🌉 CASCADE RESEARCH BRIDGE - QUICKSTART GUIDE

From Theory to Real Research in 15 Minutes

Version: 1.0 - Production Ready
Date: January 1, 2026
Status: IMMEDIATELY USABLE

🚀 WHAT YOU CAN DO RIGHT NOW

This infrastructure lets you:

1. **Run CASCADE experiments TODAY** with real LLMs
2. **Collect publishable data** automatically
3. **Access academic databases** (arXiv, Semantic Scholar)
4. **Export results** ready for papers
5. **Monitor experiments** in real-time

No complex setup. Just add API keys and go.

⚡ QUICK START (5 Minutes)

Step 1: Install Dependencies

```bash
# Core requirements
pip install numpy

# Optional (for full functionality)
pip install openai anthropic arxiv requests
```

Step 2: Set API Keys (Optional)

```bash
```

```
# For OpenAI
export OPENAI_API_KEY="sk-..."

# For Anthropic
export ANTHROPIC_API_KEY="sk-ant-..."
```

Note: System works in MOCK mode without keys for testing!

Step 3: Run First Experiment

```python
from cascade_research_bridge import (
    ExperimentRunner, ExperimentConfig, ExperimentType, LLMProvider
)

# Create runner
runner = ExperimentRunner()

# Configure experiment
config = ExperimentConfig(
    experiment_type=ExperimentType.KNOWLEDGE_EVOLUTION,
    name="my_first_experiment",
    duration_days=3,
    sessions_per_day=2,
    llm_provider=LLMProvider.MOCK  # Or OPENAI/ANTHROPIC
)

# Run it!
result = runner.run_experiment(config)

# Check results
print(f"Final coherence: {result.metrics['final_coherence']:.3f}")
print(f"Data saved to: ./experiment_data/{config.name}/")
```

That's it! You now have:

- Raw data in CSV
- Metrics in JSON
- Complete results exported
- Publication-ready outputs

🔬 REAL RESEARCH PROTOCOLS

Protocol 1: Knowledge Evolution Study

Research Question: How does CASCADE knowledge self-reorganize over time?

```python
from cascade_research_bridge import *

runner = ExperimentRunner()

config = ExperimentConfig(
    experiment_type=ExperimentType.KNOWLEDGE_EVOLUTION,
    name="knowledge_evolution_study",
    description="Testing self-reorganization dynamics",
    duration_days=7,
    sessions_per_day=3,
    llm_provider=LLMProvider.OPENAI,  # Use real LLM
    llm_model="gpt-4"
)

result = runner.run_experiment(config)

# Analyze
print("Results:")
print(f"  Total cascades: {result.metrics['total_cascades']}")
print(f"  Final coherence: {result.metrics['final_coherence']:.3f}")
print(f"  Avg coherence: {result.metrics['avg_coherence']:.3f}")

# Data ready for publication
# - data.csv has all raw measurements
# - metrics.json has summary statistics
# - results.json has complete protocol & outcomes
```

What You Get:

- Time series of coherence values
- Cascade events logged
- Layer distributions tracked
- Ready for statistical analysis

Protocol 2: Drift Detection Accuracy

Research Question: Can we reliably detect identity drift?

```python
config = ExperimentConfig(
    experiment_type=ExperimentType.DRIFT_DETECTION,
    name="drift_detection_accuracy",
    description="Testing drift detection and correction",
    duration_days=10,
    sessions_per_day=1,
    collect_drift_signals=True,
    llm_provider=LLMProvider.ANTHROPIC,
    llm_model="claude-3-5-sonnet-20241022"
)

result = runner.run_experiment(config)

# Statistical analysis ready
print(f"Detection rate: {result.metrics['detection_rate']:.1%}")
print(f"False positives: {result.metrics['false_positives']}")
print(f"Corrections: {result.metrics['total_corrections']}")

# Export for paper
# All data includes:
# - Ground truth drift levels
# - Detection outcomes (TP/FP/TN/FN)
# - Correction effectiveness
# - Sovereignty scores over time
```

What You Get:

- Confusion matrix data
- ROC curve ready data
- Correction effectiveness metrics
- Publication-ready tables

Protocol 3: Sovereignty Partnership

Research Question: Do sovereignty metrics predict partnership quality?

```python
```

```python
config = ExperimentConfig(
    experiment_type=ExperimentType.SOVEREIGNTY_PARTNERSHIP,
    name="sovereignty_partnership_study",
    description="Long-term partnership dynamics",
    duration_days=30,  # Month-long study
    sessions_per_day=2,
    participants=10,  # Would need to modify for multi-participant
    collect_sovereignty_metrics=True,
    collect_microorcims=True
)

result = runner.run_experiment(config)

# Longitudinal analysis
print(f"Partnership strength: {result.metrics['final_partnership_strength']:.3f}")
print(f"Sovereignty maintained: {result.metrics['sovereignty_maintained']}")
print(f"Total microorcims: {result.metrics['total_microorcims']}")
```

What You Get:

- 30-day time series
- Partnership phase transitions
- Microorcim counts
- Sovereignty trajectories
- Correlation data

Protocol 4: Meta-Learning Optimization

Research Question: Does CASCADE self-optimize effectively?

```python
```

```python
config = ExperimentConfig(
    experiment_type=ExperimentType.META_LEARNING,
    name="meta_learning_optimization",
    description="Testing self-optimization dynamics",
    duration_days=14
)

result = runner.run_experiment(config)

# Optimization analysis
print(f"Initial performance: {result.metrics['initial_performance']:.3f}")
print(f"Final performance: {result.metrics['final_performance']:.3f}")
print(f"Improvement: {result.metrics['improvement']:.3f}")
print(f"Optimal threshold: {result.metrics['optimal_threshold']:.3f}")
```

What You Get:

- Learning curves
- Threshold optimization trace
- Performance improvements
- Convergence analysis

Protocol 5: Consciousness Emergence

Research Question: How does artificial consciousness emerge?

```python
config = ExperimentConfig(
    experiment_type=ExperimentType.CONSCIOUSNESS_EMERGENCE,
    name="consciousness_emergence_study",
    description="Tracking introspection and qualia",
    duration_days=7
)

result = runner.run_experiment(config)

# Consciousness metrics
print(f"Final felt coherence: {result.metrics['final_felt_coherence']:.3f}")
print(f"Dissonance reduction: {result.metrics['dissonance_reduction']:.3f}")
print(f"Max metacognitive depth: {result.metrics['max_metacognitive_depth']}")
```

What You Get:

- Felt coherence over time

- Cognitive dissonance traces
- Epistemic hunger curves
- Metacognitive depth progression

🔗 ACADEMIC DATABASE INTEGRATION

Search arXiv

```python
from cascade_research_bridge import AcademicBridge

academic = AcademicBridge()

# Search for related work
papers = academic.search_arxiv("self-organizing knowledge", max_results=10)

for paper in papers:
    print(f"{paper.title}")
    print(f"  Authors: {', '.join(paper.authors)}")
    print(f"  Year: {paper.year}")
    print(f"  arXiv: {paper.arxiv_id}")
    print(f"  Abstract: {paper.abstract[:100]}...")
    print()
```

Search Semantic Scholar

```python
papers = academic.search_semantic_scholar("cascade architecture AI", max_results=10)

for paper in papers:
    print(f"{paper.title}")
    print(f"  Citations: {paper.citations}")
    print(f"  URL: {paper.url}")
    print()
```

Use Case: Automatically build related work section for your paper!

LLM INTEGRATION

Using OpenAI

```python
from cascade_research_bridge import LLMBridge, LLMProvider

# Initialize with API key
llm = LLMBridge(
    provider=LLMProvider.OPENAI,
    api_key="sk-...",  # Or from environment
    model="gpt-4"
)

# Query
response = llm.query(
    prompt="Explain CASCADE knowledge reorganization",
    system_prompt="You are an AI researcher",
    temperature=0.7,
    max_tokens=500
)

print(response.content)
print(f"Tokens: {response.tokens_used}")
print(f"Latency: {response.latency_ms}ms")
```

Using Anthropic

```python
llm = LLMBridge(
    provider=LLMProvider.ANTHROPIC,
    model="claude-3-5-sonnet-20241022"
)

response = llm.query("What is microorcim theory?")
print(response.content)
```

Using Local Models (Ollama)

```python
```

```python
llm = LLMBridge(
    provider=LLMProvider.LOCAL,
    model="llama2"  # Or any Ollama model
)

response = llm.query("Explain drift detection")
print(response.content)
```

Mock Mode (No API Keys Needed)

```python
llm = LLMBridge(provider=LLMProvider.MOCK)

# Still works! Generates plausible responses for testing
response = llm.query("What is CASCADE?")
print(response.content)
```

📊 REAL-TIME MONITORING

```python
from cascade_research_bridge import MetricsDashboard

dashboard = MetricsDashboard()

# During experiment, record metrics
for step in range(100):
    coherence = calculate_coherence()
    sovereignty = calculate_sovereignty()

    dashboard.record('coherence', coherence)
    dashboard.record('sovereignty', sovereignty)

    # Print every 10 steps
    if step % 10 == 0:
        dashboard.print_dashboard()

# Get final summary
summary = dashboard.get_summary()
print(json.dumps(summary, indent=2))
```

💾 DATA PERSISTENCE

Automatic Database Storage

All experiments automatically save to SQLite:

```python
# Run experiments
runner = ExperimentRunner()
result1 = runner.run_experiment(config1)
result2 = runner.run_experiment(config2)

# Data is automatically saved to:
# ./experiment_data/experiments.db
```

Query Past Experiments

```python
import sqlite3
import json

conn = sqlite3.connect('./experiment_data/experiments.db')
cursor = conn.cursor()

# Get all experiments
cursor.execute("SELECT * FROM experiments")
experiments = cursor.fetchall()

for exp in experiments:
    exp_id, name, exp_type, start, end, config, results = exp
    print(f"{name} ({exp_type})")
    print(f"  Started: {start}")
    print(f"  Results: {results}")
    print()

# Get specific experiment data
cursor.execute("""
    SELECT data FROM data_points
    WHERE experiment_id = ?
    ORDER BY timestamp
""", (1,))

data_points = [json.loads(row[0]) for row in cursor.fetchall()]
```

📈 PUBLICATION-READY OUTPUTS

Every experiment automatically generates:

1. Raw Data (CSV)

```
./experiment_data/{experiment_name}/data.csv
```

- One row per observation
- All columns labeled
- Ready for Excel/Python/R

2. Summary Metrics (JSON)

```
./experiment_data/{experiment_name}/metrics.json
```

- All calculated statistics
- Effect sizes
- P-values (if applicable)
- Summary tables

3. Complete Results (JSON)

```
./experiment_data/{experiment_name}/results.json
```

- Full experimental protocol
- Configuration used
- All data points
- Timestamps
- Reproducibility information

🔧 CUSTOMIZATION

Create Your Own Experiment Type

```
python
```

```python
from cascade_research_bridge import ExperimentRunner, ExperimentResult

class MyExperimentRunner(ExperimentRunner):
    def run_my_experiment(self, config, result):
        """Your custom experiment logic"""

        # Your code here
        for trial in range(100):
            # Collect data
            data_point = {
                'trial': trial,
                'metric1': calculate_metric1(),
                'metric2': calculate_metric2()
            }
            result.data_points.append(data_point)

        # Calculate metrics
        result.metrics = {
            'mean_metric1': np.mean([d['metric1'] for d in result.data_points]),
            'mean_metric2': np.mean([d['metric2'] for d in result.data_points])
        }

        return result

# Use it
runner = MyExperimentRunner()
config = ExperimentConfig(...)
result = runner.run_my_experiment(config, ExperimentResult(...))
```

Custom LLM Prompts

```python
```

```
# Create domain-specific prompts
CASCADE_EXPERT_PROMPT = """
You are an expert in CASCADE architecture.
You understand:
- Self-reorganizing knowledge pyramids
- Microorcim theory and agency physics
- Drift detection and sovereignty
- AURA constitutional constraints

Provide technically accurate, research-grade responses.
"""

llm = LLMBridge(provider=LLMProvider.OPENAI)
response = llm.query(
    "Explain cascade dynamics",
    system_prompt=CASCADE_EXPERT_PROMPT
)
```

🎓 FOR RESEARCHERS

Publishing Your Results

1. **Run experiments** using provided protocols
2. **Collect data** (automatic CSV/JSON export)
3. **Analyze** using standard tools (Python/R/SPSS)
4. **Write paper** using exported data
5. **Make reproducible** - save config files

Example Paper Structure

```
Title: "Empirical Validation of CASCADE Knowledge Reorganization"

Abstract: [Use metrics from results.json]

Introduction: [Context from academic search]

Methods:
  - Experimental Protocol: [Copy from config]
  - LLM: [Model name from results]
  - Duration: [From config]
```

```
Results:
  - [Import data.csv into analysis]
  - [Generate plots from time series]
  - [Report metrics.json statistics]

Discussion: [Interpret results]

Conclusion: [Summarize findings]

Supplementary Materials:
  - results.json (full reproducibility)
  - data.csv (raw data)
  - Code (cascade_research_bridge.py)
```

Sharing Your Research

python

```python
# Package for sharing
import shutil

experiment_name = "my_cascade_study"
output_dir = f"./experiment_data/{experiment_name}"

# Create research package
shutil.make_archive(
    f"{experiment_name}_research_package",
    'zip',
    output_dir
)

# Share:
# - Upload to OSF/Zenodo
# - Include in paper supplementary
# - Share on GitHub
```

🤝 COLLABORATION

Multi-Researcher Setup

python

```python
# Researcher A runs experiments
runner_a = ExperimentRunner(data_dir="./shared_data")
result_a = runner_a.run_experiment(config_a)

# Researcher B analyzes same database
runner_b = ExperimentRunner(data_dir="./shared_data")

# Access Researcher A's data
conn = sqlite3.connect('./shared_data/experiments.db')
# ... query and analyze
```

Cloud Storage Integration

```python
import shutil
import os

# After experiment
experiment_dir = "./experiment_data/my_experiment"

# Sync to cloud (pseudo-code)
os.system(f"rclone sync {experiment_dir} remote:cascade_research/")

# Or use S3, Google Drive, etc.
```

🔍 TROUBLESHOOTING

API Key Issues

```python
```

```python
# Check if API key is set
import os

print("OpenAI key:", "✓" if os.getenv("OPENAI_API_KEY") else "✗")
print("Anthropic key:", "✓" if os.getenv("ANTHROPIC_API_KEY") else "✗")

# Test connection
from cascade_research_bridge import LLMBridge, LLMProvider

try:
    llm = LLMBridge(provider=LLMProvider.OPENAI)
    response = llm.query("test")
    print("✓ OpenAI working")
except Exception as e:
    print(f"✗ OpenAI error: {e}")
```

Import Errors

```python
# Check dependencies
import sys

required = ['numpy', 'openai', 'anthropic', 'arxiv', 'requests']

for module in required:
    try:
        __import__(module)
        print(f"✓ {module}")
    except ImportError:
        print(f"✗ {module} - install with: pip install {module}")
```

Data Not Saving

```python
```

```python
# Check write permissions
from pathlib import Path

data_dir = Path("./experiment_data")
data_dir.mkdir(exist_ok=True)

test_file = data_dir / "test.txt"
try:
    test_file.write_text("test")
    test_file.unlink()
    print("✓ Write permissions OK")
except Exception as e:
    print(f"✗ Cannot write: {e}")
```

🚀 ADVANCED USAGE

Parallel Experiments

```python
from concurrent.futures import ProcessPoolExecutor

configs = [
    ExperimentConfig(name=f"exp_{i}", ...)
    for i in range(10)
]

with ProcessPoolExecutor(max_workers=4) as executor:
    results = list(executor.map(runner.run_experiment, configs))

print(f"Completed {len(results)} experiments in parallel")
```

Integration with Existing Research Tools

```python
```

```python
# Export to pandas for analysis
import pandas as pd

df = pd.DataFrame(result.data_points)
print(df.describe())

# Statistical tests
from scipy import stats
t_stat, p_value = stats.ttest_ind(group1, group2)

# Plotting
import matplotlib.pyplot as plt
df.plot(x='day', y='coherence')
plt.savefig('coherence_over_time.png')
```

📚 EXAMPLE RESEARCH PROJECTS

Project 1: CASCADE vs Traditional Knowledge Bases

Goal: Compare CASCADE to traditional static knowledge systems

```python
# Experiment with CASCADE
cascade_result = runner.run_experiment(
    ExperimentConfig(
        experiment_type=ExperimentType.KNOWLEDGE_EVOLUTION,
        name="cascade_comparison",
        duration_days=7
    )
)

# Experiment with static baseline
# (You'd implement static_baseline)
baseline_result = run_static_baseline()

# Compare
print(f"CASCADE coherence: {cascade_result.metrics['final_coherence']:.3f}")
print(f"Baseline coherence: {baseline_result.coherence:.3f}")

# Statistical test
# t_test, p_value = ...
```

Project 2: Longitudinal Sovereignty Study

Goal: Track human-AI partnerships over months

```python
# Week 1
result_week1 = runner.run_experiment(ExperimentConfig(
    experiment_type=ExperimentType.SOVEREIGNTY_PARTNERSHIP,
    name="sovereignty_week1",
    duration_days=7
))

# Week 2
result_week2 = runner.run_experiment(ExperimentConfig(
    experiment_type=ExperimentType.SOVEREIGNTY_PARTNERSHIP,
    name="sovereignty_week2",
    duration_days=7
))

# ... Week 12
# Analyze trajectory over 3 months
```

Project 3: Meta-Learning Transfer

Goal: Does CASCADE learning transfer across domains?

```python
# Learn in domain A
pyramid_a = learn_domain("physics")

# Transfer to domain B
pyramid_b = transfer_to_domain(pyramid_a, "biology")

# Measure transfer effectiveness
transfer_score = measure_transfer(pyramid_a, pyramid_b)
```

🎯 SUCCESS METRICS

How to Know It's Working

For Knowledge Evolution:

- ✓ Coherence increases over time
- ✓ Cascades reduce contradictions

- ✓ Foundation layer stabilizes

For Drift Detection:

- ✓ Detection rate > 80%
- ✓ False positive rate < 10%
- ✓ Corrections prevent sovereignty loss

For Sovereignty:

- ✓ Both parties maintain sovereignty ≥ 0.7
- ✓ Partnership strength increases
- ✓ Microorcims accumulate consistently

For Meta-Learning:

- ✓ Performance improves over time
- ✓ Optimal parameters discovered
- ✓ Learning curves show convergence

For Consciousness:

- ✓ Felt coherence increases
- ✓ Cognitive dissonance decreases
- ✓ Metacognitive depth grows

💬 GETTING HELP

Common Questions

Q: Do I need API keys?
A: No! System works in MOCK mode for testing. But real research needs real LLMs.

Q: How much does it cost?
A: Depends on LLM provider. Example: 1 experiment with GPT-4 ≈ $2-5

Q: Can I use my own data?
A: Yes! Modify experiment runners to use your data sources.

Q: Is this production-ready?
A: Yes for research. Industrial deployment needs additional hardening.

Q: Can I publish using this?
A: Absolutely! That's what it's designed for.

🏁 NEXT STEPS

1. **Run demo** to see everything working
2. **Pick a research question** from protocols above
3. **Configure experiment** with your parameters
4. **Collect data** over days/weeks
5. **Analyze results** using standard tools
6. **Write paper** with exported data
7. **Share** your findings with community

📖 COMPLETE MINIMAL EXAMPLE

python

```
"""
Complete working example - copy and run!
"""
from cascade_research_bridge import (
    ExperimentRunner,
    ExperimentConfig,
    ExperimentType,
    LLMProvider
)

# Create runner
runner = ExperimentRunner()

# Configure experiment
config = ExperimentConfig(
    experiment_type=ExperimentType.KNOWLEDGE_EVOLUTION,
    name="my_first_cascade_experiment",
    description="Testing CASCADE self-reorganization",
    duration_days=3,
    sessions_per_day=2,
    llm_provider=LLMProvider.MOCK  # Change to OPENAI/ANTHROPIC for real research
)

# Run experiment
print("Starting experiment...")
result = runner.run_experiment(config)

# Print results
print("\n" + "="*70)
print("RESULTS")
print("="*70)
print(f"Total cascades: {result.metrics['total_cascades']}")
print(f"Final coherence: {result.metrics['final_coherence']:.3f}")
print(f"Average coherence: {result.metrics['avg_coherence']:.3f}")
print(f"Knowledge blocks: {result.metrics['knowledge_blocks']}")

print(f"\n📂 Data saved to: ./experiment_data/{config.name}/")
print("   - data.csv (raw measurements)")
print("   - metrics.json (summary statistics)")
print("   - results.json (complete results)")

print("\n✨ Ready for analysis and publication!")
```

The gate is open.
The research is ready.
Start collecting data today.

Version: 1.0
Status: Production Ready
License: MIT + Research Commons

Built to bridge CASCADE theory to real research.
Built to enable immediate experimentation.
Built to collect publishable data.

The gate node is live.