Step-by-Step Implementation Plan: Process vs. Outcome Supervision for Trustworthy and Interpretable LLM Reasoning

Project Overview

Goal: Compare process-based supervision (step-by-step reasoning) vs. outcome-based supervision (final answers only) in terms of (a) factual reliability and (b) interpretability/auditability for multi-step reasoning tasks.

Timeline: 6 weeks
Scope: Evaluation study (no model training required)
Target: 150-200 math problems, 50-70 annotated in depth

Phase 1: Setup and Data Collection (Week 1-2)

Week 1: Environment Setup and Model Selection

Step 1.1: Set Up Development Environment

Tasks:

- [ ] Set up Python environment (Python 3.9+)
- [ ] Install core libraries:

```bash
pip install transformers torch datasets openai anthropic pandas numpy scikit-learn jupyter matplotlib seaborn --break-system
```

- [ ] Set up API access:
  - Option A: OpenAI API (GPT-4o-mini for cost-efficiency)
  - Option B: Anthropic API (Claude 3.5 Sonnet)
  - Option C: Open-weight model via HuggingFace (Qwen2.5-Math-7B-Instruct or Llama-3.1-8B-Instruct)
- [ ] Create project directory structure:

```
project/
├── data/
│   ├── raw/
│   ├── processed/
│   └── annotations/
├── outputs/
├── scripts/
├── notebooks/
└── results/
```

Resources:

- HuggingFace Transformers documentation
- API documentation (OpenAI/Anthropic)

Deliverable: Working development environment with API access confirmed

---

Step 1.2: Select and Download Dataset

Tasks:

- [ ] Download GSM8K dataset from HuggingFace:

```python
from datasets import load_dataset
gsm8k = load_dataset("openai/gsm8k", "main")
```

- [ ] Sample 150-200 problems from test set:
  - Use stratified sampling if possible (by difficulty/length)
  - Ensure problems have clear ground-truth answers
  - Verify solutions have step-by-step reasoning
- [ ] Create data manifest (CSV with problem_id, question, answer, solution_steps)

Alternative datasets if GSM8K insufficient:

- MATH dataset (more challenging): `load_dataset("hendrycks/competition_math")`
- SVAMP (variations): `load_dataset("ChilleD/SVAMP")`

Resources:

- GSM8K: https://huggingface.co/datasets/openai/gsm8k
- MATH: https://huggingface.co/datasets/hendrycks/competition_math

Deliverable: CSV file with 150-200 selected problems including ground truth

---

Step 1.3: Design Prompts for Both Conditions

Tasks:

- [ ] Create **Outcome-only prompt template**:

```
Answer the following math problem. Provide ONLY the final numerical answer, without showing any work or reasoning.

Problem: {question}

Answer:
```

- [ ] Create **Process prompt template** (standard CoT):

```
Solve the following math problem step by step. Show all your reasoning clearly, then provide the final answer.

Problem: {question}

Solution:
```

- [ ] Create **Structured Process prompt template** (Process+):

```
Solve the following math problem using the following format:

Step 1: [First reasoning step]
Step 2: [Second reasoning step]
...
Final Answer: [Numerical answer]

Problem: {question}

Solution:
```

- [ ] Test prompts on 5 sample problems to verify format

Resources:

- Chain-of-Thought Prompting paper examples
- OpenAI/Anthropic API prompt engineering guides

Deliverable: Three validated prompt templates

Week 2: Data Generation

Step 2.1: Generate Model Responses

Tasks:

- [ ] Write data collection script with:
  - API rate limiting and retry logic
  - Progress tracking and checkpointing
  - Response parsing and storage
  - Error handling
- [ ] Generate responses for ALL conditions:

```python
for problem in problems:
    # Outcome condition
    outcome_response = query_model(outcome_prompt.format(question=problem))

    # Process condition
    process_response = query_model(process_prompt.format(question=problem))

    # Process+ condition (optional)
    structured_response = query_model(structured_prompt.format(question=problem))

    # Store with problem_id, condition, response, timestamp
```

- [ ] Implement response storage format:

```python
{
  "problem_id": "gsm8k_001",
  "question": "...",
  "ground_truth": "42",
  "ground_truth_solution": "...",
  "outcome_response": "42",
  "process_response": "Step 1: ... Step 2: ... Therefore, 42",
  "structured_response": "Step 1: ... Final Answer: 42",
  "timestamp": "2025-01-15T10:30:00"
}
```

Cost estimation (OpenAI GPT-4o-mini):

- ~200 problems × 3 conditions × ~500 tokens avg = ~300K tokens
- Input: ~$0.45, Output: ~$1.80, Total: ~$2.25

Resources:

- OpenAI API: https://platform.openai.com/docs/api-reference
- Anthropic API: https://docs.anthropic.com/claude/reference

Deliverable: JSON/CSV file with all model responses for all conditions

---

Step 2.2: Parse and Extract Final Answers

Tasks:

- [ ] Implement answer extraction logic:
  - For outcome condition: direct extraction
  - For process/structured: parse last line or "Final Answer:" marker
  - Handle edge cases (no answer, multiple numbers, formatting issues)
- [ ] Create answer normalization function:

```python
def normalize_answer(text):
    # Extract numerical value
    # Handle units, fractions, decimals
    # Standardize format
    return normalized_value
```

- [ ] Validate extraction quality manually on sample (20 problems)
- [ ] Create processed dataset with extracted answers:

```python
```

```
{
  "problem_id": "gsm8k_001",
  "ground_truth_normalized": 42.0,
  "outcome_answer": 42.0,
  "process_answer": 42.0,
  "structured_answer": 42.0,
  "outcome_correct": True,
  "process_correct": True,
  "structured_correct": True
}
```

Resources:

- Regex patterns for number extraction
- GSM8K evaluation code: https://github.com/openai/grade-school-math

Deliverable: Processed dataset with extracted and normalized answers

---

Phase 2: Metrics Implementation (Week 3)

Week 3: Implement Core Metrics

Step 3.1: Calculate Final Answer Accuracy

Tasks:

- [ ] Implement accuracy calculation:

```python
def calculate_accuracy(predictions, ground_truth):
    correct = sum([pred == gt for pred, gt in zip(predictions, ground_truth)])
    return correct / len(predictions)
```

- [ ] Compute accuracy for each condition:
  - Accuracy(Outcome)
  - Accuracy(Process)
  - Accuracy(Process+)
- [ ] Perform statistical significance testing:

```python
```

```python
from scipy.stats import mcnemar
# McNemar's test for paired binary outcomes
contingency_table = [[both_correct, process_correct_outcome_wrong],
                     [outcome_correct_process_wrong, both_wrong]]
statistic, pvalue = mcnemar(contingency_table)
```

- [ ] Create accuracy comparison visualization

Deliverable: Accuracy metrics with statistical significance tests

---

Step 3.2: Error-Type Analysis (Subset)

Tasks:

- [ ] Select 30-50 incorrectly answered problems across conditions
- [ ] Define error taxonomy:
  1. **Arithmetic error**: Calculation mistake (e.g., 3 × 5 = 11)
  2. **Conceptual error**: Wrong approach or formula
  3. **Reading comprehension error**: Misunderstood problem
  4. **Incomplete reasoning**: Missing steps or logic gaps
  5. **Formatting/parsing error**: Answer present but not extracted
- [ ] Manually label errors by type for each condition
- [ ] Create error distribution comparison:

python

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Create stacked bar chart showing error types per condition
```

Deliverable: Error taxonomy with distribution across conditions

---

Step 3.3: Hallucination Rate in Process Responses

Tasks:

- [ ] Select 50 process/structured responses for hallucination analysis
- [ ] Define hallucination categories:
  1. **Factual error**: Objectively false statement (3 × 5 = 11)

2. **Irrelevant information**: Introduces unrelated facts

3. **Logical inconsistency**: Contradicts earlier steps

4. **Confabulation**: Makes up non-existent details

- [ ] Binary label: Does response contain ≥1 hallucination?
- [ ] Calculate hallucination rate:

```python
hallucination_rate = num_with_hallucinations / total_responses
```

- [ ] Correlate with correctness:
  - Hallucination rate among correct answers
  - Hallucination rate among incorrect answers

Deliverable: Hallucination rate metric with analysis

Phase 3: Interpretability Metrics (Week 4)

Week 4: Deep Annotation for Interpretability

Step 4.1: Sample Selection for Deep Annotation

Tasks:

- [ ] Select 50-70 problems for intensive annotation:
  - Stratified by correctness (correct/incorrect answers)
  - Stratified by condition (ensure coverage)
  - Include diverse problem types
- [ ] Prepare annotation interface (spreadsheet or custom tool):
  - Problem text
  - Ground truth solution (step-by-step)
  - Model response (with steps highlighted)
  - Annotation fields (see below)

Deliverable: Annotated problem set (50-70 problems)

Step 4.2: Implement Step Correctness Score

Tasks:

- [ ] Segment process responses into individual steps:
  - For structured: parse by "Step N:" markers
  - For unstructured: manual or heuristic segmentation (by sentence/line)
- [ ] Design annotation rubric:

```
Score 0: Incorrect or unjustified
  - Factual error (wrong calculation, false premise)
  - Missing justification
  - Illogical inference

Score 1: Partially correct or incomplete
  - Correct direction but imprecise
  - Partially justified
  - Minor arithmetic errors with correct approach

Score 2: Correct and well-justified
  - Factually accurate
  - Clear reasoning
  - Properly follows from previous steps
```

- [ ] Annotate each step for 50-70 problems
- [ ] Calculate Step Correctness Score:

```python
step_correctness_score = mean([step_scores for all problems])
# Average score per problem, then average across problems
```

- [ ] Inter-annotator reliability check:
  - If 2 annotators: Calculate Cohen's kappa on 20% overlap
  - Resolve disagreements through discussion

Deliverable: Step correctness scores with inter-rater reliability

---

Step 4.3: Calculate Faithfulness / Expert Alignment Score

Tasks:

- [ ] For each problem, align model steps with expert solution steps:

```
Expert Step 1: Find total items → 3 + 5 = 8
Model Step 1: Add the quantities → 3 + 5 = 8
Alignment: Match (Score 2)

Expert Step 2: Divide by cost → 8 / 2 = 4
Model Step 2: Multiply by cost → 8 × 2 = 16
Alignment: Different operation (Score 0)
```

- [ ] Design faithfulness rubric:

```
Score 0: Different and incorrect/irrelevant
  - Model uses different operation and gets wrong result
  - Introduces irrelevant reasoning

Score 1: Similar operation but imprecise/partially wrong
  - Correct operation type but sloppy execution
  - Right general approach but inefficient

Score 2: Essentially same operation and correct
  - Model reasoning matches expert reasoning
  - May use different words but same logic
```

- [ ] Calculate Faithfulness Score:

```python
faithfulness_score = mean([alignment_scores for all problems])
```

- [ ] Optional: Compute automatic similarity proxy using embeddings:

```python
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')

expert_embedding = model.encode(expert_step)
model_embedding = model.encode(model_step)
similarity = cosine_similarity([expert_embedding], [model_embedding])[0][0]
```

Deliverable: Faithfulness/expert alignment scores

Step 4.4: Human Auditability Evaluation

Tasks:

- [ ] Design auditability questionnaire (5-point Likert scale): **Clarity:** 1 - Very unclear / 2 - Unclear / 3 - Neutral / 4 - Clear / 5 - Very clear "How easy is it to understand what the model is doing at each step?" **Verification Effort:** 1 - Very difficult to verify / 5 - Very easy to verify "How much effort would it take to check if each step is correct?" **Coherence:** 1 - Incoherent / 5 - Highly coherent "Do the steps flow logically from one to the next?"
- [ ] Conduct evaluation with 2-3 raters (including yourself):
  - Rate same 50-70 problems
  - Provide brief justification for scores
- [ ] Calculate inter-rater reliability (Fleiss' kappa or ICC)
- [ ] Aggregate scores:

```python
mean_clarity = mean([clarity_scores across raters and problems])
mean_verification_effort = mean([effort_scores across raters and problems])
mean_coherence = mean([coherence_scores across raters and problems])
```

- [ ] Compare across conditions (Outcome has no reasoning to audit)

Deliverable: Human auditability scores with reliability metrics

---

Phase 4: Analysis and Qualitative Study (Week 5)

Week 5: Comprehensive Analysis

Step 5.1: Statistical Analysis

Tasks:

- [ ] Compare metrics across conditions:

```python
```

```python
import pandas as pd
from scipy import stats

# Create comparison DataFrame
results_df = pd.DataFrame({
    'Condition': ['Outcome', 'Process', 'Process+'],
    'Accuracy': [outcome_acc, process_acc, structured_acc],
    'Step_Correctness': [None, process_step_corr, structured_step_corr],
    'Faithfulness': [None, process_faith, structured_faith],
    'Clarity': [None, process_clarity, structured_clarity],
    'Verification_Effort': [None, process_verify, structured_verify]
})

# Statistical tests
# Paired t-test for accuracy
t_stat, p_value = stats.ttest_rel(outcome_correct, process_correct)

# Effect sizes (Cohen's d)
cohens_d = (mean_process - mean_outcome) / pooled_std
```

- [ ] Create correlation analysis:
  - Accuracy vs. Step Correctness
  - Accuracy vs. Faithfulness
  - Step Correctness vs. Faithfulness
  - Verification Effort vs. Accuracy
- [ ] Generate summary statistics table

Deliverable: Statistical analysis report with significance tests

---

Step 5.2: Qualitative Failure Case Analysis

Tasks:

- [ ] Identify interesting failure patterns:
  1. **Correct answer, garbage reasoning**: Final answer correct but steps invalid
  2. **Wrong answer, good reasoning**: Sound approach but arithmetic error
  3. **Hallucination with confidence**: Confidently stated false facts
  4. **Incomplete reasoning**: Jumps steps, still reaches correct answer
- [ ] Select 10-15 representative examples for each pattern

☐ Document detailed case studies:

```
Case ID: gsm8k_042
Problem: [text]
Ground Truth: 15

Outcome Response: "15"
Process Response: "First, we calculate 3 × 5 = 11. Then..."

Analysis:
- Final answer correct in both conditions
- Process condition shows arithmetic error in Step 1
- Despite error, arrives at correct answer (compensating error in Step 3)
- Demonstrates process supervision reveals hidden errors
```

☐ Create categorized failure taxonomy with examples

Deliverable: Qualitative analysis document with case studies

Step 5.3: Visualization Creation

Tasks:

☐ Create comprehensive visualizations:

1. **Accuracy comparison bar chart**
2. **Error type distribution (stacked bars)**
3. **Step correctness distribution (box plots)**
4. **Faithfulness vs. Accuracy scatter plot**
5. **Verification effort comparison**
6. **Correlation heatmap** (all metrics)

☐ Use consistent color scheme and style:

```python
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("whitegrid")
colors = {'Outcome': '#FF6B6B', 'Process': '#4ECDC4', 'Process+': '#95E1D3'}
```

☐ Add error bars and statistical annotations where relevant

- [ ] Export high-resolution figures (300 DPI for paper)

Deliverable: Complete set of publication-quality visualizations

---

Phase 5: Synthesis and Documentation (Week 6)

Week 6: Final Report and Presentation

Step 6.1: Connect Findings to Legal/Regulatory Context

Tasks:

- [ ] Analyze results through regulatory lens: **If Process Supervision Shows Higher Reliability:**
  - Discuss implications for GDPR Article 22 (right to explanation)
  - Connect to AI Act transparency requirements
  - Argue for process-based explanations in high-stakes domains

If Process Supervision Shows Higher Interpretability:

  - Discuss auditability for algorithmic accountability
  - Connect to requirements for meaningful human oversight
  - Argue transparency enables non-arbitrary decision-making

If Reasoning Traces Are Unfaithful:

  - Warn about liability risks of misleading explanations
  - Discuss dangers of "illusion of explainability"
  - Recommend caution in using CoT for compliance

- [ ] Reference specific legal frameworks:
  - EU GDPR (Articles 13-15, 22)
  - EU AI Act (Articles 13, 52)
  - U.S. Administrative Procedure Act (§ 706)
  - Emerging case law on algorithmic accountability
- [ ] Draft 2-3 page legal implications section

Deliverable: Legal/regulatory implications analysis

Step 6.2: Write Final Report

Tasks:

- [ ] Structure report following standard format: **1. Introduction (2 pages)**
  - Research question and significance

- Connection to Responsible AI and Law

2. Related Work (2-3 pages)

- Process supervision literature
- Interpretability and faithfulness work
- Legal frameworks for AI explainability

3. Methodology (3-4 pages)

- Experimental design (conditions, prompts)
- Dataset description
- Metrics definitions with rubrics
- Annotation procedures and reliability

4. Results (4-5 pages)

- Quantitative findings (accuracy, interpretability metrics)
- Statistical analyses
- Visualizations

5. Qualitative Analysis (2-3 pages)

- Failure case studies
- Error pattern analysis
- Representative examples

6. Legal and Regulatory Implications (2-3 pages)

- Connections to legal frameworks
- Implications for accountability
- Policy recommendations

7. Discussion (2 pages)

- Interpretation of findings
- Limitations
- Future work

8. Conclusion (1 page)

- [ ] Total length: ~20-25 pages (double-spaced)
- [ ] Include appendices:
  - Complete annotation rubrics
  - Prompt templates
  - Additional visualizations

- Inter-rater reliability calculations

Deliverable: Complete research paper draft

Step 6.3: Prepare Presentation

Tasks:

- [ ] Create slide deck (15-20 slides, 15-minute presentation):

1. **Title & Motivation** (1 slide)
2. **Research Question** (1 slide)
3. **Why This Matters for Law** (2 slides)
4. **Methodology Overview** (2 slides)
5. **Experimental Design** (2 slides)
6. **Key Results - Accuracy** (2 slides)
7. **Key Results - Interpretability** (2 slides)
8. **Qualitative Analysis** (2-3 slides with examples)
9. **Legal Implications** (2 slides)
10. **Limitations & Future Work** (1 slide)
11. **Conclusions** (1 slide)

- [ ] Design clear, minimal slides (avoid text walls)
- [ ] Prepare speaker notes
- [ ] Practice presentation timing (aim for 12-13 minutes to allow Q&A)
- [ ] Create backup slides for anticipated questions

Deliverable: Presentation slides with speaker notes

Step 6.4: Documentation and Reproducibility

Tasks:

- [ ] Clean and document code:

```python
```

```
# Add docstrings to all functions
# Include usage examples
# Add requirements.txt
# Create README.md with setup instructions
```

- [ ] Organize GitHub repository (if sharing):

```
repo/
├── README.md (setup, usage, results summary)
├── requirements.txt
├── data/
│   ├── dataset_manifest.csv
│   └── responses_sample.json (not full dataset if API-generated)
├── notebooks/
│   ├── 01_data_collection.ipynb
│   ├── 02_metric_calculation.ipynb
│   └── 03_analysis_visualization.ipynb
├── scripts/
│   ├── generate_responses.py
│   ├── calculate_metrics.py
│   └── visualize_results.py
├── results/
│   ├── tables/
│   └── figures/
└── paper/
    └── final_report.pdf
```

- [ ] Write comprehensive README:
  - Project description
  - Installation instructions
  - Usage guide
  - Results summary
  - Citation information
- [ ] Archive final outputs:
  - All data files
  - All code
  - Final report PDF
  - Presentation slides

- Supplementary materials

Deliverable: Complete, documented, reproducible project package

---

Resource Requirements Summary

Computational Resources

- **Local machine**: Sufficient for most tasks (data processing, annotation, analysis)
- **API access**: OpenAI/Anthropic OR HuggingFace Inference API
- **Estimated API cost**: $2-10 depending on model choice and problem count

Human Resources

- **Primary researcher** (you): ~15-20 hours/week for 6 weeks
- **Optional co-annotators**: 2-3 volunteers for 3-5 hours total (reliability checks)

Software/Tools

- Python 3.9+ with standard ML/data science stack
- Jupyter notebooks for exploratory analysis
- LaTeX or Microsoft Word for report writing
- Presentation software (PowerPoint, Google Slides, or Beamer)

Datasets (All Free & Public)

- GSM8K: https://huggingface.co/datasets/openai/gsm8k
- MATH (backup): https://huggingface.co/datasets/hendrycks/competition_math
- PRM800K (reference): https://github.com/openai/prm800k

Models (Choose One Path)

Option A: API-based (Recommended for simplicity)

- OpenAI GPT-4o-mini (~$2.25 for 200 problems × 3 conditions)
- Anthropic Claude 3.5 Sonnet (~$15 for same workload)

Option B: Open-weight (Free but requires more setup)

- Qwen2.5-Math-7B-Instruct (strong on math)
- Llama-3.1-8B-Instruct (general purpose)
- DeepSeek-Math-7B (specialized on math)

Risk Mitigation

Potential Issues & Solutions

Issue 1: API rate limits or cost overruns

- Solution: Use open-weight models via HuggingFace
- Solution: Reduce problem count to 100 minimum viable set
- Solution: Use GPT-4o-mini instead of GPT-4

Issue 2: Response parsing difficulties

- Solution: Implement robust regex patterns with fallbacks
- Solution: Manual review and correction of edge cases
- Solution: Simplify prompts to encourage more consistent formatting

Issue 3: Low inter-rater reliability in annotations

- Solution: Refine annotation rubrics with more specific criteria
- Solution: Conduct training session with co-annotators using examples
- Solution: Adjudicate disagreements through discussion and consensus

Issue 4: Insufficient differences between conditions

- Solution: This is still a valid result - document null findings
- Solution: Qualitative analysis becomes more important
- Solution: Focus on failure modes and edge cases

Issue 5: Time constraints

- Solution: Reduce annotation scope to 30-40 problems minimum
- Solution: Simplify interpretability metrics (drop one metric if needed)
- Solution: Focus on most important aspects for legal implications

---

Success Criteria

Minimum Viable Project

✅ 100+ problems evaluated across 2+ conditions
✅ Accuracy comparison with statistical tests
✅ 30+ problems annotated for step correctness
✅ Qualitative analysis of 10+ failure cases

✅ 15-page report connecting findings to legal frameworks
✅ 15-minute presentation

Target Project (Full Scope)

✅ 150-200 problems evaluated across 3 conditions
✅ All factual reliability metrics computed
✅ 50-70 problems deeply annotated
✅ All three interpretability metrics calculated
✅ Inter-rater reliability ≥0.70 (substantial agreement)
✅ 20-25 page comprehensive report
✅ Publication-quality visualizations
✅ Reproducible code and documentation

Stretch Goals

✅ Comparison across two different models
✅ Analysis on both GSM8K and MATH datasets
✅ Integration with existing process supervision tools
✅ Interactive visualization dashboard
✅ Submission to workshop or conference

Timeline Overview (Gantt Chart Format)

```
Week 1: [████████ Setup ████████][███ Dataset ███][███ Prompts ███]
Week 2: [███████████████████ Data Generation ███████████████████]
Week 3: [██████ Accuracy ██████][███ Errors ███][█ Hall. █]
Week 4: [██████ Sample ██████][█ Steps █][█ Faith █][█ Audit █]
Week 5: [████ Stats ████][████ Qual ████][████ Visuals ████]
Week 6: [██████ Legal ██████][████████████ Report ████████████]
```

Milestones:

- End of Week 2: All model responses collected ✓
- End of Week 3: Factual reliability metrics complete ✓
- End of Week 4: Interpretability annotations complete ✓
- End of Week 5: All analysis and visualizations complete ✓
- End of Week 6: Final report and presentation ready ✓

Key Decisions to Make Now

Decision 1: Model Selection

Option A: OpenAI GPT-4o-mini (Recommended)

- ✅ Easy API access
- ✅ Low cost ($2-5)
- ✅ Reliable, consistent outputs
- ❌ Closed-source (less reproducible)

Option B: Open-weight (Qwen2.5-Math-7B)

- ✅ Free inference
- ✅ Fully reproducible
- ✅ Math-specialized
- ❌ Requires more setup
- ❌ May need GPU access

Recommendation: Start with GPT-4o-mini for speed, optionally add Qwen2.5-Math if time permits

---

Decision 2: Annotation Scope

Option A: Solo annotation (50 problems)

- ✅ Faster, no coordination overhead
- ❌ No inter-rater reliability
- ❌ Potential bias

Option B: Multi-rater (30 problems with 2-3 raters)

- ✅ Stronger validity
- ✅ Inter-rater reliability metrics
- ❌ Requires recruiting and coordinating annotators

Recommendation: Start solo, recruit 1-2 volunteers for 20-30% overlap if possible

---

Decision 3: Dataset Choice

Option A: GSM8K only (Recommended)

- ✅ Grade-school level (easier to annotate)

- ✅ High-quality step-by-step solutions
- ✅ Standard benchmark

Option B: MATH dataset

- ✅ More challenging (greater differentiation)
- ❌ More difficult to annotate accurately
- ❌ Requires stronger math background

Recommendation: Use GSM8K as primary dataset

Next Immediate Actions (Today)

1. **Set up Python environment and install dependencies**
2. **Create API account** (OpenAI or Anthropic) and verify access
3. **Download GSM8K dataset** and explore structure
4. **Sample 150-200 problems** using stratified random sampling
5. **Draft and test prompt templates** on 5 example problems
6. **Create project directory structure** and initialize version control

Time estimate: 2-3 hours

Questions to Resolve

1. **Which model(s) will you use?** → Decide by end of today
2. **Will you have co-annotators?** → Decide by Week 3
3. **What is your actual weekly time budget?** → Adjust timeline if needed
4. **Do you have GPU access if using open models?** → Determines model choice
5. **What format for final submission?** → Paper length, presentation format

---

Conclusion

This plan provides a **realistic, achievable path** to completing your project on process vs. outcome supervision in 6 weeks. The phased approach ensures:

✅ **Weeks 1-2:** Core infrastructure and data collection complete
✅ **Weeks 3-4:** All quantitative and qualitative metrics computed

✅ **Week 5:** Comprehensive analysis connecting technical findings
✅ **Week 6:** Polished final deliverables with legal implications

The plan is flexible: You can scale down to the "Minimum Viable Project" if time is tight, or pursue "Stretch Goals" if ahead of schedule.

Key success factors:

- Start with reliable, easy-to-use tools (GPT-4o-mini API + GSM8K)
- Focus on quality over quantity in annotations (50 deep > 200 shallow)
- Connect every finding back to legal/regulatory implications
- Document as you go (don't wait until the end)

Good luck with your project! 🚀