Nandinī: A Female Voice for Vāgdhenu, a Vṛtta (Meter) Aware Śloka-to-Chant (TTS) System for Sanskrit

Ram G Athreya
Independent Researcher
ramgathreya@gmail.com

November 2026

Abstract

We present Nandinī, a female chanting voice for Vāgdhenu. Vāgdhenu turns a Sanskrit verse into chant that follows its meter, in the voice of a single male reciter. A female voice would normally need recordings of a trained female reciter, but Nandinī is built only from Vāgdhenu's released data. Each of its 1,467 recorded verses is converted into a voice taken from a single 25-second text-to-speech sample, and the reciter's melody is then put back in the new voice's range. Up to ten such candidates are made for each verse, and the first in which a chant recogniser finds at most one syllable wrong is kept. By that recogniser's count, 1,298 of the 1,467 clips (88.5%) meet this bar, which is optimistic because the same recogniser chose them. IndicF5 is then fine-tuned on the clips with Vāgdhenu's recipe, together with a vocoder for the new voice. On 60 verses new to both systems, Nandinī's character error rate is 3.81% against Vāgdhenu's 3.67%, and its chant is 4.5% longer. By ear, its tune matches Vāgdhenu's fully or mostly in 5 of 8 blind pairs. In the other 3 the melody is also similar, and only the pauses fall in different places. We release the dataset, the code and the model, including an ONNX version.

1 Introduction

Sanskrit recitation is chanted rather than read. In chant, the *vṛtta* (meter) of a verse is carried by the sound itself. Heavy (*guru*) syllables are held longer than light (*laghu*) ones, the voice pauses at the *yati* (caesura) and lingers at the end of each *pāda*, and the *visarga* is sustained or breathed. A synthesiser can learn this only from recordings of chanted verse, and very few are available. Vāgdhenu (Prathosh, 2026a) did the hard work of creating such data by hand. Its corpus was designed for metrical and phonetic balance, recorded by one reciter and openly released. About five hours of these recordings were enough to fine-tune a flow-matching model (Chen et al., 2024; Varadhan et al., 2025) into a chant synthesiser that follows the meter of the verse it is given.

The resulting system chants only in the reciter's own voice. Vāgdhenu has no female chanting voice. Its paper notes that chanting in an arbitrary voice would require a multi-reciter fine-tune (Prathosh, 2026a, Section 14), which would mean recording more trained reciters.

This work addresses that gap. We present Nandinī, a female chanting model built only from the data Vāgdhenu has already released, with no recordings from a second reciter. The female voice comes from a single 25-second sample of plain reading by a text-to-speech model (Bodhan AI and AI4Bharat, 2026). The released recordings are converted into that voice, and IndicF5 is then fine-tuned on them with Vāgdhenu's recipe, as Vāgdhenu was on the original recordings. The difficulty lies in the data. A training clip must keep every syllable of the verse and the reciter's timing and melody while sounding like a different person.

On 60 verses that neither system was trained on, Nandinī chants about as accurately as Vāgdhenu and at nearly the same pace. The Su-śrotā recogniser (Prathosh, 2026b) finds a character error rate of 3.81% against 3.67%, and Nandinī's renditions are 4.5% longer. In 8 blind pairs the tune was judged the same as Vāgdhenu's in 4 and mostly the same in 1. In the other 3 the melody was also similar, and only the pauses fell in different places. An automatic melody test was inconclusive, because its measure could not separate a change of tune from a change of voice (Section 5).

1.1 Contributions

- **A female-voice chant dataset and its generation method.** We convert all 1,467 clips of the Vāgdhenu dataset (5.33 hours) into a female voice and drop no verse. By construction, each clip has the reciter's timing and melody, with the melody transposed into the new voice's range. Each verse keeps the first of up to ten candidates with at most one syllable wrong. Su-śrotā, counting only syllables it reads correctly on the reciter's own recording, finds that 1,298 of the 1,467 verses (88.5%) meet that bar. Because the same recogniser chose each clip, this figure is optimistic (Section 3.3).
- **A corresponding female chanting model.** We fine-tune IndicF5 (Varadhan et al., 2025) on this data with Vāgdhenu's recipe, as far as it is released, and fine-tune a vocoder for the new voice. The result gives Vāgdhenu's system a female voice. It is compared with Vāgdhenu's model on verses new to both.

We release the dataset, the generation and training code, and the model weights in PyTorch and ONNX formats (Section 7).

2 Related work

Sanskrit chant synthesis and Vāgdhenu. Earlier work synthesised chanted Sanskrit verse by joining pre-recorded audio units under rules derived by hand (Rama and Lakshmanan, 2014). Vāgdhenu (Prathosh, 2026a) turns a verse into chant with a pipeline in which only two parts are learned. A text frontend first converts any Brahmic script to Kannada script, so that the multilingual backbone does not read Sanskrit with Hindi pronunciation rules. It then applies Sanskrit sound-change rules and scans the meter. Next, the system takes a reference clip from a bank of the reciter's chant whose pattern of heavy and light syllables matches the verse. This clip supplies the voice, melody and pace. The verse itself is generated by IndicF5 (Varadhan et al., 2025) fine-tuned on about five hours of the reciter's chant. A BigVGAN-v2 vocoder (Lee et al., 2023) fine-tuned on that model's output turns it into audio. The author chose this vocoder because IndicF5's default vocoder Vocos (Siuzdak, 2023) left an artefact on long vowels. The corpus was designed for balance across meters and sounds and recorded in two styles. It is released under CC-BY 4.0 together with the code and weights. The paper names other voices as an open problem.

We reuse most of this pipeline. The training data is the whole released corpus of 1,467 clips, converted into the new voice. IndicF5 is fine-tuned with Vāgdhenu's training code and its recipe, as far as it is released. To chant a new verse, Nandinī runs through Vāgdhenu's frontend, meter detector and renderer unchanged. Two parts are new. The first is a reference bank cut from the converted clips of the same verses as the reciter's bank. The second is a vocoder fine-tuned further for the new voice, because Vāgdhenu's own vocoder adds an echo to it (Section 4). The comparison in Section 5 runs Vāgdhenu's released weights through the same path.

Indic text-to-speech. General Indic text-to-speech models read Sanskrit as speech rather than chant it. IndicF5 covers eleven Indian languages but not Sanskrit. It reads Devanagari with Hindi pronunciation rules, which is why Vāgdhenu converts its text to Kannada script (Prathosh, 2026a). Indic Parler-TTS (Sankar et al., 2025) and Indic-Speak (Bodhan AI and AI4Bharat, 2026) list Sanskrit among their languages, and Indic-Speak offers named voices for it. The target voice comes from one of these. Readings of Sanskrit verses by Indic-Speak's voice Bharati were joined and cut to a single 25-second sample (Section 3.1). We chose Indic-Speak over Indic Parler-TTS for two reasons. On the same 12 verses, its readings had a character error rate of 3.2% against 11.6%. Its readings also kept one recognisable voice, whereas Indic Parler-TTS gave a different speaker on each verse when prompted with a description rather than a named speaker.

Synthetic speech as training data. Training one text-to-speech model on another's output is also established. Hwang et al. (2020) expand five hours of recordings into 179 hours of speech generated by an autoregressive model and train a faster model on it. Finkelstein et al. (2022) obtain accent transfer by training a robust model on the output of a less robust one built for that transfer. In both, the generated voice models speakers who were recorded. Here the target voice exists only as the output of a text-to-speech model. In each converted clip only the voice is synthetic; the words, timing and melody come from the reciter's chant.

Voice conversion as training data for TTS. Converting recordings to build training data is established for read and expressive speech. Huybrechts et al. (2021) convert supporting speakers' expressive recordings into a target voice and train a TTS model on the result. They need as little as 15 minutes of the target's real speech. Marques et al. (2024) use singing-voice conversion for the same purpose with 5 minutes of the target's speech. They also shift each source speaker's pitch by its average distance from the target's, much as the melody step here transposes the reciter's melody (Section 3.1). Singing-voice converters keep a tune by conditioning on the source's fundamental frequency, and Seed-VC (Liu, 2024) offers such a mode. This work instead converts with a speech model and puts the reciter's melody back afterwards (Section 3.1). Nosek et al. (2025) are closest in setting. They convert one annotated Serbian speaker into new voices so that its annotations can be reused. Their converter is trained on 3 to 30 minutes of each target speaker. All of these methods use at least some real audio of the target speaker, whereas the target voice here comes only from a single sample generated by a text-to-speech model.

Damage to Indic sounds. PSP (Menta, 2026) measures retroflex collapse in Indic text-to-speech with phoneme-level embedding probes. Across commercial systems it reports the collapse rising from Hindi to Telugu to Tamil. Its author cautions that the Telugu and Tamil scores are best read as relative rankings, because native audio itself scores high on them. We saw the same substitution from a different cause. When converting Sanskrit chant, the default Seed-VC model (Liu, 2024) collapsed retroflexes into dentals, which is why we use a converter fine-tuned on Indic speech (Section 3.1).

Selecting among generated samples with a recogniser. Generating several candidates and keeping the one a recogniser transcribes best is a known way to improve content accuracy. Yu and Kang (2026) study this for zero-shot TTS, choosing among whole utterances from F5-TTS. They identify a pitfall. The gain from choosing appears larger when the recogniser that scores the result comes from the same family as the one that chose it. They therefore recommend reporting scores from at least two recogniser families with separate training lineages, so that one of them is unrelated to the recogniser that chose. The method here makes the same kind of choice for voice conversion,

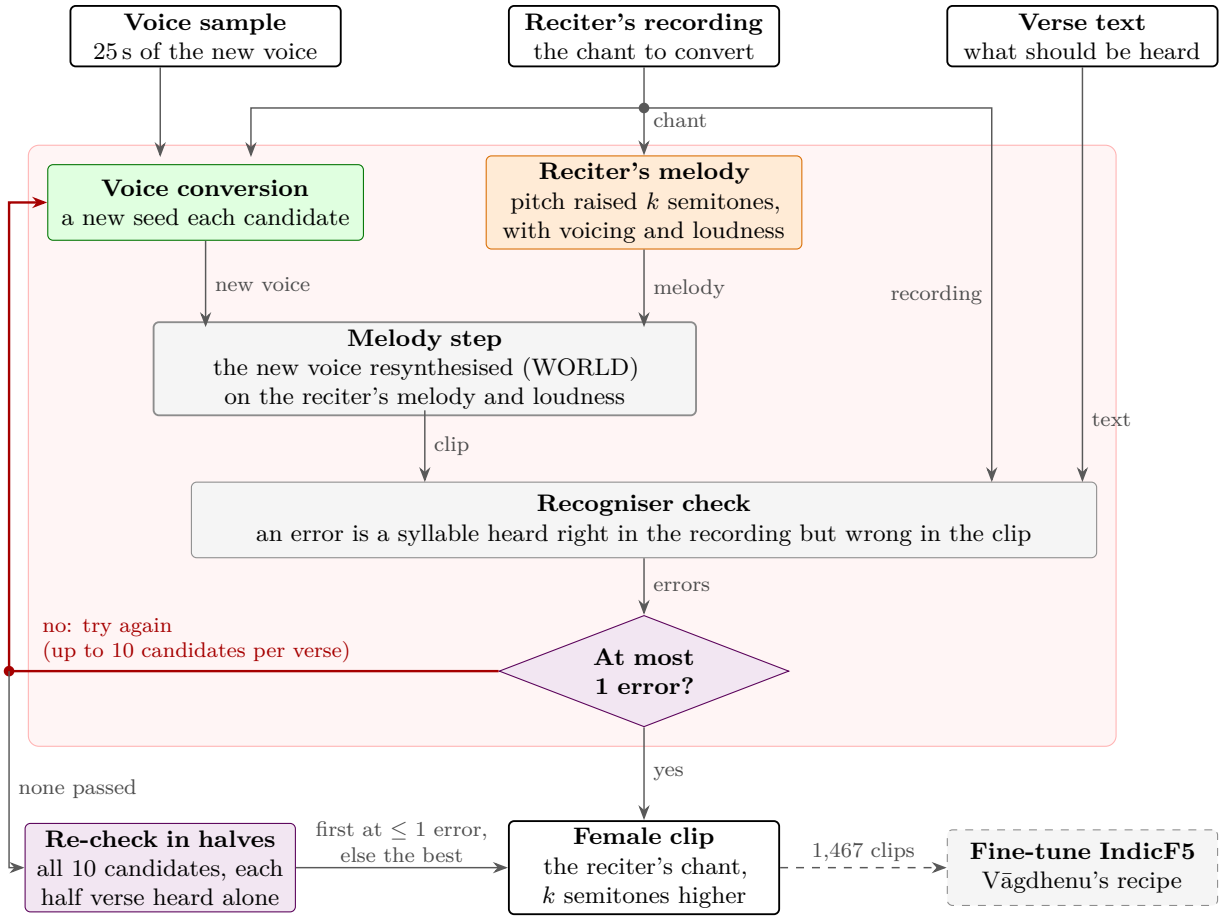


Figure 1: Data generation for one verse. Each candidate (shaded) converts the recording with a new seed, puts back the reciter's melody and is checked by the recogniser; the first with at most one error is kept. Training (dashed) is outside data generation.

although errors are counted against the source recording rather than the text. It also shares the pitfall. Su-śrotā (Prathosh, 2026b) both chooses the clips and scores them, so their error rates are optimistic. No recogniser from a second family exists for chant (Section 3.2), and our only check from outside the recogniser is targeted spot checks by ear on the cases it flagged as most at risk.

3 Data generation

Our approach takes each of the reciter's recordings and changes only the voice. The words, the timing and the melody all stay the reciter's own. Here the timing means how long each sound is held and where the pauses fall. The melody means the pitch contour, how the pitch rises and falls from one syllable to the next. A melody can be moved to a higher pitch without changing its shape, as when a song is sung in another key. This matters because chant has features that ordinary speech lacks. The visarga is sung or breathed depending on where it falls, a final m is hummed, word gaps are chanted through, and the last syllable of a verse is drawn out. A text-to-speech voice reading the verse would have none of these, whereas a converted recording keeps them. Each clip must therefore satisfy four requirements:

1. Every syllable of the verse pronounced correctly.
2. The reciter's timing.
3. The reciter's melody.
4. One consistent new voice.

Voice conversion supplies the new voice but damages a few syllables, and the damaged syllables differ from one conversion to the next. A second conversion of the same recording, with a different random seed, can therefore get right what the first got wrong. For each verse we generate up to ten such candidates, check each one with a speech recogniser, and keep the first with at most one syllable wrong (Figure 1). Verses where no candidate passes get a fallback, described in Section 3.3.

3.1 Generating a candidate

The input is the released dataset of 1,467 clips, about 5.3 hours in two recording styles (Prathosh, 2026a). Each clip is one verse chanted by the reciter. The target voice comes from a single 25-second sample of plain readings by Indic-Speak's named voice Bharati (Bodhan AI and AI4Bharat, 2026). A candidate for a clip is made in two steps. First, the reciter's recording is converted into the target voice. This gives the new voice, but the converted pitch no longer follows the reciter's chant. Second, that pitch is replaced with the reciter's own, transposed into the new voice's range.

Voice conversion. Each recording is converted zero-shot to the voice of the sample by the Whisper checkpoint of IndicVoiceChanger (DreamSyncCo, 2025). It is a Seed-VC (Liu, 2024) model fine-tuned on Indian-language speech from IndicVoices (Javed et al., 2024) and on proprietary data. In preliminary trials the default Seed-VC model often collapsed retroflex consonants into dentals, a contrast that is phonemic in Sanskrit. IndicVoiceChanger did not show the same collapse on the clips tried, so we chose it instead.

Conversion uses 30 diffusion steps, the lower end of the range Seed-VC recommends for best quality. It uses Seed-VC's default guidance rate of 0.7. Neither setting was tuned. The converter starts from random noise, so the seed is fixed. Candidate j of the verse with index i uses the seed $1000j + i$. Every run then produces the same candidates on the same hardware.

The converter keeps the reciter's timing by design. It stretches the recording's content features to the recording's own length before generating the voice, so each moment of the conversion is built from the same moment of the recording. The converted voice also stays steady within a clip, which the fourth requirement needs. On 61 verses converted ten times each, 2-second windows of a conversion match that conversion's average about as closely as windows of the reciter's recording match its own. Closeness here, and throughout, is the cosine similarity between ECAPA speaker embeddings (Desplanques et al., 2020; Ravanelli et al., 2021), and the two values are 0.821 against 0.807.

The pitch shift. The reciter chants at about 170 Hz on average, and the target voice reads at about 250 Hz. The melody step therefore raises the reciter's pitch by k semitones, so that the clips sit where the target voice naturally speaks.

We measure k rather than choose it. For each chant style, k is the difference in mean log pitch between the target voice reading verses and the reciter chanting them. The reciter's side uses all 1,467 clips. The target voice's side uses 61 verses from the dataset read in the target voice, 31 from the first style and 30 from the second. Within each style, these verses are spread across meter

	Reciter's clips		Target voice's readings	
Meter family	First style	Second style	First style	Second style
anuṣṭubh	311 (40.7%)	87 (12.4%)	12 (38.7%)	4 (13.3%)
triṣṭubh	125 (16.4%)	120 (17.1%)	5 (16.1%)	5 (16.7%)
other	78 (10.2%)	119 (16.9%)	3 (9.7%)	5 (16.7%)
jagatī	70 (9.2%)	124 (17.6%)	3 (9.7%)	5 (16.7%)
vasantatilakā	55 (7.2%)	92 (13.1%)	2 (6.5%)	4 (13.3%)
sragdharā	42 (5.5%)	50 (7.1%)	2 (6.5%)	2 (6.7%)
mālinī	34 (4.5%)	39 (5.5%)	1 (3.2%)	2 (6.7%)
rucirā	24 (3.1%)	29 (4.1%)	1 (3.2%)	1 (3.3%)
17-syllable	13 (1.7%)	28 (4.0%)	1 (3.2%)	1 (3.3%)
śārdūlavikrīḍita	12 (1.6%)	15 (2.1%)	1 (3.2%)	1 (3.3%)
Total	764 (100%)	703 (100%)	31 (100%)	30 (100%)

Table 1: The clips and readings behind k, by meter family and chant style, with each column's shares in brackets.

families in proportion to the reciter's clips, in case reading pitch depends on the text. Table 1 shows the result, and each meter family's share of the readings is within 2 points of its share of the clips. Rounding would leave *śārdūlavikrīḍita* out of the first style, so one verse is added there. Pitch is tracked with the autocorrelation method (Boersma, 1993) of Praat (Boersma and Weenink, 2021), run through Parselmouth (Jadoul et al., 2018), in 10 ms frames, in two passes per speaker. The second pass searches only between 0.75× the first quartile and 1.5× the third quartile of the first pass, which guards against octave errors.

	Target voice reading		Reciter chanting			
Style	Readings	Mean (Hz)	Clips	Mean (Hz)	k (semitones)	95% interval
First style	31	251	764	170	+6.74	[+6.55, +6.92]
Second style	30	249	703	170	+6.58	[+6.38, +6.78]
Pooled	61	250	1,467	170	+6.66	[+6.52, +6.80]

Table 2: The pitch shift k per chant style, computed from the target voice's readings and the reciter's clips.

Table 2 gives k for each style, +6.74 semitones for the first and +6.58 for the second. Its 95% intervals show how much k could change with a different sample of readings. They come from a bootstrap, which resamples the readings and clips 10,000 times while keeping the number in each meter family fixed. The two differ by less than the precision of either, and the reciter's own pitch differs between the styles by only 0.02 semitones and the pooled k was calculated to be +6.66. After the shift, the highest clip's 95th-percentile pitch is 312 Hz. At the same percentile the target voice's own readings reach a median of 333 Hz, and 50 of its 61 readings reach at least 312 Hz. The shifted clips therefore stay within the range the target voice uses when it reads. We chose this check after seeing the numbers.

Melody step. When Vāgdhenu chants a new verse, its renderer takes the melody and pace from a reference clip, chosen by meter from a bank of clips. Nandinī uses the same renderer, with a bank cut from the converted clips (Section 4). The converted clips therefore have to carry the reciter's melody twice over, as training data for the model and as references when the model chants.

The converter does not keep that melody, because its output follows a pitch contour of its own. IndicVoiceChanger is a speech model with no pitch input, so the reciter's pitch cannot be given to it and has to be put back afterwards. The melody step therefore keeps the converter's voice and puts back the reciter's pitch.

The step uses the WORLD vocoder (Morise et al., 2016), which splits speech into three parts. The pitch carries the melody, the spectral envelope carries the timbre of the voice, and the aperiodicity carries the noise-like part of the sound. Both the reciter's recording and the conversion are analysed at 24 kHz in 5 ms frames, and the candidate is synthesised from parts of both. Pitch is tracked with WORLD's Harvest method (Morise, 2017) and refined with StoneMask. The spectral envelope and aperiodicity come from the conversion, trimmed or padded to the recording's length. The pitch comes from the recording, raised by the k semitones measured above. The envelope already belongs to the target voice, and k puts the pitch where that voice speaks, so the two fit together. The recording also decides which frames are voiced. A frame is voiced where the vocal folds vibrate, as in a vowel, and unvoiced where they do not, as in the hiss of a sibilant such as *s* or *ś*, or a breath. The loudness of each frame is matched to the reciter's, with the gain smoothed over about 20 ms. Finally, the whole candidate is set to an average level of −20 dBFS. Whatever pitch the converter produced, the candidate therefore has the reciter's timing and melody, transposed into the new voice's range.

Two parts of this step were tested on 61 verses with ten conversions each before the full run. The first is the loudness match. Where the conversion is nearly silent but the recording is not, matching the reciter's loudness boosts the conversion by 20 to 74 dB and turns faint noise into a loud spike. The spike then dominates the clip's level, so the rest of the clip ends up too quiet once the level is set and the peaks are limited. Without a limit on the boost, 54.8% of the 610 clips ended more than 3 dB below −20 dBFS. The boost is therefore capped at 6 dB above the clip's typical gain, which brings that share to 0.2%. The cap adds no errors. With it, 3.35% of syllables are wrong, against 3.53% without it, a difference of −0.18 points (95% interval $[-0.51, +0.09]$).

The second is the voicing. Taking the reciter's voicing forces voiced sound where the conversion is unvoiced in a median 4.97% of frames, and unvoiced sound where it is voiced in 1.53%, counting only frames where either is voiced. The four longest such stretches, 305 to 430 ms, were judged fine in human verification, so the reciter's voicing is kept.

In pronunciation the step costs almost nothing. Counting every syllable, the recogniser flags 7.13% of syllables on the converter's output and 7.18% after the melody step, on the same 610 conversions. In voice the step has a cost. Closeness to the voice sample, measured as above, falls from 0.840 for the conversion to 0.688 after the melody step. The reciter's own recordings score 0.136, so the candidates remain far closer to the new voice than to his. This is probably because the reciter's pitch and loudness replace the converter's, and speaker embeddings respond to both. We accept the cost, because the step is what keeps the chant's timing and melody.

3.2 Scoring a candidate

The model learns pronunciation from these clips, so a sound that is reliably wrong in the data may be learned wrong by the model. The difficult cases are the sound distinctions that Sanskrit depends on, such as retroflex against dental consonants, aspirated against unaspirated stops, the three sibilants and dense conjuncts. Every candidate is therefore checked syllable by syllable with Su-śrotā (Prathosh, 2026b), the only recogniser we know of that was built for chant (Section 2). It is AI4Bharat's IndicConformer (AI4Bharat, 2024) fine-tuned on 17.4 hours of śāstric and recitational Sanskrit, and its model card reports a character error rate of about 6% on Bhāgavata chant.

A recogniser also makes mistakes on the original recording. A plain error rate against the text

would mix the conversion's damage with these mistakes, so each candidate is compared with the reciter's recording instead. The score follows six rules.

1. Su-śrotā transcribes the candidate and the reciter's recording with greedy CTC decoding.
2. Each transcript and the verse text are written in SLP1, a romanisation with one character per sound, without spaces or avagraha. A transcript that divides words differently is therefore not counted as wrong.
3. Each transcript is aligned with the text by edit distance.
4. A syllable is a vowel with the consonants before it and any anusvāra or visarga after it. Consonants at the end of a word join its last syllable.
5. A syllable is flagged if any of its characters is changed or dropped. An extra character in the transcript flags the syllable just before it, or the first syllable if it comes at the very start.
6. An *error* is a syllable flagged on the candidate but not on the reciter's recording.

Syllables flagged on the recording are left out, because there the recogniser already disagrees with the text and a flag cannot be blamed on the conversion. Across all 1,467 of the reciter's recordings, the recogniser misreads 4.25% of syllables (2,911 of 68,572). For these syllables the score cannot tell whether a candidate pronounces them correctly, so if the conversion damaged one of them, the score would not show it.

The score has limits, because it reflects only what the recogniser hears. First, it can miss real errors. It compares transcripts with the verse text, so a candidate that follows the text where the reciter chanted something else still passes. In one sample human quality check of 38 converted clips, for instance, the recogniser accepted all 13 instances of *kṣa*, including two that the check found mispronounced. Second, a single flag is weak evidence. At these error rates, nearly identical versions of the same audio can gain or lose a flag by chance (Section 3.3). Third, the same recogniser both selects each clip and scores it, so the error rates it reports for our method are optimistic. Human verification was limited to targeted spot checks on the cases the measurements flagged as most at risk, such as the longest changes in voicing and the largest loudness boosts.

3.3 Keeping a candidate

Candidates for a verse are generated one at a time, and the first with at most one error is kept. If none of the ten gets there, the verse keeps the one with the fewest errors, and the earliest on ties. No verse is dropped, so the dataset keeps all the verses, styles and meters of the original.

We allow one error rather than none for two reasons, both measured on 61 verses with ten candidates each before the full run. First, one error is within the recogniser's own resolution. On these verses Su-śrotā misreads 4.66% of syllables on the reciter's own recordings, which are taken as correct, while one error is about 2.2% of a median verse of 46 syllables. Second, a stricter rule would need many more candidates for little gain, and each candidate costs a full conversion, melody step and transcription. Stopping only at zero errors would have taken 335 candidates on these verses instead of 195, or 1.72 times as many, and would have lowered the error rate of the kept clips only from 2.09% to 1.69%. Because the seeds are fixed, a stricter rule could still be applied later by continuing each verse from its next candidate.

Each candidate is judged on its final audio, after the melody step. This matters because a choice flatters the version of the audio it was made on. On the same 61 verses, the best of ten conversions, judged on the converter's output, had 1.33% of its syllables flagged there. After the melody step,

which changes the audio only slightly, the same conversion had 3.27% flagged, close to the 3.53% of an average conversion. Most of the apparent gain from choosing was therefore the recogniser's luck on that version. Judging the final audio removes this mismatch, but not the luck itself, so the error rate of the kept clips is still optimistic.

Re-check in halves. On some long verses the recogniser misreads a long stretch of the whole clip, the same way on every candidate. In one verse of 32 syllables it found 16 to 22 errors on all ten candidates. When each half of the verse was transcribed on its own, it found only 1 to 3, and none on the reciter's recording. A good clip can therefore miss the bar because of the recogniser.

	Verses	At most 1 error, first pass	After re-check in halves	Gain
First style	764	680 (89.0%)	713 (93.3%)	+33
Second style	703	532 (75.7%)	585 (83.2%)	+53
All	1,467	1,212 (82.6%)	1,298 (88.5%)	+86

Table 3: Verses whose kept clip has at most one error, after the first pass on whole clips and after the re-check in halves.

So for each verse that missed the bar after ten candidates, every candidate is scored again in halves. The clip is cut where the verse's second half begins. That point is taken from a timed transcription of the reciter's recording, then moved in each clip to the quietest 20 ms nearby. Each half is transcribed on its own, and the two transcripts are joined and scored against the whole verse as before. If a candidate now has at most one error, the first such candidate replaces the kept one. Otherwise the verse keeps its best of ten. We added this step after seeing the full run. It replaces a clip only when a candidate reaches the bar in halves. A verse that still misses keeps the candidate chosen on whole clips. Switching it to the candidate with the fewest errors in halves would gain little, because for these verses the two scores usually differ by only one syllable, which is within the recogniser's noise. Of the 255 verses that missed the bar on the first pass, the re-check gives 86 a candidate at the bar. This raises the share of verses with at most one error from 82.6% to 88.5% (Table 3), and the second style, which starts lower, gains more.

3.4 Error patterns

Kind of error	Syllable errors	Share	Most frequent (errors)
Long *o* heard as *ū*	125	9.7%	*do* → *dū* (17), *ro* → *rū* (12), *so* → *sū* (9)
Voiced and unvoiced swapped	105	8.1%	*ta* → *da* (15), *ti* → *di* (9), *ca* → *ja* (7)
Vowel length changed	62	4.8%	*nā* → *na* (10), *rā* → *ra* (4), *vā* → *va* (3)
Long *e* heard as *ī*	47	3.6%	*de* → *dī* (19), *ve* → *vī* (10), *ṇe* → *ṇī* (4)
Syllable not heard	34	2.6%	*śi* (4), *va* (3), *pa* (3)
Aspiration added or dropped	25	1.9%	*ta* → *tha* (6), *da* → *dha* (5), *de* → *dhe* (2)
Other	891	69.1%	scattered
All syllable errors	1,289	100%	

Table 4: Syllable errors in the released dataset's kept clips, grouped by kind.

In the released dataset, after the re-check in halves, the kept clips have 1,289 errors, 1.97% of the 65,594 syllables checked. Table 4 groups them by kind, assigned by rule from what the recogniser

heard. About two thirds are scattered, with no substitution repeated often. The largest pattern is a long *o* heard as *ū*, as in *do* heard as *dū*, in 125 errors. It is frequent for a few syllables, at 18.9% of the checked *do* syllables, but it amounts to about 0.2% of all syllables checked. A long *e* heard as *ī* goes the same way, in 47 errors. Whether the conversion really raises these vowels, or Su-śrotā mishears them in a higher voice, was not tested. The next group is consonants heard with the wrong voicing, such as *ta* heard as *da*.

4 Model training

	Vāgdhenu	Nandinī
Training script and settings	released	same
Starting checkpoint	IndicF5	same
Training text	its own	same
Steps	14,800	same
Audio	reciter's recordings	converted clips
Reference bank	reciter's recordings	converted clips, same verses
Vocoder	BigVGAN-v2, fine-tuned on the reciter's recordings	BigVGAN-v2, starting from Vāgdhenu's and fine-tuned on the converted clips

Table 5: Nandinī's training against Vāgdhenu's.

Nandinī is trained as Vāgdhenu was, as closely as Vāgdhenu's release allows, and differs only where the new voice requires it, in the audio, the reference bank and the vocoder (Table 5). The model is IndicF5 fine-tuned on the converted clips with Vāgdhenu's own training script and settings, unchanged. The training text is Vāgdhenu's, prepared with its own script, and training starts from the same IndicF5 checkpoint, with a learning rate of 10^{-5}, 500 warm-up steps and 19,200 frames per GPU in bf16. It runs for 14,800 steps, the step of Vāgdhenu's released model. Training took 216 minutes on two RTX A6000 GPUs, and the run cost $5.67 in all.

The training loss fell from 0.746 over the first 50 logged steps to 0.590 over the last 50. On 61 training verses, the character error rate changed by +0.19 points (95% interval $[-0.16, +0.54]$) between steps 12,000 and 14,800, so the model had levelled off and step 14,800 was kept. Its renders of these verses score 0.905 in closeness to the converted clips, against 0.220 for the reciter's recordings. These are training verses, so the checks show that the model speaks in the new voice, not that it generalises. Section 5 tests it on verses new to both systems.

Reference bank. When Nandinī chants, Vāgdhenu's renderer takes the melody and pace from a reference clip for the verse's meter (Section 3.1). Nandinī's bank has the same 20 entries as Vāgdhenu's, cut from the converted clips of the same verses, so mainly the voice changes. 18 of the cuts use the same window as Vāgdhenu's. The other 2 use the second half of the verse, because Su-śrotā marked the same syllables wrong in the first half of all ten candidates. Scored by Su-śrotā as in Section 3.2, 11 of the cuts have no errors, 8 have one and 1 has two.

Vocoder. The vocoder turns the model's spectrogram into audio. Neither available vocoder suited the new voice. Vāgdhenu's BigVGAN was fine-tuned on the reciter's voice, and it adds an echo to the new one. IndicF5's default vocoder is Vocos, and it makes held notes shiver. Vāgdhenu solved the same problem for the reciter by fine-tuning NVIDIA's BigVGAN-v2 (Lee et al., 2023)

on the reciter's recordings. For the new voice, we fine-tuned Vāgdhenu's BigVGAN further on the converted clips. Vāgdhenu's paper gives three facts about its fine-tune, and we follow all three. First, the vocoder is BigVGAN-v2. Second, it is trained on IndicF5's spectrograms. Third, its final weights average the last checkpoints. We started from Vāgdhenu's BigVGAN rather than NVIDIA's because it already handles IndicF5's spectrograms. Training uses NVIDIA's code, default settings and released discriminators on all 1,467 converted clips, because Vāgdhenu's release does not include its own. Training ran on two A40 GPUs and stopped at step 10,000. The final weights average the checkpoints from step 5,000 to 10,000. The fine-tune cost $4.32.

Vocoder	Echo	Shiver	CER (%)
Vocos (IndicF5's default)	0.007	0.402	3.07
Vāgdhenu's BigVGAN	0.231	0.315	3.14
BigVGAN, fine-tuned here	0.000	0.257	3.00

Table 6: The three vocoders, averaged over the same spectrograms of 20 training verses with long held notes. Lower is better in each column.

Table 6 compares the three vocoders on the same spectrograms of 20 verses. The echo is the cepstral peak at the model's frame rate of 93.75 Hz. The shiver is the share of a held note's amplitude modulation between 5 and 30 Hz. The fine-tune has no echo, unlike Vāgdhenu's BigVGAN. Its shiver is about a third lower than Vocos's, a change of -0.144 (95% interval $[-0.217, -0.073]$). Its character error rate is no higher than either.

5 Model evaluation

Meter family	Dataset clips	Test verses
anuṣṭubh	398 (27.1%)	23 (38.3%)
triṣṭubh	245 (16.7%)	14 (23.3%)
jagatī	194 (13.2%)	11 (18.3%)
vasantatilakā	147 (10.0%)	8 (13.3%)
sragdharā	92 (6.3%)	–
mālinī	73 (5.0%)	–
rucirā	53 (3.6%)	–
17-syllable	41 (2.8%)	2 (3.3%)
śārdūlavikrīḍita	27 (1.8%)	2 (3.3%)
other	197 (13.4%)	–
Total	1,467 (100%)	60 (100%)

Table 7: Meter families of the 60 unseen test verses used to compare Nandinī with Vāgdhenu, with the training clips for comparison.

To test whether Nandinī generalises, we compare it with Vāgdhenu on 60 verses that neither model was trained on. The verses come from the Bhāgavata Purāṇa (56) and the Mukundamālā (4) in the Digital Corpus of Sanskrit (Hellwig, 2010–2024), released under CC-BY 4.0. They are drawn at random among verses whose meter Vāgdhenu's detector recognises, in proportion to the dataset's meter families. None of these verses, or any of their lines, are in the training data. At most, a test verse shares a word or two with the training text. The draw covers six of the dataset's

meter families (Table 7). The other four hold 415 of the 1,467 clips (28.3%) and had no recognised verse in these works, so the test does not cover them.

All three systems (Nandinī, Vāgdhenu and Vāgdhenu's steered model) chant each verse once, through the same path as Vāgdhenu's website. This path is Vāgdhenu's frontend, its meter detector with its fallback meter, and its renderer with its default settings and seed. Only the weights, the reference bank and the vocoder differ between systems. Nandinī is the model of Section 4. Vāgdhenu is its released fine-tune, the same kind of model as Nandinī. Vāgdhenu's website serves a third model, which we call the steered model. It is the same fine-tune, retrained on 179 paired clips to follow the reference clip more closely (Prathosh, 2026a). Vāgdhenu's release does not include the code for this retraining, so Nandinī has no such steered model. The renders for this section ran on one A40 GPU and cost $1.55.

Measure	Bar	Nandinī vs Vāgdhenu	Steered model vs Vāgdhenu	Met
CER difference (points)	$\leq +2.0$	$+0.14$ $[-0.38, +0.64]$	-0.32 $[-0.85, +0.17]$	yes
Melody correlation	≥ 0.8	0.237 $[0.182, 0.292]$	0.604 $[0.537, 0.662]$	no
Duration ratio	1 ± 0.1	1.045 $[1.039, 1.052]$	1.022 $[1.017, 1.028]$	yes

Table 8: Character errors, melody and pace, averaged over the 60 test verses, with 95% confidence intervals.

Errors, pace and melody. Nandinī meets two of the three bars in Table 8. The bars were set before any verse was rendered. Each interval in this section is a 95% confidence interval from a bootstrap that resamples the verses 10,000 times. Its character error rate (CER) is 3.81% against Vāgdhenu's 3.67%, and the interval of the difference includes zero. Its renders are 4.5% longer than Vāgdhenu's. It fails the melody bar, with a correlation of 0.237 against a bar of 0.8. So Nandinī does not meet all three bars. Su-śrotā chose the training clips and also scores these renders, so the two share its blind spots (Section 6).

We dropped the melody measure after seeing this result, but not because Nandinī missed its bar. The measure itself cannot support a conclusion, for three reasons. First, the steered model scores only 0.604 against Vāgdhenu, although both are the reciter's voice, so 0.8 was out of reach for any system. Second, the measure cannot separate a different tune from a different voice. It tracks each render's pitch with pYIN (Mauch and Dixon, 2014) in semitones around the render's median. It then aligns the two renders by dynamic time warping on MFCCs and correlates the aligned pitch curves. MFCCs carry the voice's timbre, so a voice 6.6 semitones higher aligns worse. Third, the measure is low even where the tune should match. Against the reciter's own recordings of 61 training verses, Vāgdhenu scores 0.204 and Nandinī 0.159, a difference of -0.045 (95% interval $[-0.108, +0.017]$). On the six demo verses of Vāgdhenu's website, the steered model run through the same path scores between 0.262 and 0.749 against the published renders. Each replacement we considered needed a reference with flaws of its own, so none was adopted. The tune was judged in blind pairs instead.

System	Steadiness	Median pitch (Hz)
Nandinī	0.739 $[0.729, 0.748]$	251
Vāgdhenu	0.736 $[0.722, 0.749]$	170
Steered model	0.725 $[0.713, 0.736]$	170

Table 9: Each system's voice on the 60 test verses, as averages with 95% confidence intervals.

Voice. On the test verses, Nandinī's voice is as steady as Vāgdhenu's (Table 9). Steadiness here is the lowest closeness of any 2-second window to the render's own average. Nandinī's pitch is 6.64 semitones above Vāgdhenu's on the same verses (95% interval $[6.56, 6.70]$). The shift applied to the training data was 6.66 semitones (Section 3.1), so the model reproduces it almost exactly.

Blind pairs. By ear, Nandinī's tune matches Vāgdhenu's fully or mostly in 5 of 8 blind pairs. In the other 3 the melody is also similar, and only the pauses fall in different places. In each pair, one listener compared Nandinī's and Vāgdhenu's renders of the same verse. Two pairs had the largest gap in character errors against Nandinī, two had the lowest melody correlation, and four were drawn at random. The order within each pair was random, and loudness was matched. The voices are audibly different, so the pairs are blind to order only. The listener judged the tune and rhythm the same in 4 pairs and mostly the same in 1. In the other 3, the differences were in the pauses, not in the rise and fall of the melody. Nandinī paused more than Vāgdhenu in two of them and less in one. A mistake was heard from Nandinī in 2 pairs and from Vāgdhenu in 1, and none in the two pairs drawn for Nandinī's largest error gap. The test verses have no daṇḍas (verse-end marks) or line breaks, so the renderer passes each verse as one line and the model chooses where to pause. Whether Nandinī's pauses follow the verse's quarters (*pāda*) less well than Vāgdhenu's is untested.

6 Limitations

Evaluation. As with Vāgdhenu (Prathosh, 2026a), the evaluation is the main limitation of this work. Every error rate comes from Su-śrotā, and Su-śrotā also chose the training clips. The rates are therefore optimistic, and they share the recogniser's blind spots. For example, whether a long *o* heard as *ū* comes from the conversion or from the recogniser was not tested (Section 3.4). The tune was judged by ear, because the melody measure could not separate a change of tune from a change of voice (Section 5). One listener judged it in 8 blind pairs, which is not a controlled listening study. The same listener compared the vocoders in 3 pairs, and the fine-tune happened to be played first in all 3. Human verification of the dataset was limited to targeted spot checks.

Pauses. Nandinī and Vāgdhenu pause in different places in 3 of the 8 blind pairs. The test verses have no verse-end marks, so the model chooses where to pause. Whether Nandinī's pauses follow the verse's quarters as well as Vāgdhenu's was not tested.

Untuned converter settings. The converter's settings were not tuned (Section 3.1). Each test verse was rendered with one seed, so the variation between renders was not measured.

7 Release

Everything needed to use and rebuild Nandinī is released. The code is on GitHub[1]. It makes the dataset from Vāgdhenu's recordings, trains the model and the vocoder, and chants a verse through Vāgdhenu's renderer. The dataset of 1,467 converted clips is on Hugging Face[2]. The model, the reference bank and the vocoder are in a Hugging Face model repository[3], together with ONNX

[1] https://github.com/ram-g-athreya/nandini
[2] https://huggingface.co/datasets/ramgathreya/nandini-data
[3] https://huggingface.co/ramgathreya/nandini

versions of the model and the vocoder. Sample chants by Nandinī and Vāgdhenu are on the project website[4].

ONNX. The ONNX version runs Nandinī with ONNX Runtime (ONNX Runtime developers, 2021) instead of PyTorch. It follows Vāgdhenu's renderer step by step, and Hemanth's browser version of Vāgdhenu (HM, 2026) served as a reference for splitting the model into graphs. On five test verses rendered from the same noise, it reproduces the PyTorch renderer. The audio has the same length to the sample, and Su-śrotā makes the same number of errors on every verse. A compact set of files with weights stored as int8 is released too. On the same five verses, its renders differ from the full set's about as much as another seed's do. That was enough to miss the bars set for it, so it is labelled as an approximation.

8 Conclusion

Nandinī is a female chanting voice for Vāgdhenu that needed no new recordings. Its training data is Vāgdhenu's own chant, converted into a voice taken from a single 25-second text-to-speech sample, with the reciter's melody put back. Su-śrotā finds at most one syllable wrong in 1,298 of the 1,467 converted clips. Fine-tuned on these clips with Vāgdhenu's recipe, Nandinī chants 60 new verses about as accurately as Vāgdhenu, with character error rates of 3.81% and 3.67%, and at nearly the same pace. By ear, its tune matches Vāgdhenu's fully or mostly in 5 of 8 blind pairs. In the other 3 the melody is also similar, and only the pauses fall in different places. Two questions remain open. The first is whether Nandinī pauses within a verse as well as Vāgdhenu does. The second is whether a long *o* heard as *ū* comes from the conversion or from the recogniser.

Use of AI

Claude (Anthropic) assisted throughout this work. It wrote and ran most of the code, including the dataset conversion, the training runs and the scoring, and drafted much of the text. The author set the research question and goals, chose the voice, decided and approved every experiment, verified the outputs, and directed and revised the writing.

References

AI4Bharat. IndicConformer: Speech recognition for the 22 official Indian languages. https://github.com/AI4Bharat/IndicConformerASR, 2024.

Bodhan AI and AI4Bharat. Indic-Speak: Text-to-speech for 22 Indian languages and English. https://huggingface.co/bodhan-ai/indic-speak, revision 76e0814, 2026.

Paul Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proceedings of the Institute of Phonetic Sciences, University of Amsterdam*, volume 17, pages 97–110, 1993.

Paul Boersma and David Weenink. Praat: doing phonetics by computer. Computer program, version 6.1.38 (as bundled in Parselmouth 0.4.7), https://www.praat.org, 2021.

[4] https://ram-g-athreya.github.io/nandini/

Yushen Chen et al. F5-TTS: A fairytaler that fakes fluent and faithful speech with flow matching, 2024.

Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification. In *Proc. Interspeech*, 2020.

DreamSyncCo. IndicVoiceChanger. https://huggingface.co/DreamSyncCo/IndicVoiceChanger, revision 6fb95c7, 2025.

Lev Finkelstein, Heiga Zen, Norman Casagrande, Chun-an Chan, Ye Jia, Tom Kenter, Alexey Petelin, Jonathan Shen, Vincent Wan, Yu Zhang, Yonghui Wu, and Rob Clark. Training text-to-speech systems from synthetic data: A practical approach for accent transfer tasks. arXiv:2208.13183, 2022.

Oliver Hellwig. DCS: The digital corpus of Sanskrit. https://github.com/OliverHellwig/sanskrit, 2010–2024.

Hemanth HM. sanskrit-tts-web: Vāgdhenu in the browser with ONNX Runtime Web. https://github.com/hemanth/sanskrit-tts-web; weights https://huggingface.co/gnumanth/sanskrit-tts-onnx, 2026.

Goeric Huybrechts, Thomas Merritt, Giulia Comini, Bartek Perz, Raahil Shah, and Jaime Lorenzo-Trueba. Low-resource expressive text-to-speech using data augmentation. In *Proc. ICASSP*, 2021.

Min-Jae Hwang, Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. TTS-by-TTS: TTS-driven data augmentation for fast and high-quality speech synthesis. arXiv:2010.13421, 2020.

Yannick Jadoul, Bill Thompson, and Bart de Boer. Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*, 71:1–15, 2018.

Tahir Javed, Janki Atul Nawale, Eldho Ittan George, Sakshi Joshi, Kaushal Santosh Bhogale, et al. IndicVoices: Towards building an inclusive multilingual speech dataset for Indian languages. arXiv:2403.01926. Dataset: https://huggingface.co/datasets/ai4bharat/IndicVoices, 2024.

Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. BigVGAN: A universal neural vocoder with large-scale training, 2023.

Songting Liu. Zero-shot voice conversion with diffusion transformers, 2024.

Leonardo B. de M. M. Marques, Lucas H. Ueda, Mário U. Neto, Flávio O. Simões, Fernando Runstein, Bianca Dal Bó, and Paula D. P. Costa. Improving data augmentation-based cross-speaker style transfer for TTS with singing voice, style filtering, and F0 matching, 2024.

Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, 2014.

Venkata Pushpak Teja Menta. PSP: An interpretable per-dimension accent benchmark for indic text-to-speech, 2026.

Masanori Morise. Harvest: A high-performance fundamental frequency estimator from speech signals. In *Proceedings of Interspeech*, pages 2321–2325, 2017.

Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884, 2016.

Tijana Nosek, Siniša Suzić, Nikola Simić, Milan Sečujski, Darko Pekar, and Vlado Delić. Exploiting voice conversion in creating new TTS voices. In *Proc. 32nd International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2025. https://ieeexplore.ieee.org/document/11151916.

ONNX Runtime developers. ONNX Runtime. https://onnxruntime.ai, 2021.

A. P. Prathosh. Vāgdhenu: A vṛtta (meter) aware śloka-to-chant (TTS) system for Sanskrit, 2026a.

A. P. Prathosh. Su-śrotā: Scholar-grade Sanskrit ASR and metre-aware chant practice. https://huggingface.co/prathoshap/sushrota-sanskrit-asr, 2026b.

N. Rama and Meenakshi Lakshmanan. An algorithm based on empirical methods, for text-to-tuneful-speech synthesis of Sanskrit verse. arXiv:1409.4169, 2014.

Mirco Ravanelli et al. SpeechBrain: A general-purpose speech toolkit. arXiv:2106.04624. Model used: https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb, 2021.

Ashwin Sankar, Yoach Lacombe, Sherry Thomas, Praveen Srinivasa Varadhan, Sanchit Gandhi, and Mitesh M. Khapra. RASMALAI: Resources for adaptive speech modeling in Indian languages with accents and intonations. arXiv:2505.18609. Model released as Indic Parler-TTS, https://huggingface.co/ai4bharat/indic-parler-tts, 2025.

Hubert Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis, 2023.

Praveen Srinivasa Varadhan, Srija Anand, Soma Siddhartha, and Mitesh M. Khapra. Phir Hera Fairy: An English fairytaler is a strong faker of fluent speech in low-resource Indian languages. arXiv:2505.20693. Model released as IndicF5, https://huggingface.co/ai4bharat/IndicF5, 2025.

Taehyung Yu and Seongjae Kang. Best-of-N TTS evaluation is confounded by ASR family alignment, 2026. ICML 2026 Workshop on Machine Learning for Audio.