.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates Inc.

V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi, A. Baevski, Y. Adi, X. Zhang, W.-N. Hsu, A. Conneau, and M. Auli. 2023. Scaling speech technology to 1,000+ languages. *Journal of Machine Learning Research*, 25(97):1–52.

A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202, pages 28492–28518. PMLR.

P. Regmi, A. Dahal, and B. Joshi. 2019. Nepali speech recognition using RNN-CTC model. *International Journal of Computer Applications*, 178(31):1–6.

S. Regmi and B. K. Bal. 2021. An end-to-end speech recognition for the Nepali language. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON 2021)*, pages 180–185. NLP Association of India.

S. Rijal, S. Adhikari, M. Dahal, M. Awale, and V. Ojha. 2024. Whisper fine-tuning on Nepali language. *arXiv preprint arXiv:2411.12587*.

S. Singh, F. Hou, and R. Wang. 2023. A novel self-training approach for low-resource speech recognition. In *Proceedings of INTERSPEECH 2023*, pages 1588–1592. ISCA.

Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert. 2020. Iterative pseudo-labeling for speech recognition. In *Proceedings of INTERSPEECH 2020*, pages 1006–1010. ISCA.

Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli. 2021. Self-training and pre-training are complementary for speech recognition. In *ICASSP 2021 – IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3030–3034. IEEE.

Z. Zheng, Z. Ma, Y. Wang, and X. Chen. 2023. Unsupervised active learning: Optimizing labeling cost-effectiveness for automatic speech recognition. *arXiv preprint arXiv:2308.14814*.

Language Resource References

Oddur Kjartansson, Supheakmungkol Sarin, Knot Pipatsrisawat, Martin Jansche, and Linne Ha. 2018. OpenSLR Nepali Speech Dataset (SLR54). In *Proceedings of the 6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*, pages 52–55. ISCA. https://www.openslr.org/54/