RESEARCH · APPLE METAL · VIDEO MODELS

Video Models at Full Load: Unpacked Weights, Fused Epilogues and Fewer Small Ops on Apple Silicon

Q8_0 weights unpacked to F16 · scale and bias inside the matrix kernel · modulation, gates and casts in one op · VAE borders, concatenations and pixel shuffles without per-frame copies · every change bit-identical

Exe — Exe AI, developer studio

Measured on Apple M2 Max (38-core GPU, 64 GB unified memory)

−13 %
MiniMax H3 864×480×56, one sampling step: 36.68 s → 31.95 s; GPU ops per step 6,381 → 1,298

−10 %
LTX-2.3 1280×704×121, one sampling step: 57.29 s → 51.36 s; GPU ops per step 14,590 → 10,734

−23 %
HunyuanVideo 1.5 480p, video decode: 42.31 s → 32.52 s; VAE load 84.7 % → 96.5 %

17,940
ggml op tests on Metal, all passed, final build `f8aaf98b`

SETUP

Start point	ggml fork `de043708`, the end state of Report 4 · engine `7bdf150`
End point	ggml fork `f8aaf98b` · engine `07f8c72`
Question	where the video model families leave the GPU below 70 % load, and which of these losses can be removed without changing a single output bit
Target	every part at ≥ 70 % load · no model slower than before · video bit-identical for every change built in (Section 8)
Rules	every kernel in our ggml fork · stable-diffusion.cpp graph code changed in four files (Sections 4–6) · every figure from a log or raw file · one run per figure unless stated · model measurements on their own sheet (Section 11)

1 Where the time went

Profiler	`GGML_METAL_OP_PROFILE=2`: the labels of Report 4 plus the matrix shape (M, N, K, batch) of every product · `=3`: in addition every op below 0.5 % of the time, one command buffer per op
Load	of an op and of a part as in Report 4, Section 1 (`nightrun/part_load.py`)
Runs	2 sampling steps at the large size of each family; settings and files as in the night run (Section 11)

FAMILY, PROFILE AT THE START	STEPS	VAE	LARGEST ITEMS IN THE STEPS
Wan 2.1 T2V 1.3B, 832×480×33	78.5 % (7.52 s per call)	77.6 % (14.47 s)	attention 51 % of the time at 72.5 % · MUL_MAT f16 at 87–88 %
LingBot Dense 1.3B, 832×480×33	77.9 % (8.09 s)	77.6 % (same VAE)	attention 50 % at 71 %
Wan 2.2 TI2V 5B, 1280×704×33	84.2 % (8.88 s)	86.4 % (45.8 s)	MUL_MAT f16 65 % at 88 % · attention 25 % at 81 %
MiniMax H3, 864×480×56	76.8 % (36.7 s)	83.3 % (1.03 s)	MUL_MAT q8_0 70 % at 78–79 % · attention 20 % at 72 % · **6,381 ops per step**
HunyuanVideo 1.5, 480p image → video	71.5 % (18.9 s)	84.7 % (37.7 s)	VAE: **CONCAT 24.3 % of the decode, 2,808 calls** · steps: attention 55 % at 65 %
LTX-2.3, 1280×704×121, F16 connectors and VAEs	74.7 % (57.3 s)	88.7 % (18.7 s)	MUL_MAT q8_0 ≈ 60 % at 78 % · attention 29 % at 75 % · VAE: **CONCAT 10 % at 38 % · 14,590 ops per step**
Iris 3B (image, pixel space), 1024²	**49 %** (3.86 s)	no VAE	**MUL_MAT f16 80 % of the time at 45 %**
Ming-Image 0.1 Design, 1024²	85 % (5.57 s)	94 % (1.12 s)	MUL_MAT f16 at 88 % · attention 18 % at 72 %
MOSS-Image Base, 1024²	85 % (4.75 s)	93 % (1.04 s)	MUL_MAT at 88 % · attention 18 % at 73 %

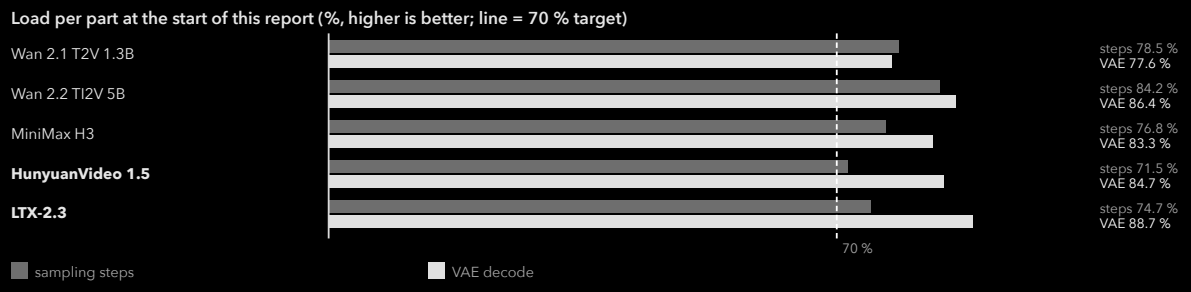


Figure 1. Load per part at the start, one profile each, `GGML_METAL_OP_PROFILE=2` (ops below 0.5 % count as 0, so the figures are lower bounds). LingBot as Wan 2.1 within 0.6 points. ggml `87d6c0aa`; LTX-2.3 on `c31be469`.

- Every video family starts above 70 % in both parts; the time of a video run sits in matrix products at 78–88 % and in attention at 65–81 %.
- The losses are therefore not one slow kernel but many small ones: VAE concatenations (Hunyuan, LTX), thousands of copies, scales and element-wise ops between the products (MiniMax, LTX), and one product with an unaligned inner dimension (Iris).
- Video VAE without `--conv-direct`: the 2D convolutions of the Wan VAE run through IM2COL (13 % of the decode at 11 % load). With `--conv-direct auto,vae=on`, a setting only: 77.6 → 88.9 %, 14.47 → 12.42 s (−14 %), video 42.96 dB (maximum 25/255, WebM encoding included). The night run does not set the switch for video models.

- Number formats (weight statistics of the loader): MiniMax q8_0/f16/f32, Wan 2.1, Wan 2.2 and LingBot f16/f32 – no BF16. The Q8_0 block scale is FP16 (`ggml_half d`, `ggml-common.h:253`). The night run of LTX-2.3 still loads the BF16 connectors and VAEs, although F16 files lie beside them.
- Logs: `measurements/profil-video/`, `profil-…-ming/`, `profil-…-moss/`, `profil-…-iris/`.

2 Matrix products with an unaligned inner dimension

- Cause (profile with matrix shapes, `iris_1024_shapes.log`): the Iris MLP with K = 6,826 ran at 2.1–2.2 TFLOP/s and took 43 % of a step; every other shape ran at 11.7–12.0. For K % 32 ≠ 0, ggml's `kernel_mul_mm` loads both matrices value by value with a bounds check in **every** K block, not only in the last one.
- Change (`87d6c0aa`, `kernels/mul_mm.metal`): the bounds check only in the last, partial K block; full blocks load single values without it (the rows are not aligned for vector loads).

MEASURE	BEFORE	AFTER
`test-backend-ops perf`, f16, m = 2,560, n = 4,396, k = 6,826	72.3 ms, 2.12 TFLOP/s	**14.6 ms, 10.51 TFLOP/s** (4.95×)
Iris 1024², 8 steps, application setting, release `82a37b3` against laboratory	8.89 s/it · 74.67 s	**5.01 s/it** (−44 %) · **43.60 s** (−42 %)
Load of the Iris steps	49 %	**79 %** (2.47 s per call)

- Image: 59.45 dB against the release.
- Kernel time before the change: the previous kernel file restored for one run of the same build.
- Tests: six new cases (f16, f32, bf16 × K = 206 and K = 6,826, batch 2) · MUL_MAT 2,458/2,458 · all ops 17,768/17,768.
- Not changed: `kernel_mul_mm_id` and the tensor-API variant (Apple M5 and later) carry the same check in every block; no model of this report uses them.
- In the model: Section 11.1.

3 Quantised weights unpacked to F16

- Observation: on the M2 Max, `kernel_mul_mm` reaches 11.7 TFLOP/s with F16 weights and 10.5 with Q8_0 weights at the same shape.
- Change (`20edc198`): a Q8_0 product with at least 1,024 rows unpacks its weights once to F16 into scratch memory behind the destination and runs the F16 kernel · `GGML_METAL_MM_Q8_UNPACK=0` switches it off.
- Price: the compute buffer grows by the F16 weights of the products alive at the same time (LTX-2.3: +459 MB).

WEIGHTS × ROWS, `TEST-BACKEND-OPS PERF`	Q8_0 KERNEL	UNPACK + F16 KERNEL
4096×4096 × 512	1.76 ms	1.78 ms (no gain below 1,024 rows)
4096×4096 × 1,024	3.41 ms	**3.28 ms**
4096×4096 × 14,080 (LTX-2.3)	44.9 ms	**40.9 ms (−9 %)**
28672×5376 × 7,095 (MiniMax H3)	207.4 ms	**189.6 ms (−9 %)**
5376×14336 × 7,095 (MiniMax H3)	104.4 ms	**95.2 ms (−9 %)**

MODEL, PROFILE OF ONE STEP	BEFORE	AFTER	LOAD OF THE STEPS	COMPUTE BUFFER	VIDEO
LTX-2.3, 1280×704×121	57.29 s	**53.86 s (−6 %)**	74.7 → 79.4 %	2,299 → 2,758 MB	bit-identical
MiniMax H3, 864×480×56	36.68 s	**35.31 s (−4 %)**	76.8 → 79.8 %	1,863 → 1,871 MB	bit-identical

3.1 Larger tiles: no gain

VARIANT, MINIMAX SHAPES (7,095 TOKENS), TFLOP/S	Q8_0 28672×5376	Q8_0 5376×14336	F16 28672×5376	F16 5376×14336
ggml `kernel_mul_mm`, tile 64×32, 4 simdgroups	**10.55**	**10.47**	**11.67**	**11.62**
tile 64×64, 4 simdgroups of 32×32	10.51	10.27	11.11	10.91
tile 64×64, 8 simdgroups of 32×16	9.71	9.18	10.39	9.86

SECOND ATTEMPT: OWN F16 KERNEL WITH GGML'S FRAGMENT LAYOUT, TFLOP/S	F16 28672×5376	F16 5376×14336
ggml `kernel_mul_mm` 64×32, 128 threads	11.12–11.21	10.58–10.70
own kernel 64×32, 128 threads	11.10	10.67
own kernel 128×64, 256 threads, 32×32 per simdgroup	11.02	10.51
own kernel 128×64, 256 threads, 64×16 / 16×64 per simdgroup	10.16–10.18	9.66–9.74
own kernel 256×128, 512 threads, 64×32 per simdgroup	8.94	8.65

- Hypothesis tested: Q8_0 loses because every weight tile is unpacked again for every 32 tokens. Refuted – twice the reuse is not faster.
- Reason: the product is bound by the arithmetic units, not by memory. 38 cores × 128 ALUs × 2 × 1.398 GHz ≈ 13.6 TFLOP/s; 11.1 TFLOP/s are ≈ 82 %. More reuse saves reads that do not limit; larger tiles cost registers and resident threadgroups.
- Own kernel: same fragment layout and summation order as ggml, hence the same bits; identical speed at the same tile confirms the layout. All F16 MUL_MAT tests passed for every variant (285/285); nothing built in.
- Neural Engine: reachable only through Core ML with fixed shapes, without Q8_0 and without bit-identical results; ggml computes on the GPU only.
- Raw files: `measurements/mm-kachel/kern_versuch.patch`, `perf_varianten.txt`.

4 Video VAEs: borders, concatenations and pixel shuffles

4.1 Replicated borders in one op (HunyuanVideo 1.5)

- Cause: not the kernel (227 GB/s) but the graph. `CausalConv3d` assembled the temporal border from one slice per frame and a chain of concatenations, the spatial border from one concatenation per edge – the input was copied once per frame and once per edge.
- Change: ggml `c31be469` – a PAD mode "replicate edge" (`ggml_pad_ext_replicate`) on CPU and Metal; other GPU backends reject it and leave it to the CPU · stable-diffusion.cpp `hunyuan_vae.hpp`: one PAD for the temporal and spatial border, one concatenation only with the frames carried over from the previous chunk.
- Kernel: Hunyuan shape {424, 240, 6, 256}: 3.54 ms, 331 GB/s · PAD tests 52/52 (new: mode 2, axes of size 1, border wider than the source).

4.2 A flat concatenation (LTX-2.3)

- Cause: the LTX VAE copies once per convolution (two border frames before the input); slow was ggml's CONCAT kernel, which derives the four-dimensional source index of every element.
- Change (`6841f2e6`): `heavy_metal_exe_concat` – for contiguous, 16-byte-aligned tensors a flat copy in 16-byte units, one division per unit · everything else on ggml's kernel · `GGML_METAL_CONCAT_KERNEL=ggml` switches it off · CONCAT tests 218/218 with both kernels.

LTX-2.3	GGML KERNEL	HEAVY_METAL_EXE_CONCAT
`test-backend-ops perf`, 160×88×(2+8)×512 f32	4.02 ms, 134 GB/s	**1.60 ms, 335 GB/s**
VAE GPU time	18.74 s	**17.86 s**

LTX-2.3	GGML KERNEL	HEAVY_METAL_EXE_CONCAT
Video decode	21.32 s	**20.17 s (−5 %)**
Load of the VAE	88.7 %	**94.1 %**

4.3 A narrow copy for the pixel shuffle (HunyuanVideo 1.5)

- Cause: the pixel shuffle copies tensors whose first axis is the factor 2 and comes from the slowest axis of the source (CONT 1203, 3.9 % of the decode at 12 % load). The 32×32 tile kernel of Report 2 filled 2 of its 32 rows.
- Change (`5cf5f9bb`): `kernel_cpy_narrow` for first axes up to 8 – each thread copies the values of one element · CPY 223/223, CONT 78/78, DUP 7/7.

TEST-BACKEND-OPS PERF, 424×240×F×(512/F) F32	32×32 TILE	NARROW COPY
f = 2, permutation {1,2,0,3}	9.14 ms, 43 GB/s	**1.12 ms, 348 GB/s**
f = 4	4.68 ms	**1.11 ms**
f = 8	2.49 ms	**1.11 ms**
f = 16 (stays on the tile)	**1.37 ms**	1.81 ms
f = 2, permutation {2,1,0,3}	9.14 ms	**1.80 ms**

HUNYUANVIDEO 1.5, 480P IMAGE → VIDEO	START (87D6C0AA)	+ PAD (C31BE469)	+ CONCAT + NARROW COPY (5CF5F9BB)
CONCAT in the decoder	2,808 × · 9.17 s	288 × · 1.14 s	–
VAE GPU time	37.7 s	29.1 s	**27.6 s**
Video decode	42.31 s	34.49 s	**32.52 s (−23 %)**
Image encode (start frame)	2.81 s	2.58 s	–
Load of the VAE	84.7 %	91.8 %	**96.5 %**
Video against the build before	–	bit-identical	bit-identical

- Logs: `measurements/profil-video-pad/`, `profil-video-concat/`, `profil-video-cpy/`.

5 MiniMax H3: fewer and fused small ops

Profile	864×480×56 · the second, undisturbed step of two · level 2 for the first two rows of the table below, level 3 (every op) from the views onwards
Starting picture	after the unpacking of Section 3: 6,381 ops per step, of them 2,874 plain copies (CONT 0123, 1.0 s) · about 0.8 s per step in small ops around the attention

CHANGE	WHERE	MECHANISM
Views instead of copies	stable-diffusion.cpp `c4b9a8e`	`ggml_ext_slice` and `ggml_ext_chunk` copy by default; the token segments, modulation rows, MLP halves and q/k/v are now views that the next op reads in place
Scale inside the matrix product	ggml `00a53065`	the MLP divides its input by 128 before each product and multiplies the result by 128 afterwards to stay inside F16; the mat-mat kernel applies both factors while loading and storing, with the formula of the SCALE kernel
Partial rotation in one op	ggml `9749c54d`	MiniMax rotates only the leading part of each head; q and k were sliced (two copies), rotated, reordered and concatenated · `ROPE_APPLY` now rotates the first n_{rot} values and copies the rest

CHANGE	WHERE	MECHANISM
k and v in one pass to F16	ggml `1effb59b`, stable-diffusion.cpp `ggml_extend.cpp`	k: scale, then cast (two passes); v: reorder, scale, cast (three) · one pass of the unary kernel reads the source, scales and writes F16 with the same rounding; v is reshaped only after the cast, so the three ops stand next to each other
Modulation of all segments in one op	ggml `f610f3e8`, stable-diffusion.cpp `minimax_h3.hpp`	each segment (text runs, conditions, audio, video) has its own modulation row; every block sliced the segments, multiplied and added per segment and concatenated them again, four times per block · new op `ADD_MUL_ID`: (a + b·s[id]) + t[id] for all tokens at once, the row of each token from an I32 table

MINIMAX H3 864×480×56, ONE STEP	STEP	OPS	COMPUTE BUFFER	VIDEO
start (`87d6c0aa`)	36.68 s	6,381	1,863 MB	–
+ Q8_0 unpacked (`20edc198`)	35.31 s	6,381	1,871 MB	bit-identical
+ segments as views	34.4 s	3,973	–	bit-identical
+ q/k/v and MLP halves as views (`c4b9a8e`)	33.90 s	3,713	1,731 MB	bit-identical
+ scale inside the product (`00a53065`)	33.58 s	3,505	–	bit-identical
+ partial rotation (`9749c54d`)	33.36 s	3,105	–	bit-identical
+ k and v in one pass (`1effb59b`)	33.09 s	2,949	–	bit-identical
+ ADD_MUL_ID (`f610f3e8`)	**31.95 s**	**1,298**	–	bit-identical

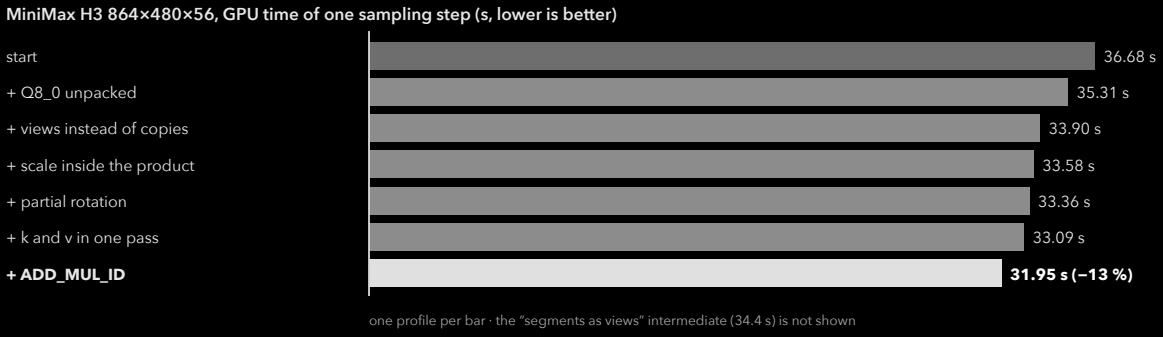


Figure 2. Cumulative, each bar on the build of its row in the table; one profile each. The drop at ADD_MUL_ID (1.14 s) is 0.43 s of the op and ≈ 0.7 s of a quieter GPU: the untouched product M21504 ran 7,165 → 7,100 ms and the attention 7,590 → 7,194 ms in the same pair.

- Small ops of the last step (all ops except matrix products and attention): 1,477 → 1,048 ms; ADD + MUL + CONCAT 345 + 345 + 178 ms → MUL 170 ms (MLP) and ADD_MUL_ID 203 ms.
- ADD_MUL_ID alone (`test-backend-ops perf`, 5,376 × 7,095): 0.81 ms for the modulation, 1.24 ms for the gate, ≈ 350 GB/s.
- Backends: CPU and Metal; CUDA and Vulkan report the op as unsupported, and stable-diffusion.cpp then builds the previous graph. A token outside every segment also leads to the previous graph.
- Tests: ADD_MUL_ID 12/12 at zero tolerance (Section 8).
- Logs: `measurements/profil-video-q8/`, `profil-video-mm5/`, `profil-video-mms2/`, `profil-video-rope/`, `profil-video-kv/`, `profil-video-amid/`.

6 LTX-2.3: bias, modulation, gates and casts

Profile	`GGML_METAL_OP_PROFILE=3` · 1280×704×121 · F16 connectors and VAEs · the second step of two
Starting picture	14,590 ops per step; ADD alone 4,018 calls and 2.06 s

CHANGE	WHERE	MECHANISM
Bias inside the matrix product	ggml `f1f655f2`	every linear layer added its bias with a separate ADD over the whole product · the mat-mat kernel adds one F32 row while storing (fusion `MUL_MAT_SCALE`, now also with a trailing bias ADD, alone or after the scales); each thread adds the bias of the two columns it holds in every 8×8 fragment; partial tiles add it while copying
Gate without REPEAT	stable-diffusion.cpp `b8403ab`	the gated attention repeated the gate of each head over its values before the MUL; the MUL broadcasts it itself
Modulation and gated residuals in one op	ggml `f8aaf98b`, stable-diffusion.cpp `07f8c72`	`ADD_MUL_ID` without ids: s and t are broadcast over a as MUL and ADD broadcast their second operand · LTX modulates (x + x·scale + shift) at 16 places and adds its gated branches (x + y·gate) at 9 with it
CONT + CPY to F16 in one copy	ggml `f8aaf98b`	attention makes keys and values contiguous (CONT), often followed by a RESHAPE, and casts them to F16 (CPY) · the fusion `CONT_CPY` copies the F32 source of the CONT directly into the F16 destination; the F32 copy is neither written nor read again

Q8_0 4096×14080×4096, `test-backend-ops perf`, FASTEST OF FIVE ALTERNATING RUNS	TIME
product only	40.03 ms
product with the bias in the kernel	**40.35 ms**
product, then a separate ADD	41.27 ms

- Placement matters: with the addition inside the branch that stores straight to memory, the whole kernel ran 3 % slower, even with a constant instead of the bias; next to the scale epilogue, before that branch, it is nearly free.

LTX-2.3 1280×704×121, ONE STEP	START OF THE SERIES	+ BIAS, GATE	+ ADD_MUL_ID	+ CONT_CPY
ggml	before `f1f655f2`	`f1f655f2`	`f8aaf98b` without CONT_CPY	`f8aaf98b`
Ops per step	14,590	12,538	10,998	**10,734**
ADD	4,018 × · 2,058 ms	2,270 × · 1,191 ms	730 × · 8 ms	730 × · 8 ms
MUL	1,266 × · 813 ms	1,266 × · 768 ms	304 × · 198 ms	304 × · 198 ms
REPEAT (gate)	304 × · 316 ms	–	–	–
ADD_MUL_ID	–	–	962 × · 688 ms	962 × · 690 ms
CONT 0213 + CPY to F16	–	–	721 ms	**621 ms**
All ops except matrix products and attention	5,130 ms	3,837 ms	2,742 ms	**2,664 ms**
Matrix products (unchanged code except the bias)	32,014 ms	32,752 ms	32,174 ms	32,351 ms
Attention (unchanged)	16,280 ms	17,206 ms	16,070 ms	16,340 ms
Video against the build before	–	bit-identical	bit-identical	bit-identical

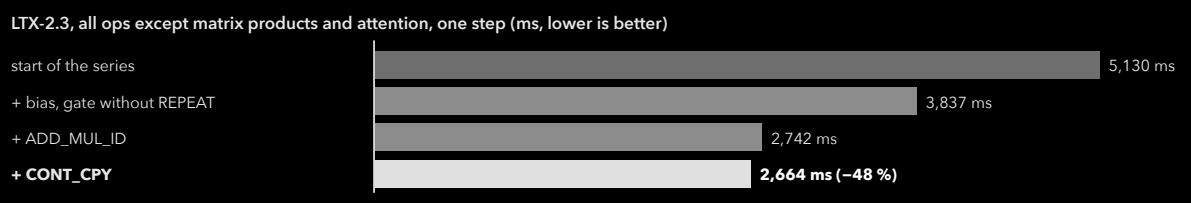


Figure 3. One profile each, counted the same way for all four (GPU time of the step minus MUL_MAT and FLASH_ATTN_EXT). The untouched matrix products and attention move by up to 1.14 s between these runs: other processes shared the GPU (Section 9).

- CONT_CPY: 264 of 608 casts fused. The micro-benchmark (128×14,080×32) showed no gain; the gain appears only in the model run – cause not established.
- Remaining CPY to F16: 344 calls, 162 ms per step.
- Tests: MUL_MAT_BIAS 18/18 · MUL_MAT 2,462/2,462 · MUL_MAT_SCALE 50/50 · ADD 84/84 · ADD_MUL_ID 28/28 (new: broadcast per batch and per token through a permuted view) · CONT_CPY 4/4 · CPY 223/223 · CONT 78/78.
- Logs: `measurements/profil-video-ltx3/`, `profil-video-ltxg/`, `profil-video-ltxm/`, `profil-video-ltxc/`, `profil-video-ltxb2/bias_perf_rounds.txt`.

7 Attention

- Share after this report: LTX-2.3 16.3 s of a 51.4 s step (32 %), MiniMax H3 7.6 s of 32.8 s (23 %), at 74–77 % load – the largest single item after the matrix products.
- A larger block for long sequences (Metal FlashAttention, head size 128, `test-backend-ops perf`): the block of Report 4, 32/128/32 with 128 threads, against 64/128/32, 64/64/32, 64/128/16 (256 threads each) and 16/128/32 (64 threads).

BLOCK (QUERIES / KEYS / HEAD COLUMNS)	THREADS	32 HEADS × 14,080 TOKENS
32/128/32 (Report 4, kept)	128	**319.4 ms**
16/128/32	64	321.5 ms
64/128/32	256	335.6 ms
64/128/16	256	345.1 ms
64/64/32	256	350.3 ms

- Other shapes (2,304 tokens; Krea 2 shape, 4,096 tokens): every alternative slower as well. The block stays.
- Rejected by measurement: folding the ×128 after the attention into the output projection. The memory planner of stable-diffusion.cpp (`ggml_gallocr`) places the output of the projection in the freed buffer of the attention output; the fusion refuses overlapping input and output, and the in-place alternative is a view of foreign memory, which `ggml_can_fuse_subgraph_ext` refuses. Possible gain by the profile: ≈ 52 ms per step. Withdrawn.

8 Exactness

Criterion	every change built in gives a video identical to the build before it: `nightrun/psnr.py` on all frames, maximum absolute difference 0
Op tests	new fusions and ops are tested at zero tolerance (`max_nmse_err` 0) against the unfused graph or the CPU
Fused multiply-add	Metal compiles with fast math and contracts a product followed by a sum into one FMA, which rounds once instead of twice · the kernels that must reproduce MUL followed by ADD switch it off (`#pragma METAL fp contract(off)`); without the pragma the zero-tolerance test of ADD_MUL_ID fails, with it 12/12
Same order	the fused bias, the fused scales and the own F16 kernel keep ggml's summation order and fragment layout

REJECTED FOR EXACTNESS	GAIN	DEVIATION
`ggml_swiglu` instead of SILU and MUL (MiniMax H3), also as a float4 kernel	0.25 s per step	video 43.5 dB, maximum 22/255 – not from `exp`, but from how the compiler contracts division and multiplication inside one kernel

- Not built, by estimate: RMS norm and modulation in one op (MiniMax H3) – the norm writes its result and the modulation reads it again; ≈ 0.8 ms per place, at most ≈ 0.08 s per step (0.25 %), and only half of that without restructuring the graph.

9 Measurement conditions

DISTURBANCE	SEEN IN	TREATMENT
Other processes on the GPU (a local language-model server, the Exe AI Metal application, further work sessions)	`gpu before the run`: mean 4–9 %, maximum up to 99 % · untouched ops 3–6 % slower in the same run	untouched ops (attention, unchanged products) serve as the control of every profile pair; a change counts only beyond their drift
Profile level 3: one command buffer per op	the GPU time of a step includes no overlap between ops	used for op counts and shares; absolute step times only between profiles of the same level
First sampling step	loads weights and compiles pipelines	the second step of two is reported
Spread of micro-benchmarks	up to 3 % between consecutive runs	fastest of five runs, old and new build alternating

- Example of a false alarm: after the broadcast form of ADD_MUL_ID, the op took 223 ms in a MiniMax profile instead of 205 ms, but the untouched products of the same run were 5 % slower. Old and new kernel alternating, fastest of five: 797.65 against 800.84 µs and 1,213.86 against 1,216.26 µs – equal.
- Rules adopted: the rules of Report 4, Section 9, plus the control ops above.

10 Open points and next steps

OPEN POINT	NEXT STEP	SECTION
Whole runs of the video models	Original, Release and Laboratory in the night-run format on an idle GPU; this report gives profiles of single steps only	11
Attention	23–32 % of a video step at 74–77 % load; no faster block found	7
Matrix products	compute-bound at ≈ 82 % of the arithmetic peak; larger tiles do not help	3
HunyuanVideo 1.5 steps	73.7 % load; attention 55 % of the time at 65 %	1, 11
Wan 2.1, Wan 2.2, LingBot	not profiled again on the end state	1
LTX-2.3 in the night run	still loads the BF16 connectors and VAEs; F16 files exist	1
Application defaults	`--conv-direct auto`, `vae=on` also for video VAEs (Wan: −14 % decode)	1
ADD_MUL_ID on CUDA and Vulkan	reported as unsupported; the previous graph runs there	5
CONT_CPY	the gain in the model run is not reproduced by the micro-benchmark	6
Repeats	most profile figures are one run, micro-benchmarks five	all

MiniMax H3 licence

- The MiniMax H3 licence, in the version consulted for Report 3, excludes the European Union, the United Kingdom, the Republic of Korea and the United States, and forbids using or displaying the model's outputs outside its territory; a licence can be requested from MiniMax.
- This report therefore gives timings only and shows no output of the model.

Licences and acknowledgements

COMPONENT	ORIGIN
stable-diffusion.cpp, ggml, ccv's 3D convolution kernel	Report 1
Metal FlashAttention kernels (`kernels/mfa`), generated by the MFA generator in ccv	ccv (BSD-3-Clause, © 2010 Liu Liu) · Metal FlashAttention (MIT, © 2024 Philip Turner); both texts in `kernels/mfa/LICENSE`

COMPONENT	ORIGIN
K-tail bounds check, Q8_0 unpacking, replicate PAD, `heavy_metal_exe_concat`, `kernel_cpy_narrow`, partial `ROPE_APPLY`, the scale, bias, cast and copy fusions, `ADD_MUL_ID`, the graph changes in stable-diffusion.cpp, measurements, analysis	Exe AI, own work; ggml's licence (MIT) applies to the ggml files, stable-diffusion.cpp's licence (MIT) to its files
GPL-licensed code	none used

References

1. Exe AI, *Fast 3D Convolutions for Local Video Generation on Apple Silicon*, Technical Report 1.
2. Exe AI, *Memory and Speed Together: Optimising Local Video Generation on Apple Silicon*, Technical Report 2.
3. Exe AI, *Beyond the Convolution: Attention, Small Ops and Image Models on Apple Silicon*, Technical Report 3.
4. Exe AI, *Image Models at Full Load: Normalisation, Rotary Embeddings and Attention Blocks on Apple Silicon*, Technical Report 4.
5. P. Turner, Metal FlashAttention; L. Liu, ccv (`lib/nnc/mfa`, commit `3564f9f`), github.com/liuliu/ccv.
6. ggml, github.com/ggml-org/ggml; stable-diffusion.cpp, github.com/leejet/stable-diffusion.cpp.
7. Project measurements: `measurements/profil-video*/`, `measurements/mm-kachel/`, `measurements/ab-…-iris/`, `measurements/tests-f8aaf98b/`; step log `paper/paper4-schritte.md`; engine notes `engine/ACCELERATOR.md`; overview `UEBERSICHT.html`.

11 Model measurements

Machine	Apple M2 Max, 38-core GPU, 64 GB · macOS 26.5.2 (25F84) · Xcode 26.6 (17F113) · models on the external USB SSD
Command	`sd-cli … --steps N --sampling-method euler --flash-attn on -s 42 -W w -H h -p "$(cat measurements/prompt.txt)"` · profiles: `nightrun/ab_check.py`, cases `vprof_<family>` and `vproff_<family>` · night-run format: `nightrun/run_matrix.py`
Builds	Original = stable-diffusion.cpp `master-1000-95ff762`, unmodified · Release = `master-1000-82a37b3` (ggml `34e0dc1c`) · Laboratory = `engine/build-edge`, ggml as named per row

MODEL	FILES
MiniMax H3	`MiniMaxH3_FL2VA_Q_8_F16.gguf` · `Qwen3VL32B_MiniMaxH3_Q_8_F16.gguf` · VAE `MiniMaxH3_VideoVAE_F16` · audio VAE `MiniMaxH3_AudioVAE_F16`
LTX-2.3	`LTX2.3_22BDistilled1.1_Q_8_F16.gguf` · Gemma 3 12B Q8_0 · profiles with F16 connectors and VAEs, night run with BF16
HunyuanVideo 1.5	`HunyuanVideo1.5_480pI2VStepDistilled_F16.safetensors` · VAE `HunyuanVideo1.5_VAE_F16`
Wan 2.1 / Wan 2.2 / LingBot	`Wan2.1_T2V1.3B_F16` / `Wan2.2_TI2V5B_F16` / `LingBotVideoDense1.3B_F16` · UMT5-XXL F16 (Wan), Qwen3-VL 4B F16 (LingBot)
Ming-Image / MOSS-Image / Iris	`MingImage0.1_Design_F16` + Ling-mini 2.0 F16 / `MossImage_Base_F16` + Qwen3.5 9B Q4_K_M / `Iris3B_F16` + Qwen3-VL 4B F16

11.1 Night-run format: Original, Release, Laboratory

WHOLE RUN	ORIGINAL	RELEASE	LABORATORY	LAB VS. RELEASE	DECODE O / R / L	IMAGE O-R / O-L
Ming-Image 512², 12 steps	167.2 s	123.2 s	**121.6 s**	−1 %	23.7 / 0.81 / 0.75 s	55.3 / 49.1 dB
Ming-Image 1024², 12 steps	341.7 s	175.0 s	**166.3 s**	−5 %	93.7 / 1.92 / 1.64 s	51.9 / 43.3 dB
MOSS-Image 512², 50 steps, CFG 3	191.6 s	116.0 s	**113.2 s**	−2 %	13.6 / 1.00 / 0.53 s	52.8 / 56.9 dB
MOSS-Image 1024², 50 steps, CFG 3	741.9 s	513.3 s	**492.8 s**	−4 %	53.5 / 3.61 / 1.37 s	57.2 / 44.3 dB
Iris 512², 100 steps, CFG 3	427.2 s	265.2 s	**150.8 s**	**−43 %**	no VAE	44.2 / 44.1 dB
Iris 1024², 100 steps, CFG 3	1,360.1 s	893.7 s	**505.4 s**	**−43 %**	no VAE	59.6 / 58.4 dB

- Laboratory build: ggml `87d6c0aa` (Section 2); one run per cell, each alone on the GPU.
- Ming-Image: its text encoder (36 GB F16) reads ≈ 70 s from disk in every run — most of the time at 512².
- Images at 43–49 dB: inspected, the same picture; the differences lie in snowflakes and edges. Iris: original against release only 44 dB as well — a pixel-space model over 100 steps.
- Logs: `measurements/nightrun-…/{ming,moss,iris}_*_{orig,ours,lab}.log`.

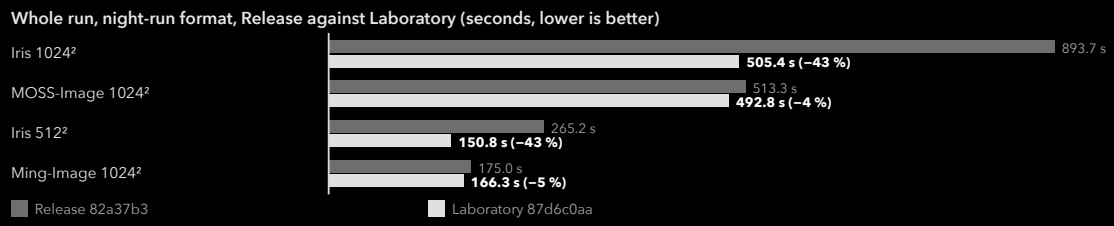


Figure 4. Whole runs from the night-run logs, one run each, at the larger size of each family (Iris also at 512²).

11.2 Video models: one sampling step, start against end

MODEL, SIZE	MEASURE	START	END	GGML START → END	VIDEO
MiniMax H3, 864×480×56	step (GPU) · ops	36.68 s · 6,381	**31.95 s · 1,298**	`87d6c0aa` → `f610f3e8`	bit-identical at every change
MiniMax H3, re-run on the end state	step (GPU) · ops	–	33.82 s · 1,294	`f8aaf98b`	bit-identical; GPU not idle (Section 9)
LTX-2.3, 1280×704×121	step (GPU) · ops	57.29 s · 14,590	**50.33–51.36 s · 10,734**	`c31be469` → `f8aaf98b`	bit-identical at every change
LTX-2.3, 1280×704×121	VAE GPU time	18.74 s	**17.48 s**	`c31be469` → `f8aaf98b`	bit-identical
HunyuanVideo 1.5, 480p image → video	video decode · VAE GPU time	42.31 s · 37.7 s	**32.52 s · 27.6 s**	`87d6c0aa` → `5cf5f9bb`	bit-identical at every change
Wan 2.1, Wan 2.2, LingBot	–		not measured again	`87d6c0aa`	–

- Step times are the GPU time of one profiled step (second of two), the start at profile level 2, the end at level 3; LTX-2.3 end: both steps of the last profile. Whole-run comparisons in the night-run format are open (Section 10).
- Images: `nightrun/psnr.py` against the build before each change; maximum absolute difference 0 on every frame.
- Logs: `measurements/profil-video/` (start), `profil-video-amid/`, `profil-video-amidc/`, `profil-video-ltxc/`, `profil-video-cpy/` (end).

11.3 Load per part against the 70 % target

FAMILY	STEPS BEFORE	STEPS AFTER	VAE BEFORE	VAE AFTER	GGML AFTER
MiniMax H3	76.8 %	**83.3 %** (31.95 s)	83.3 %	84.4 % (1.02 s)	`f610f3e8`
LTX-2.3	74.7 %	**82.7 %** (51.36 s)	88.7 %	**95.8 %** (17.48 s)	`f8aaf98b`
HunyuanVideo 1.5	71.5 %	73.7 % (18.3 s)	84.7 %	**96.5 %** (27.6 s)	`5cf5f9bb`
Iris 3B	49 %	**79 %** (2.47 s)	–	–	`87d6c0aa`
Wan 2.1 / LingBot / Wan 2.2	78.5 / 77.9 / 84.2 %	not measured again	77.6 / 77.6 / 86.4 %	not measured again	–

- Counting: “before” at profile level 2 (ops below 0.5 % count as 0), “after” at level 3 (every op); the coverage rises from 97–99 % to 100 %, so part of the rise in the steps is counting.
- Target met: every measured part at ≥ 70 %; the lowest is the HunyuanVideo 1.5 steps at 73.7 %, held down by the attention (Section 10).