

TECHNICAL REPORT

知音 (Zhiyin) : 面向消費級硬體的可商用、 情緒可控中文語音合成系統——設計、引擎評 比與效能實證

江家寓 Chia-Yu Chiang

銘傳大學 生物醫學工程學系

yuyulsc881209@icloud.com · github.com/yu8812

2026-09-29

本文記錄單機環境下的工程決策與量測結果；樣本規模與統計限制詳見「效度威脅」一節。

目錄

摘要 (Abstract)

1. 緒論 (Introduction)

2. 相關工作與候選引擎 (Related Work)

3. 系統設計 (System Design)

4. 引擎評比：方法與結果 (Engine Evaluation)

5. 情緒控制設計 (Emotion Control)

6. 效能實證與硬體限制 (Performance & Hardware Constraints)

7. 討論 (Discussion)

8. 結論與未來工作 (Conclusion & Future Work)

附錄：可重現性 (Reproducibility)

參考文獻(References)

技術報告 v0.3。本文記錄有聲書/敘事 TTS 產品底層自建語音引擎「知音」的設計決策、多引擎評比，以及在 8GB 消費級 GPU 上的效能量測。目前狀態：雙軌音色（真人克隆/微調 × 設計音）、跨引擎配音標記系統（情緒/停頓/氣口/音效/讀音）、ASR-QC 重抽產線、雙說話者真人資產管線與資料量實驗均已完成，並通過多題材壓測。

摘要 (Abstract)

單一現成的中文語音合成 (TTS) 方案難以同時滿足情緒豐富、可克隆音色、可商用、8GB 本地的聯合需求，且「最佳引擎」隨開源生態快速變動。本研究提出知音 (Zhiyin)：一個模型無關 (model-agnostic) 的 TTS 後端抽象層，以統一介面 + 跨環境隔離讓相依互斥的多個引擎熱插拔，使「引擎評比與替換」從架構改動降為換一個欄位；並以此對四個候選引擎 (GPT-SoVITS、CosyVoice2、seed-vc 串接、IndexTTS-2) 在「克隆 × 情緒 × 中文品質 × 可商用 × 8GB 本地」的聯合約束下，進行以客觀度量為主、母語者耳測為輔的系統性評比（沿用 Seed-TTS eval 協定：Whisper 拼音-CER 量內容正確性 + ERes2NetV2 說話者餘弦量音色保真）。本文的重點不在功能數量，而在受限硬體與實際授權條件下，以可重現的量測支持每一個工程決策。

主要結果與可轉移的負結果：（1）以聲音轉換 (VC) 串接情緒會逐層稀釋，直接淘汰整條 cascade 路線；（2）情緒與音色解耦的單模型 (IndexTTS-2) 能同時達成克隆、強情緒與中文品質；（3）在 8GB 上模型載入而非推論主導延遲，常駐 worker 將每句降至純推論 ~87 秒；（4）fp32 溢出顯存導致崩潰式變慢 (52→350 秒/步)，證明半精度在此硬體為必要而非選用；（5）合成音無法被零樣本二次克隆（拼音 CER 退化 11–57%），但微調可解（留出句 CER 3.3%、音色餘弦 0.892，且比來源 clip 更穩定）；（6）音色相似度在 ~5 句 (30 秒) 即飽和，而咬字正確率與素材量無關、由取樣方差主導（同句換 seed CER 在 3%–58% 間波動），故量產可靠性的正解是 ASR 驗證 + 重抽迴圈而非堆素材；（7）配音的細部控制（讀音/停頓/氣口/音效）可整體上移到引擎層跨模型共用，其中讀音控制以「內嵌拼音」與「同音字替換」兩種機制在三引擎上全部達成。全部結論以量測為依據並附效度威脅分析；過程中有兩次由量測推翻了我原先的假設（「40 分/句」實為外部 GPU 競用、「28 句版壞掉」實為取樣方差）。

1. 緒論 (Introduction)

1.1 問題與動機

「知音 TTS」是一個以商用為目標開發的中文有聲書/故事產品，本文討論的是其自建語音引擎「知音」。其語音層需同時滿足：（a）音色克隆（角色一致性）、（b）情緒表現（敘事張力，也是與同類產品區隔的關鍵）、（c）中文母語級清晰度、（d）可永久本地、免授權費、可自主升級。單一現成方案難以全部滿足，且「最佳引擎」會隨開源生態快速變動。

1.2 研究問題 (Research Questions)

- **RQ1 (架構)** : 能否以單一抽象層讓多個相依互斥的 TTS 引擎熱插拔，使引擎評比與替換成為低成本操作？
- **RQ2 (引擎選型)** : 在「克隆 × 情緒 × 中文品質 × 可商用 × 8GB 本地」的聯合約束下，開源引擎的真實能力邊界為何？組合式 (VC 串接) 與端到端路線孰優？
- **RQ3 (可控性)** : 配音級的細部控制 (讀音/停頓/氣口/音效/情緒/語調轉移) 能否獨立於單一引擎、在引擎層統一提供，包括缺乏對應輸入通道的引擎？
- **RQ4 (可靠性)** : 取樣式 TTS 的單次輸出不可靠 (吞字/截斷)，量產級「近 100% 正確」應由什麼機制保證？
- **RQ5 (資料效率)** : 業餘使用者需要錄多少素材，克隆品質才「夠」？品質各維度 (音色/咬字/情緒) 對素材量的敏感度是否相同？

1.3 貢獻 (Contributions)

1. **可抽換後端架構 + 跨環境隔離** (對應 RQ1) : Backend 介面 + 每引擎自帶 venv、subprocess/常駐 worker 橋接；換模型只需加一個檔案。此架構後來兩度派上用場：四引擎評比 (§4) 與主力對調 (§4.4) 皆零遷移成本。
2. **聯合約束下的系統性引擎評比** (對應 RQ2) : 統一介面、統一參考音、ASR+耳測雙軌；記錄淘汰原因與兩個可轉移的負結果：VC 串接稀釋情緒 (§4.2)、合成音不可零樣本二次克隆 (§6.8)。
3. **引擎層配音控制系統** (對應 RQ3) : 標記系統 (情緒/停頓/氣口/音效) + 跨引擎讀音控制 (內嵌拼音注入與同音字替換雙機制)；提出「能力上移/能力模仿 vs 能力移植」的設計原則 (§3.4–3.5)。
4. **ASR 驗證重抽產線** (對應 RQ4) : 常駐 ASR worker + 引擎級「生成→驗證→換 seed 重抽」迴圈，貫穿單句到整書 CLI；以多題材壓測驗證 (§6.12)。
5. **資料量甜蜜點實證** (對應 RQ5) : 音色相似度 ~5 句 (30 秒) 飽和、咬字正確率由取樣方差主導與素材量無關，據此導出分級錄音台本與「穩定性交給 QC」的產線設計 (§6.11)。

2. 相關工作與候選引擎 (Related Work)

零樣本音色克隆自編解碼語言模型路線 (VALL-E [1]) 與流式/多語系統 (YourTTS [2]、XTTS [3]) 以降快速演進；中文語境的代表性開源系統包括少樣本微調路線的 GPT-SoVITS [4]、指令控制的 CosyVoice 2 [5]，以及情緒與音色解耦、支援情緒向量/文字/參考音三通道的 IndexTTS-2 [6]。「從文字描述生成全新音色」由 MOSS-TTS 系列 (VoiceGenerator) [7] 提供，規避真人肖像授權。評測方法上，本文沿用零樣本 TTS 的主流客觀協定 (Seed-TTS eval [8])：ASR 字錯率 (內容正確性；裁判用 Whisper [9]) + 說話者嵌入相似度 (音色保真；用 ERes2NetV2 [10])；少樣本語者適應的資料效率問題可上溯 AdaSpeech 等遷移學習工作 [11]。與上述工作不同，本文關注的是工程整合層：單機 8GB 消費硬體、可商用授權鏈、跨引擎統一控制面與量產可靠性，這一層在文獻中較少被系統性記錄。

引擎	路線	中文	情緒控制	授權	本研究結論
GPT-SoVITS v2Pro	少樣本克隆	佳	無原生情緒旋鈕	寬鬆	初判為備案，真人微調後升為日常主力 (§4.4)
CosyVoice2	指令式 + 克隆	實測偏方言、不穩定	指令文字	寬鬆	淘汰 (中文品質不足)
seed-vc (VC 串接)	對任意 TTS 做情緒轉換	視來源	轉換式	寬鬆	淘汰 (情緒稀釋，見 §4.2)
IndexTTS-2	零樣本克隆 + 解耦情緒	原生佳	emo_vector / emo_text / emo_audio + 強度 α	bilibili 自訂授權 (可商用，設有營收/月活門檻)	採用為主力

3. 系統設計 (System Design)

3.1 後端抽象 (the swappable seam)

核心定義 Backend 介面 (`is_up()`、`synthesize(voice, text, lang, params) -> (wav, sr)`、`capabilities()`)。角色語音庫 (`Voice`) 以 `backend` 欄指定引擎；`engine` 只依賴介面，不知道任何特定引擎細節。新增引擎僅需實作介面 + 註冊一行。此設計使 §4 的評比成為「換一個欄位」而非「改架構」。

3.2 跨環境隔離 (multi-env isolation)

不同引擎相依互斥：GPT-SoVITS 綁 torch 2.0/cu118 + Python 3.9；IndexTTS-2 需 torch 2.8/cu126；Kokoro 需 onnxruntime $\geq$ 1.20 (Python 3.10+)。我們不強迫共存於單一環境，而是讓每個重型引擎跑在自己的 venv，主程式以 subprocess 橋接 (stdin 餵 JSON、stdout 收結果)，如同呼叫外部工具，完全不污染彼此環境。此模式同時用於 Kokoro 與 IndexTTS-2。

3.3 情緒標記 (markup)

文字以行內標記 `【情緒】` 切段 (`core/markup.py`，行為逐行重置以免情緒外溢到旁白)；情緒名對應全域情緒模板 (`voices/_emotions.json`)，解析為合成參數；角色可用同名情緒覆蓋全域版。同一處標記對任何角色、任何後端皆適用。

3.4 配音標記系統：能力上移到引擎層 (capability lifting)

情緒標記的原則被推廣為完整配音標記系統：`[停N]` (插入 N 秒靜音)、`[氣口:嘆]` (插入真人氣口聲：吸/嘆/笑/清嗓/喘/倒抽，素材自使用者錄音抽取，角色可覆寫)、`[音效:x]` (同機制、獨立素材夾)、`{pinyin}` (讀音覆寫，§3.5)。這些能力全部實作在引擎層 (標記解析→切段→逐段合成→素材插入→重採樣拼接)，因此任何後端都能直接使用。設計原則是：引擎層功能天生由所有模型共用，模型內在能力

(輸入通道)則無法移植。前者盡量上移，後者以「每角色選定單一主引擎」處理，避免一句話多模型接力造成音色斷裂。

3.5 跨引擎讀音控制：注入 vs 模仿 (reading control)

破音字/指定讀音是中文配音剛需，但三引擎僅 IndexTTS-2 具備拼音輸入通道。我們以兩種機制在三引擎全部達成 字{pinyin} 標記：(a) 內嵌拼音注入 (IndexTTS：樂{yue4} 在文字中置換為以空白包夾的拼音；ASR 實測聲調正確套用)；(b) 同音字替換 (GPT-SoVITS/MOSS 沒有拼音通道，因此在幕後把被標註字換成「主讀音=目標音」的常用同音字，如 樂{yue4} → 「悅」，顯示文字不變；映射表 = 頻率常用字 × pypinyin 主讀音反查，查無候選則退回原字)。(b) 背後的觀察是：讀音控制的本質是讓模型發出指定的音，不必依賴模型提供通道，能力可以在引擎層「模仿」而非「移植」；同一做法同時補齊了兩個引擎。兩機制皆經 ASR 驗證 (lè→yuè，顯示文本不變)。繁中使用者介面以注音 + 候選試聽呈現，幕後轉拼音，使用者全程不接觸拼音。

4. 引擎評比：方法與結果 (Engine Evaluation)

4.1 實驗設定與度量協定 (Experimental Setup & Metrics)

平台：RTX 2080 Super Max-Q (8GB, Turing)、i7-10875H、32GB RAM、Windows 11。所有引擎經同一 Backend 介面、同一參考音檔測試。

內容正確性：拼音字錯率 (pinyin-CER)。 合成音經 faster-whisper (large-v3、CPU int8、beam=5) 轉寫，與目標文本各自轉為無調號拼音音節序列 (pypinyin；數字先行中文化以消除轉寫格式假錯)，取音節級 Levenshtein 編輯距離 d 對目標長度 n 之比 $CER = d/n$ 。取拼音層的理由：繁簡同拼音 (裁判轉寫多為簡體)、直接度量「發音對不對」。已知偏差：ASR 語言模型會「腦補」修正真實誤讀 (低估錯誤)、對戲劇化語調與英文夾雜偏嚴 (高估錯誤)，故 CER 用於相對比較與迴歸偵測，關鍵結論輔以母語者耳測。

音色保真：說話者嵌入餘弦。 ERes2NetV2 (16kHz fbank) 對合成音與該說話者真實錄音集合中心取餘弦相似度；同批真實錄音彼此的平均餘弦 (intra) 與異人對照 (不同說話者) 提供上下錨點。

主觀評測：目標使用者 (母語者) 對照逐字文本盲聽；所有關鍵決策要求客觀+主觀一致方案 (§7 討論兩者互補性)。規模說明：本文屬單機案例研究，樣本數小 (見 §7.5 效度威脅)，結論定位為工程決策證據而非統計顯著性宣稱。

4.2 關鍵發現：VC 串接的情緒稀釋

「先用任意 TTS 生中性語音，再用 seed-vc 做情緒轉換」的串接路線，實測情緒逐層被稀釋：VC 為保真音色與內容，會壓抑情緒位移，最終強度不足以與競品區隔。結論：情緒應在合成階段原生產生，而非後處理轉換。此發現直接淘汰整條 cascade 路線。

4.3 採用 IndexTTS-2 的理由

單模型同時克隆 + 情緒，零中間損耗；情緒與音色解耦（可分別給音色 ref 與情緒向量/文字/參考），故「一套情緒套所有角色」為內建能力，無需 VC。中文原生品質佳，授權允許商用與微調（個人開發者遠未達其營收/月活門檻）。

4.4 真人資料到位後的主力再評估

真人錄音落地後重測推翻了初判的一部分：IndexTTS-2 的中文咬字弱為引擎固有（高品質真人 ref 亦然，ASR 與耳測一致），且零樣本會壓平說話者的抑揚；而 GPT-SoVITS 經真人微調後咬字近乎全對、中英混搭最乾淨、速度 2–8 s/句，情緒經 §5.1 的 ref 置換機制補齊。角色因此對調：日常主力改為真人微調的 GPT-SoVITS；IndexTTS-2 只在需要情緒向量、讀音注入、導演式語調轉移時使用，並搭配 §6.12 重抽 QC。三軸總結（情緒/細部控制/自然度）：沒有單一開源模型三軸全優。系統的價值在於以架構讓「每個角色選定當下最適引擎」成為低成本的決策，並能隨開源生態演進調整。

5. 情緒控制設計 (Emotion Control)

IndexTTS-2 提供三種解耦情緒輸入：emo_vector（8 維 [喜, 怒, 哀, 懼, 厭, 低落, 驚, 自然]，強且可控）、emo_text（自然語言描述，經其 Qwen 情緒模型解讀，較含蓄）、emo_audio（情緒參考音），皆配強度旋鈕 α 。知音把【情緒】標記對應到全域模板的 emo_vector（主）/ emo_text（備援）/ α ，經 SynthParams 傳入後端。設計取舍：emo_vector 直接、可重現、強度高，定為主路徑；emo_text 保留供細緻客製。§6.4 探討強度 α 與清晰度的權衡。優先序實作為 emo_audio > emo_vector > emo_text（顯式示範音蓋過標記情緒）。

5.1 無情緒引擎的情緒：參考音置換。 GPT-SoVITS 無情緒通道，但其音色/語氣強烈跟隨推理參考音，因此情緒可由「換成具名情緒變體的 ref」實現（使用者每情緒錄 2 句、3–5 秒，管線以 ASR 每對挑優）。實測七情緒（開心/憤怒/悲傷/溫柔/驚訝/恐懼/冷漠）可用；品質上限取決於使用者的演繹品質。此為「能力模仿」原則（§3.4）的又一例。

5.2 導演式語調轉移 (prosody transfer)。 IndexTTS-2 的 emo_audio 與音色解耦，故可以做到：使用者錄一段演繹示範，任何角色都能照著演，並保留自身音色。真人實測（示範：懸疑放慢拉尾音/熱情輕快兩種）：合成語調跟隨示範，使用者評「熱情（版）與我演示相當」。界線：轉移的是語調/情緒風格，非逐字音高曲線。延伸方向是演法預設庫：預錄專業演繹片段，讓不會配音的使用者「挑演法」而非自己演，搭配 §6.12 的成品母帶化，構成「錄 30 秒、挑演法、直接輸出成品」的低門檻配音路徑。

6. 效能實證與硬體限制 (Performance & Hardware Constraints)

實驗平台：RTX 2080 Super Max-Q（8GB）、i7-10875H、32GB RAM；模型置於 USB 外接碟。

6.1 延遲來源的拆解

初期觀察到 IndexTTS-2 「~40 分鐘/句」。經逐項測量，延遲實由三者構成，須分別處理：（1）外部 GPU 競用、（2）模型載入、（3）推論本身。

6.2 「40 分鐘」來自外部競用，而非引擎

在 GPU 被其他程式（遊戲）佔用 ~100% 時測得 40 分/句；GPU 空載重測，同一句降至 270 秒。教訓是：歸因前先排除環境變因，以量測而非直覺判斷。

6.3 常駐 worker：載入是主因

一次性 worker 每句重載 ~5.5GB 模型。改為 `--serve` 常駐（載一次、迴圈接單）後實測（fp16、GPU 空載）：

指標	數值
模型載入（一次性）	~183 s
純推論（每句）	~84–89 s
批次 4 句總計	529 s

推得每章 100 句 ≈ 2.6 小時（常駐）vs 7.5 小時（一次性）。載入而非推論主導小批量延遲。

6.4 精度可行域：fp32 在 8GB 上崩潰

嘗試以 fp32 換取品質，實測顯存溢出：fp32 佔 ~7.8GB，逼近 8GB 上限後溢入系統共享記憶體，擴散每步時間大幅上升（52 → 105 → 350 秒/步），單句外推 >2 小時。結論：8GB 上半精度（fp16）為必要條件，非選用。提速應改走減少擴散步數或換更大顯存，而非提高精度。

6.5 清晰度問題的歸因：文本個案 + 參考音，非引擎缺陷

初期某句（短促驚嘆 + 強情緒）母語者判定咬字不清，一度懷疑 fp16 或引擎。逐項排除後結論：（1）換成正常敘事長句，母語者判定咬字清晰，可見失敗屬於特定文本個案（短促驚嘆 + 高情緒疊加），非系統性；（2）官方文件與用例證實 fp16 為合法選項，且本系統 infer 呼叫與官方一致；（3）殘留的音色不討喜源於以合成語音（edge-tts）作克隆參考，也就是「複製的複製」造成的退化。因此修正方向不是調整精度，而是以乾淨的真人中性錄音作克隆參考（規格見系統文件 VOICES §9）；情緒由 `emo_vector` 解耦提供，故參考音應為中性。方法上的意義是：在歸咎模型/精度前，先以受控變因（文本、參考音）逐一排除，否則會誤砍 fp16（在此硬體為必要）而走錯路。

6.6 長文穩定度的控制手段與 8GB 約束的再現

神經 TTS 為取樣式輸出，同句多次生成結果不同，偶發咬字瑕疵，長文（數千句）會累積成品質問題。為此把可靠性相關參數接入後端：`seed`（固定即可重現，更換即重抽）、`gen_kwargs` 直通（如 `temperature`）。IndexTTS 有自己的參數尺度（`temperature=0.8`、`repetition_penalty=10.0`），與

GPT-SoVITS 的通用欄位尺度不能混用，所以另走獨立直通，不共用欄位。量產可靠性的做法是以微調降低錯誤率，再加上 ASR 自動 QC 重抽（撰寫本節時仍在規劃，後於 §6.12 完成）。

8GB 約束的再現：本研究多次觀察到，主機被其他程式佔用（RAM 82%、GPU 競用）時，fp16 顯存由 ~7.5GB 升至 ~7.8GB，超過 8GB 邊緣後溢出到共享記憶體，擴散每步由 ~3.5s 增至 ~240s（慢約 70 倍）；GPU 空載時同一工作為 87s/句。這再次確認延遲對機器是否空載非常敏感，8GB 顯存是固定的瓶頸。生產應在空載機/離峰批次，或換更大顯存/雲端 GPU。

6.7 安全性 (Security posture)

新增的 IndexTTS-2 後端 + 常駐 worker 經審查，維持既有的安全做法：不使用 eval、exec、pickle、os.system 或 shell=True；所有 subprocess 一律清單參數（無 shell 注入）；常駐 worker 走 stdin/stdout 管線（子行程，非對外網路服務）；torch.load 僅發生於 IndexTTS 套件載入其自有可信 checkpoints（沿用「只載可信模型檔」原則，因 torch.load 等同 pickle 反序列化）。生成參數 gen_kwargs 源自本地可信設定（voice.json / _emotions.json），非任意程式執行。既有原則不變：HTTP 服務僅綁 127.0.0.1（從不 0.0.0.0）。

6.8 語音「設計」與 ASR 客觀 QC：合成音不可二次克隆

為解「一人難備大量角色」，評估 MOSS-VoiceGenerator（從文字描述零樣本設計音色，無需參考音，Apache 授權、非真人音色，沒有肖像授權疑慮）。關鍵在以 ASR (Whisper) + 拼音字錯率 (CER) 做客觀量測（取代主觀聽辨），驗證「MOSS 設計音 → IndexTTS-2 克隆 + 情緒」這條管線：

	MOSS 自身直接合成	IndexTTS-2 克隆該合成音
中性 CER	0%	11-57%
強情緒 CER	（走 instruction）	35-39%（emo_alpha 由 1.0 降至 0.5 仍崩）

發現：合成語音再被另一模型克隆會嚴重退化（「複製的複製」），且降 temperature / emo_alpha / 換 seed 皆無法挽救。因此設計音色須由其生成模型直接合成，不可作為另一克隆模型的參考音。職責切分明確：克隆既有真人音用 IndexTTS-2（以 emo_vector 控制情緒）；設計虛構新音用 MOSS 直接合成（情緒走 instruction）。此 ASR-CER 流程同時提供了生成可靠性的客觀量測，後來成為量產自動品管（攔截不良樣本並重抽）的基礎。

已實作並驗證 (2026-07-01)：backends/moss.py（設計音色直接合成，【情緒】→instruction）實測 CER=0%，含多角色 + 多情緒 + 多句的完整有聲書 pipeline（main.py book）整段亦 0%。ASR-QC 收成可重用工具 zhiyin/qc/asr_qc.py。至此雙軌取得音色的做法定案：真人音以 IndexTTS-2 克隆並用 emo_vector 控制情緒；設計音以 MOSS 直接合成並用 instruction 控制情緒（兩者不混用）。「讓一般人也能配音（有真人錄音用真人，沒有也能做）」所需的引擎基礎已完成；規模化（音色目錄）則透過微調（§6.9）。

6.9 以微調解決克隆退化 (2026-07-02 驗證)

§6.8 證明合成音不可被零樣本克隆；本節證明微調可解。以一個 MOSS 設計音生成 24 句訓練資料 (固定 seed 鎖音色) → ASR-QC 清洗 (剔除生成瑕疵句，留 21) → GPT-SoVITS v2 本地微調 (8GB、~15 分)。留出句結果：拼音 CER 3.3% (零樣本克隆同一音色為 11–57%)、音色餘弦 0.892 (不同人對照 0.423)，且顯著高於來源 clip 彼此的一致性 (0.716)，表示微調不只保真，還修正了生成式音色的跨句漂移；速度較 MOSS 直接合成快 ~20×。機制解釋：零樣本克隆是「照著唸」(參考音瑕疵被繼承並放大)，微調是「學會這個音色的分布」(瑕疵被平均掉)。此結果使「設計音經微調成為可穩定使用的音色」成為音色目錄的量產路徑。

6.10 真人語音資產管線：從業餘錄音到雙說話者角色 (2026-07-03/05)

真人音色的品質瓶頸在錄音工程而非模型。我們把從業餘環境錄音到可用角色音色的過程固定為可重複的管線，並以第二位說話者 (1 小時內完成全程) 驗證其可轉移性：1. **切分對齊**：whisper 逐字時間戳 + 滑動視窗在字流上為每句台詞找最佳匹配 (拼音 CER 最低、同分取最晚，即自動選重唸後的版本)，可處理「句中換氣切斷 VAD」與「句間空隙不足」(樸素分段比對在實錄上大量失敗，逐字視窗版不需人工介入)；2. **靜音壓縮**：業餘朗讀在逗號處停頓 3–4 秒，直接訓練會讓模型學會異常長的停頓，因此把句內超過 0.7s 的靜音壓至 0.45s；3. **除噪**：低電平錄音 (RMS -34dB) 正規化會同步放大底噪並滲入模型，因此以原始檔安靜段取噪音指紋做頻譜門控 (48kHz)，重訓後耳測「雜音變小很多」且無可聞 artifacts；修音邊界：修復性處理 (除噪/電平/靜音) 用於素材，美化性處理 (EQ/壓縮器) 禁止 (模型會學到假聲)；4. **情緒挑優**：每情緒錄 2 句，逐 clip ASR 打分自動取優；5. **品管防呆**：檔案時長+峰值+RMS 統計比對可自動偵測「匯出事故」(實際抓到三檔皆為同一音檔的複本)；數字句的 ASR 假錯 (「三十」vs「30」) 以中文數字正規化消除。這些經驗整理成分級錄音守則 (電平 -12~-6dB、句間 ≥2s、句中停頓 <1s、情緒句 ≥3s)，並做成使用者可直接照著錄的台本。

6.11 素材量實驗：音色約 30 秒即飽和 (2026-07-05)

「要錄多少素材？」以同一說話者資料的子集 (0=零樣本/5/10/20/28 句) 各訓一版，同留出句量測：

素材量	音色餘弦 (vs 真人中心)	留出句 CER (單次)
零樣本 (僅 1 句 ref)	0.767	~9%
5 句 (~30 秒)	0.885	~9%
10 / 20 / 28 句	0.886 / 0.889 / 0.878 (持平)	7%–32% (高方差)

發現一：音色相似度在約 5 句即飽和，「像本人」的成本很低 (約 30 秒)；其後素材的價值在情緒 ref、氣口素材與穩定性，不在音色。**發現二**：咬字正確率與素材量無關，由取樣隨機性主導：同一模型同一句換 seed，CER 在 3%–58% 間波動 (一次整段英文被吞、一次近全對)。方法論警示：單次 CER 不足以評判模型好壞 (我們曾因兩次失敗的取樣誤判「28 句版壞掉」，第三個 seed 即近全對)；量產穩定性的正解是逐段 ASR 驗證 + 重抽 (§6.12)，而非堆素材。此曲線直接轉化為產品的分級錄音台本 (1 句可玩 / 5–10 句很像本人 / 完整版含情緒與氣口)。

6.12 生產閉環：QC 重抽產線、多題材壓測與成品母帶化 (2026-07-04/06)

§6.6 規劃的可靠性方案已完成：常駐 whisper worker (載一次，查核由 30–60s/次降至秒級) + 引擎級 QC (`synthesize(qc=True)`)：生成→拼音 CER→超標換 seed 重抽→用盡取最佳) 貫穿單句/標記混排/有聲書 CLI (`--qc`)。多題材壓測 (武俠/科幻雙章、雙真人角色互演、全標記混排) 全程通過，並抓出一個真實整合 bug (行首 `[氣口:x]` 被章節解析器誤判為角色標記，改以 cast 感知的解析修復，同時保留打錯字防呆)，這也說明壓測應該用產品實際的使用形態來跑。成品端提供 opt-in 母帶化 (EBU loudnorm，實測 -21.6→-18.8 LUFS，達有聲書響度規格)；母帶化只作用於成品，與「訓練素材保持乾聲」的原則不衝突。

7. 討論 (Discussion)

- **不自訓底模**：消費級個人無法在資料量/算力上勝過大廠開源底模；理性策略為在強開源模型上微調自有角色，差異化交給內容與角色設定，而非重造輪子。
- **架構先行的回報**：因先有後端抽象，四引擎評比的成本僅是「換欄位 + 寫一個檔」，使「快速淘汰」可行；架構投資換來了實驗速度。後期的主力對調 (§4.4) 同樣零遷移成本。
- **測量驅動除錯**：多個關鍵決策 (放棄 cascade、常駐 worker、放棄 fp32、主力對調、甜蜜點) 皆由量測而非假設決定；且兩度修正自己的判斷 (「40 分/句」歸因外部競用；「28 句版壞掉」歸因取樣方差)。
- **能力上移 vs 能力移植**：配音控制 (停頓/氣口/音效/讀音/QC) 全數實作於引擎層而跨模型共用；模型內在能力無法移植，但常可模仿 (同音字替換之於讀音、ref 置換之於情緒)；缺少輸入通道，不等於做不到該功能。
- **客觀量測與耳測互補、缺一不可**：ASR-CER 抓吞字/錯字但會「腦補」修正誤讀、也聽不出雜音與節奏 (誤讀「昏→分」CER 為 0；底噪與拉長音只有耳朵抓到)；耳測則受單次印象與期望偏誤影響。本研究所有驗收均雙軌並行。

7.5 效度威脅與限制 (Threats to Validity)

- **樣本規模**：說話者 $n=2$ 、留出句每組 3 句、聽者 1 人；甜蜜點曲線為單說話者案例。結論應讀作「工程決策證據 + 可重現的實驗設計」，非族群統計推論；擴大樣本 (說話者/句子/聽者/多 seed 統計) 是升級為正式論文的首要工作。
- **裁判偏差**：單一 ASR 裁判 (Whisper large-v3)；其 LM 修正效應對「誤讀類」錯誤系統性低估。多裁判 (如 Paraformer) 交叉可緩解。
- **單次取樣方差**：§6.11 顯示同條件 CER 可在 3%–58% 間波動；本文凡單次數字均應視為點估計。生產路徑 (重抽 QC) 本身即為此方差的工程對策。
- **同音字替換的語義副作用**：替換後文本供引擎的 BERT 前端讀取，語義改變對韻律的影響未量化 (聽感未見異常)；且映射表基於「主讀音」，聲調連讀變化 (sandhi) 未建模。
- **授權分析非法律意見**：各引擎授權 (含 bilibili 門檻條款) 為工程側閱讀，商用發布前應法務複核。

- **硬體外推**：全部效能結論綁定 8GB Turing 單卡；更大顯存/新架構下（如 fp32 可行性）結論可能不同。

8. 結論與未來工作 (Conclusion & Future Work)

知音以可抽換後端 + 跨環境隔離，支撐了一場可重現（且可翻案）的多引擎評比；以引擎層標記系統把配音控制（情緒/停頓/氣口/音效/讀音）做成跨模型共用能力；以 ASR-QC 重抽產線把取樣式生成的可靠性從依賴單次結果，改為由重抽機制控制；並以雙說話者實錄管線與甜蜜點實驗，把「一般人也能配音」量化成可執行的分級台本（30 秒素材即達音色飽和）。8GB 消費級硬體上的全部結論（精度邊界、載入主導、競用敏感）均有量測依據。

未來工作：（1）演法預設庫（\$5.2）：以專業演繹片段庫讓使用者「挑演法」而非自己演；（2）雲端微調 IndexTTS-2：讓設計音在 IndexTTS-2 上也能兼顧三軸（目前沒有現成工具，需自行建立訓練流程）；（3）實時變聲/歌聲（RVC 系，與現有錄音資產共用）；（4）背景音混音層（疊加式音效）、影片生成接入（口型同步優先）；（5）音色目錄量產與 Studio 產品化收尾。

附錄：可重現性 (Reproducibility)

- 系統碼（私有儲存庫，可現場展示）：`zhiyin/`（後端抽象 `backends/base.py`、各引擎 backend + worker、標記系統 `core/markup.py` / `core/reading.py` / `core/breaths.py`、QC `qc/asr_qc.py` + `qc/retry.py` + `qc/_asr_worker.py`、產線 `pipeline/book.py`）。單元與整合測試在 `zhiyin/tests/`（13 個檔案）。
- **追蹤版流程腳本**：真人錄音管線六步 `docs/examples/recording/`（切分對齊/複檢/靜音壓縮/除噪/微調/氣口抽取，附零判斷執行手冊與失敗分支）；設計音鑄造 `docs/examples/mint/`。
- **度量**：pinyin-CER 與 SV 餘弦協定見 §4.1；甜蜜點實驗子集清單與評測腳本方法同 §6.11 所述，以 `synthesize()` API + 固定留出句可重現。
- **架構與檔案地圖**：`docs/ARCHITECTURE.md`、`docs/FILES.md`；錄音台本（分級語料庫）`docs/recording-list.md`；引擎現況矩陣 `docs/tts-status.md`。

參考文獻(References)

[1] C. Wang et al., “Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers (VALL-E),” arXiv:2301.02111, 2023. [2] E. Casanova et al., “YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for Everyone,” ICML 2022. [3] E. Casanova et al., “XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model,” arXiv:2406.04904, 2024. [4] RVC-Boss, “GPT-SoVITS: few-shot voice cloning TTS,” open-source project, github.com/RVC-Boss/GPT-SoVITS. [5] Z. Du et al., “CosyVoice 2: Scalable Streaming Speech Synthesis with Large Language Models,” arXiv:2412.10117, 2024. [6] IndexTeam (bilibili), “IndexTTS2: A Breakthrough in Emotionally Expressive and Duration-Controlled Auto-Regressive Zero-Shot Text-to-Speech,”

arXiv:2506.21619, 2025. [7] OpenMOSS, "MOSS-TTS family (含 VoiceGenerator : 文字描述→音色設計) · " open-source, 2025. [8] P. Anastassiou et al., "Seed-TTS: A Family of High-Quality Versatile Speech Generation Models," arXiv:2406.02430, 2024. (CER+SIM 客觀評測協定) [9] A. Radford et al., "Robust Speech Recognition via Large-Scale Weak Supervision (Whisper)," ICML 2023. [10] Y. Chen et al., "ERes2NetV2(3D-Speaker toolkit)說話者驗證模型 · " ModelScope/Alibaba, 2024. [11] M. Chen et al., "AdaSpeech: Adaptive Text to Speech for Custom Voice," ICLR 2021. (少樣本語者適應) [12] EBU R128, "Loudness normalisation and permitted maximum level of audio signals," EBU Recommendation.