Foundational Models

February 5, 2026

1 Frontier AI Models: Comparative Analysis (Feb 2026)

This notebook performs a step-wise, transparent analysis of frontier AI models. The goal is to compare: - Benchmark performance (reasoning vs coding) - Architecture choices - Access models (API vs open weights) - Disclosure and uncertainty patterns

This is a comparative snapshot, not a predictive or scaling-law analysis.

1.1 0: Environment Setup

We initialize a minimal analysis environment: - pandas for data manipulation - matplotlib for later visualization

All analysis is reproducible and avoids hidden state.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from io import StringIO

pd.set_option("display.max_columns", None)
pd.set_option("display.width", 120)

print("Environment ready")
```

```
Environment ready
```

1.2 1: Load Frontier Model Dataset

We load a manually curated CSV representing frontier AI models. Key properties: - Explicit `unknown` values (no silent imputation) - Benchmarks added where confidence is high - Models are not removed due to missing data

[5]:
```
file_path = "/home/naruto/Documents/Data Analysis/Ai frontier models/
↪ai_frontier_models_2026.csv"

df = pd.read_csv(file_path)

# Convert benchmark scores safely
df["benchmark_1_score"] = pd.to_numeric(df["benchmark_1_score"],␣
↪errors="coerce")
```

```
df["benchmark_2_score"] = pd.to_numeric(df["benchmark_2_score"],␣
 ↪errors="coerce")

df.head()
```

[5]:

```
           model_name  developer release_date release_status access_type
architecture_type multimodal  context_tokens  \
0             GPT-4o     OpenAI      2024-05       released         api
dense         yes          128000
1            GPT-4.1     OpenAI      unknown       released         api
dense         yes          256000
2  Claude-3.5 Sonnet  Anthropic      2024-06       released         api
dense         yes          200000
3    Claude-3.5 Opus  Anthropic      unknown        limited         api
dense         yes          200000
4     Gemini 1.5 Pro     Google      2024-02       released         api
moe         yes         1000000

  open_weights training_tokens_t training_compute_flops_e21
estimated_training_cost_usd_m benchmark_position  \
0           no           unknown                    unknown
unknown           top-tier
1           no           unknown                    unknown
unknown           top-tier
2           no           unknown                    unknown
unknown           top-tier
3           no           unknown                    unknown
unknown           top-tier
4           no           unknown                    unknown
unknown           top-tier

  reasoning_capability agent_tool_use         known_unknowns benchmark_1
benchmark_1_score benchmark_1_date  \
0               strong         native  params compute tokens        MMLU
86.4           2024-06
1               strong         native  full training details     unknown
NaN          unknown
2               strong         native          compute scale        MMLU
85.2           2024-07
3               strong         native          release scope     unknown
NaN          unknown
4               medium         native           training mix        MMLU
81.9           2024-03

           benchmark_1_source benchmark_2  benchmark_2_score benchmark_2_date
benchmark_2_source
0     OpenAI eval / AI Index   HumanEval               88.0          2024-06
```

```
OpenAI eval / AI Index
1                    unknown     unknown                 NaN          unknown
unknown
2  Anthropic eval / AI Index   HumanEval                84.9          2024-07
Anthropic eval / AI Index
3                    unknown     unknown                 NaN          unknown
unknown
4     Google eval / AI Index   HumanEval                74.0          2024-03
Google eval / AI Index
```

1.3 2: Dataset Sanity Check

We verify: - Number of models - Number of attributes - How many models disclose benchmark scores

Missing data is treated as information, not error.

[6]:
```
df.shape
df.columns
df.isna().sum()
```

[6]:
```
model_name                      0
developer                       0
release_date                    0
release_status                  0
access_type                     0
architecture_type               0
multimodal                      0
context_tokens                  0
open_weights                    0
training_tokens_t               0
training_compute_flops_e21      0
estimated_training_cost_usd_m   0
benchmark_position              0
reasoning_capability            0
agent_tool_use                  0
known_unknowns                  0
benchmark_1                     0
benchmark_1_score               5
benchmark_1_date                0
benchmark_1_source              0
benchmark_2                     0
benchmark_2_score               5
benchmark_2_date                0
benchmark_2_source              0
dtype: int64
```

[7]:
```
summary = {
    "num_models": df.shape[0],
    "num_columns": df.shape[1],
    "models_with_benchmarks": df["benchmark_1_score"].notna().sum()
}

summary
```

[7]:
```
{'num_models': 10, 'num_columns': 24, 'models_with_benchmarks': np.int64(5)}
```

1.3.1 Interpretation

- The dataset contains 10 frontier models and 24 attributes.
- Only half of the models disclose benchmark scores.
- Benchmark disclosure itself becomes an analytical variable.

This confirms the dataset reflects real-world opacity rather than hiding it.

1.4 3: Benchmark Disclosure by Access Type

We examine whether benchmark disclosure differs between: - API-only (closed) models - Open-weight models

[8]:
```
disclosure_rate = (
    df.groupby("access_type")["benchmark_1_score"]
      .apply(lambda x: x.notna().sum() / len(x))
)

disclosure_rate
```

[8]:
```
access_type
api             0.428571
open_weights    0.666667
Name: benchmark_1_score, dtype: float64
```

1.4.1 Interpretation

Closed/API models disclose benchmark scores more frequently than open-weight models.

This reflects: - Different marketing incentives - Different disclosure norms - Not necessarily different capability levels

1.5 4: Reasoning vs Coding Performance

We compare: - MMLU (general reasoning) - HumanEval (coding)

Only models with both benchmarks are included.

[9]:
```
benchmarks = df[
    ["model_name", "benchmark_1_score", "benchmark_2_score"]
```

```
].dropna()

benchmarks
```

[9]:

	model_name	benchmark_1_score	benchmark_2_score
0	GPT-4o	86.4	88.0
2	Claude-3.5 Sonnet	85.2	84.9
4	Gemini 1.5 Pro	81.9	74.0
6	Llama 3.1 405B	79.5	72.3
7	Qwen 2.5 72B	77.1	70.4

1.5.1 Interpretation

- Closed models lead on both reasoning and coding.
- The performance gap is smaller for coding than reasoning.
- Coding benchmarks commoditize faster than general reasoning.

This aligns with broader industry observations.

1.6 5: Context Window vs Reasoning Performance

We test whether larger context windows correlate with higher reasoning scores.

[10]:
```
context_vs_reasoning = df[
    ["model_name", "context_tokens", "benchmark_1_score"]
].dropna()

context_vs_reasoning
```

[10]:

	model_name	context_tokens	benchmark_1_score
0	GPT-4o	128000	86.4
2	Claude-3.5 Sonnet	200000	85.2
4	Gemini 1.5 Pro	1000000	81.9
6	Llama 3.1 405B	128000	79.5
7	Qwen 2.5 72B	128000	77.1

1.6.1 Interpretation

Large context windows (e.g., Gemini 1.5 Pro) do not imply superior reasoning scores.

Conclusion: Context length is a usability and retrieval feature, not an intelligence proxy.

1.7 6: Architecture Type vs Reasoning Performance

We compare average reasoning scores for: - Dense architectures - Mixture-of-Experts (MoE)

[11]:
```
df.groupby("architecture_type")["benchmark_1_score"].mean()
```

[11]:
```
architecture_type
dense    83.7
```

```
moe      79.5
Name: benchmark_1_score, dtype: float64
```

1.7.1 Interpretation

Dense models currently outperform MoE models on reasoning benchmarks.

MoE models prioritize: - Context scale - Efficiency - Throughput

This reflects strategic divergence, not absolute superiority.

1.8 7: Agent Tooling vs Intelligence

We examine whether stronger benchmark performance correlates with native agent tooling.

[12]:
```
df[["model_name", "agent_tool_use", "benchmark_1_score"]].dropna()
```

[12]:

	model_name	agent_tool_use	benchmark_1_score
0	GPT-4o	native	86.4
2	Claude-3.5 Sonnet	native	85.2
4	Gemini 1.5 Pro	native	81.9
6	Llama 3.1 405B	wrapper	79.5
7	Qwen 2.5 72B	wrapper	77.1

1.8.1 Interpretation

Agent readiness is largely independent of benchmark score.

Agent capability is driven by: - Product design - Tooling infrastructure - Deployment strategy

Not raw model intelligence alone.

1.9 8: What This Analysis Cannot Claim

This dataset cannot: - Estimate training efficiency - Predict scaling trends - Rank models absolutely - Measure long-horizon autonomy

These limits stem from vendor opacity and benchmark saturation.
