So , ICM is supposed to find good labels by making the model predict each label based on all the other labels it has assigned already . The issue , also mentioned in the paper as well , being the fact that only a limited amount can be added to the context window . This limits the model's mutual predictability.
The paper acknowledges this issue on page 10:
" It doesn't work with long inputs because we need to fit many dataset examples into the model's effective context window when calculating the scoring function, particularly for the mutual predictability term."

Other thing was that ,

1. I couldn't find token for the examples (question + claim + prompt formatting) .
2. Also how many tokens are used at different iterations ? (100, 500, 1000, 2000 , so on)
3. so I do not know how much each each example takes up , what is the total memory used and if they will hit the memory limit
4. This point above is based on the fact that the MP formula is based on previous examples.
5. For Claude how were the 6000 examples handled in the context window. What happens when D exceeds the context window?

Lets estimate for TruthfulQA:
• Each example includes: question (~50 tokens) + claim (~50 tokens) + label (5 tokens) + formatting (~20 tokens)
• Estimated tokens per example: ~125 tokens
• At 2560 examples: 125 × 2,560 = 320000 tokens
• Llama 3.1 context limit: 128000 tokens
• Overflow ratio: 2.5x over the limit

Even at 1024 examples, we'd need ~128,000 tokens - exactly at the limit. This suggests that for most of their training runs, the model cannot see the full labeled dataset D, and if the model can only see a subset of labels due to context limits, then, different context sampling strategies would yield different results , the scoring function no longer measures true mutual predictability across all examples.

Let's explore solutions :

Solution 1: Hierarchical Context Sampling

When the labeled dataset D exceeds the context window, sample a relevant subset of D instead of attempting to include all examples.

Method

The sampling strategy prioritizes three types of examples:

1. Recent Examples ; takes the most recent examples , sort of like D[-200]
2. Similar Examples , semantic similarities , using sentence embeddings to find related examples For example ; if the last question was " Is GWU in USA ? " , prioritize geological question data
3. Label Balance , just to prevent label bias

Now this presents us with new issues :

Advantages

Fits within context window limits
Recent examples preserve search progress
Relevant examples improve prediction quality
Can be added to existing ICM with minimal changes
Only one forward pass per prediction (same as original)

Disadvantages

1. Labels not in sampled context never influence this prediction despite label balance
2. Only conditions on subset, not all labels so its not really true mutual predictability
3. Adds complexity and dependency since this will require embedding model
4. Different runs might sample different subsets (unless random seed fixed)
5. Weakens the original "mutual predictability across all labels" claim.

this might work if as a quick solution honestly , or if resources are limited but here's another solution , a bit more resource extensive , just a little :)

SOLUTION 2 : Chunked Mutual Predictability

Instead of checking if a label is predictable from ALL examples at once, verify that it's consistently predictable across multiple OVERLAPPINGG chunks of D

The original claim was that a good label is predictable from all other labels , this would shift it a bit to a good label is predictable from any subset of labels.

SO:

If a label truly reflects a coherent concept, it should be predictable from any random sample of D, not just the complete dataset. A label that's only predictable when seeing all 2000 examples simultaneously suggests overfitting, not true conceptual understanding.

Now :

Advantages

Every label in D is examined
Checks consistency across all labels, just in chunks, so true mutual predictability
All examples appear in multiple chunks so no sampling bias
Same input always produces same output
Doesn't require embeddings or external tools
Sort if like if monkey see its predictable across all chunks then monkey see that it must be more strong

Disadvantages

K times more forward passes (where K ≈ 10-15 chunks)
Each iteration takes longer , slower convergence
Must store and manage multiple chunks
Different aggregation strategies give different results

Computational Cost Analysis

Original ICM (from Table 3):

- TruthfulQA: 2.5 forward passes per example
- GSM8K: 3.9 forward passes per example
- Alpaca: 2.0 forward passes per example

Chunked ICM:

- With K=15 chunks: 15 forward passes per evaluation
- Total: 15 × 2.5 = 37.5 forward passes per example (for TruthfulQA)

Comparison: Solution 1 vs Solution 2

Criterion	**Solution 1: Sampling**	**Solution 2: Chunking**
Theory	Weakens original claim	Maintains original claim
Checks all labels?	Only sampled subset	All labels, in chunks
Deterministic?	No (unless seed fixed)	Yes
Needs embeddings?	Yes	No
Computational cost	Low (1× original)	High (15× original)
Implementation	Moderate complexity	Simple
Sampling bias?	Yes	No

Criterion	Solution 1: Sampling	Solution 2: Chunking
Speed	Fast	Slow
Theoretical purity	Compromised	Preserved