Unified Offline Voice Assistant Architecture

ASR (Hindi) + Translation (Hindi → English)
Target: **Raspberry Pi 4 (ARM Cortex-A72, 4GB RAM, Offline, INT8 optimized)**

1.COMPLETE SYSTEM ARCHITECTURE

End-to-End Signal Flow

16kHz Audio
↓
WebRTC VAD (20 ms frames)
↓
Feature Pipeline (Model dependent)
↓
ASR Model (CTC or Seq2Seq)
↓
Hindi Text
↓
Tokenizer (SentencePiece/BPE)
↓
Translation Model (Transformer)
↓
English Text
↓
Intent Model (Optional)
↓
Action + TTS

---

2. AUDIO FRONTEND (COMMON FOR ALL MODELS)

Sampling:

$$f_s = 16000 \text{ Hz}$$

Frame Duration:

$$T = \frac{320}{16000} = 20ms$$

VAD:

- WebRTC VAD
- Removes silence
- Reduces compute 40–60%

3.ASR MODELS – UNIFIED TECHNICAL ANALYSIS

A) COQUI STT (DeepSpeech2 Based)

Architecture

$$X \rightarrow \text{CNN} \rightarrow \text{Bi-GRU} \rightarrow \text{Dense} \rightarrow \text{CTC}$$

Spectrogram Input:

$$X \in \mathbb{R}^{T \times F}$$

CTC Probability:

$$P(y \mid X) = \sum_{\pi \in B^{-1}(y)} \prod_{t} P(\pi_t \mid X)$$

Loss:

$$L_{CTC} = -\log P(y \mid X)$$

Optimization Strategy

Convert to TFLite

- Strip training ops
- Freeze graph

INT8 Quantization

Weight scaling:

$$W_q = \text{round}(W/s)$$

Reduces:

- Memory ×4
- CPU latency ×2

Decode Optimization

- Disable beam search
- Use greedy decoding

Best Use:

✓ Lightweight edge deployment
✓ When RAM < 1GB

B) WAV2VEC 2.0 (Self-Supervised Transformer)

Architecture

$$X \rightarrow Conv1D \rightarrow TransformerEncoder \rightarrow Linear \rightarrow CTC$$

Feature encoder:

$$Z = Conv1D(X)$$

Transformer:

$$H = \text{SelfAttention}(Z)$$

CTC output:

$$Y = Softmax(WH)$$

Fine-Tuning

- Replace CTC head
- Freeze feature encoder (optional)
- Train with CTC loss

Metric:

$$WER = \frac{S + D + I}{N}$$

Optimization Strategy

Layer Reduction

Reduce transformer layers (12 → 6)

ONNX Export

torch.onnx.export(...)

Dynamic INT8 Quantization

$$W_{int8} = \text{round}(W/scale)$$

ONNX Runtime ARM Build

Memory After Quantization:

~300–500MB

Best Use:

✓ High accuracy Hindi
✓ Moderate Pi deployment

C) WHISPER (Encoder-Decoder Transformer)

Architecture

$$Mel(X) \rightarrow Encoder \rightarrow Decoder \rightarrow Tokens$$

Autoregressive:

$$P(y \mid X) = \prod_{t} P(y_t \mid y_{<t}, X)$$

Loss:

$$L = -\sum \log\ P(y_t \mid y_{<t}, X)$$

Optimization Strategy

Use Tiny / Base

Avoid Large models.

Disable:

- Language detection
- Large beam search

Export to ONNX

INT8 Quantization

Tradeoff:

Higher accuracy than Wav2Vec2
Slightly slower due to decoder loop

D) VOSK (Kaldi Hybrid HMM-DNN)

Architecture

$$MFCC \rightarrow DNN \rightarrow HMM \rightarrow WFSTDecoder$$

Bayesian decoding:

$$P(W \mid X) = P(X \mid W)P(W)$$

Optimization

- Small Hindi model (~50MB)
- Grammar-restricted decoding
- No transformer overhead

Latency:

< 50ms

Best Use:

✓ Ultra low CPU
✓ Production stability

4.TRANSLATION MODEL

(Hindi → English)

Model:

OPUS-MT / MarianMT (Transformer Seq2Seq)

Architecture

$$X_{hi} \rightarrow Encoder \rightarrow Decoder \rightarrow Y_{en}$$

Attention:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Loss:

$$L = -\sum \log\ P(y_t \mid y_{<t}, X)$$

Fine-Tuning

- Parallel corpus
- SentencePiece tokenization
- CrossEntropy loss

- Evaluate BLEU

BLEU:

$$BLEU = BP \cdot \exp\left(\sum w_n \log\, p_n\right)$$

Optimization

1. **Reduce max_length=64**
2. **Reduce attention heads**
3. **ONNX export**
4. **Dynamic INT8 quantization**

Final Memory:
~80–120MB

5.COMMON OPTIMIZATION PIPELINE

Pruning

$$W' = W \cdot \mathbb{1}(|\ W\ | > \tau)$$

Removes low magnitude weights.

Quantization

Linear mapping:

$$x_{int8} = \frac{x_{fp32}}{scale}$$

Memory:
4 bytes → 1 byte

Speed:
2–4× on ARM

ONNX Simplification

onnxsim model.onnx simplified.onnx

Threaded Execution

Threads:

1. Audio Capture
2. ASR Inference
3. Translation Inference
4. Intent Execution

Queue streaming:

Mic → Queue → ASR → Queue → Translation

Lock interpreter (important for TFLite).

6.DEPLOYMENT STRATEGY MATRIX

Hardware Constraint	**Recommended Stack**
Very Low RAM	Vosk only
Balanced	Wav2Vec2 INT8 + MarianMT INT8
Highest Accuracy	Whisper Tiny INT8 + MarianMT INT8

7. EXPECTED PERFORMANCE (RPI 4)

After INT8:

Latency:
50–120 ms (depending model)

Memory:
< 500MB

CPU:
60–80%

8.FINAL PRODUCTION ARCHITECTURE (RECOMMENDED)

Balanced System:

Audio (16kHz)
↓
WebRTC VAD
↓
Wav2Vec2 INT8 ONNX
↓
Hindi Text
↓
MarianMT INT8 ONNX

↓

English Text

Low Hardware Mode:

Audio

↓

Vosk

↓

Hindi Text

↓

MarianMT INT8

---

9. COMPLETE LIBRARY STACK

Training:

- torch
- transformers
- datasets
- evaluate
- accelerate
- sentencepiece

Optimization:

- onnx

- onnxruntime
- onnxsim
- torch.nn.utils.prune

Deployment:

- onnxruntime (ARM build)
- vosk
- tflite-runtime
- webrtcvad
- sounddevice / pyaudio
- numpy

10.**ENGINEERING CONCLUSION**

For Raspberry Pi 4 offline assistant:

Most stable:

Vosk + MarianMT INT8

Best balanced:

Wav2Vec2 INT8 + MarianMT INT8

Maximum accuracy:

Whisper Tiny INT8 + MarianMT INT8

Avoid:

- Full TensorFlow runtime
- Large transformer models
- Unquantized inference

If you want next, I can provide:

• Complete ONNX export + INT8 quantization code pipeline
• Full multithreaded production-ready inference architecture
• Memory and latency benchmarking framework