LocalVQE: A Compact, GGML-Native Joint AEC, Noise Suppression and Dereverberation Model — Technical Report

Richard Palethorpe* Ettore Di Giacinto† Claude Opus 4.6 / 4.7‡

April 2026

Abstract

LocalVQE is a 1.3 M-parameter neural model for joint acoustic echo cancellation (AEC), noise suppression, and dereverberation of 16 kHz speech. It is a derivative of DeepVQE [1] with two substantive substitutions relative to the published architecture: an S4D diagonal state-space module in place of the original GRU bottleneck, and a real-valued reformulation of the complex convolving mask. The analysis/synthesis filterbank is realised as a frozen `Conv1d` initialised at the STFT-256 basis, keeping every intermediate tensor real-valued without changing the underlying representation. The shipped reference runs as a hand-written GGML graph in real time (˜9.6 × realtime on a single Zen 4 core). This is a technical report rather than a results paper — the validation metrics on the ICASSP 2022 AEC Challenge blind set are reproduced for context only. We document the exact mathematical definitions of each non-standard block, the streaming-state contract, the streaming-causal normalisation operator that runs at deployment, and the conventions used to load the released checkpoints in either PyTorch state-dict or GGUF format — enough to port the model to a different runtime (PyTorch from scratch, ONNX/CoreML/TFLite, custom CUDA, ARM NEON, or a streaming DSP framework). The inference engine and weights are open source (Apache-2.0).

1 Introduction

Voice-quality enhancement (VQE) on a typical voice call has three jobs that classical signal processing has historically split across separate modules: cancel the echo of the far-end participant played back through the local speakers and re-captured by the microphone; suppress ambient noise (fan, keyboard, traffic); and remove reverberation introduced by the room. DeepVQE [1] did the three in a single causal neural network, which avoided cascade-error between a separate AEC and NS, but its 7.5 M-parameter STFT-based architecture and fused complex-valued mask layer are inconvenient for deployment on commodity CPUs.

LocalVQE is a smaller, GGML-native re-derivation of that idea. The substitutions relative to the DeepVQE paper are:

- **Filterbank.** The STFT is realised as a frozen `Conv1d` filterbank initialised at the STFT-256 basis (Section 2.1); this keeps every intermediate tensor real-valued and lets us avoid `torch.complex` end-to-end without changing the underlying representation.
- **Bottleneck.** The GRU is replaced by an S4D [2] diagonal state-space module (Section 2.5), which is causal-by-construction, prunable per channel, and in our experiments matches GRU quality at fewer parameters and lower compute.

*`richard@localaisrl.com`
†`ettore@localaisrl.com`; guidance.
‡Anthropic.

- **Complex Convolving Mask (CCM).** We retain DeepVQE's 3×3 complex-valued mask but rewrite the arithmetic with a real-valued cube-root-of-unity decomposition (Section 2.6) so the deployment graph contains only real-valued ops.
- **AlignBlock.** Cross-attention soft-delay block as described by DeepVQE; we re-derive the streaming-causal form in Section 2.4.

A common thread through these substitutions is keeping the model prunable and quantisation-friendly: kernel area 16, real-valued arithmetic end-to-end, a per-channel-diagonal SSM whose state channels can be dropped independently, and no complex-tensor ops that would have to be unpacked at quantisation time. Channel widths are aligned to small multiples (the shipped widths come out of a pruning architecture search and end up at 32 and 40 rather than power-of-two values); a fresh training run with strictly power-of-two widths is a configuration change rather than a model change. The shipped 1.3 M-parameter checkpoint is approximately the smallest configuration we found to remain functional on the ICASSP 2022 AEC Challenge blind set; it is intended as the starting point for further compression rather than the end point. Future releases may include larger model sizes and calibrated quantised variants (Q4_K, Q8_0, BitNet-style ternary).

The inference engine and pretrained F32 GGUF weights are open source[1] under Apache-2.0. This report documents the parts that are not self-evident from reading the source.

2 Architecture

The model takes a microphone waveform $x_{\text{mic}} \in \mathbb{R}^N$ and a far-end reference $x_{\text{ref}} \in \mathbb{R}^N$ at 16 kHz and produces an enhanced waveform $\hat{y} \in \mathbb{R}^N$ in a single causal pass. Hyperparameters of the shipped configuration are listed in Table 1.

Sample rate	16 000 Hz
Frame size / hop	512 / 256 samples
Frequency bins (F)	256
Power-law c	0.3
Mic encoder channels	$[2, 32, 40, 40, 40, 40]$
Far-end encoder channels	$[2, 32, 40]$
Conv kernel (T,F)	$(4, 4)$
Encoder freq stride	2 (per stage)
AlignBlock $d_{\max}$ / hidden	32 frames (512 ms) / 32 ch
AlignBlock softmax temperature τ	0.1 (Section 2.4)
S4D bottleneck hidden H	162
CCM kernel	3×3 (causal)
Decoder layout	5 sub-pixel-conv stages, mirror of encoder
Total parameters	1.29 M ($\approx$ 1.03 M trainable; the 262 k STFT-init encoder weights are frozen)

Table 1: Shipped `localvqe-v1.1-1.3M` hyperparameters. The reference checkpoint can also be loaded with deeper-channel widths (e.g. 128) by editing the config; the GGUF stores per-stage channel counts as metadata so a re-implementer can read them out from the file rather than hard-coding.

2.1 STFT-256 analysis/synthesis filterbank

The encoder is a single `Conv1d`, kernel size K=512, stride S=256, K output channels, no bias, with weights frozen at the windowed STFT basis for the lowest F=256 FFT bins. Concretely,

[1] https://github.com/localai-org/LocalVQE (inference engine); https://huggingface.co/LocalAI-io/LocalVQE (weights).

with k indexing the frequency bin ($1 \le k \le F$) and n indexing within the kernel ($0 \le n < K$), the analysis window is the sqrt-Hann $w[n] = \sqrt{\frac{1}{2} - \frac{1}{2}\cos\big(2\pi(n{+}0.5)/K\big)}$ and the two filters per bin are

$$W_{2k-2,\,n} = w[n]\,\cos\Big(\frac{2\pi kn}{K}\Big), \qquad W_{2k-1,\,n} = -w[n]\,\sin\Big(\frac{2\pi kn}{K}\Big), \tag{1}$$

so each consecutive pair of output channels carries the (real, imag) parts of the same FFT bin. The DC and Nyquist bins are dropped to keep F a power of two. The input is zero-padded by $K/2{=}256$ samples on each side so the first frame is centred at $n{=}0$; the convolution itself is unpadded, so the output time length is $T = \lfloor N/S \rfloor + 1$. The 512 output channels are reshaped to $(F, 2)$ with $F{=}256$, matching the layout a complex STFT would produce, so any block downstream that expects $(B, F, T, 2)$ is unchanged.

The decoder is the matched synthesis filterbank: a frozen `Linear` projection from K channels back to K samples per frame whose rows are $(2/K)\,w[n]\cos(\cdot)$ and $-(2/K)\,w[n]\sin(\cdot)$ (the inverse-FFT scale times the synthesis sqrt-Hann window so that the sqrt-Hann analysis $\times$ sqrt-Hann synthesis gives a constant overlap-add). The encoder zero-padding is removed at the end.

Re-implementation notes.

- The STFT basis and synthesis weights are fixed; neither the encoder `Conv1d` nor the decoder `Linear` is a trained parameter, so a re-implementer can ship them as constants or recompute them on load. The GGUF export includes them as tensors for symmetry.
- Streaming: each output frame depends on the next $K{=}512$ future samples of the input (centre-padding), so a causal-streaming wrapper must defer output by $K/2{=}256$ samples. This is reflected in the 16 ms per-hop latency.

2.2 Power-law feature extraction

After encoding, both mic and ref pass through a power-law magnitude compressor (DeepVQE-style, $c{=}0.3$) before any learned layer. Letting $z = (z_r, z_i)$ be the two channels at a given (F, T) bin,

$$|z| = \sqrt{z_r^2 + z_i^2 + \epsilon}, \qquad \tilde{z} = \frac{z}{|z|^{1-c} + \epsilon}. \tag{2}$$

This preserves phase direction while compressing magnitude. The output is permuted to $(B, 2, T, F)$ for the encoder.

2.3 Encoder/decoder blocks

Each encoder stage is

$$y = \mathrm{Res}\big(\mathrm{ReLU6}(\mathrm{Conv2d}(\mathrm{Pad}(\mathrm{N}(x))))\big), \tag{3}$$

where N is the streaming-causal normalisation defined in Section 3, Conv2d has kernel $(4, 4)$, stride $(1, 2)$ (i.e. no stride in time, halving frequency every stage), and `Pad` is causal: $k_h{-}1{=}3$ **rows on the time axis, all on the past side, and** $\lfloor (k_w{-}1)/2 \rfloor$ **left and** $\lceil (k_w{-}1)/2 \rceil$ **right on the frequency axis.** The asymmetric padding on the *frequency* axis is purely a layout convenience; frequency is non-causal. Res is a single residual $y' = \mathrm{ReLU6}(\mathrm{Conv2d}(\mathrm{Pad}(\mathrm{N}(y)))) + y$ of the same shape. Decoders mirror this with sub-pixel-conv upsampling: the inner Conv2d outputs $2C$ channels and a `rearrange("b (r c) t f -> b c t (r f)", r=2)` doubles the frequency resolution while keeping channels at C; the last decoder block (`dec1`) drops the trailing ReLU6 since its output is the 27-channel CCM mask, not a feature map.

The pre-norm placement (norm before the conv, not after) keeps every input to a conv bounded across batches, which matters under quantisation-aware training because the activation quantiser's step size is then driven by the norm's learnable affine rather than by upstream variance. ReLU6 caps the post-conv path at 6, complementing the norm to bound both ends.

The skip connection at each decoder stage is $y_d \mathrel{+}= \mathrm{Conv2d}_{1\times1}(y_e)$ where y_e is the matching encoder feature, before the residual block. The decoder may produce one more frequency bin than the encoder skip (because the sub-pixel doubling is exact while the encoder used floor-strided downsampling), so a trim `[..., :encoder.shape[-1]]` is applied at every stage.

CCM identity initialisation. Without an explicit initialisation the cube-root basis defined in Section 2.6 sums to zero and the freshly initialised network produces silence. Set the bias of `dec1.deconv.conv` to zero except at two indices: 7 and 34 (current (t, f) kernel position, real-basis component, both even and odd freq sub-pixel slots). After the cube-root decomposition this gives a unit real-mask — i.e. the input passes through.

2.4 AlignBlock — cross-attention soft delay

The far-end speaker is delayed (relative to the playback path) by the device's audio buffer plus the room time-of-flight, typically 50–500 ms. AlignBlock estimates a soft per-frame delay distribution $\alpha(t, d)$ over $0 \le d < d_{\max}$ frames ($d_{\max}$=32, i.e. $32 \times 16\,\mathrm{ms} = 512\,\mathrm{ms}$ of look-back) and produces an aligned far-end feature

$$\tilde{x}_{\mathrm{ref}}(t) = \sum_{d=0}^{d_{\max}-1} \alpha(t, d)\, x_{\mathrm{ref}}(t - d), \tag{4}$$

which is concatenated channel-wise to the mic feature before `enc3`.

The distribution itself is computed from a cross-attention similarity between mic and ref encoder features at level 2:

$$Q = \mathrm{Conv2d}^{\mathrm{mic}}_{1\times1}(x_{\mathrm{mic}}), \quad K = \mathrm{Conv2d}^{\mathrm{ref}}_{1\times1}(x_{\mathrm{ref}}) \in \mathbb{R}^{B\times h\times T\times F}, \tag{5}$$

$$V_{btd} = \frac{1}{\sqrt{F}} \sum_{f} Q_{btf}\, K_{b,t-d,f}, \quad 0 \le d < d_{\max}, \tag{6}$$

$$\alpha = \mathrm{softmax}\big(\mathrm{Conv2d}_{(5,3)}(V)/\tau\big) \quad (\text{over } d). \tag{7}$$

The shifted K tensor is built with `nn.Unfold` after zero-padding $d_{\max}-1$ frames on the past side so the operation stays causal. The smoothing convolution is along (t, d) with a $(5, 3)$ causal kernel and reduces the head dimension to 1. The shipped checkpoint trains and runs at τ=0.1, which sharpens the soft-delay distribution toward a near-one-hot peak; matching the trained temperature at inference is worth several dB of FE-ST blind ERLE on real recordings, so the value must survive both the state-dict and the GGUF round-trip.

Temperature persistence and folding. τ is non-trainable but state-dependent. In the PyTorch model it is held as a `Module` buffer (`align.temperature`) so `state_dict` save/load and resumed training carry the trained value. Two equivalent inference paths apply it:

- **PyTorch.** Evaluate $\alpha = \mathrm{softmax}(V/\tau)$ directly, reading τ from the buffer.
- **GGUF/GGML.** The reference C++ runtime calls `ggml_soft_max`, which has no temperature argument. Before export, the trained temperature is folded into the preceding smoothing-conv weights and bias by dividing both by τ (`model.align.fold_temperature()` in the training repository); plain $\mathrm{softmax}\big((W/\tau) \star V + b/\tau\big)$ then produces the trained distribution. After folding, the buffer is reset to 1.0 so the call is idempotent. A GGUF exported without folding will silently run at τ=1 and lose several dB of FE-ST ERLE.

Re-implementation note. The weighted sum is over x_{ref}, the *full-channel* encoder feature, not the projected K. An easy slip is to reshape the unfolded tensor using h (the projection's hidden channels) where the in-channel count C is required, which silently produces a wrong-shape result on most non-trivial channel widths. Concretely: unfold x_{ref} along the time axis with `Tensor.unfold(2, dmax, 1)` for the weighted sum.

2.5 S4D bottleneck

A diagonal complex-valued state-space recurrence operating along the time axis at the deepest encoder stage. Let $u_t \in \mathbb{R}^{C\,F'}$ be the flattened deepest-feature vector at time t; with input projection $W_{\text{in}} \in \mathbb{R}^{H\times CF'}$, output projection $W_{\text{out}} \in \mathbb{R}^{CF'\times H}$, and $D \in \mathbb{R}^{CF'}$ the per-channel skip,

$$v_t = W_{\text{in}}\, u_t \in \mathbb{R}^H, \tag{8}$$

$$h_t = a \odot h_{t-1} + b \odot v_t \quad (\text{complex, } h_0{=}0,\ h_t \in \mathbb{C}^H), \tag{9}$$

$$y_t = \mathrm{Re}(c \odot h_t), \tag{10}$$

$$o_t = W_{\text{out}}\, y_t + D \odot u_t. \tag{11}$$

We parametrise a in polar form for unconditional stability:

$$a_n = r_n \cdot e^{j\theta_n}, \quad r_n = \exp(-\max(\mathrm{softplus}(\rho_n), 0.01)), \quad 0 < r_n < 1, \tag{12}$$

with ρ_n initialised uniformly on $[-6, 2]$ (so r spans $\sim$0.12 to $\sim$0.99, i.e. moderately fast to very slow decay) and θ_n initialised on the linspace $[0, \pi]$. $b, c \in \mathbb{C}^H$ are trained with small Gaussian init. D is initialised to ones (identity skip).

Numerical precision under AMP. The recurrence accumulates over T frames and is sensitive to FP16 underflow when r is small. The shipped reference promotes the entire SSM body to FP32 and only casts back to the AMP dtype after the output projection. The input/output projections themselves can run in BF16/FP16.

Re-implementation notes.

- The C++ reference precomputes $a_{\text{real}} = r\cos\theta$ and $a_{\text{imag}} = r\sin\theta$ at GGUF export time and ships those instead of ρ, θ, so the inference runtime does not need `softplus`/`cos`/`sin`. State-dict-loading paths in PyTorch keep the polar form.
- The deepest-stage layout is $(B, C, T, F')=(B, 40, T, 8)$ with $CF' = 320$, which is the input dimension to W_{in}. The $F'{=}8$ value comes from five floor-strided halvings of $F{=}256$ under the causal-padding rule in Section 2.3; verify against the actual `bottleneck.input_proj.weight` shape in the GGUF. The hidden dimension H is configurable; 162 is a hand-tuned compromise between expressive power and per-step cost.
- Each state channel is independently prunable since a is diagonal — this matters for further compression but is outside the scope of the shipped checkpoint.

2.6 Real-valued Complex Convolving Mask (CCM)

DeepVQE's CCM applies a learned 3×3 complex-valued convolutional mask to the encoded spectrum. We rewrite the same arithmetic with only real-valued tensors.

The decoder produces a 27-channel mask $m \in \mathbb{R}^{B\times 27\times T\times F}$. Reshape to $m \in \mathbb{R}^{B\times 3\times 9\times T\times F}$ (3 basis indices, 9 kernel taps). Project onto cube-root-of-unity:

$$H^{\mathrm{re}}_{btf}[k] = \sum_{r=0}^{2} v^{\mathrm{re}}_r \, m_{brktf}, \qquad H^{\mathrm{im}}_{btf}[k] = \sum_{r=0}^{2} v^{\mathrm{im}}_r \, m_{brktf}, \tag{13}$$

with $v^{\mathrm{re}} = (1, -\frac{1}{2}, -\frac{1}{2})$ and $v^{\mathrm{im}} = (0, \frac{\sqrt{3}}{2}, -\frac{\sqrt{3}}{2})$ (real and imaginary parts of $1, e^{j2\pi/3}, e^{-j2\pi/3}$). Reshape $H^{\mathrm{re/im}}$ to a 3×3 kernel $M^{\mathrm{re/im}} \in \mathbb{R}^{B\times 3\times 3\times T\times F}$.

The encoded mic spectrum $x \in \mathbb{R}^{B\times F\times T\times 2}$ is permuted to $(B, 2, T, F)$ and unfolded with a causal 3×3 kernel (pad: 2 above on time, 1 left and 1 right on frequency). Indexing the unfolded patch by (m, n) (kernel offsets: m=0 is $t-2$, m=1 is $t-1$, m=2 is current t; n=0 is $f-1$, n=1 is f, n=2 is $f+1$), the enhanced spectrum is

$$\hat{x}^{\mathrm{re}}_{btf} = \sum_{m,n} M^{\mathrm{re}}_{bmntf} \, x^{\mathrm{re}}_{b,t+m-2,f+n-1} - M^{\mathrm{im}}_{bmntf} \, x^{\mathrm{im}}_{b,t+m-2,f+n-1}, \tag{14}$$

$$\hat{x}^{\mathrm{im}}_{btf} = \sum_{m,n} M^{\mathrm{re}}_{bmntf} \, x^{\mathrm{im}}_{b,t+m-2,f+n-1} + M^{\mathrm{im}}_{bmntf} \, x^{\mathrm{re}}_{b,t+m-2,f+n-1}. \tag{15}$$

The kernel position for "current (t, f)" is index m=2, n=1, which after 3×3 flattening is index 7 (not 4 — a frequent re-implementation slip). See `ggml/localvqe_graph.cpp` in the inference repository for the fused GGML implementation.

3 Streaming inference

The reference runtime is a true streaming graph: a 256-sample audio hop is consumed per step, all internal state is held in named buffers, and there is no notion of a "length" input. The 16 ms algorithmic latency is the encoder centre-padding alone. Three classes of state must be held between hops:

1. **Encoder/decoder time-context.** Each $(4, 4)$ Conv2d has $k_h - 1 = 3$ time taps in its receptive field. A streaming port keeps a 3-frame ring buffer of the previous activations *per Conv2d* (the encoder/decoder blocks each have two such convs, the main and the residual), and prepends them to the current frame before the conv. The top of the ZeroPad2d (`kh-1, 0`) is replaced by the ring buffer at runtime.

2. **S4D state.** Two $\mathbb{R}^{H=162}$ vectors $h^{\mathrm{re}}, h^{\mathrm{im}}$, persisted across hops. The recurrence in Section 2.5 is run for exactly one time step per hop.

3. **AlignBlock time-context.** Three buffers: the last $d_{\max}-1$=31 frames of K (the projected ref), the last 31 frames of `x_ref` (used for the weighted sum), and a $(d_{\max}, k_h-1=4, h)$ history for the smoothing convolution's time axis (kernel $(5, 3)$, so 4 past frames are needed). The reference C++ graph names these `align_K_hist`, `align_ref_hist`, `align_smooth_hist`.

4. **CCM time-context.** The last 2 frames of the encoded mic spectrum (its 3×3 kernel covers $t-2, t-1, t$).

Streaming-causal normalisation (CausalGroupNorm). The training graph normalises every conv *input* (pre-norm) with a causal-friendly variant of GroupNorm: per-frame statistics over (C, F), no reduction across time, and per-channel learnable affine. For an input $x \in \mathbb{R}^{B\times C\times T\times F}$ the operation is

$$\mu_{bt} = \frac{1}{CF}\sum_{c,f} x_{bctf}, \qquad \sigma^2_{bt} = \frac{1}{CF}\sum_{c,f}(x_{bctf} - \mu_{bt})^2, \tag{16}$$

$$\mathrm{N}(x)_{bctf} = \gamma_c \frac{x_{bctf} - \mu_{bt}}{\sqrt{\sigma_{bt}^2 + \epsilon}} + \beta_c, \tag{17}$$

with $\epsilon{=}10^{-5}$ and $\gamma, \beta \in \mathbb{R}^C$. Because the reduction does not span T, frame t's statistics depend only on frame t; no running buffers, no warm-up. This is what makes the op streaming-causal — standard GroupNorm with $G{=}1$ would implicitly reduce across time and break the causal contract.

The op runs at deployment: it is *not* folded into the preceding conv because μ_{bt}, σ_{bt} are per-frame functions of the input, not constants. In the GGUF every block has `*.norm.weight` and `*.norm.bias` tensors carrying γ, β; an inference runtime computes the reduction itself.

4 Validation

Evaluated on the full 800-clip ICASSP 2022 AEC Challenge blind test set [3] — real recordings, not synthetic mixes.

Scenario	n	AECMOS echo ↑	AECMOS deg ↑	blind ERLE (dB)	DNSMOS OVRL ↑
doubletalk	115	4.70	2.35	8.4	2.85
doubletalk-with-movement	185	4.63	2.35	8.3	2.80
farend-singletalk	107	2.98	4.91	44.7	1.93
farend-singletalk-with-movement	193	3.40	4.95	45.0	1.91
nearend-singletalk	200	4.99	4.05	2.5	3.13

AECMOS [4] is reported as the per-scenario mean over all clips in the blind set. Per-clip blind ERLE is energy-pooled, $10\log_{10}(\sum_{f\in\mathcal{S}} |x_{\mathrm{mic},f}|^2 / \sum_{f\in\mathcal{S}} |\hat{y}_f|^2)$, where $\mathcal{S}$ is the set of frames satisfying both *far-end-active* ($\|\mathrm{ref}\|_{\mathrm{rms}} > p_{75}$) and *near-end-silent* ($\|\mathrm{mic}\|_{\mathrm{rms}} < 2\|\mathrm{ref}\|_{\mathrm{rms}}$, i.e. a 6 dB margin). The per-scenario number is the arithmetic mean of those per-clip values. Blind ERLE is only really meaningful on FE-ST, where the input is echo-only and the near-end-silent gate trivially holds. CPU realtime: 1.66 ms per 16 ms frame on a Zen 4 desktop with 24 threads (˜9.6× realtime); 1.11 ms per frame on Apple M4 with 4 P-cores.

5 Reproducibility

5.1 Checkpoint conventions

The PyTorch state-dict uses dotted module paths (e.g. `mic_enc1.conv.weight`, `bottleneck.A_log_rate`). The GGUF export keeps the same names with two adjustments:

- Every encoder and decoder block carries a CausalGroupNorm pair (`*.norm.weight`, `*.norm.bias`) which the inference runtime executes per frame; see Section 3 for the formula. No running mean/variance buffers exist (the reduction is over (C, F) per frame) so the GGUF carries only (γ, β).
- `bottleneck.A_log_rate` and `bottleneck.A_theta` are replaced by precomputed `bottleneck.a_real` and `bottleneck.a_imag` (Section 2.5).
- `align.temperature` is a buffer in the PyTorch state-dict carrying the trained τ. In the GGUF the AlignBlock smoothing-conv weight and bias are pre-divided by τ (Section 2.4); the buffer itself is absent from the GGUF and the temperature is implicit in the conv weights.

GGUF metadata keys (under the `localvqe.` namespace) document sample rate, K, S, F, $d_{\max}$, kernel size, bottleneck hidden, and the per-stage channel counts. A re-implementer should read these out rather than hard-code anything from this paper.

5.2 Numerical reference points

For verification of a re-implementation, the following bit-exact checks are useful (run them with the released F32 checkpoint at N=48000):

- STFT encoder output $\| \cdot \|_2$ on the FE-ST sample `ne_st_zeros.wav` (in the released test data).
- AlignBlock $\arg\max_d \alpha(t, d)$ on a clip with a known 320 ms latency: should track d=20 within ±1 frame.
- CCM identity-init: feed the encoded mic with the dec1 bias set as in Section 2.3; output should match the input within FP error.
- S4D recurrence on a single hop with all parameters at init: h_t should remain bounded by $\|b\|_\infty$ for all t.

5.3 What is and isn't required for a port

- **Required:** STFT analysis/synthesis (frozen `Conv1d` per Section 2.1), AlignBlock with the causal unfold and the cross-attention smoothing, S4D recurrence in FP32 (or higher), real-valued CCM with the cube-root basis, CausalGroupNorm pre-norm + ReLU6 around every encoder/decoder conv (Sections 3, 2.3).
- **Not required:** the FE block (a stateless arithmetic op), the residual connections (standard), and the encoder time-axis padding (replaced by a ring buffer in streaming).
- **Easy to get wrong:** the CCM kernel index for "current (t, f)" is 7, not 4 (Section 2.6); the AlignBlock weighted-sum reshape must use `x_ref.shape[1]`, not the projected K (Section 2.4); the AlignBlock softmax temperature τ must be applied — by dividing V before `softmax` (PyTorch path), or by folding $1/\tau$ into the preceding smoothing-conv weights and bias (GGUF/GGML path, whose `ggml_soft_max` has no temperature argument); the shipped value τ=0.1 is non-trivial and ignoring it loses several dB of FE-ST ERLE; the synthesis filterbank rows must include the $(2/K)$ inverse-FFT scale and the sqrt-Hann synthesis window so that COLA holds without a divisor; the S4D body must be FP32 under AMP.

6 Limitations

LocalVQE is 16 kHz only. The training data was DNSMOS-filtered, which can misclassify distressed speech (screaming, crying) as noise; the model can attenuate or distort such signals and must not be used on emergency-call pipelines. AECMOS DT degradation is 2.35 on the blind set — a deliberate trade favouring near-end preservation and noise suppression over more aggressive echo suppression during simultaneous double-talk. The released weights are F32 only; calibrated Q4_K / Q8_0 / ternary variants and other model sizes have not yet been released.

7 Acknowledgements

We thank the authors of DeepVQE [1] for the architecture LocalVQE derives from, and the authors of GGML [5] for the deployment runtime.

References

[1] E. Indenbom, N. C. Ristea, A. Saabas, T. Pärnamaa, J. Gužvin, R. Cutler. *DeepVQE: Real Time Deep Voice Quality Enhancement for Joint Acoustic Echo Cancellation, Noise Suppression and Dereverberation.* Interspeech 2023. https://arxiv.org/abs/2306.03177.

[2] A. Gu, K. Goel, A. Gupta, C. Re. *On the Parameterization and Initialization of Diagonal State Space Models (S4D)*. NeurIPS 2022. https://arxiv.org/abs/2206.11893.

[3] R. Cutler, et al. *ICASSP 2022 Acoustic Echo Cancellation Challenge*. ICASSP 2022. https://github.com/microsoft/AEC-Challenge.

[4] M. Purin, et al. *AECMOS: A Speech Quality Assessment Metric for Echo Impairment*. ICASSP 2022.

[5] G. Gerganov, et al. *GGML — Tensor library for machine learning*. GitHub. https://github.com/ggml-org/ggml.