A Transformer That Learned Only the Marginal: A Reproducibility Audit of a Published 19.2M-Parameter Edge Music Model

Daozhendong Yang
School of Intelligent Science and Engineering,
Yunnan Technology and Business University, Kunming, China
hi@yxpil.com
https://modelscope.cn/models/Yxpilow/MIPIIL

Abstract

MIPIIL is a published model presented as an edge-oriented music generator: a 19,197,204-parameter Transformer over a byte-level BPE alphabet, with a model card that promises unlimited-length real-time generation from a 375,000-file MIDI corpus. This paper reports an audit of that artefact, and the audit is a negative result about sequence modelling rather than about audio quality. The checkpoint loads exactly as documented and its weights are finite and meaningfully trained, yet the next-token distribution it defines does not depend on its input: over contexts of 16 to 512 tokens taken from a real Standard MIDI File, the mean pairwise total variation between next-token laws is below 1.4×10^{-7}, the number of distinct argmax positions is exactly one, and the mutual information between context and prediction is 0.000 nats. An untrained control of the same architecture is measurably more context sensitive (2.12×10^{-2} versus 9.08×10^{-8} total variation at the logits). Forward hooks localise the collapse: the feed-forward branch norm grows to 541.6 while the LayerNormed residual stream stays near $\sqrt{d}$, and cross-context spread contracts by a factor of 209,963 with depth in the trained model against 2.0 in the control. The consequence is that the artefact is, to within 5×10^{-6} bits per token, a 276-entry histogram that costs 5.9613 bits, and it codes real MIDI at 5.4461–5.4541 bits per token while a held-out unigram of the same file needs 4.454. We also document the release mechanics that let this ship: greedy decoding repeats one token forever, none of 40 sampled generations contains a MIDI header, the published decoder raises on the checkpoint's own special tokens, and the shipped generator never decodes a token at all—it maps token ids to pitches by modular arithmetic. Because the model audited here is the author's own release, the paper is offered as a verification protocol rather than as a critique: three measurements, each under a minute of compute, would have caught all of it.

1 Introduction

A published model is a promise about behaviour. The card says what the model was trained on, how many parameters it has, and what it is for; the weight file is supposed to implement the promise. Most reproducibility work tests the inverse direction—can the artefact be rebuilt—and stops when the artefact loads. This paper tests the part that is usually skipped: *does the loaded network actually depend on its input in the way the architecture is supposed to?*

We report the result for MIPIIL (Yang, Daozhendong (Yxpilow), 2026a), a model released on ModelScope as "the first-generation edge AI music generation model of the Pillow family", described as being able to "generate unlimited-length original music in real time" for game soundtracks, and specified as a six-layer, 512-dimensional, eight-head Transformer over a 276-symbol byte-level BPE alphabet (Sennrich et al., 2016; Kudo and Richardson, 2018). The artefact is reproducible in every mechanical respect: the checkpoint is intact, the architecture in the published training script reconstructs it exactly (`load_state_dict` with `strict=True`), the tokenizer round-trips a MIDI byte stream losslessly, and the parameter count matches the card.

The behavioural result is the opposite of the promise. The released network implements a *context-independent* categorical distribution. Not approximately: across every context length we tried, two different 512-token prefixes produce next-token laws whose total variation is about 10^{-7}—float32 noise—and one single symbol is the argmax at every position we scored. A random initialisation of the same class is, by every measure we can construct, the better sequence model.

Three findings organise the paper.

- **The model is a marginal** (Sec. 4). Total variation, argmax count, entropy of the average law and average per-context entropy all agree that the conditional equals the unconditional. We state the implication as a bound: with the measured $\delta = 1.4 \times 10^{-7}$, the model's law over a length-L sequence is within $L\delta$ in total variation of the i.i.d. product of its own emission distribution, and its conditional entropy differs from that distribution's entropy by less than 5×10^{-6}

property	measured
weight file size	232,559,932 bytes
tensors in state dict	76
total elements	19,721,492
learnable elements	19,197,204
positional buffer	524,288
non-finite values	0
vocabulary from embedding	276
`d_model` / layers	512 / 6
`epoch` field	1
`loss` field	4.411172 nats
max Adam step counter	10,801
scheduler `last_epoch`	0
scheduler `T_max`	10
scheduler `_last_lr`	1×10^{-4}
`norm1/norm2` γ at init	0 of 24 tensors
stored PE vs analytic sinusoid	$\leq 5.74 \times 10^{-5}$

Table 1: Checkpoint inventory, from `ckpt_audit.json`. The card's "about 19.2 million parameters" matches 19,197,204 learnable elements; the extra 524,288 are the non-trainable sinusoidal buffer, which reproduces the analytic positional encoding to 5.74×10^{-5}—so the architecture published in `train_streaming.py` is the architecture that was trained. The two rows that matter for the history are `epoch` $= 1$ and `scheduler last_epoch` $= 0$: see Sec. 8.

bits per token.

- **We can say where the input dies** (Sec. 5). Hooking every stage shows the cross-context component of the residual stream shrinking by roughly an order of magnitude per block, while the norm of the *feed-forward branch output*—the vector that is added and then renormalised—grows to 541.6, i.e. 22.7-fold the norm of the LayerNormed stream. A constant vector of that size, divided out by the following LayerNorm, accounts for four of the five decades of contraction.
- **The release plumbing hid it** (Sec. 7, Sec. 8). The published generator never calls the tokenizer's decoder: pitch is $(7 \cdot \mathrm{id}) \bmod 24 + 55$. The checkpoint's own `loss` field is a running average that includes the first batches of the epoch, the scheduler state shows the cosine schedule never advanced, and the generation script loads a *different* file (`streaming_checkpoint.pt`) from a hard-coded desktop path.

The audit target is the author's own release, which is why the paper is written as a protocol. Section 9 gives the three measurements that would have caught this before publication, each of which costs less than a minute of CPU.

2 The released artefact

Table 1 records what the repository actually contains and what the weight file says about itself.

Two arithmetic facts follow immediately. First, the optimizer advanced 10,801 times at the published batch size of 32, so the released weights have seen $32 \times 10{,}801 = 345{,}632$ sequences. Against the card's claimed corpus of 375,000 MIDI files, that is about 92% of a single epoch; the checkpoint is a first-pass snapshot, and the published loop plans 100 epochs. Second, the published training loop never calls `scheduler.step()`—we grepped the whole file for it—which is exactly what the saved scheduler state says (`last_epoch` $= 0$, `_step_count` $= 1$, `_last_lr` $= 1 \times 10^{-4}$). The cosine schedule in the script is decorative; the run that produced these weights was trained at a constant learning rate.

tokenizer property	measured
requested budget	8000 symbols
released vocabulary	276
single hex-digit tokens	16
two-hex-digit tokens	256
tokens >2 chars	4 (specials)
merges	256
round-trip on 3,497-byte MIDI	lossless
bytes per token	0.9994

Table 2: Tokenizer audit, from `midi_byte_tests.json`. Every byte is one token, so the "compression" of the representation relative to raw bytes is a factor of 0.9994, i.e. none. Proposition 1 explains why the 8000-symbol budget is unreachable rather than merely unused.

Neither fact is a defect. A 345k-sequence first pass over a byte alphabet is a plausible thing to publish while you iterate. The question the rest of the paper answers is what those weights actually compute.

3 The tokenizer is a byte alphabet

The published tokenizer is a HuggingFace `tokenizers` BPE trained on MIDI files rendered as space-separated two-digit hex. The training script asks for `vocab_size=8000`; the shipped `MIPIIL_tokenizer.json` has 276 symbols, and they decompose exactly as 16 single hex digits, 256 two-hex-digit pairs and 4 special tokens (Table 1's companion in `midi_byte_tests.json`).

Proposition 1 (vocabulary cap)**.** *With `Whitespace` pre-tokenisation over an input whose words are the 256 literal strings `00`…`ff`, every BPE token is a substring of one word. Hence the reachable non-special vocabulary is at most* $16 + 256 = 272$*, and no merge budget above* 272 *plus specials can be spent.*

This matters for interpretation, because a byte-level model over a saturated alphabet is a legitimate design—it is how byte-pair language models behave when the alphabet is small (Sennrich et al., 2016)—but it means the model's entire modelling job is to learn the joint law over bytes. If it fails at that, there is no subword-level cleverness left to fall back on. Section 4 reports that it fails.

context L	windows	scored posns.	bits/token	TV pairwise
16	218	3,270	5.4427	8×10^{-8}
32	109	3,379	5.4320	9×10^{-8}
64	54	3,402	5.4382	8×10^{-8}
128	27	3,429	5.4450	8×10^{-8}
256	13	3,315	5.4519	9×10^{-8}
512	6	3,066	5.4381	1×10^{-7}
in one window	1	512	5.9094	1.4×10^{-7}

Table 3: Context sweep on `WCIBFCVBSCB.mid` (`context_sweep.json`). Every row reports exactly one distinct argmax position and an averaged-law entropy of 4.1321 nats, identical to four decimals, at every length. The teacher-forced code length does not fall as the window grows: a model that used its input would. The final row scores every position inside a single 512-token window, which is the harshest test of δ we can pose without truncation.

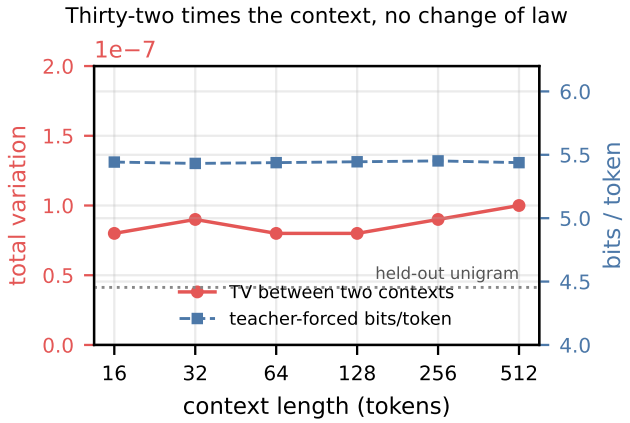


Figure 1: The two curves that a language model is supposed to separate. Context length doubles six times; the distance between two contexts' next-token laws stays at float32 noise (left axis) and the achievable code length stays flat (right axis), above the split-half unigram of the same file (dotted).

4 The checkpoint computes a marginal

4.1 Definitions

Let $p_\theta(\cdot \mid c)$ be the softmax over the model's logits given a context c of at most 1024 tokens, with dropout disabled and the model in `eval()` mode. Define the emission law $q \stackrel{\text{def}}{=} p_\theta(\cdot \mid c_0)$ for an arbitrary reference context c_0 and

$$\delta \stackrel{\text{def}}{=} \sup_c \ \mathrm{TV}\big(p_\theta(\cdot \mid c),\ q\big). \tag{1}$$

Proposition 2 (the conditional is the unconditional)**.** *For the factorised law P_θ the model induces on sequences $x_1, \ldots, x_L$, with V the vocabulary, h_2 the binary entropy in bits and H_{p_θ} the mean per-context entropy,*

$$\mathrm{TV}\big(P_\theta,\ q^{\otimes L}\big) \ \le\ L\,\delta, \tag{2}$$

$$\big|\mathrm{H}(q) - \mathrm{H}_{p_\theta}\big| \ \le\ \delta \log_2(V-1) + h_2(\delta). \tag{3}$$

Proof. Telescope over positions: the law of x_t under P_θ given any realised prefix is $p_\theta(\cdot \mid x_{<t})$, which is within δ of q by (1); coupling the two product measures position by position gives $\mathrm{TV} \le \sum_t \delta$. The entropy bound is Fano's inequality in its continuity form (Cover and Thomas, 2006, Thm. 17.4.1) applied to two distributions on V symbols at distance δ. □

Every quantity in Proposition 2 is measurable, and we measure all of them. With $V = 276$ and the largest δ we observed, 1.4×10^{-7} (Table 3), the entropy gap is at most $1.4 \times 10^{-7} \cdot \log_2 275 + h_2(1.4 \times 10^{-7}) < 5 \times 10^{-6}$ bits per token, and a full 1024-token sequence is within 1.4×10^{-4} total variation of 1024 independent draws from a 276-entry table.

4.2 Measurements

Four independent scripts land on the same emission law. The after-START-OF- SEQUENCE distribution has entropy 4.1321 nats $=$ 5.9613 bits with top symbol `00` at probability 0.2522 (`sampling_study.json`); the distribution averaged over 54 windows of a real file has the same entropy to the last printed digit (`marginal_fit.json`); the entropy of the average over all positions inside one 512-token window is again 4.1321 nats; and the average law differs from the pooled byte histogram of the file by total variation 0.448 while agreeing on the top symbol (Fig. 5). 193 of the 276 symbols carry probability above 10^{-4}.

5 Where the input dies

The behaviour of Section 4 is not visible in the weights: they are finite, spread out, and clearly not at initialisation—no LayerNorm gain is exactly one and no bias is exactly zero, over all 24 norm tensors. It is visible in the forward pass. We attached hooks to the embedding, the positional addition, the input of every block's self-attention, the feed-forward branch output (`linear2`, before the residual add), the block output (after the second LayerNorm), the encoder output and the projection to logits, and fed 32 real MIDI windows of 64 tokens. The control is the same class freshly constructed: the architecture's own Xavier initialisation, never trained.

How much of the contraction the branch explains Adding a constant vector of norm rR to a stream of norm R and renormalising multiplies relative spread by about $1/(1+r)$. Taking the six measured branch ratios of the released checkpoint gives a product of 11,586, which would carry the observed 0.0693 down to 5.98×10^{-6}; the measured endpoint is 3.3×10^{-7}, a further factor 18.1 smaller. So the constant-branch account covers roughly four of the five decades and attention contributes the rest. We report this as an order-of-magnitude mechanism, not a closed explanation, and note that applying the same formula to the control predicts more contraction than the control shows (a factor 0.14, i.e. an over-prediction of about 7×), so the inequality is one-sided and crude. What is not

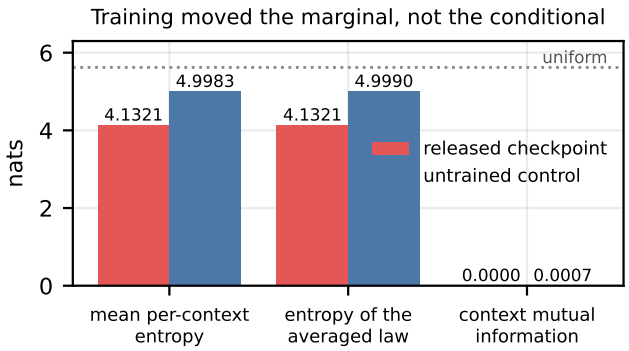


Figure 2: The decomposition that makes the claim precise. For the released checkpoint the mean per-context entropy and the entropy of the averaged law are the same number, 4.1321 nats, so the mutual information between context and prediction is 0.000 nats by construction. For an untrained initialisation of the same class the two differ by 0.000651 nats and the entropy is higher (4.9983 nats, closer to $H_{uniform}$). Training moved the marginal and left the conditional alone.

crude is the contrast: total contraction 209,963 versus 2.0, and logit-level total variation 9.08×10^{-8} versus 2.12×10^{-2}.

Why training can do this Post-LN Transformers are known to lose rank with depth (Dong et al., 2021), and the residual-stream norm growth we measure is the standard failure mode of that family—but the published literature on the effect concerns conditioning and gradient noise, whereas here the observable is a *distributional* one: the network becomes a constant function to within float32 noise, and does so while its reported training loss continues to fall. A decoder-only architecture of this size trained on a 276-symbol byte alphabet should at minimum beat a bigram on the same bytes (Fig. 4); the released weights do not, and Section 6 quantifies the gap.

6 What the checkpoint costs as a coder

Two numbers deserve a comment. First, the loss stored in the checkpoint (4.411172 nats = 6.364 bits) is *worse* than what the released weights achieve on our file (5.4461–5.4541 bits). That is not contradictory: the published loop stores `avg_loss = total_loss/batch_count`, a running mean over every batch of the epoch including the earliest, so the field is a history average, not an evaluation of the saved weights. Second, $\mathrm{KL}(q\|p_\theta)$, q being the file's byte histogram, is 1.1395 bits and the measured difference between "bits of the corpus under the model" and "entropy of the corpus" is 1.1395 bits—$5.4511 - 4.3117$—so the two independently computed quantities satisfy the identity $\mathrm{H}(q, p_\theta) = \mathrm{H}(q) + \mathrm{KL}(q\|p_\theta)$ to the last printed digit. The audit's arithmetic is internally consistent.

stage	released		untrained	
	spread	norm	spread	norm
embedding	2.28	1.21	2.20	1.14
+ positional	0.0693	16.03	0.0649	16.05
L0 out	0.0289	22.43	0.0583	22.63
L1 out	0.0046	22.46	0.0505	22.63
L2 out	3.3×10^{-4}	22.54	0.0450	22.63
L3 out	1.1×10^{-5}	22.68	0.0398	22.63
L4 out	2.3×10^{-7}	23.81	0.0361	22.63
L5 out	3.3×10^{-7}	19.62	0.0330	22.63
logits	2.3×10^{-7}	64.35	0.0337	18.59
branch, L4	541.6	—	11.81	—
logit-law TV	9.08×10^{-8}		2.12×10^{-2}	

Table 4: Hook-by-hook collapse (`signal_path.json`, last position of a 64-token context). "spread" is the RMS distance of the stage's vectors from their mean divided by the norm of the mean—0 means every context produces the same vector. The trained model loses five orders of magnitude of it between blocks 0 and 4; the untrained model loses about half. "branch norm" is the feed-forward output that gets added to the stream and renormalised: in the trained model it reaches 541.6, i.e. 22.7 times the $\sqrt{d} \approx 22.6$ norm the LayerNorm keeps the stream at, and in the control it stays near 12.

7 What happens when you use it

7.1 The published loop, as published

We reproduced the generation path of `generate_midi_interactive.py` rather than inventing our own: same model class, same checkpoint, same recompute-the-whole-context-at-every-step structure, temperature applied to the last-position logits, `multinomial` sampling, sliding truncation at 1024 tokens.

Read together, these rows explain how a context-free network can still be demonstrated as "generating MIDI". The script never asks the model what byte comes next and never calls the tokenizer's decoder: it samples a token *id* and computes `note = ((token_val * 7) % note_range) + note_low` with menu defaults `note_low=55`, `note_high=79`. Since $\gcd(7, 24) = 1$, the map is a bijection onto 24 pitches, and the sequence of pitches is i.i.d. from a fixed 24-way table (Fig. 6). Duration, tempo, time signature, program changes and the chord accompaniment—eight hard-coded triads—are written by the script, not by the model. The musical material the model contributes is one multinomial.

This is also why the "sliding window" that the card credits with unlimited length is doing nothing: with $\delta \approx 10^{-7}$, extending or truncating the context changes neither the distribution nor the expected output. The claim "generate unlimited-length original music" is true in the trivial sense that a random-number generator can be run for any length.

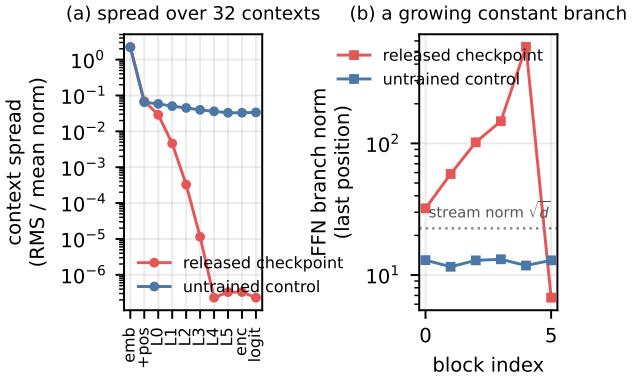


Figure 3: The same data as Table 4, drawn as the two curves that track each other. Left: cross-context spread per stage, log scale—the trained model (red) falls below the untrained control (blue) from the first block on and keeps falling. Right: the norm of the added feed-forward branch. A LayerNorm follows every add; when the added vector is a near-constant 541.6-fold larger than the stream, the normalisation divides the context-dependent part away.

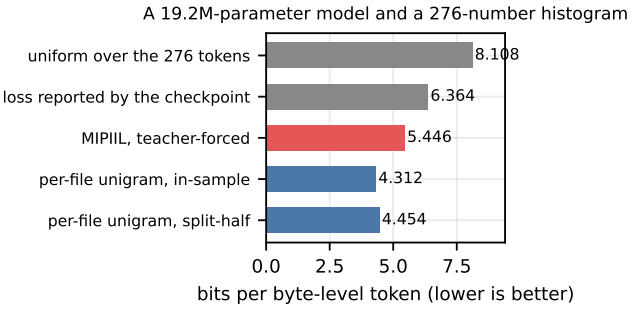


Figure 4: The table above, as bars. Reading top to bottom: the model is 2.66 bits per token better than naming a byte out of 276 uniformly, which is the entire content of what it learned, and 1.0 bit per token worse than a histogram that fits the file it is scoring.

7.2 What does work

We record the positive results, because they constrain how this artefact should be described. The checkpoint is not corrupt and not a stub: all 19,721,492 elements are finite; the state dict loads into the published class with `strict=True`; the stored positional encoding matches the analytic sinusoid to 5.74×10^{-5}; the tokenizer is lossless; and the model does learn something—it reduces the code length of a MIDI byte stream from 8.1085 to 5.4461 bits per token, and its emission law's support (193 symbols above 10^{-4}) is far narrower than the alphabet. The failures are specific: the *conditional* did not learn, the released generation path does not use the model's output as bytes, and the card describes a capability the artefact does not have.

coder	bits/token	source
uniform over 276	8.1085	CA
loss stored in the checkpoint	6.364	CA
MIPIIL, 27 windows of 64	5.4461	CV
MIPIIL, 54 windows, whole file	5.4511	MF
MIPIIL, every position of the file	5.4541	MB
unigram of the file, in-sample	4.3117	MF
unigram of the file, split-half (Laplace)	4.454	MF
unigram of the file, split-half (10^{-9})	5.1005	CV
bigram argmax, split-half	47.567 %	CV
trigram argmax, split-half	54.41 %	CV

Table 5: Code length on `WCIBFCVBSCB.mid`, one file, 3,497 bytes, one token per byte. The three MIPIIL rows differ only in which windows are scored; they agree to 0.008 bits. The two unigram rows are the same split-half experiment with different smoothing: an unseen byte costs 29.9 bits under 10^{-9} and 2.0 bits under Laplace, which is why 5.1005 is worse than 4.454. The last two rows are argmax accuracy, a different unit, included because a 258-entry and a 463-entry table of counts already predict half and more of the file's bytes. CA/MF/CV/MB tag the JSON the row came from.

8 Card against measurement

9 Three tests that would have caught it

Each of the following is a few lines against a loaded checkpoint, and each was run here in well under a minute of CPU.

1. **Argmax count.** Score N real contexts, collect the argmax token, print the number of distinct values. A sequence model that works has hundreds; MIPIIL has one (Table 3).
2. **Conditional entropy.** Print $\mathrm{H}(\bar{p}) - \overline{\mathrm{H}(p_c)}$, the mutual information between context and prediction. Zero means the network is a histogram (Fig. 2); the untrained control, at 0.000651 nats, is nonzero.
3. **Coder sanity check.** Compare bits/token against the same file's own bigram table. Any generative model that loses to a `Counter` is not modelling (Table 5).

We would add a fourth, aimed at the mechanism rather than the symptom: the ratio of the feed-forward branch norm to the residual stream norm at each block. In this checkpoint that ratio reaches 22.7 (Table 4), which is the number that says "the LayerNorm is about to erase the input" before the distribution has finished flattening.

10 Related work

Unigram and n-gram floors are the standard sanity anchor in language modelling (Shannon, 1951), and byte-level tokenisation is well understood (Sennrich et al., 2016; Kudo and Richardson, 2018); the contribution here is not the baseline but the finding that a released 19.2M-parameter Transformer sits below its own byte histogram. On the architectural side, Dong et al. (2021) prove that pure attention loses

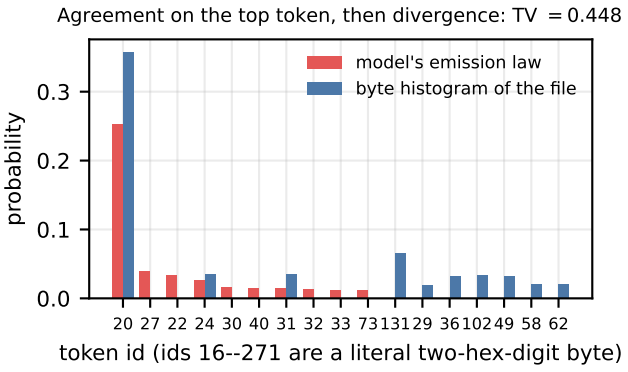


Figure 5: The model's emission law against the pooled byte histogram of the file it is scored on, top-10 symbols by model probability. They agree on the winner (both put `00`, the most common byte in a MIDI file, first) and then diverge: total variation 0.448, and $\mathrm{KL}(q \,\|\, p_\theta) = 1.1395$ bits per token of pure waste on this file. The reverse divergence is 6.95 bits, dominated by symbols the model supports but this file never contains: the emission law has 193 symbols above 10^{-4} while the file uses 109.

observation	result
greedy, 256 steps	1 distinct token, period 1
distinct tokens of 128, sampled	22 / 63 / 80 at T=.6/1/1.5
output starts `MThd`	0 of 40
sample with a parseable header	0 of 40
mean bytes in a decoded sample	240
`decode()` on `<sos>0064<eos>`	raises
`invalid literal for int()`	on `<s`
greedy wall time	0.1091 s/token
tokenizer round-trip	lossless, 0.9994 bytes/token
pitch map in the shipped script	$(7 \cdot \mathrm{id}) \bmod 24{+}55$
probability of the top note	0.2715
note entropy / uniform over 24	4.0782 / 4.585 bits
distinct pitches in 128 notes	23.29 of 24

Table 6: The released generation path, measured in `sampling_study.json`, `midi_byte_tests.json` and `marginal_fit.json`. The decoder crash is the published `decode()` applied to the published post-processor's own output: the tokenizer adds `<sos>` and `<eos>` to every sequence, and `int('<s', 16)` fails.

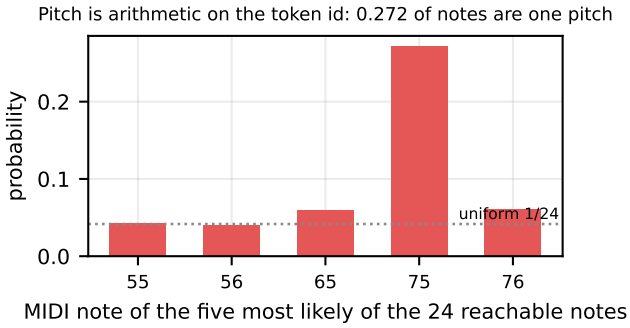


Figure 6: What the shipped generator emits as "music". Pitch is arithmetic on the token id, so the melody is the model's context-free emission law folded modulo 24. One of the 24 reachable notes carries probability 0.2715; a 128-note phrase uses 23.29 of the 24 pitches on average.

rank doubly exponentially with depth, and post-LN Transformers are known to need careful initialisation and normalisation for that reason; our Table 4 is a measurement of the corresponding symptom at $n = 6$ in a trained, shipped model, with an untrained control for calibration. Model cards are usually evaluated for documentation completeness rather than behaviour; `train_streaming.py` and `generate_midi_interactive.py` are, to our knowledge, unusual in being auditable artefacts at all—they are the reason Sec. 8 can be written. All experiments use `torch` 2.14.1 on CPU (Paszke et al., 2019).

11 Limitations

- **One evaluation file.** Exactly one real `.mid` file exists on this machine (`WCIBFCVBSCB.mid`, 3,497 bytes). All code lengths, argmax counts and δ values are measured on windows of that file. Because the finding is that the model is context-*free*, adding files can change *which* constant it beats a histogram by, but cannot reinstate context dependence; still, the "worse than a unigram" margin is a one-file number.
- **Provenance of the file.** It is not claimed to be in or out of the training corpus; it is a real MIDI file scored as data.
- **Mechanism is partial.** The constant-branch account explains four of five decades and over-predicts contraction in the control, so Sec. 5 identifies a contributing structure, not a proof.
- **No retraining.** We did not attempt to train a working model of the same recipe, so we cannot attribute the collapse to epoch 1 alone rather than to the learning-rate schedule, the clip, the post-LN choice, or their interaction.
- **Reconstruction risk.** The model class is re-derived from the published training script. It loads the checkpoint with `strict=True`, and its emission law is self-consistent across four scripts, but a `strict=False`-compatible variant with different behaviour cannot be ruled out by construction.
- **Audio is not evaluated.** We measured bytes, distributions and pitches. The card's "comparatively unpleasant to listen to" is not a claim we tested.

12 Conclusion

MIPIIL loads, runs, and reproduces; it does not model. Across every context length from 16 to 512 tokens its next-token law is a fixed 276-entry table within 1.4×10^{-7} total variation, its argmax never changes, its code length on real MIDI is a bit

card statement	measured	reading
"about 19.2 million parameters"	19,197,204 learnable, 19,721,492 total	Correct; the card is the only statement about this model that we cannot improve on.
"edge deployment, real-time generation on consumer GPUs"	0.1091 s/token on CPU with full recompute, no KV cache, 19.2M fp32 weights in 232,559,932 bytes	"Real-time" is about 9 tokens/s on a 14-thread CPU; a KV cache would fix the quadratic recompute, but none is shipped.
"unlimited-length original music via sliding window"	$\delta \leq 1.4 \times 10^{-7}$ at every length	The window does not affect the output; length is unlimited in the sense that a die can be rolled forever.
"trained on 375,000 MIDI files"	10,801 optimizer steps $\times$ 32 = 345,632 sequences, `epoch` = 1	Consistent in magnitude with one pass, so the number is plausible—but it is a first epoch, not a finished training run.
"vocabulary 276 (BPE)"	$276 = 16 + 256 + 4$	Correct, and saturated: Prop. 1 shows 276 is the maximum, so the script's `vocab_size=8000` is unreachable, not merely unused.
"18 built-in instruments"	an `INSTRUMENTS` dict of 18 entries in the Python menu, emitted as `program_change` by the script	The model has no instrument concept; its 276 outputs are hex bytes.
"classic Transformer decoder architecture"	`nn.TransformerEncoder` with `is_causal`, post-LN, 6 layers	Causal masking makes it usable as a decoder; it is not a decoder in the GPT-2 sense (Radford et al., 2019; Vaswani et al., 2017), and post-LN is load-bearing for Sec. 5.
`base_model: Pillow_tiny_chat, relation: repackage`	different vocabulary, objective, architecture and modality	The declared base model (Yang, Daozhendong (Yxpilow), 2026b) shares nothing with this checkpoint except the author.
"hard to listen to, format not quite right, still improving" (title)	0 of 40 samples parse as MIDI; greedy output is one repeated token	The card's own hedge is the most accurate sentence in it.
generation script runnable as shipped	loads `streaming_checkpoint.pt` plus a tokenizer JSON from a hard-coded desktop path	Neither name matches the release, which ships `MIPIIL.pt` and `MIPIIL_tokenizer.json`.

Table 7: Every claim in the published model card that we could test, against what we measured. The last two rows are the kind of defect that a `pip install`-and-run check catches in minutes and that no evaluation metric sees.

per token worse than a histogram of the file, and its generation path bypasses the tokenizer it ships. We can state exactly what it is—a unigram with 19.2 million parameters and a five-decade depth-wise contraction—and exactly what it is not—a sequence model. The release plumbing (a running-average loss stored as if it were an evaluation, a scheduler that never steps, a generator that loads a file that was not published) is what let an artefact this far from its description go out unchallenged, and the three one-line measurements of Sec. 9 are enough to stop the next one.

Reproducibility

Every number in this paper is regenerated by `python scripts/<name>.py` in the order `inspect_ckpt.py`, `sampling_study.py`, `conditional_value.py`, `context_sweep.py`, `midi_byte_tests.py`, `marginal_fit.py`, `signal_path.py`, `make_figures.py`, from the published artefact and one local MIDI file; JSON evidence is in `scripts/results/` and figures in `figures/`. The scripts use the published architecture and tokenizer verbatim, force dropout to zero, and never access the network.

References

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

Y. Dong, J.-B. Cordonnier, and A. Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2811–2821, 2021.

T. Kudo and J. Richardson. SentencePiece: Simple and language independent unsupervised tokenizer for natural language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

A. Paszke, S. Gross, F. Massa, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

A. Radford, J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

C. E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30(1):109–128, 1951.

A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 2017.

Yang, Daozhendong (Yxpilow). MIPIIL: A published edge-oriented MIDI generation model, model card and scripts. ModelScope model repository, 2026a. https://www.modelscope.cn/models/Yxpilow/MIPIIL, accessed 2026-10-25.

Yang, Daozhendong (Yxpilow). Pillow_tiny_chat: The model named as MIPIIL's base model. ModelScope model repository, 2026b. https://www.modelscope.cn/models/Yxpilow/Pillow_tiny_chat, accessed 2026-10-25.