60-second takeaway

We tested four bidirectional language encoders under one fixed protocol: LFM2.5-Encoder-230M, LFM2.5-Encoder-350M, ModernBERT-base and ModernBERT-large.

On our RTX 3090 Ti at 8,192 tokens, the two LFM models were 2.69x and 2.85x faster than their ModernBERT comparison models. On CPU, the advantage appeared mainly at long context. The LFM models also performed well in our small sentiment and legal-document classification pilots.

These are promising deployment results, not a universal model ranking. We tested one machine, small task subsets, three seeds and one learning rate. We also tested base encoder classification, not retrieval embeddings.

Why compare these four models?

Many document workflows do not need a model to generate prose. They need a model to read a passage and produce a useful representation for classification, extraction, reranking or search.

That is the role of a bidirectional encoder. Unlike a decoder that reads only the text to its left when predicting the next token, an encoder can use words on both sides. This often makes it a better fit for understanding a complete document.

Liquid AI's LFM2.5 encoders are especially interesting because they were converted from compact decoder backbones. Liquid replaced causal attention with bidirectional attention, made the short convolutions non-causal and continued training with masked language modelling.

The practical claim is not simply that the models are small. It is that their architecture should remain efficient when a document grows to thousands of tokens.

We compared them with ModernBERT, a strong modern encoder family with native support for 8,192-token inputs.

Model	Parameters	Role in our comparison
LFM2.5-Encoder-230M	230M	Compact LFM model
LFM2.5-Encoder-350M	350M	Larger LFM model

© 2025 Instavar · CC-BY 4.0

ModernBERT-base	149M Primary established baseline
ModernBERT-large	395M Rough size match for LFM 350M

The question was narrow:

When all four models use the same machine and test protocol, what do we gain or lose by choosing LFM2.5 instead of ModernBERT?

What we tested

We ran five checks:

1. **Compatibility:** load every checkpoint and verify masked-language-model and hidden-state outputs.
2. **CPU inference:** measure forward-pass latency at 128, 512, 1,024, 2,048, 4,096 and 8,192 tokens.
3. **GPU inference:** repeat the length sweep on one NVIDIA RTX 3090 Ti with 24 GB VRAM.
4. **Memory:** record model storage, peak VRAM and memory release after unloading.
5. **Classification quality:** fine-tune every model with the same recipe on SST-2 sentiment and ECtHR-A legal cases across three fixed seeds.

The CPU runs used FP32, batch size one and eight threads. GPU runs used batch size one. The timing inputs were synthetic token IDs so tokenizer cost did not distort model-forward latency.

This makes the comparison controlled, but not complete. Real production throughput also depends on tokenization, batching, document length distribution and the serving runtime.

First finding: loading the right class matters

The most important compatibility finding appeared before any benchmark.

Loading an LFM encoder with generic `AutoModel` returned tensors with the expected shape, but the base model weights were newly initialized. A shape-only smoke test could therefore pass while the model was effectively random.

The correct path in our environment was:

```
from transformers import AutoModelForMaskedLM

masked_lm = AutoModelForMaskedLM.from_pretrained(model_id)
encoder = masked_lm.base_model
```

This is a general lesson for converted or custom checkpoints: a successful import and plausible tensor shape do not prove that pretrained weights loaded correctly. Read the loader warnings, inspect missing and unexpected keys, and run a semantic sanity check before benchmarking.

We excluded the invalid random-weight smoke from all results below.

CPU result: ModernBERT-base wins short inputs, LFM wins long ones

Median forward-pass latency in seconds:

Tokens	LFM 230M	LFM 350M	ModernBERT-base	ModernBERT-large
128	0.116	0.203	**0.087**	0.274
512	0.430	0.720	**0.335**	1.015
1,024	0.935	1.627	**0.793**	2.117
2,048	**1.967**	3.051	2.075	4.625
4,096	**4.572**	7.352	5.358	12.234
8,192	**11.484**	17.343	15.966	37.633

ModernBERT-base was fastest through 1,024 tokens. LFM 230M crossed it around 2,048 tokens and was 1.39x faster at 8,192 tokens.

The size-matched result was stronger. LFM 350M was 2.17x faster than ModernBERT-large at 8,192 tokens.

This supports the architectural efficiency claim in our environment, but not every published headline. Liquid reported about a 3.7x CPU advantage for LFM 230M over ModernBERT-base at 8,192 tokens. We measured 1.39x under our protocol.

That difference is not necessarily a contradiction. CPU model, software versions, thread settings, precision and timing method can move the result substantially. It is a reason to benchmark on the machine that will actually run the workload.

© 2025 Instavar · CC-BY 4.0

GPU result: both LFM models pulled ahead at long context

At 8,192 tokens on our RTX 3090 Ti:

Model	Median latency	Peak VRAM
LFM2.5-Encoder-230M	**0.1005 s**	1,802 MiB
LFM2.5-Encoder-350M	**0.1341 s**	2,514 MiB
ModernBERT-base	0.2707 s	**1,466 MiB**
ModernBERT-large	0.3824 s	2,543 MiB

LFM 230M was 2.69x faster than ModernBERT-base. LFM 350M was 2.85x faster than ModernBERT-large.

The memory result needs equal weight. ModernBERT-base used about 336 MiB less peak VRAM than LFM 230M. LFM 350M and ModernBERT-large were nearly equal in peak VRAM.

The practical interpretation is:

- choose ModernBERT-base when inputs are usually short and the smallest memory footprint matters;
- consider LFM 230M when long documents and latency matter more;
- consider LFM 350M when you want a larger encoder without ModernBERT-large's long-context latency on this GPU.

After each model was unloaded, PyTorch reported only 8 to 9 MiB allocated and 20 to 22 MiB reserved. After the benchmark process exited, the GPU returned to its 47 MiB display-only baseline. None of the four models needed to occupy VRAM permanently.

Model storage

The downloaded model snapshots occupied approximately:

Model	Local storage
LFM2.5-Encoder-230M	881 MiB
LFM2.5-Encoder-350M	1.4 GiB
ModernBERT-base	574 MiB

ModernBERT-large	1.5 GiB

ModernBERT-base remained the smallest artifact. The two larger models were close enough that storage alone would not decide between them.

General classification result

We fine-tuned all four models on the same fixed SST-2 subset: 2,000 training examples and a separate 500-example report set. Each model used context length 256, three epochs and seeds 45, 46 and 47.

Mean accuracy across the three seeds:

Model	SST-2 accuracy
LFM2.5-Encoder-230M	0.9053 +/- 0.0064
LFM2.5-Encoder-350M	0.9147 +/- 0.0081
ModernBERT-base	0.8693 +/- 0.0429
ModernBERT-large	**0.9240 +/- 0.0139**

ModernBERT-large had the highest mean. Both LFM models were competitive, and ModernBERT-base varied much more across the three runs.

Do not treat a 2,000-example subset as a replacement for the full benchmark. This pilot tests whether the models can learn under one shared recipe and whether the result is obviously unstable.

Long legal-document classification result

For a long-document task, we used ECtHR-A, a multilabel dataset derived from European Court of Human Rights cases. Every model used 500 training examples, 200 separate threshold-calibration examples and a separate 200-example report set. Context length was 2,048 tokens, with two epochs and the same three seeds.

Model	Micro F1	Macro F1
LFM2.5-Encoder-230M	0.5188 +/- 0.0254	0.2492
LFM2.5-Encoder-350M	**0.5661 +/- 0.0533**	**0.2865**
ModernBERT-base	0.4083 +/- 0.0090	0.1871
ModernBERT-large	0.4594 +/- 0.0494	0.2378

© 2025 Instavar · CC-BY 4.0

LFM 350M led this small legal pilot. LFM 230M also exceeded both ModernBERT models under the fixed recipe.

This is useful evidence for further legal-document evaluation. It is not evidence that LFM is universally better for law. ECtHR-A covers European human-rights cases, not contracts, Singapore judgments, discovery bundles or our own local legal corpus. Legal language and label structure vary sharply between those settings.

These are encoders, not ready-made embedding models

An encoder produces hidden states. An embedding system needs more than hidden states:

- a defined pooling method;
- task-specific training for similarity or retrieval;
- query and document instructions when the model expects them;
- normalization rules; and
- retrieval evaluation on the target corpus.

Our experiment used the models for masked-language compatibility and supervised classification. It did not compare vector search quality against models such as BAAI or Jina embeddings.

You can build an embedding model from an encoder, but mean-pooling an untrained base model is not a fair retrieval benchmark. Teams evaluating these checkpoints for search should fine-tune or use an officially trained embedding variant, then measure recall and ranking quality on their own queries.

Which model would we choose?

There is no single winner because the workloads differ.

Mostly short documents on CPU

Start with ModernBERT-base. It was fastest through 1,024 tokens and had the smallest model file and lowest 8,192-token peak GPU memory.

Long documents on CPU

Test LFM2.5-Encoder-230M. Its advantage appeared as context grew, reaching 1.39x over ModernBERT-base at 8,192 tokens on our CPU.

Long documents on a 24 GB NVIDIA GPU

Both LFM models were compelling. LFM 230M offered the best absolute latency, while LFM 350M combined strong long-context speed with the best legal-pilot result.

Highest quality without a domain-specific study

Do not decide from this pilot alone. ModernBERT-large led SST-2, while LFM 350M led ECtHR-A. That reversal is exactly why deployment choices should follow the real task rather than one combined score.

What this benchmark proved

- All four checkpoints can produce valid masked-language and hidden-state outputs when loaded correctly.
- LFM2.5's long-context speed advantage is real on our tested CPU and RTX 3090 Ti setup.
- The advantage grows with input length and is stronger against ModernBERT-large than against ModernBERT-base.
- All four models release almost all GPU memory after unloading in a short-lived process.
- LFM 350M is a credible candidate for a larger legal-document trial.

What it did not prove

- a universal speed ranking across CPUs, GPUs, runtimes or batch sizes;
- Liquid AI's exact published CPU ratio;
- superiority across classification tasks;
- retrieval or semantic-search quality;
- performance on our private legal documents;
- production throughput including tokenization and request batching; or
- statistical certainty from only three seeds and one learning rate.

The next useful test is not a larger generic leaderboard. It is a corpus-specific comparison using real document lengths, labels and retrieval questions from the

© 2025 Instavar · CC-BY 4.0

intended workflow.

References

- Liquid AI: LFM2.5 Encoders
- LFM2.5-Encoder-230M model
- LFM2.5-Encoder-350M model
- Liquid encoder evaluation repository
- Liquid encoder classification cookbook
- ModernBERT paper
- ModernBERT-base model
- ModernBERT-large model