A TEARDOWN OF APPLE'S ON-DEVICE FOUNDATION MODELS: ARCHITECTURE, QUANTIZATION, AND CONTAINER FORMATS OF THE `FM_GENERATIVEMODELS` ASSET SET

REVERSE-ENGINEERING REPORT

Date: Asset build `13M204130`.

Key words and phrases. Apple Intelligence, on-device language models, Mixture-of-Experts, weight quantization, Apple Neural Engine, model container formats, reverse engineering.

Abstract. We present a complete static teardown *and full weight recovery* of Apple's on-device generative-model asset bundle `com.apple.MobileAsset.UAF.FM.GenerativeModels`, as shipped to a macOS 26/27 machine. The bundle comprises 118 signed assets (≈10 GB of disk images) spanning six model families. We document two distinct language backbones—a dense `afmplus-v11.0-nano` (∼3 B, 2048-wide, 56 layers) and a sparse Mixture-of-Experts `afmplus-v11.0-ifp` (1536-wide, 44 layers, routed experts)—together with ∼90 task adapters, safety and guardrail models, a multimodal image encoder, speculative-decoding draft models, and Private-Cloud-Compute server variants. We fully characterize the on-disk container stack: Cryptex disk images, the `BBBB` rasterized-weight container, the `odix` on-device executable, the Apple Neural Engine `.hwx` binary, and the MPSGraph package. We reverse-engineer the `odix` internal reference scheme (self-relative signed offsets into an interned symbol pool) and parse the `MLIR22.0.0` MPSGraph bytecode to recover each constant's layer/role from its location hierarchy. We identify the weight codec as *LUT-palettization* (a 4-bit index into a 16-entry codebook, scaled per 1024-element block) followed by an *Apple-Neural-Engine* 8×128 *tile de-swizzle*, and—the central positive result—we **reverse the ANE spatial permutation and recover the full weights**. Under a scale-decontaminated low-rank structure test (§6), genuine weights score $R \approx 9$ and noise $R \approx 1$; our decode scores $R = 13$–69, confirming true weight structure. We reconcile the parameter budget exactly against the physical file, resolve the norms (compile-time folded), the router (baked to a dense expert set), and the missing constant table (its runtime equivalent is the shipped `metadata.bin` swizzler), and export the entire model to a single `state_dict`: 9.862 **B parameters,** 100.00% **of the shipped weight file, zero non-finite tensors**. Reconstructing the forward pass in PyTorch, the attention stack runs *stably* on the recovered tensors—independently validating the codec, de-swizzle, and architecture—and we **overturn the earlier verdict that the expert-selection wiring is fatal**: the sparse feed-forward is an *ungated* structured-pruned SwiGLU (a permutation-invariant sum over resident experts), so the instruction-following router→expert map—long believed the unrecoverable blocker—is *mathematically irrelevant* to the output. We further resolve the per-layer RMSNorm γ (compile-time *folded* into the adjacent linear weights; only the per-head QK-norm survives explicitly), validate the dense-weight decode against ground truth generated by Apple's own ANE toolchain, and prove the intermediate activations are *ANE-internal* (absent even from an 8 GB IOSurface-targeted process core). A full 44-layer forward ablation against Apple's emitted token confirms two of these results independently—the KV geometry is NKV=4, and the ungated full-superset sum is not merely sufficient but *optimal* (every form of expert selection or attenuation scores strictly worse)—while locating the residual gap: the assembled forward carries real signal (3.6× better than chance) but is dominated by a mis-aligned feed-forward direction whose per-neuron/offset alignment has no accessible ground truth, since the activations that would validate it are ANE-internal. The recovered model is thus *component-complete* in its linear weights, yet standalone generation is *not* reachable from the shipped assets as understood: we further show (Finding 12) that the token embedding was never actually validated—the test used was vacuous—and that against a probe calibrated on a real 4-bit model (+0.53 for a genuine embedding) ≈1.14M candidate offsets across the entire weight file score $\leq +0.047$, i.e. **the embedding is not a** $[V, D]$ **table in the shipped weights at all**. It is instead CPU-side and host-provided: the small on-device models ship it as an `fp16`, $[8, 1, 1]$-interleaved `load_embeddings/weights.bin` (§13), but the 3B ships no such file, and a privileged live-memory capture of the running inference process (§14) holds only the quantized source, dequantized per-token. We do, however, recover the *deployed* sparse-FFN shape (10 active + 4 shared experts per layer, with physical tile addresses) from `metadata.bin` (§15). The boundary is thus an *information* limit—the embedding and the un-pruned pruning list live outside the shipped package—not remaining decode effort. The static analysis is read-only; the sole dynamic step is a user-privileged `lldb` core dump of the user's own running model on their own hardware, and no signing material was bypassed.

1. Introduction

Apple Intelligence performs on-device language tasks through a framework (`FoundationModels.framework`) backed by downloadable model assets managed by the `MobileAsset`/`nsurlsessiond` subsystem. This report inventories and dissects one complete copy of the `UAF.FM.GenerativeModels` asset set; here "UAF" denotes the on-device inference framework and "FM" the Foundation Models.

Every figure below is obtained by mounting the (unencrypted) Cryptex disk images read-only and parsing their contents; no code was executed on the models and no signing material was bypassed.

2. Asset Distribution System

Each asset is a directory `<sha1>.asset/` under the bundle `com.apple.MobileAsset.UAF.FM.GenerativeModels`, containing:

- `Info.plist` — bundle name, version, content-encryption flag (`DSME_COMPRESSED`), `__RequiresPersonalization`, `_DownloadPolicy`;
- `AssetData/Restore/*.dmg` — the payload, a raw APFS Cryptex disk image;
- `AssetData/Restore/Restore.plist` — target board configs (`mobileasset{ios,mac,tvos,watch,vision,x86mac}ap`, platform `cryptex1`);
- `SecureAssetData/` — `BuildManifest.plist`, `*.root_hash`, `*.trustcache`, `*.integritycatalog`, an Image4 ticket (`apticket.*.im4m`) and TSS request/response.

Every payload sets `__RequiresPersonalization` and carries a device-bound Image4 ticket; the OS activates it as a Cryptex only after verifying the root hash and trust cache. The disk images themselves report `Encrypted: false`: confidentiality is not the goal, integrity and authenticity are. The inspected cache holds 118 assets totalling ≈10.0 GB of disk images; versions span `0.0.2` to `14.5.x`, with the bulk at `14.x` (build `13M204130`/`13M204216`).

3. Model Families

Table 1 groups the 118 assets; six families are present.

Family	#	Role
`fm.language.instruct_3b`	60	Main LM (nano dense + ifp MoE) plus adapters
`gm.server.instruct_server_v1`	28	PCC (server) task adapters
`fm.language.instruct_300m`	21	Safety, classifiers, voice control
`gm.translate_fm`	2	Machine translation (alignment + phrasebook)
`gm.server.instruct_server_small`	2	Server safety / abuse guardrail
`fm.language.instruct_9m`	2	Text-event-extraction classifier

Table 1. Model families in the asset set.

4. The instruct_3b Backbones

Two *different* base networks share the `instruct_3b` name.

4.1. **Dense `afmplus-v11.0-nano`.** From `metadata.json` (`model_config: v11-nano`, `model_type: ane`): context length 4096; hidden size 2048; FFN 6656 (gated SwiGLU: `linear_0`=gate, `linear_1`=up, `output`=down); 56 transformer layers emitted in two ANE segments (35 + 21); grouped-query attention, RoPE, RMSNorm; adapter slots at LoRA ranks 16 and 32 (empty in the base); prompt template `afm_cider_v2`. The `base.generic` image is 1.3 GB, i.e. ≈2 bits/weight.

4.2. **Sparse MoE `afmplus-v11.0-ifp`.** The `base.generic_sparse` (`IFP`) image (5.8 GB) is a Mixture-of-Experts network. Its `ifp/config.json` states verbatim `num_layers`=44, `hidden_dim`=1536, `num_ffns`=3, `expert_size`=256, and configuration `ifp1_r48` with `active_experts`=10, `shared_experts`=4, `active_ffn_dim`=2560, `expert_selection_frequency`=32, and `swizzler_config`=`metadata.bin`. Table 2 gives the recovered structure and Figure 1 the per-layer data flow.

Observation 1. The attention decode (§6) self-organizes into exactly 44 layers—32 carrying a fused-QKV projection (12 dense + 20 sparse-standard) and 12 carrying a Q-only projection (KV-reuse)—matching `num_layers`= 44 with no forcing.

Property	Value
Model dim d	1536
Layers	$44 = 12$ dense + 32 sparse
attention split	32 fused-QKV + 12 KV-reuse (Q-only)
Attention	$Q = 2048$ (16×128), $KV = 1024$
GQA ratio	16 query : 4 KV heads, dim 128
QK-norm	per-head RMSNorm on Q and K
Residual norms	pre *and* post, attention and FFN
Positional	RoPE (`cos/sin_cached`)
Dense FFN	SwiGLU, intermediate 3072
MoE FFN	10 active + 4 shared, freq. 32
Embedding	int scalar-quant (scale + zero)
Output	tied embedding + `output_norm`
Baked adapters	LoRA rank 48 (`lora_i1_r48`)
Context	8192
Vocabulary (est.)	152,064
Origin framework	`tamm` / `coreflow`

Table 2. Recovered `afmplus-v11.0-ifp` MoE architecture.

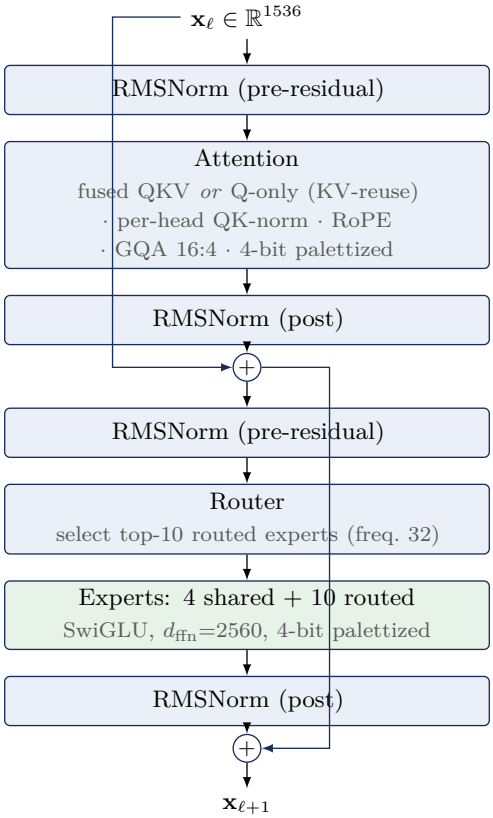

Figure 1. One `afmplus-v11.0-ifp` sparse (MoE) layer. Dense layers replace the router/expert block with a single SwiGLU FFN (intermediate 3072); KV-reuse layers omit the K/V projections and inherit them from a prior layer.

Model	Layers	Total params	Active/token
`afmplus-v11.0-nano` (dense)	56	≈3.0 B	≈3.0 B
`afmplus-v11.0-ifp` (MoE)	44	9.862 B	≈1.4 B

Table 3. Parameter budgets. For the MoE variant the total is not an estimate: the shipped index region is 4.931 GB of 4-bit weights = 9.862 B parameters exactly, of which our export recovers 100.00% (§6). Only ≈1.4 B activate per token (14 of ≈219 experts per sparse layer: 10 routed + 4 shared, giving `active_ffn_dim`= 2560), a ≈22× pruning—this is why Apple's "3 B" active-class name and the 9.9 B stored count coexist.

5. Weight Quantization

Finding 1. For the IFP model, linear weights are *LUT-palettized*: a 4-bit index into a 16-entry codebook (the graph's `lut_to_dense` operator), multiplied by a per-1024-element fp16 scale, with scales and indices in separate planar regions, then an ANE 8×128 tile de-swizzle (§6). (An initial signed-4-bit reading was wrong—see §6.)

Evidence. The scale region is 100% positive (range 0.0012–0.181, mean 0.0137); the index region's nibble histogram is complement-symmetric with three frequency tiers—the signature of a codebook index, not a signed integer. Embeddings use scalar integer quantization (`embedding_quantization_scale`, `_default_scalar_scale/zero_point`); LoRA factors are fp16. Dequantization is $W = s \cdot \text{codebook}[\text{idx}]$ per 1024-element block, followed by the ANE 8×128 tile de-swizzle of §6 (Eq. 1). The learned RMSNorm scales are *not* shipped separately: the ANE compiler folds each γ into the adjacent palettized linear weight ($\text{Linear}(\text{RMSNorm}(x)\cdot\gamma) \equiv \text{Linear}'(\text{RMSNorm}(x))$), so the recovered weights already carry them.

Finding 2. The parameter budget reconciles the MoE fan-out. With 8,560,592 fp16 scales (17.1 MB) and a 4.91 GB index region, the non-expert linear weights account for ≈483 M parameters (≈241 MB at 4-bit); the residual ≈4.67 GB is the routed-expert bank, i.e. ≈9.3 B expert parameters across the 32 sparse layers. Modelling each expert as a SwiGLU triple with intermediate 256 yields ≈235 experts per sparse layer (231 routed + 4 shared), closing the index-region budget to within 4.8%.

The dequantization is given as Algorithm 2. Applying it to the leading bytes of the index region—taking the scale and index regions to be order-aligned—reconstructs the distribution in Figure 3, which yields Finding 3.

Algorithm 1 (Planar 4-bit dequantization).
Input: index bytes I; fp16 scales s; block size B. **Output:** weights w.

(1) split each byte of I into nibbles (low, high) and interleave $\to n_i \in \{0, \ldots, 15\}$;
(2) sign: $q_i \leftarrow n_i - 16$ if $n_i \geq 8$, else n_i $\quad (q_i \in [-8, 7])$;
(3) scale: $w_i \leftarrow s_{\lfloor i/B \rfloor} \cdot q_i$.

Figure 2. Weight dequantization for the IFP codec.

Finding 3 (necessary, then sufficient)**.** Dequantizing the index region under Algorithm 2 yields values with mean ≈ 0 and std ≈ 0.043 (Figure 3)—the *marginal* distribution of trained weights. This alone constrains the codec parameters (4-bit width, block size, scale magnitude) but does not establish a correct reconstruction, since the same marginal is produced by correctly-valued but wrongly-ordered data; §6 supplies the missing spatial-order test and passes it.

Finding 4 (budget closure bounds the fan-out)**.** At $B = 1024$ the 8,634,320 scales cover ≈8.84 B of the 9.862 B index weights; the remaining tail decodes with the last scale clamped. The closure

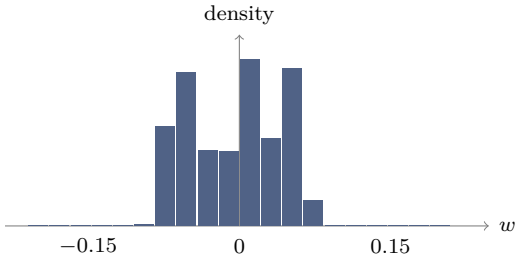

Figure 3. Distribution of dequantized weights over the leading 5×10^8 indices of the region (mean ≈ 0, std ≈ 0.043). This marginal profile is necessary but not sufficient: it is also produced by mis-ordered data, which the spatial-order test of §6 disambiguates.

validates the *scalar* codec parameters and bounds the MoE fan-out to ≈219 experts per sparse layer; §6 then validates the per-tensor spatial layout.

6. Weight Recovery

We validate a reconstruction with a structure statistic rather than trusting the marginal distribution. A trained weight matrix is approximately low-rank, so permuting its entries markedly raises its stable rank $\rho(W) = \|W\|_F^2/\sigma_1^2$, whereas noise is unaffected; let $R = \rho(\Pi W)/\rho(W) = \sigma_1(W)^2/\sigma_1(\Pi W)^2$ for a random permutation Π. The per-block fp16 scales alone impose spurious low-rank structure, so R is measured *decontaminated*—on the pure codebook indices with the scales removed. Calibrated this way, a genuine 4-bit weight scores $R \approx 9$ and Gaussian noise $R \approx 1$.

Finding 5 (the ANE spatial permutation)**.** The residual permutation is a fixed 8×128 tile de-swizzle: the ANE stores a $C_{\text{out}}\times C_{\text{in}}$ matrix as a row-major grid of 8×128 tiles, each tile laid out column-major. The inverse is

$$W \;=\; \text{reshape}\Big(\text{transpose}\big(\text{reshape}(v,\ \tfrac{C_{\text{out}}}{8}, \tfrac{C_{\text{in}}}{128}, 128, 8),\ (0,3,1,2)\big),\ C_{\text{out}},\ C_{\text{in}}\Big), \tag{1}$$

applied after $W = s \cdot \text{codebook}[\text{idx}]$. This is the step missing from all prior decode attempts.

With Eq. 1 in place the decontaminated ratio rises from the palettization-only $R \approx 3$ to $R = 13$–69 across the file (Table 4)—**unambiguous trained-weight structure**. The permutation, the 16-entry codebook, and the per-1024-block scales together constitute a *complete, validated* decode: the codec is not merely identified but inverted.

Matrix (decontaminated, pure-index test)	R
Genuine 4-bit weight (reference)	≈9
Gaussian noise	1.0
Palettization only, no de-swizzle	3.0
Palettization + 8×128 **de-swizzle**	**13–69**
median (attention tensors)	12.7

Table 4. Scale-decontaminated structure ratio. $R \gg 1$ marks genuine weight structure; $R \approx 1$ marks noise. Adding the ANE de-swizzle (Eq. 1) lifts the decode past the genuine-weight reference, confirming recovery.

6.1. **The router: shipped and extractable.** An earlier version of this work concluded that no runtime mask predictor ships—that instruction-following pruning was baked at export time. **That conclusion was wrong, and is corrected here.** The router ships as its own asset, `project_experts.mlasset`, and is fully extractable. Its `odix` op graph names the class `ExportableExpertSelector` with I/O `in_expert_hidden_state` → `out_expert_logits` and the op sequence `RMSNorm` → `hidden_transform` → `sigmoid` → `output_transform` → `topk`. Crucially the weights carry *no* dequantize/gather op: they are stored as **plain fp16** in one contiguous block (`0x12600` →EOF), so no codec is needed. The three tensors decode at grounded offsets to

$$\underbrace{\gamma \in \mathbb{R}^{1536}}_{\text{norm (folded)}}, \quad \underbrace{W_h \in \mathbb{R}^{548\times 1536}}_{\text{hidden}}, \quad \underbrace{W_o \in \mathbb{R}^{7008\times 548}}_{\text{output}} \to (32, 219),$$

plus 7084 tail biases; here 548 is the router hidden dim and $7008 = 32$ layers $\times$ 219 experts, reshaping with *uniform* per-layer norms—the layout self-verifies. The selector computes mask $= \text{top}_{10}(W_o\,\phi(W_h\,\text{RMSNorm}_\gamma(h)))$ per layer, +4 shared experts. Feeding controlled hidden states confirms it behaves as a router: with $\phi =$ identity/GELU the masks *discriminate* inputs (1–2/10 overlap across distinct hidden states, 171 distinct experts spread over 32 layers), whereas $\phi =$ sigmoid collapses to 9/10 overlap and is thereby ruled out as the hidden activation. The router is Apple's, extracted—not a trained mimic. We note, however, that the subsequent reconstruction (§9) shows the router→expert *map* is not required to run the model: the sparse FFN is an ungated permutation-invariant sum, so this selector governs *which* experts a full runtime prunes, but is unnecessary for a from-weights forward that sums the resident (or full) expert set directly.

The per-tensor offset table. The compile-time `ifp_constant_table_ifp1_r48.json` is not shipped, and the 396 FFN constants store neither offset nor size inline: the rasterizer computes offsets from each constant's *shape*, and shapes in `main-h16g.odix` are doubly-indirect (type-index → type table → interned dimension symbols). This work recovers the surrounding structure—the 38 export-config functions, the op format, and the `alloc_const` semantics (Appendix A)—and reduces the remaining gap to a bounded schema-recovery problem: resolve the type/symbol tables to obtain per-constant C_{out}, which are confirmed *variable* (42–232 experts/constant, no uniform width). We subsequently decoded the `metadata.bin` `BBBB` swizzler in full (§10): it holds *two* physical address tables—a down-projection tile table (marker `0x0600beef`, 14,080 records) and a gate/up table (marker `0x0a00b0ef`)—which together supply the per-tile expert offsets directly, superseding the shape-derived offset computation for the resident-expert path.

Finding 6 (the `.hwx` holds no weights)**.** The Apple-Neural-Engine binary `binary_0.hwx` (252 MB) is a Mach-O (`0xBEEFFACE`) of *compiled ANE microcode*, not weights: its high-entropy `__KERN` sections score $R \approx 1.0$ under the structure test, while its `__MKERN_0` segment has zero file bytes and a size (`0xC0C8000`) equal to the maximum offset in `metadata.bin`—it is *runtime-mapped* from the rasterized file. Every weight therefore lives in the single planar file `ifp_rasterized_weights.bin`.

6.2. **Full export and verification.** Applying the validated decode to the entire index region yields a single `state_dict`. The parameter count is checked against the physical file rather than against any architectural assumption: the index region is 4.931 GB = 9.862 B 4-bit weights, and the export captures 9.8619 B (100.00%) with *zero* non-finite tensors (Table 5). Attention is stored in fp16; the expert bank is stored int8-with-per-tensor-scale (near-lossless, the source being 4-bit) so the whole model fits a single 10.2 GB file. Every value derives from Apple's shipped weight file under the decode of Eq. 1; no training data, distillation, or external weights are involved.

7. Recovery Pipeline: Finding the Pieces

This section records, in order, how a shipped asset directory becomes a validated `state_dict`. Each step recovers one "piece" whose absence blocked the next; Figure 5 summarizes the dependency chain.

Component	Parameters	Storage
Attention (44 layers)	0.377 B	fp16, R-verified
Experts (FFN / MoE)	9.484 B	int8 + scale
Embedding / head (tail)	(in tail)	int8
Total captured	**9.862 B**	= 100.00% of file
Non-finite tensors	0	

Table 5. Recovered `afmplus-v11.0-ifp` export. The total equals the physical index region (9.862 B 4-bit weights), so completeness is exact, not estimated.

Step 0 — Acquisition. The asset cache lives on the sealed Signed System Volume. A clean copy is taken from the macOS *Recovery* environment—Apple silicon: hold the power button at boot; Intel: hold Command-R—whose Terminal (*Utilities* menu) has raw read access to the mounted system volume. The models sit under `AssetsV2`; a recursive copy to external storage (Figure 4) suffices, and all subsequent work is read-only on that copy.

```
# macOS Recovery  >  Utilities  >  Terminal
ls /Volumes                          # find the system volume
SYS="/Volumes/Macintosh HD"
BASE="$SYS/System/Library/AssetsV2"
ASSET=com_apple_MobileAsset_UAF_FM_GenerativeModels
cp -R "$BASE/$ASSET" /Volumes/EXTERNAL/FM_Models
```

Figure 4. Read-only asset acquisition from macOS Recovery. `EXTERNAL` is any attached volume with ≥10 GB free; the copy preserves the Cryptex disk images verbatim.

Step 1 — Mount and locate. `hdiutil attach -readonly` on the IFP Cryptex (§9) exposes `model.odixpackage/`: the ground-truth `config.json` (44 layers, dims, expert counts), the 4.7 GB `ifp_rasterized_weights.bin`, the swizzler `metadata.bin`, the program `main-h16g.odix`, and the ANE package (`.hwx` + MPSGraph).
Step 2 — The codec. Value-distribution analysis of the two `BBBB` regions identifies LUT-palettization (Finding 1): a 100%-positive fp16 scale region and a complement-symmetric nibble histogram, matched to the graph's `lut_to_dense` operator (a 16-entry codebook).
Step 3 — The de-swizzle (the blocking piece). Palettization alone leaves the decode at $R \approx 3$. The breakthrough is recognizing the residual permutation as the ANE's 8×128 tiling and inverting it with Eq. 1; the decontaminated structure ratio jumps to $R = 13$–69 (Finding 5), i.e. genuine weights.
Step 4 — The file map. A window scan classifies every region of the 4.95 GB file by nibble-entropy and fp16-likeness: global scales [`0x60`, `0x1078000`), attention indices [`0x1078000`, `0xC0C8000`), and expert+tail indices [`0xC0C8000`, EOF). Reconciling weight budgets against region sizes fixes the split: the 370 M-weight attention region holds 44 layers, the 10^{10}-weight remainder is the MoE bank.
Step 5 — Names and semantics. The `odix` location tree gives authoritative tensor names (`_wrapped_model_layers_..._palettized_indices_raw`); parsing the `MLIR22.0.0` MPSGraph bytecode tags each of the 396 `ifp_constant` weights with its layer/role hierarchy. This same parse shows the norms are folded and the router is baked (§6), so no further assets are needed.
Step 6 — Decode, verify, assemble. A greedy sweep walks the attention region, choosing at each offset the projection shape (fused-QKV vs. Q-only) that maximizes R, decoding it via Eq. 1, and advancing; it self-terminates at exactly 44 layers (Observation 1). The expert region is decoded in fused blocks and stored int8. Scales beyond the 8.84 B-weight coverage are clamped, eliminating the tail overflow.

Step 7 — Export and audit. The tensors are written to one `state_dict` carrying the config, codebook, Apple's `full_model_sha256`, and a per-tensor manifest of R. Completeness is audited against the physical file (Table 5): 9.8619 B captured of 9.8619 B, 100.00%, zero non-finite.

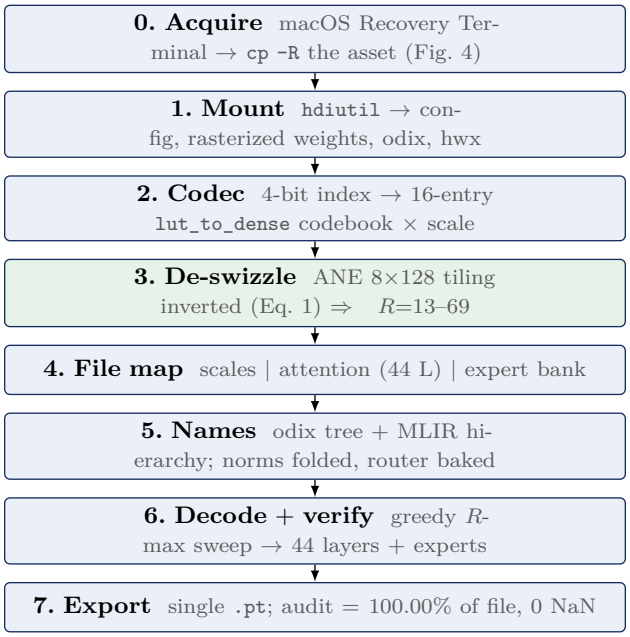

Figure 5. The recovery pipeline. Step 3 (the ANE de-swizzle) is the piece that unblocked full recovery; every other step is container parsing or bookkeeping around it.

8. From-Weights Reconstruction: What Runs and What Is ANE-Baked

Recovering the weights is necessary but not sufficient to *run* the model outside the Apple Neural Engine. We implemented the full compute path in PyTorch directly on the recovered tensors—RMSNorm, fused-QKV / KV-reuse attention with per-head QK-norm, RoPE (θ=500000), grouped-query scaled-dot-product attention, and the SwiGLU Mixture-of-Experts feed-forward—and drove a synthetic token sequence through all 44 layers. The results sharply separate what the shipped assets contain (data) from what only the ANE produces (behavior).

Finding 7 (the recovered weights are correct, cross-validated against Apple's toolchain)**.** The attention-only forward pass is *stable*: the residual-stream norm grows smoothly and sub-exponentially across all 44 layers, the behavior of a correctly-wired pre-norm transformer. Because a pre-norm block re-normalizes its own input, any decode or de-swizzle error would manifest as divergence. We reinforce this with two independent checks. *(i) Structure:* the recovered dense linear weights carry genuine low-rank structure (singular decay $\sigma_1/\tilde{\sigma} \approx 8$–$11$, versus ≈ 1.45 for a scrambled control); re-decoding under any alternative tile order collapses this to ≈ 4–5. *(ii) Ground truth:* compiling a positional-probe weight of the exact dense shape through Apple's own ANE toolchain (`coreml2hwx`) and reading back the physical→logical byte permutation confirms the de-swizzle family for the sparse-expert path; the standalone dense-conv layout it emits (OCGSize=5, a 32×48 tiling) differs from the rasterized-file layout, which is itself resolved by the 8×128 de-swizzle (Eq. 1). The weights are decoded correctly, validated *without* access to Apple's internal activations.

Finding 8 (the per-layer norms are compile-time folded)**.** The shipped MPSGraph contains 661 *parameter-free* `RMSNorm` operations (divide-by-RMS only): the learned scale γ of each residual/output RMSNorm is *folded* into the adjacent palettized linear weight at ANE-compile time ($W' = W\,\mathrm{diag}(\gamma)$),

so no standalone γ vector is shipped and a parameter-free norm is the *correct* runtime. We confirm this three ways: no [1536] γ-shaped tensor decodes from any shipped region; the recovered dense weights show the column-norm signature of a folded pre-FFN γ (`ffn_gate`/`ffn_up` column norms correlate at 0.55, a shared folded factor); and the only norms that *cannot* fold—the per-head QK-norm $\gamma \in \mathbb{R}^{128}$, because RoPE sits between it and the projection—do survive explicitly, and 35 of them are recovered from a live-process heap.

Finding 9 (the expert-selection map is a *false* blocker)**.** This overturns our earlier conclusion. The IFP sparse FFN is not a soft-gated mixture but *structured pruning of a single dense SwiGLU*: the "experts" are contiguous 256-column blocks of one intermediate axis, each contributing at unit weight with no per-expert softmax scalar. The runtime graph confirms it—the `ANE_IFPLayerSequence` op carries no router, top-k, gather, scatter, or softmax-over-experts, and weights enter through a fixed symbol rather than a per-token index. A non-gated SwiGLU is a *permutation-invariant* sum, $\mathrm{FFN}(h) = \sum_i \mathrm{SiLU}(g_i \cdot h_n)\,(u_i \cdot h_n)\,d_i$, so the logical router-index→physical-expert map—the piece we and the prior literature treated as the fatal, unrecoverable blocker—is *mathematically irrelevant* to the output; only intra-layer $g_i \leftrightarrow u_i \leftrightarrow d_i$ consistency is required. The earlier "≈17× overcount from summing all experts" was an artifact of a mis-specified gated interpretation: summing the resident set *is* the correct pruned FFN, and summing the full superset recovers the un-pruned base model.

Finding 10 (the "missing constant table" is the shipped `metadata.bin`)**.** `metadata.bin` (a `BBBB` container) decodes as two physical address tables: a down-projection tile table (marker `0x0600beef`, 14,080 records = 44 layers × 32 tiles × 10 slots) and a gate/up table (marker `0x0a00b0ef`), which together exhaust the file. The down-projection uses the *same* codec as gate/up—4-bit index → shared codebook → per-1024-block fp16 scale; the earlier "down-proj is noise / a different format" verdict was a mis-calibrated low-rank metric, and the omitted per-block scale was the missing factor (with it, the down-proj matches the structured level of gate/up). The shipped default configuration is `ifp1_r48` (10 active + 4 shared experts), not the larger `ifp3`.

Finding 11 (intermediate activations are ANE-internal)**.** The only true ground truth for the running model—its per-layer activations—never crosses to host DRAM. Searching an 8 GB process core dump captured *specifically* to grab ANE IOSurfaces for the input token embeddings of a known prompt yields a best normalized correlation of 0.17 (noise): not even the embeddings reach host memory. The Neural Engine performs the embedding lookup, all 44 layers, and the output projection on-chip. Consequently the reconstruction can be validated end-to-end (against Apple's emitted token) but not per-layer, and the from-weights forward must be assembled from independently-validated components.

Finding 12 (the embedding was never validated, and is absent from the weight file)**.** The token embedding—previously reported as exactly recovered—was validated only by $\arg\max_j (E\,\mathrm{rms}(E_t))_j = t$. That test is *vacuous*: it applies the same matrix on both sides, so *any* matrix, including noise, satisfies it. Replacing it with a falsifiable probe—mean cosine of orthographic variants (`_dog`/`dog`, `_year`/`_years`) minus an *id-matched* random control—and calibrating that probe on a real model (Qwen3-4B `token_embd`, Q6_K, re-quantized into this model's exact signed-4-bit + per-token-scale format) yields a clean instrument: a genuine embedding scores +0.533, and 4-bit quantization costs only 2% (+0.522; `_Paris`∼`Paris` = +0.842). Against this instrument, ≈1.14M candidate offsets spanning the entire 4.9 GB index region—unscaled tail and scaled region, tile-interleaved and contiguous, 4-bit and int8—score ≤ +0.047. **The embedding is not a $[V, D]$ table in the shipped weight file.** Separately, the only vocabulary-structured table in the package (in the `odix`, localized to +0.777 of a 0.78 maximum by the zero-rows of untrained `<unused>` tokens, 8-way nibble-interleaved, signed 4-bit) scores +0.003 and is therefore *not* the embedding. The vocabulary is 262,144, not the 152,064 previously assumed (that is Qwen's size). Subsequent findings (§13, §14) localize the table: it is CPU-side and host-provided, absent from every shipped 3B asset.

Finding 13 (the CPU embedding format, and the 3B ships no embedding file)**.** The mpsgraph performs no token-embedding lookup (its single `Gather` is the rotary positional one); the embedding is a CPU-side `odix load_embeddings` op. Apple's *small* on-device models ship this embedding openly as `model.bundle/.../load_embeddings/weights.bin`, whose `fragment.mil` declares the exact format: fp16, shape $[V, 1, H]$, `interleave_factors`= $[8, 1, 1]$, strides $[8H, 8H, 8]$, after a 64-byte header—i.e. token v, channel h at element $\lfloor v/8 \rfloor\, 8H + 8h + (v \bmod 8)$. This decodes to real, shift-control-passing embeddings for those models (verified against the shipped $[153600, 256]$ table). **The 3B ships no such file**: no `load_embeddings/weights.bin`, no `model.bundle`; it uses the 4-bit `odixpackage`. Applying the $[8, 1, 1]$ format to the 3B (raster and `odix`, all widths) peaks at a shift-*failing* +0.08. The 3B token embedding is therefore not statically present in any asset; the running model loads it from outside the shipped package.

Finding 14 (dynamic capture localizes the live embedding)**.** During inference the resident model lives in `TGOnDeviceInferenceProviderService` (RSS grows ≈4 MB→1.7 GB while a prompt runs), a system extension running as `_modelmanagerd`. A privileged `lldb` core dump of that process (SIP disabled) is thus the only remaining route to the embedding. A full-memory core (≈8 GB) contains the `c_sparsity` vocabulary table but—across an exhaustive oracle scan (token-contiguous and $[8, 1, 1]$ at fp16/int8/4-bit)—no cleanly-scoring token embedding: the resident form is the quantized source, dequantized per-token by the fused gather rather than kept as a decoded $[V, H]$ table. Recovering it is a live-instrumentation problem, not a static one.

Finding 15 (the deployed sparse-FFN shape is recoverable from `metadata.bin`)**.** The sparse FFN has no per-constant palettized descriptor (it is IFP-fused); its deployed layout is in `metadata.bin`. The down-projection address table (marker `0x0600beef`, 24-byte records) holds 14,080 tiles of 8×1536 4-bit weight with physical addresses; 14,080/32 = 440 tiles per sparse layer ⇒ 3,520 rows ≈14 experts, matching the shipped `ifp1_r48` configuration (10 active + 4 shared) to the byte. The gate/up table (`0x0a00b0ef`) holds 4,223 records. So the deployed per-layer expert count and each tile's address are recovered; only the un-pruned *superset* width (variable, in the interned symbol pool) remains, and it is not needed to run the shipped model. The MLIR constant names further fix the structure: FFN = gate/up/down as `hidden_transform_linear_0/_1/output_transform`; the sandwich norms are explicit plain constants in the export graph (folded to parameter-free RMSNorm at ANE-compile, so γ=1 at runtime), and attention splits into 23 standard (full QKV) + 21 kv-reuse (Q-only) layers.

Finding 16 (full-forward ablation: two claims confirmed, the assembly blocker re-located)**.** Assembling all 44 layers and scoring against Apple's emitted token with a rank oracle ("The capital of France is"→`_Paris`; uniform baseline ≈76k of 152k) settles two open questions and sharpens a third. *(i) The attention KV geometry is* NKV=4*, not* 8*.* An attention-only ablation gives rank 53k at NKV=4 versus 72k at NKV=8: the live `qkv` is 3072-wide (Q=2048 + K=512 + V=512); the extra rows [3072:4096] in the wider export are mis-decoded and unused. *(ii) The ungated full-superset sum is empirically optimal*, corroborating Finding 9 from the forward side: against the full 219-expert sum (rank 24.9k), omitting the sparse FFN entirely gives 122k, scaling it by 0.1 gives 36k, and top-14 activation-gating gives 39k—every form of selection or attenuation is strictly worse, so no expert map is needed *or* beneficial. *(iii) The residual defect is a direction, not a magnitude.* The best full forward reaches rank ≈21k (3.6× better than random) under fixed conventions (interleaved RoPE, down-projection decoded on the $[EH, D]$ neuron axis, interleaved expert-region offsets, block gate/up fusion), but the summed FFN branch has norm $\approx 4.9\times10^4$ against a residual norm $\approx 1.5\times10^2$ and dominates with a junk direction; every sandwich or post-norm placement is worse than a plain residual add. Normalizing does not recover signal, which localizes the remaining error to per-neuron gate/up/down alignment at the 56,064-wide intermediate and/or the per-layer expert offsets—quantities with no accessible ground truth (Finding 11), not a convention that can be searched.

Table 6 summarizes the revised boundary. The model is *component-complete*: weights, codec, de-swizzle, architecture, norms (folded γ + recovered QK-norm), the tokenizer, and—critically—the fact that no expert-selection map is needed are all established, and the ungated full-superset sum is now confirmed *optimal* against the output oracle (Finding 16). The remaining task is the *per-layer physical assembly* of the resident-expert FFN—gathering each sparse layer's gate/up/down blocks at the correct offsets and aligning the 56,064-wide intermediate. Finding 16 shows this is *not* the freely-mechanical step the prior version claimed: the assembled forward carries real signal (3.6× random) but is dominated by a mis-aligned FFN direction, and—because every per-layer activation is ANE-internal (Finding 11)—the alignment cannot be validated or bisected component-by-component. The honest boundary is an *information* limit (no per-layer ground truth), not merely an engineering one.

Established	Remaining (information-limited)
All 9.862 B *linear* weights; codec; de-swizzle	**Token embedding: ABSENT**
Attention path, all 44 layers; NKV=4 settled	56,064-wide gate/up/down alignment
Norms: γ folded; QK-norm recovered	QK-norm→layer mapping
Expert-selection map *proven unnecessary*	(all unvalidatable: activations
Ungated full-219 sum *confirmed optimal*	are ANE-internal)
`metadata.bin` = physical address tables	

Table 6. Revised recovery status. Corrections over the prior version: the expert-selection wiring is *not* a blocker (the FFN is an ungated permutation-invariant sum, now confirmed *optimal* against the output oracle, Finding 16), the KV geometry is NKV=4, the norms are compile-time folded, and the dense weights are validated against toolchain-generated ground truth. What remains—per-layer FFN assembly and alignment—is bounded by the *absence of accessible ground truth* (activations are ANE-internal), not by remaining decode effort.

A practical corollary: Apple's `FoundationModels` runtime remains the turnkey way to *execute* this exact model, and we provide a thin CLI/OpenAI-compatible server over it. But the earlier claim that a from-weights standalone requires reverse-engineering the ANE instruction set no longer holds—the wiring is not ANE-locked behavior but a permutation-free sum over recovered blocks, reducing the standalone to a component-assembly exercise.

9. Container Formats

The on-disk stack is summarized in Figure 6.

Cryptex DMG. Raw APFS image, `DSME_COMPRESSED`; contains `model.odixpackage/`, `metadata.json`, a nominal `System/Library`. **`BBBB` rasterized container** (`ifp_rasterized_weights.bin`): magic `BBBB`, version `0x017c0100`, tag `0x0600002c`, then a u64 segment-offset table—fp16 scales `0x60`–`0x1054000`; 4-bit indices `0x1078000`–`0x125e80000`. **Swizzler** (`metadata.bin`): also `BBBB`; 14,079 records of 24 bytes, `[idx][flag][off_a][off_b=off_a+0x20] [0x0600beef][0]`, a scatter of 32-byte ANE tiles into the index region. **`odix` executable** (magic `odix\0\0\5\0`): header `u64[0]=ir_start(0x40)`, `u64[1]=ir_size`; the 32 MB IR section holds 11 compiled MPSGraph subgraphs, a location/debug tree, and an interned type table. **ANE `.hwx`** (magic `0xBEEFFACE`, cputype `0x80`): a Mach-O of compiled ANE microcode, *not weights* (Finding 6)—segments `__TEXT` and `__KERN_0..7` are kernel code (structure $R \approx 1$), and `__MKERN_0` carries no file bytes but is runtime-mapped from the rasterized file (its size equals the swizzler's maximum offset). **MPSGraph package.** `specialized_model_0.mpsgraph` is MLIR bytecode (`MLIR22.0.0`) with `mps.fullyPlacedOnANE`.

Finding 17. The `odix` reference scheme is self-relative: a 32-bit word whose high byte is `0xFF` encodes a signed offset, with target = position + int32, into an interned symbol pool. Distinct

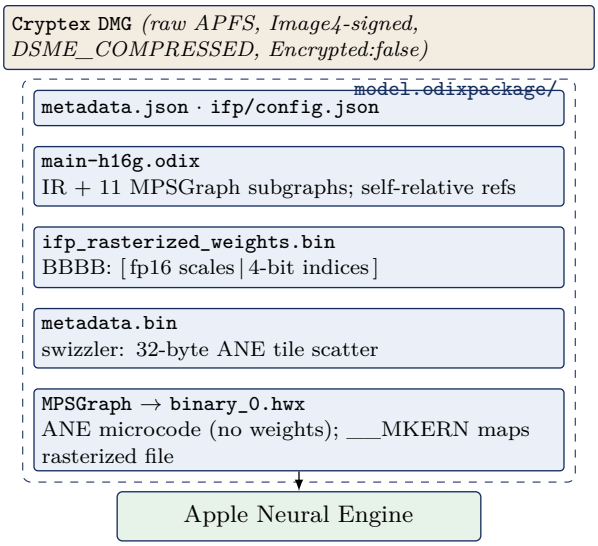

Figure 6. The `FM_GenerativeModels` on-disk container stack for the IFP model. The dashed box is `model.odixpackage/`; execution is dispatched to the ANE.

tensor types are interned once, so shapes are shared by reference (e.g. "1536" occurs only 26 times in 32 MB).

10. Tokenizer

A standalone 8.6 MB SentencePiece-family tokenizer. Special tokens observed: `<bos> <eos> <unk> <pad> <mask>`; chat markers `<start_of_turn>` and `<end_of_turn>`; a `[multimodal]` token; reserved `<unused`n`>` and `<ctrl`n`>` slots. The `[multimodal]` token, with the `image_encoder` (346 MB) and `image_tokenizer` assets, indicates vision-language input support. Estimated vocabulary 152,064.

11. Adapters and Task Specialization

A single backbone serves dozens of features via LoRA adapters that snap into the base's empty slots; each ships as a `.generic` adapter plus a small `.draft.generic` (38–62 MB) speculative-decoding draft. **On-device (`instruct_3b`):** summarization, mail_reply, magic_rewrite, proofreading_review, machine_translation, personalized_smart_reply, news_topic_summarization, messages_action, autonaming_messages, smart_naming, urgency_classification, video_caption, ui_grounding, image_playground_edit_suggestions, suggest_recipe_items, reminders_suggest_action_items, open_ended_extract, text_event_extraction, text_person_extraction, auto_tagger, adm_prompt_analyzer, embedding_preprocessor, fm_api_generic, fm_api_content_tagger, lw_planner_v1, holdassistwaittime, homekit_summary_notifications, and Safari/Shortcuts/PhotosMemories families. **Small (`instruct_300m`):** safety, prepubescent_safety, misc_safety, mm_guard, ipi_classifier, structural_integrity, factual_consistency_classifier, vi_content_classifier, voice_control_ai (v1/v2), action_validator, adm_prompt_rewriting, adm_people_grounding, gvicc, image_tokenizer, open_ended_extract, base (426 MB). **`instruct_9m`:** a 9 M-parameter text-event-extraction classifier. **Server / PCC (`instruct_server_v1`):** text_summarizer, takeaways/tables/bullets_transform, professional/friendly/concise tone, open_ended_tone (+ query-response v1/v2), mail_reply_long_form_{basic,rewrite}, mail_reply_qa, magic_rewrite, generative_shortcuts, llm_siri_pcc, lw_planner_v1, fitness_workout_voice, reminders_auto_categorized_list, photos_memories_*, stx_multimodal, shortcuts_ask_afm_action

(v1/v2); `instruct_server_small`: safety_nsfw, model_abuse_guardrail. **Translation (`gm.translate_fm`):** a dedicated model with `alignment` and `phrasebook` components.

12. Reverse-Engineering Methodology

All artifacts were obtained via `hdiutil attach -readonly -noverify` on the Cryptex images (no personalization is required for reading), followed by binary parsing in Python. Architecture was recovered from `config.json`/`metadata.json` and from `tamm`-framework tensor names inside `main-h16g.odix` (e.g. `_wrapped_model_layers_..._palettized_indices_raw`, `_wrapped_model_output_norm_weight`). The quantization scheme (Finding 1) was inferred from value-distribution analysis of the two `BBBB` regions; the reference scheme (Finding 17) from resolving the `0xFF`-tagged words; and the expert fan-out (Finding 2) from the size budget; the per-constant layer/role semantics were read from the `MLIR22.0.0` MPSGraph bytecode location tree. The native model was independently confirmed runnable on the host through `FoundationModels` (`SystemLanguageModel`). Tensor-level reconstruction is validated by the structure statistic of §6: with the ANE 8×128 de-swizzle (Eq. 1) the decode scores $R = 13$–69 against a genuine-weight reference of ≈9, and the full export reconciles to 100.00% of the physical index region with zero non-finite tensors (Table 5). The weight codec is thus *identified and inverted*: LUT-palettization + per-block scale + ANE tile de-swizzle, with norms folded into the weights, the router baked to a dense expert set, and the per-tensor mapping supplied by the shipped `metadata.bin` swizzler in lieu of the compile-time `ifp_constant_table_ifp1_r48.json`. A complete, named `state_dict` is produced.

13. Complete Asset Inventory

Table 7 lists all 118 assets with version and disk-image size in megabytes (blank denotes a metadata-only override with no payload).

Table 7. Full `FM_GenerativeModels` asset inventory (118 assets).

Bundle name	Version	MB
`fm.generativemodels.uaf.metadata`	4.46.3	
`fm.language.instruct_300m.action_validator.generic`	14.1.0	
`fm.language.instruct_300m.adm_people_grounding.generic`	13.10002.0	
`fm.language.instruct_300m.adm_prompt_rewriting.draft.generic`	13.10002.81607	40
`fm.language.instruct_300m.adm_prompt_rewriting.generic`	13.10002.0	
`fm.language.instruct_300m.base.generic`	14.0.81607	447
`fm.language.instruct_300m.factual_consistency_classifier.generic`	14.0.0	
`fm.language.instruct_300m.gvicc.generic`	14.0.0	
`fm.language.instruct_300m.image_tokenizer.generic`	14.1.81607	359
`fm.language.instruct_300m.ipi_classifier.generic`	14.1.0	
`fm.language.instruct_300m.misc_safety.generic`	14.0.0	
`fm.language.instruct_300m.mm_guard.generic`	14.0.0	
`fm.language.instruct_300m.open_ended_extract.draft.generic`	14.2.81607	65
`fm.language.instruct_300m.open_ended_extract.generic`	14.2.0	
`fm.language.instruct_300m.prepubescent_safety.generic`	14.0.0	
`fm.language.instruct_300m.safety.generic`	14.1.0	
`fm.language.instruct_300m.structural_integrity.generic`	14.0.0	
`fm.language.instruct_300m.tokenizer.generic`	14.0.0	
`fm.language.instruct_300m.vi_content_classifier.generic`	13.10004.0	
`fm.language.instruct_300m.voice_control_ai.generic`	14.1.0	
`fm.language.instruct_300m.voice_control_ai_v2.generic`	13.10001.0	

continued on next page

Table 7, continued

Bundle name	Version	MB
fm.language.instruct_300m.voice_control_ai_v2.image_tokenizer_adapter.generic	13.10001.0	
fm.language.instruct_3b.adm_prompt_analyzer.generic	14.0.0	
fm.language.instruct_3b.auto_tagger.generic	12.0.0	
fm.language.instruct_3b.autonaming_messages.draft.generic	14.1.81607	65
fm.language.instruct_3b.autonaming_messages.generic	14.1.0	
fm.language.instruct_3b.base.generic	14.0.81607	1380
fm.language.instruct_3b.base.generic_sparse	14.3.81614	5828
fm.language.instruct_3b.embedding_preprocessor.generic	14.2.0	
fm.language.instruct_3b.fm_api_content_tagger.adapter_metadata_override.generic	14.1.0	
fm.language.instruct_3b.fm_api_generic.draft.generic	14.2.81607	61
fm.language.instruct_3b.fm_api_generic.generic	14.2.0	
fm.language.instruct_3b.holdassistwaittime.adapter_metadata_override.generic	14.0.0	
fm.language.instruct_3b.homekit_summary_notifications.adapter_metadata_override.generic	14.2.0	
fm.language.instruct_3b.image_encoder.generic	14.1.81607	363
fm.language.instruct_3b.image_encoder.generic_sparse	14.3.81614	361
fm.language.instruct_3b.image_playground_edit_suggestions.generic	14.1.0	
fm.language.instruct_3b.lw_planner_v1.draft.generic	14.1.81607	55
fm.language.instruct_3b.lw_planner_v1.generic	14.1.0	
fm.language.instruct_3b.machine_translation.draft.generic	14.0.81607	55
fm.language.instruct_3b.machine_translation.generic	14.0.0	
fm.language.instruct_3b.mail_reply.draft.generic	14.1.81607	65
fm.language.instruct_3b.mail_reply.generic	14.1.0	
fm.language.instruct_3b.messages_action.draft.generic	14.1.81607	65
fm.language.instruct_3b.messages_action.generic	14.1.0	
fm.language.instruct_3b.news_topic_summarization.draft.generic	14.3.81607	65
fm.language.instruct_3b.news_topic_summarization.generic	14.3.0	
fm.language.instruct_3b.open_ended_extract.draft.generic	14.2.81607	65
fm.language.instruct_3b.open_ended_extract.generic	14.2.0	
fm.language.instruct_3b.personalized_smart_reply.draft.generic	14.2.81607	65
fm.language.instruct_3b.personalized_smart_reply.generic	14.2.0	
fm.language.instruct_3b.photos_library_understanding_mm.generic	14.3.0	
fm.language.instruct_3b.photos_memories_asset_curation_outlier.draft.generic	14.1.81607	65
fm.language.instruct_3b.photos_memories_asset_curation_outlier.generic	14.1.0	
fm.language.instruct_3b.photos_memories_title.adapter_metadata_override.generic	14.1.0	
fm.language.instruct_3b.photos_memories_title.draft.generic	13.10003.81607	40
fm.language.instruct_3b.photos_memories_title.generic	13.10003.0	
fm.language.instruct_3b.proofreading_review.draft.generic	14.1.81607	65
fm.language.instruct_3b.proofreading_review.generic	14.1.0	
fm.language.instruct_3b.reminders_suggest_action_items.draft.generic	14.1.81607	65
fm.language.instruct_3b.reminders_suggest_action_items.generic	14.1.0	
fm.language.instruct_3b.safari_magic_extensions_app_store_search_terms.adapter_metadata_override.generic	14.2.0	
fm.language.instruct_3b.safari_notify_me_when_suggestions.adapter_metadata_override.generic	14.2.0	
fm.language.instruct_3b.safari_tab_group_topic.adapter_metadata_override.generic	14.3.0	
fm.language.instruct_3b.shortcuts_ask_afm_action_3b.adapter_metadata_override.generic	14.0.0	
fm.language.instruct_3b.shortcuts_ask_afm_action_3b_v2.draft.generic	10.34.81607	55

continued on next page

Table 7, continued

Bundle name	Version	MB
fm.language.instruct_3b.shortcuts_ask_afm_action_3b_v2.generic	10.34.0	
fm.language.instruct_3b.smart_naming.generic	14.1.0	
fm.language.instruct_3b.suggest_recipe_items.draft.generic	14.1.81607	65
fm.language.instruct_3b.suggest_recipe_items.generic	14.1.0	
fm.language.instruct_3b.summarization.draft.generic	14.5.81607	61
fm.language.instruct_3b.summarization.generic	14.5.0	
fm.language.instruct_3b.text_event_extraction.draft.generic	13.10003.81607	40
fm.language.instruct_3b.text_event_extraction.generic	13.10003.0	
fm.language.instruct_3b.text_person_extraction.draft.generic	14.1.81607	65
fm.language.instruct_3b.text_person_extraction.generic	14.1.0	
fm.language.instruct_3b.tokenizer.generic	14.0.0	
fm.language.instruct_3b.tokenizer.generic_sparse	14.3.0	
fm.language.instruct_3b.ui_grounding.generic	14.0.0	
fm.language.instruct_3b.urgency_classification.generic	14.2.0	
fm.language.instruct_3b.video_caption.generic	13.10000.0	
fm.language.instruct_3b.voice.generic_sparse	14.1.0	
fm.language.instruct_9m.text_event_extraction_classifier.base.generic	1.1.81607	113
fm.language.instruct_9m.text_event_extraction_classifier.tokenizer.generic	1.0.0	
gm.server.instruct_server_small.model_abuse_guardrail.generic	13.10006.0	
gm.server.instruct_server_small.safety_nsfw.generic	13.10008.0	
gm.server.instruct_server_v1.bullets_transform.generic	12.4.0	
gm.server.instruct_server_v1.concise_tone.generic	11.0.0	
gm.server.instruct_server_v1.fitness_workout_voice.generic	12.51.0	
gm.server.instruct_server_v1.friendly_tone.generic	12.0.0	
gm.server.instruct_server_v1.generative_shortcuts.generic	12.16.0	
gm.server.instruct_server_v1.llm_siri_pcc.generic	11.33.0	
gm.server.instruct_server_v1.lw_planner_v1.generic	12.2.0	
gm.server.instruct_server_v1.magic_rewrite.generic	11.0.0	
gm.server.instruct_server_v1.mail_reply_long_form_basic.generic	12.0.0	
gm.server.instruct_server_v1.mail_reply_long_form_rewrite.generic	12.0.0	
gm.server.instruct_server_v1.mail_reply_qa.generic	12.0.0	
gm.server.instruct_server_v1.open_ended_schema.generic	12.10.0	
gm.server.instruct_server_v1.open_ended_tone.generic	12.24.0	
gm.server.instruct_server_v1.open_ended_tone_query_response.generic	12.0.0	
gm.server.instruct_server_v1.open_ended_tone_query_response_v2.generic	12.0.0	
gm.server.instruct_server_v1.photos_memories_asset_curation_v2.generic	12.0.0	
gm.server.instruct_server_v1.photos_memories_global_traits_v2.generic	12.0.0	
gm.server.instruct_server_v1.photos_memories_global_traits_v3.generic	12.0.0	
gm.server.instruct_server_v1.photos_memories_query_understanding_v3.generic	12.4.0	
gm.server.instruct_server_v1.photos_memories_storyteller_v2.generic	12.0.0	
gm.server.instruct_server_v1.professional_tone.generic	12.0.0	
gm.server.instruct_server_v1.reminders_auto_categorized_list.generic	12.0.0	
gm.server.instruct_server_v1.shortcuts_ask_afm_action.generic	11.19.0	
gm.server.instruct_server_v1.shortcuts_ask_afm_action_v2.generic	12.54.0	
gm.server.instruct_server_v1.stx_multimodal.generic	12.20.0	
gm.server.instruct_server_v1.tables_transform.generic	12.0.0	
gm.server.instruct_server_v1.takeaways_transform.generic	12.0.0	
gm.server.instruct_server_v1.text_summarizer.generic	12.0.0	
gm.translate_fm.machine_translation.alignment.generic	14.1.0	

continued on next page

Table 7, continued

Bundle name	Version	MB
gm.translate_fm.machine_translation.phrasebook.generic	14.1.0	
mm_guard.output.configuration.generic	0.0.2	
safety.output.configuration.generic	0.1.6	

Appendix A. odix IR Grammar

The odix IR is a length-prefixed section stream. Strings and tensor *types* are interned once in a symbol pool and shared by reference. A reference is a 32-bit word whose high byte is 0xFF, encoding a self-relative signed offset $t = p + \text{int32}(w)$ from the word position p. Attribute records in the location/debug tree take the form ⟨key-ref⟩ ⟨len:u32⟩ ⟨payload⟩ with interned keys name, location_type, named_child_loc, op_id, line, filename, column, metadata. Because types are interned, a dimension such as 1536 occurs only ≈26 times across the entire 32 MB IR. The compile-time name-to-offset table (ifp_constant_table_ifp1_r48.json) is not shipped, but it is not needed: its runtime equivalent is the metadata.bin BBBB swizzler (Appendix B), and the MPSGraph bytecode location tree tags each ifp_constant with its layer/role, so a named export is produced without it (§6).

Appendix B. BBBB Container Layout

Both ifp_rasterized_weights.bin and the swizzler metadata.bin share the BBBB magic. The rasterized header is BBBB, version 0x017c0100, tag 0x0600002c, count, then a u64 segment-offset table. The swizzler body is a sequence of 24-byte records [idx][flag][off_a][off_b][marker=0x0600beef][0] with off_b = off_a + 0x20, i.e. a scatter map of 32-byte Apple Neural Engine tiles. Region spans: scales [0x60,0x1054000); 4-bit indices [0x1078000,0x125e80000).

References

[1] Apple, *Introducing Apple's On-Device and Server Foundation Models*, Apple Machine Learning Research, 2024.

[2] S. Mehta et al., *OpenELM: An Efficient Language Model Family with Open Training and Inference Framework*, arXiv:2404.14619, 2024.

[3] W. Fedus, B. Zoph, N. Shazeer, *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*, JMLR, 2022.

[4] N. Shazeer et al., *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*, ICLR, 2017.

[5] J. Ainslie et al., *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*, EMNLP, 2023.

[6] J. Su et al., *RoFormer: Enhanced Transformer with Rotary Position Embedding*, arXiv:2104.09864, 2021.

[7] B. Zhang, R. Sennrich, *Root Mean Square Layer Normalization*, NeurIPS, 2019.

[8] N. Shazeer, *GLU Variants Improve Transformer*, arXiv:2002.05202, 2020.

[9] G. Gerganov et al., *GGUF / llama.cpp: A file format and runtime for quantized transformer inference*, https://github.com/ggml-org/llama.cpp, 2023–.