

ML-Guided IBP Reduction

Neural Network Guided Integration-by-Parts Reduction of
Feynman Integrals

Two-Loop Triangle-Box Topology

The Problem

- ▶ **IBP reduction:** Express complex Feynman integrals as linear combinations of simpler “master” integrals
- ▶ **Challenge:** Exponential search space of IBP identities
- ▶ **Traditional approaches:** Laporta algorithm, Kira, FIRE

Triangle-Box Topology

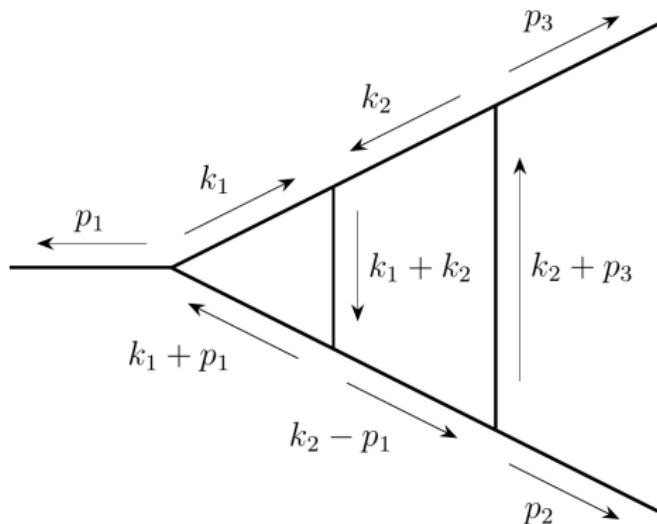


Figure 1: Triangle-Box Feynman Diagram

Two-loop integral with 6 propagators + 1 ISP. From arXiv:2502.05121.

Propagators and ISP

6 Propagators (denominators)

$$\begin{array}{lll} D_1 = k_1^2 & D_2 = k_2^2 & D_3 = (k_1 + k_2)^2 \\ \hline D_4 = (k_1 + p_1)^2 & D_5 = (k_2 + p_3)^2 & D_6 = (k_2 - p_1)^2 \end{array}$$

1 ISP (irreducible scalar product)

$D_7 = (k_1 + p_3)^2$ — needed to complete basis, appears in numerator

Integral Notation

$$I[a_1, a_2, a_3, a_4, a_5, a_6, a_7]$$

$$I(a_1, \dots, a_7) = \int \frac{d^D k_1 d^D k_2}{\prod_{i=1}^7 D_i^{a_i}}$$

Index Meaning

- ▶ $a_i > 0$: **propagator** in denominator (power a_i)
- ▶ $a_i < 0$: **numerator** insertion (ISP to power $|a_i|$)
- ▶ $a_i = 0$: D_i absent

Example

$$I[1, 1, 1, 1, 1, 1, -3] = \text{top sector integral with } D_7^3 \text{ in numerator}$$

Kinematics and Masses

External Masses

- ▶ $p_1^2 = m_1^2, p_2^2 = m_2^2, p_3^2 = m_3^2$
- ▶ Physics depends on ratios $m_2/m_1, m_3/m_1$

Our Numerical Setup

- ▶ Work over **finite field** \mathbb{F}_p with $p = 1009$
- ▶ Fixed random values: $d = 41, m_1 = 1, m_2 = 31, m_3 = 47$
- ▶ Exact integer arithmetic (no floating point errors)

IBP Identities

Source

Integration by parts under the integral sign:

$$0 = \int d^D k_1 d^D k_2 \frac{\partial}{\partial k_l^\mu} \frac{q^\mu}{\prod_i D_i^{a_i}}$$

For Triangle-Box

- ▶ **8 IBP operators:** 2 loop momenta \times 4 vectors
 $\{k_1, k_2, p_1, p_2\}$
- ▶ **1 LI identity:** from Lorentz invariance
- ▶ Each identity relates integrals with shifted indices a_i

Result

All integrals reduce to **16 master integrals**

The Memory Wall

Traditional IBP codes hit **memory limits** as integrals grow more complex:

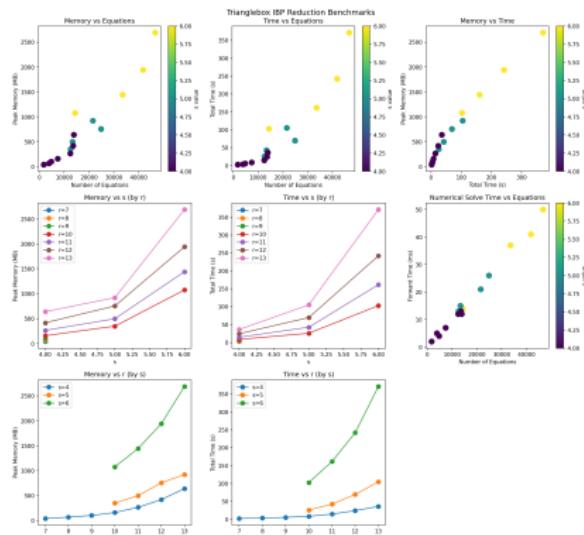


Figure 2: Kira Memory Scaling

Kira benchmarks: Memory grows exponentially with integral weight (r), reaching 2.5+ GB for $r=13$.

Our Solution

- ▶ **ML-guided beam search:** Train a neural network to score IBP actions
- ▶ **Hierarchical reduction:** Process sectors from highest to lowest
- ▶ **Parallel execution:** Distribute across Condor cluster for ~10x speedup
- ▶ **Constant memory:** Each one-step reduction is independent - no system accumulation

The Action Space: Key Innovation

Why Traditional IBP is Hard

- ▶ Infinite possible IBP identities (any seed, any operator)
- ▶ No clear way to choose which identity to apply
- ▶ Laporta: build giant linear system, solve globally

Our Approach: Finite Action Space

- ▶ Restrict to actions that **solve for a specific target integral**
- ▶ Each action = (IBP operator, seed shift)
- ▶ Enumerate only **valid actions** that eliminate the target
- ▶ Enables ML: classification over finite action set

Action Representation

Action = (ibp_op, delta)

- ▶ **ibp_op**: Which IBP/LI operator (0-15)
- ▶ **delta**: Shift from target to seed integral

Example

Target: $I[2,0,1,0,1,1,0]$, Action: (3, (-1,0,0,0,0,0,0))

IBP eq (op=3, seed=[1,0,1,0,1,1,0]):

$$c_1 \cdot I[2,0,1,0,1,1,0] + c_2 \cdot I[1,0,1, \dots] + \dots = 0$$

↑ target

Solve for target → express in terms of simpler integrals

Direct vs Indirect Actions

Direct Actions

- ▶ Target appears directly in the raw IBP equation
- ▶ Straightforward: apply IBP, solve for target

Indirect Actions (Key Innovation)

- ▶ Target does **not** appear in raw IBP equation
- ▶ But appears **after applying substitutions** from previous steps

Raw IBP: $c_1 \cdot I[A] + c_2 \cdot I[B] = 0$ (no target)

After subs: $c_1 \cdot I[A] + c_2 \cdot (\dots + c_3 \cdot I[\text{target}] + \dots) = 0$
↑ target appears!

Indirect actions leverage reduction history for deeper reductions

Subsector Filtering

The Problem

Arbitrary IBP actions can introduce integrals in **higher sectors** → explosion

Solution: Subsector Constraint

Only allow actions where all resulting integrals are in **subsectors** of target

Sector Hierarchy

Sector 63 [1,1,1,1,1,1] (6 propagators)

↓ subsectors

Sector 62 [0,1,1,1,1,1] (5 propagators)

Sector 61 [1,0,1,1,1,1] (5 propagators)

↓

...lower sectors...

Result: Reductions flow downward through sector hierarchy

Why This Works

Finite + Learnable

- ▶ Typically 10-100 valid actions per state
- ▶ Model learns which actions lead to successful reductions

Hierarchical Structure

- ▶ Subsector filtering ensures monotonic progress
- ▶ Never introduces integrals harder than current target

Substitution Chains

- ▶ Indirect actions enable multi-step reasoning
- ▶ Model implicitly learns useful substitution patterns

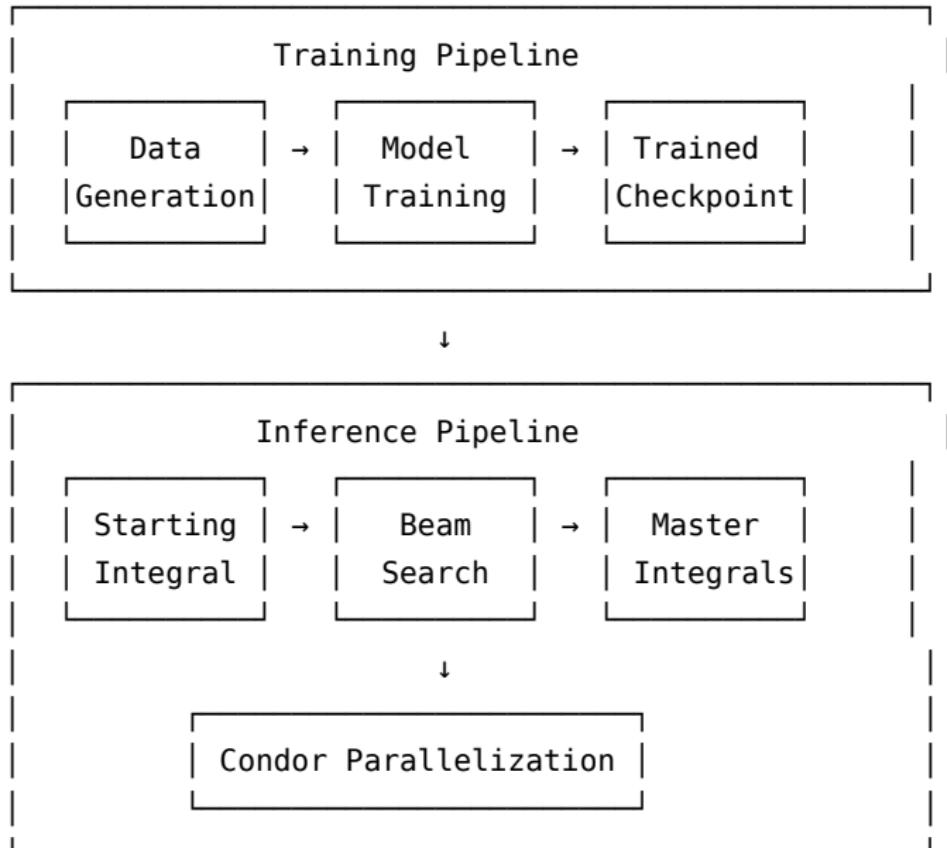
Key Results

Triangle-Box Topology (arXiv:2502.05121)

| Integral | Weight | Seq. | Par. | Speedup | Masters |
|---------------------|--------|---------------|----------------|-------------|---------|
| $I[2,0,2,0,1,1,0]$ | (6,0) | 5 min | - | - | 4 |
| $I[1,1,1,1,1,1,-3]$ | (6,3) | 73 min | 12 min | 6x | 16 |
| $I[3,2,1,3,2,2,-6]$ | (13,6) | ~20 hr | 115 min | ~10x | 16 |

- ▶ Results match Kira exactly
- ▶ Reduces to exact 16 paper masters from arXiv:2502.05121

System Architecture



Part 1: Data Generation

Data Generation: Two-Phase Approach

Phase 1: Scrambling

1. Choose a **random sector** from 63 non-trivial sectors
2. Start from random linear combination of **that sector's masters**
3. Apply random IBP identities to increase complexity
4. **Record** each IBP used: $[(op_1, seed_1), (op_2, seed_2), \dots]$
5. Only use IBPs that stay within target sector (no higher sectors)

Phase 2: Unscrambling (Oracle)

Replay the scramble in reverse to generate training samples

Unscrambling: Step by Step

At each step:

1. Find **target** = highest-weight non-master in expression
2. Look up which recorded IBP can eliminate target (**oracle**)
3. Enumerate **all valid actions** that could eliminate target
4. Record training sample: (expr, target, valid_actions, oracle_choice)
5. Apply action: add target → solution to substitutions
6. Repeat until only masters remain

Key: Oracle provides perfect labels

- ▶ No expensive search needed — answer comes from scramble record
- ▶ Model learns to predict oracle's choice among valid alternatives

Data Generation: Coverage

Sector Coverage

- ▶ All 63 non-trivial sectors covered
- ▶ Uses 16 paper masters for their respective sectors
- ▶ Uses corner integrals for remaining sectors

Dataset Statistics

| Split | Samples | Size |
|------------|----------|--------|
| Train | 946,168 | 3.8 GB |
| Validation | 118,271 | 480 MB |
| Test | ~118,000 | 480 MB |

Data Format

Each training sample contains:

```
{  
    'sector_mask': [1,0,1,0,1,1],  # 6-bit sector encoding  
    'expr': [                      # Current expression  
        ([1,0,2,0,1,1,0], 107),     # (integral, coefficient)  
        ([1,0,1,0,1,1,0], 303),  
        ...  
    ],  
    'subs': [                      # Substitution history  
        (key_integral, [(repl_int1, coeff1), ...]),  
        ...  
    ],  
    'target_integral': [1,0,2,0,1,1,0], # Integral to eliminate  
    'valid_actions': [(ibp_op, delta), ...],  
    'label': 3  # Index of correct action  
}
```

Part 2: Model Architecture

Model: IBPActionClassifierV5

Overview

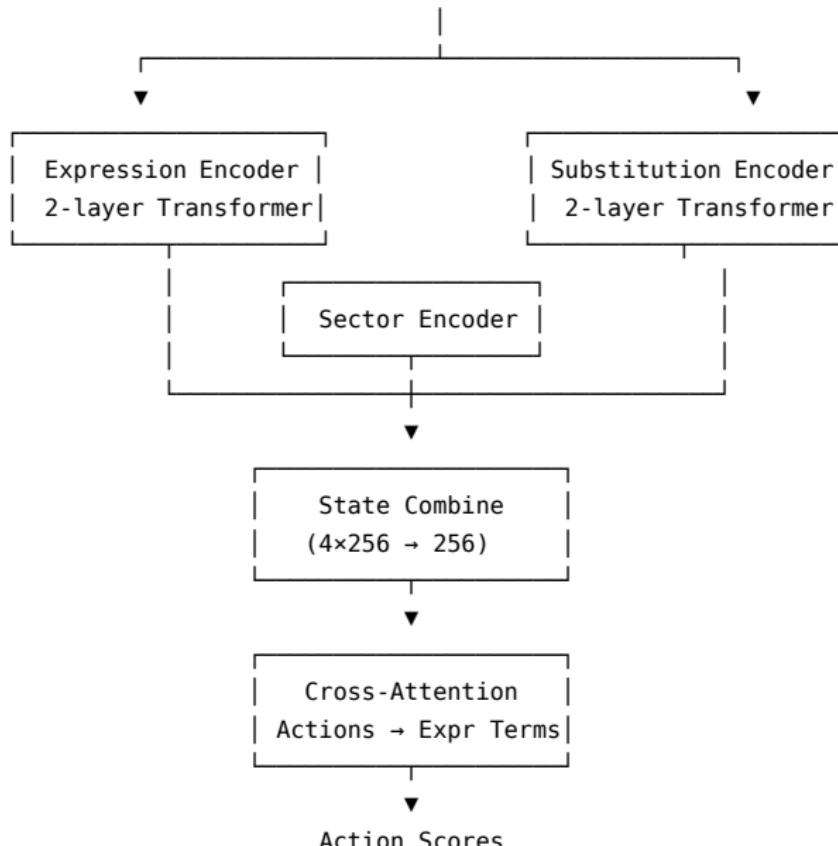
- ▶ **Task:** Score candidate IBP actions given current reduction state
- ▶ **Architecture:** Transformer-based with specialized encoders
- ▶ **Parameters:** 7.7M
- ▶ **Best validation accuracy:** 90.77%

Key Innovation (V5)

Full substitution encoding - encode not just the key integral but the complete replacement expression

Model Architecture Diagram

Inputs: Expression, Target, Substitutions, Sector, Actions



Component Details

| Component | Purpose |
|-------------------------------|-----------------------------|
| Expression Encoder | Encode expression + target |
| Substitution Encoder | Encode reduction history |
| Sector Encoder | Condition on sector |
| Cross-Attention Scorer | Score actions vs expression |

All use 2-layer architectures (Transformer or cross-attention)

Model Hyperparameters

- ▶ Embedding dimension: 256, Attention heads: 4
- ▶ Total parameters: 7,696,709

Part 3: Training

Training Configuration

```
epochs = 30
batch_size = 256
learning_rate = 0.0004
weight_decay = 1e-5
optimizer = AdamW
```

Training Results

- ▶ **Best checkpoint:** Epoch 22
- ▶ **Validation accuracy:** 90.77% (top-1)
- ▶ **Top-5 accuracy:** ~98%
- ▶ **Training time:** ~800s per epoch on GPU

Training Curves

Key Observations

- ▶ Model converges by epoch 20-25
- ▶ Validation accuracy plateaus at ~91%
- ▶ No significant overfitting observed
- ▶ Top-5 accuracy very high (~98%) - beam search can recover from top-1 mistakes

Part 4: Beam Search Inference

Beam Search Algorithm

Each Step (beam width = 20)

1. Start with **20 states** in the beam
2. For each state, identify **target** = max weight non-master
3. Model ranks valid actions; expand **top 20 actions per state**
4. This produces ~**400 candidate states**
5. Sort by (max_weight, n_non_masters, -model_score)
6. Keep **top 20** → new beam

Termination

- ▶ Stop when best state contains **only master integrals**
- ▶ Model guides *which actions to try*; weight reduction decides *what to keep*

Beam Search Optimizations

P1: Equation Caching ($\sim 3\text{-}10x$ speedup)

- ▶ IBP equation generation is expensive (sympy operations)
- ▶ Cache `get_raw_equation` results
- ▶ Reuse across beam states with shared history

P2: Batched Inference ($\sim 50x$ speedup)

- ▶ Prepare all action candidates as numpy arrays
- ▶ Single batched forward pass through model
- ▶ Eliminates per-action inference overhead

Combined Effect

- ▶ Per-step time: 10-90s \rightarrow 0.1-5s
- ▶ Makes deep reductions feasible

Hierarchical Reduction Strategy

Algorithm

1. Find highest-level sector with non-master integrals
2. Run beam search to eliminate all non-masters in that sector
3. Move to next highest sector
4. Repeat until only masters remain

Example: I[2,0,2,0,1,1,0]

| Sector | Level | Steps |
|--------------|-------|------------|
| 53 | 4 | 53 |
| 52 | 3 | 17 |
| 49 | 3 | 4 |
| 37 | 3 | 42 |
| 21 | 3 | 46 |
| ... | ... | ... |
| Total | - | 176 |

Beam Restart Strategy (V11+)

Problem

After weight improvement, beam often contains suboptimal states that will never succeed

Solution: Beam Restart

1. Run beam search until weight improves
2. **Stop and restart** with only the best state
3. Prunes dead ends, enables deeper exploration

Impact

- ▶ Essential for high-weight integrals
- ▶ $I[1,1,1,1,1,1,-3]$: 1,416 steps across 45 sectors
- ▶ $I[3,2,1,3,2,2,-6]$: 46,345 steps across 62 sectors

Part 5: Parallelization

Parallelization Motivation

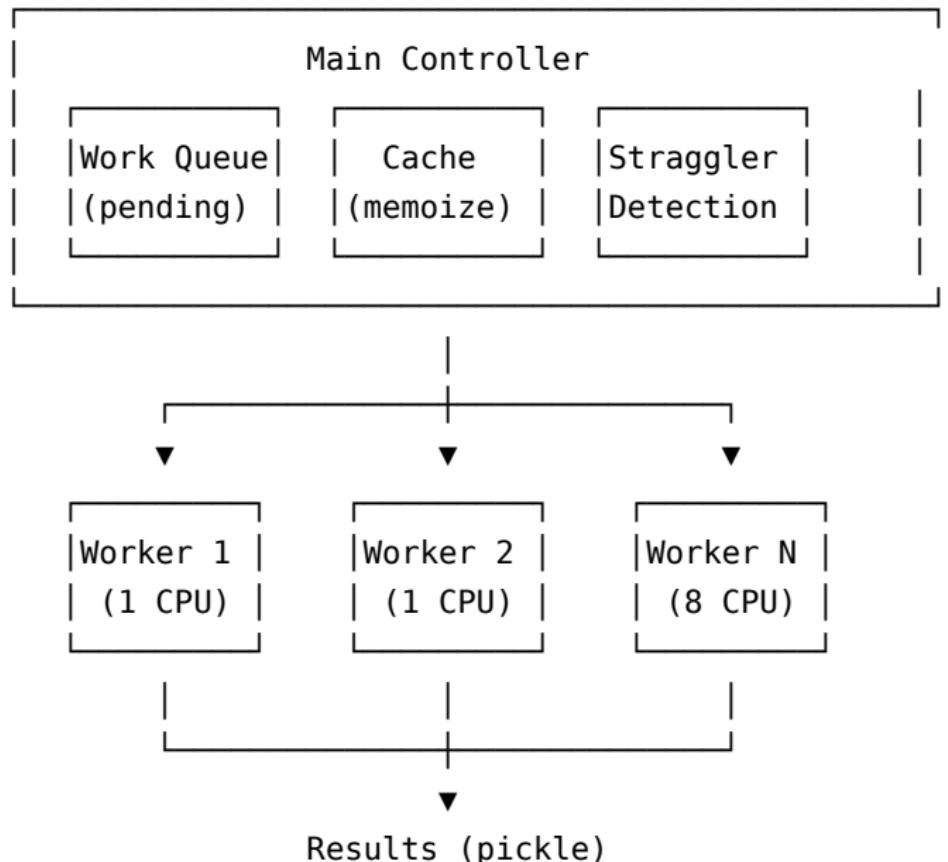
The Bottleneck

- ▶ Sequential reduction of $I[3,2,1,3,2,2,-6]$ takes ~20 hours
- ▶ Single-threaded beam search on CPU
- ▶ Many independent integrals could be processed in parallel

Key Insight

Each one-step reduction is independent - distribute across workers!

Async Parallel Architecture



Key Parallelization Features

1. Async Work Distribution

- ▶ Submit all pending non-masters immediately
- ▶ Don't wait for level synchronization
- ▶ Process results as they arrive

2. Memoization Cache

- ▶ Store: integral → reduced expression
- ▶ Avoid redundant work (~55,000 cache hits!)
- ▶ Critical when stragglers produce already-cached results

3. Straggler Detection

- ▶ Jobs >30 min are killed and resubmitted
- ▶ Resubmit with 8 CPUs (parallel beam search)
- ▶ Prevents slow integrals from blocking progress

Parallel Performance

I[3,2,1,3,2,2,-6] Results

| Metric | Value |
|------------------------|--------------------|
| Time (sequential) | ~20 hours |
| Time (parallel) | 115 minutes |
| Speedup | ~10x |
| Jobs submitted | 21,096 |
| Stragglers resubmitted | 24 |
| Cache hits | 55,075 |
| Final masters | 16 |

Part 6: Results

Validation Against Kira

$I[2,0,2,0,1,1,0]$ Coefficient Comparison (mod 1009)

| Master Integral | Kira | Our Result | Match |
|---------------------------|------|------------|----------|
| $I[1, -1, 1, 0, 1, 1, 0]$ | 107 | 107 | Exact |
| $I[1, 0, 1, 0, 1, 1, 0]$ | 303 | 303 | Exact |
| $I[1, 0, 1, 0, 1, 0, 0]$ | 752 | 752 | Exact |
| $I[0, 1, 1, 1, 0, 0, 0]$ | 915 | — | Symmetry |
| $I[1, 0, 1, 0, 0, 1, 0]$ | — | 915 | Symmetry |

4th master: $I[0, 1, 1, 1, 0, 0, 0]$ and $I[1, 0, 1, 0, 0, 1, 0]$ related by
k1 \leftrightarrow k2 loop exchange symmetry

Result: Perfect agreement with Kira!

Validation Against Kira

$I[1,1,1,1,1,1,-3]$ Coefficient Comparison (mod 1009)

| Master Integral | Our Coeff | Kira Coeff | Match |
|---------------------|-----------|------------|-------|
| $I[1,0,1,1,1,1,0]$ | 784 | 784 | Exact |
| $I[1,1,0,1,1,1,0]$ | 822 | 822 | Exact |
| $I[1,1,1,1,1,0,0]$ | 514 | 514 | Exact |
| $I[-1,1,1,1,1,0,0]$ | 45 | 45 | Exact |
| $I[0,1,1,1,1,0,0]$ | 600 | 600 | Exact |
| $I[1,-1,1,0,1,1,0]$ | 567 | 567 | Exact |
| $I[1,-1,1,1,1,0,0]$ | 239 | 239 | Exact |
| $I[1,0,0,1,1,1,0]$ | 759 | 759 | Exact |

Validation Against Kira $I[1,1,1,1,1,1,-3]$ (continued)

| Master Integral | Our Coeff | Kira Coeff | Match |
|--------------------|-----------|------------|----------|
| $I[1,0,1,0,1,1,0]$ | 948 | 948 | Exact |
| $I[1,0,1,1,1,0,0]$ | 890 | 890 | Exact |
| $I[1,1,0,1,0,1,0]$ | 761 | 761 | Exact |
| $I[1,1,0,1,1,0,0]$ | 593 | 593 | Exact |
| $I[0,0,1,1,1,0,0]$ | 742 | 742 | Exact |
| $I[1,0,1,0,1,0,0]$ | 258 | 258 | Exact |
| $I[0,1,1,1,0,0,0]$ | 97 | 77 | Symmetry |
| $I[1,0,1,0,0,1,0]$ | 989 | — | Symmetry |

Symmetry pair: $I[0,1,1,1,0,0,0]$ and $I[1,0,1,0,0,1,0]$ related by $k_1 \leftrightarrow k_2$ symmetry. Sum: $(97+989) \bmod 1009 = 77$

Result: All 16 masters validated! (14 exact, 2 symmetry-related)

Validation Against Kira

$I[3,2,1,3,2,2,-6]$ Coefficient Comparison (mod 1009)

| Master Integral | Our Coeff | Kira Coeff | Match |
|---------------------|-----------|------------|-------|
| $I[1,0,1,1,1,1,0]$ | 171 | 171 | Exact |
| $I[1,1,0,1,1,1,0]$ | 854 | 854 | Exact |
| $I[1,1,1,1,1,0,0]$ | 377 | 377 | Exact |
| $I[-1,1,1,1,1,0,0]$ | 160 | 160 | Exact |
| $I[0,1,1,1,1,0,0]$ | 100 | 100 | Exact |
| $I[1,-1,1,0,1,1,0]$ | 647 | 647 | Exact |
| $I[1,-1,1,1,1,0,0]$ | 9 | 9 | Exact |
| $I[1,0,0,1,1,1,0]$ | 197 | 197 | Exact |

Validation Against Kira

I[3,2,1,3,2,2,-6] (continued)

| Master Integral | Our Coeff | Kira Coeff | Match |
|------------------|-----------|------------|----------|
| I[1,0,1,0,1,1,0] | 38 | 38 | Exact |
| I[1,0,1,1,1,0,0] | 674 | 674 | Exact |
| I[1,1,0,1,0,1,0] | 143 | 143 | Exact |
| I[1,1,0,1,1,0,0] | 794 | 794 | Exact |
| I[0,0,1,1,1,0,0] | 502 | 502 | Exact |
| I[1,0,1,0,1,0,0] | 944 | 944 | Exact |
| I[0,1,1,1,0,0,0] | 320 | 267 | Symmetry |
| I[1,0,1,0,0,1,0] | 956 | — | Symmetry |

Symmetry pair: I[0,1,1,1,0,0,0] and I[1,0,1,0,0,1,0] related by $k_1 \leftrightarrow k_2$ symmetry. Sum: $(320+956) \bmod 1009 = 267$

Result: All 16 masters validated! (14 exact, 2 symmetry-related)

Kira time: ~10 min vs our 115 min (Kira uses symbolic reduction + optimizations)

Scaling Results

| Integral | Weight | Time | Steps | Masters |
|-------------------|--------|-----------------|---------|---------|
| I[2,0,2,0,1,1,0] | (6,0) | 5 min* | 526 | 4 |
| I[1,1,1,1,1,1,-3] | (6,3) | 12 min* | 3,784 | 16 |
| I[3,2,1,3,2,2,-6] | (13,6) | 115 min* | 131,769 | 16 |

*With parallel execution (async hierarchical)

Part 7: Conclusions

Key Contributions

1. **ML-guided IBP reduction** that matches professional software (Kira)
2. **Constant memory usage** - avoids the memory wall of traditional approaches
3. **Hierarchical beam search** with restart strategy for deep reductions
4. **Async parallel execution** with ~10x speedup
5. **Straggler handling** for robust distributed computing
6. **Paper-masters-only mode** for clean minimal basis

Technical Innovations

Model

- ▶ Full substitution encoding (V5)
- ▶ Cross-attention action scoring
- ▶ Target-aware expression encoding

Inference

- ▶ Equation caching (3-10x speedup)
- ▶ Batched model inference (50x speedup)
- ▶ Beam restart strategy

Parallelization

- ▶ Async one-step distribution
- ▶ Memoization cache (~55k hits)
- ▶ Automatic straggler resubmission

Future Work

1. **GPU workers** for faster beam search
2. **Adaptive timeouts** based on sector statistics
3. **More integrals** - test on other two-loop families
4. **Training on successful paths** - use reduction results to improve model
5. **Path optimization** - shorten saved reduction paths

Code Availability

Repository: github.com/davidshih17/RL_IBPrediction_claude

Key Files

- ▶ `models/classifier_v5.py` - Model architecture
- ▶ `scripts/eval/hierarchical_reduction_async.py` - Parallel reduction
- ▶ `scripts/eval/reduce_integral_onestep_worker.py` - Condor worker
- ▶ `docs/parallelization.md` - Detailed documentation

Thank You

Questions?

Contact

- ▶ Repository:
github.com/davidshih17/RL_IBPrediction_claude
- ▶ arXiv: 2502.05121 (paper masters reference)