Experimental Protocol:
Neural Network Track Extrapolators for LHCb

G. Scriven

January 2026

Contents

1 Introduction and Objectives

This document defines the experimental protocol for evaluating neural network-based track extrapolators as potential replacements or supplements to the classical Runge-Kutta extrapolators used in LHCb track reconstruction.

1.1 Primary Research Questions

1. Can neural networks achieve sufficient accuracy (sub-micron position, sub-microradian angle) for LHCb track extrapolation?
2. Does physics-informed training (PINN) improve generalization compared to pure data-driven approaches?
3. What is the inference speed compared to classical Runge-Kutta integration?
4. How does performance vary across the LHCb momentum spectrum (0.5–100 GeV)?

1.2 Success Criteria

- **Position accuracy**: $\sigma_x, \sigma_y < 10$ μm (target: < 1 μm)
- **Angular accuracy**: $\sigma_{t_x}, \sigma_{t_y} < 10$ μrad (target: < 1 μrad)
- **Speed**: $\geq 10\times$ faster than Runge-Kutta extrapolator
- **Bias**: Mean residuals < 0.1 μm (position), < 0.1 μrad (angle)

2 Experimental Design

2.1 Dataset Specification

2.1.1 Training Data

Table 1: Training dataset specification

Parameter	**Value**
Total samples	50,000,000 tracks
Generation method	Runge-Kutta integration (ground truth)
Field map	Real LHCb dipole (twodip.rtf, 81×81×146 grid)
Propagation	$z_{\text{start}} = 4000$ mm to $z_{\text{end}} = 12000$ mm
Step size Δz	8000 mm (single-step extrapolation)
Input features X $\in \mathbb{R}^6$	
Position	$x, y \in [-4000, 4000]$ mm
Slopes	$t_x, t_y \in [-0.3, 0.3]$
Charge/momentum	$q/p \in [-2, 2]$ GeV^{-1} (both charges)
Step size	$\Delta z = 8000$ mm
Output targets Y $\in \mathbb{R}^4$	
Final position	x', y' (mm)
Final slopes	t'_x, t'_y (dimensionless)

Table 2: Momentum-split datasets

Dataset	Range (GeV)	Samples	Mean p	Physics Regime
Low-p	0.5–5	10M	1.95 GeV	Multiple scattering dominant
Mid-p	5–20	10M	10.8 GeV	Typical LHCb tracks
High-p	20–100	10M	49.7 GeV	Minimal bending

2.1.2 Momentum-Specific Datasets

For momentum-dependent studies, the main dataset is filtered into three ranges:

2.1.3 Test/Validation Data

- 10% of training data held out for validation during training
- Separate test set generated with different random seed (1M tracks)
- Additional "stress test" datasets with extreme parameters

2.2 Model Architectures

Three model families are evaluated:

2.2.1 1. Standard MLP (Baseline)

Pure feedforward network trained on data loss only:

$$\mathcal{L}_{\text{MLP}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - f_\theta(\mathbf{x}_i)\|^2 \tag{1}$$

2.2.2 2. Physics-Informed Neural Network (PINN)

Incorporates Lorentz force equation as soft constraint:

$$\mathcal{L}_{\text{PINN}} = \mathcal{L}_{\text{data}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} \tag{2}$$

where the PDE loss enforces:

$$\frac{d^2x}{dz^2} = \frac{q}{p}\sqrt{1+t_x^2+t_y^2}\left(t_x t_y B_x - (1+t_x^2)B_y + t_y B_z\right) \tag{3}$$

$$\frac{d^2y}{dz^2} = \frac{q}{p}\sqrt{1+t_x^2+t_y^2}\left((1+t_y^2)B_x - t_x t_y B_y - t_x B_z\right) \tag{4}$$

2.2.3 3. RK-PINN (Runge-Kutta Inspired)

Hybrid architecture that internally performs multiple sub-steps mimicking RK4 integration, with physics-informed loss at each sub-step.

2.3 Architecture Sizes

3 Experiment 1: Architecture Comparison

3.1 Objective

Determine optimal network size balancing accuracy and inference speed.

Table 3: Network architecture configurations

Name	Hidden Layers	Total Params	Purpose
Tiny	[64, 64]	$\sim$5K	Minimum viable, speed test
Small	[128, 128, 64]	$\sim$20K	Lightweight production
Medium	[256, 256, 128]	$\sim$100K	Baseline comparison
Wide	[512, 512, 256]	$\sim$400K	Maximum accuracy

3.2 Experiments (12 jobs)

Table 4: Architecture comparison experiments

Model	Architecture	λ_{PDE}	λ_{IC}
mlp_tiny	[64, 64]	–	–
mlp_small	[128, 128, 64]	–	–
mlp_medium	[256, 256, 128]	–	–
mlp_wide	[512, 512, 256]	–	–
pinn_tiny	[64, 64]	1.0	1.0
pinn_small	[128, 128, 64]	1.0	1.0
pinn_medium	[256, 256, 128]	1.0	1.0
pinn_wide	[512, 512, 256]	1.0	1.0
rkpinn_tiny	[64, 64]	1.0	1.0
rkpinn_small	[128, 128, 64]	1.0	1.0
rkpinn_medium	[256, 256, 128]	1.0	1.0
rkpinn_wide	[512, 512, 256]	1.0	1.0

3.3 Metrics to Record

- Training loss curves (data, PDE, IC components)
- Validation loss at each epoch
- Final test accuracy (MAE, RMSE, max error per output)
- Training time (wall clock and GPU hours)
- Number of parameters
- Inference time (batch sizes: 1, 100, 10000)

3.4 Analysis

1. Plot accuracy vs. model size (Pareto frontier)
2. Plot accuracy vs. inference time
3. Determine minimum architecture meeting accuracy criteria

4 Experiment 2: Physics Loss Ablation

4.1 Objective

Quantify the impact of physics-informed training on accuracy and generalization.

4.2 Experiments (8 jobs)

Using medium architecture [256, 256, 128]:

Table 5: Physics loss ablation experiments

Name	$\lambda_{\mathbf{PDE}}$	$\lambda_{\mathbf{IC}}$	Description
pinn_medium_data_only	0.0	0.0	Pure data-driven (MLP baseline)
pinn_medium_pde_weak	0.1	0.1	Weak physics regularization
pinn_medium (default)	1.0	1.0	Balanced physics/data
pinn_medium_pde_strong	10.0	10.0	Strong physics emphasis
pinn_medium_pde_dominant	100.0	100.0	Physics-dominant training
rkpinn_medium_data_only	0.0	0.0	RK architecture, data only
rkpinn_medium_pde_weak	0.1	0.1	RK + weak physics
rkpinn_medium_pde_strong	10.0	10.0	RK + strong physics
rkpinn_medium_pde_dominant	100.0	100.0	RK + physics dominant

4.3 Metrics to Record

- Loss component breakdown (data vs PDE vs IC)
- Convergence speed (epochs to reach threshold)
- Generalization gap (train vs test error)
- Physical consistency metrics (energy conservation, Lorentz force satisfaction)

4.4 Analysis

1. Plot test accuracy vs. λ_{PDE}
2. Examine learning curves for signs of physics constraint helping/hurting
3. Compare generalization: train on mid-p, test on low/high-p

5 Experiment 3: Momentum-Dependent Performance

5.1 Objective

Characterize model performance across the LHCb momentum spectrum.

5.2 Experiments (9 jobs)

Train dedicated models on each momentum range:

5.3 Metrics to Record

- Accuracy as function of momentum within each range
- Cross-range generalization (train low-p, test high-p and vice versa)
- Residual distributions binned by momentum

Table 6: Momentum-specific experiments

Model	Momentum Range	Physics Challenge
mlp_medium_low_p	0.5–5 GeV	Large bending, multiple scattering
mlp_medium_mid_p	5–20 GeV	Typical LHCb tracks
mlp_medium_high_p	20–100 GeV	Small bending angles
pinn_medium_low_p	0.5–5 GeV	Strong field gradients
pinn_medium_mid_p	5–20 GeV	Moderate curvature
pinn_medium_high_p	20–100 GeV	Near-linear trajectories
rkpinn_medium_low_p	0.5–5 GeV	Sub-stepping benefits?
rkpinn_medium_mid_p	5–20 GeV	Standard regime
rkpinn_medium_high_p	20–100 GeV	Overkill for straight tracks?

5.4 Analysis

1. Is a single model sufficient, or do we need momentum-specific models?
2. Does PINN help more at low-p where physics is more complex?
3. Can high-p model be simpler (fewer parameters)?

6 Experiment 4: Timing Benchmarks

6.1 Objective

Compare inference speed of neural networks vs. classical Runge-Kutta extrapolator.

6.2 Methodology

6.2.1 Neural Network Inference

1. Load trained model (PyTorch and ONNX versions)
2. Warm-up: 100 forward passes (discard)
3. Benchmark: 1000 forward passes, record mean and std
4. Batch sizes: 1, 10, 100, 1000, 10000, 100000
5. Hardware: CPU (single core), CPU (multi-core), GPU (CUDA)

6.2.2 Runge-Kutta Baseline

1. Use existing `TrackRungeKuttaExtrapolator`
2. Same track parameters as NN benchmark
3. Record time for equivalent number of extrapolations
4. Test different step sizes (adaptive vs fixed)

6.3 Metrics to Record

- Wall-clock time per extrapolation
- Throughput (tracks/second)
- Memory usage
- Speedup factor vs. RK baseline
- Latency distribution (important for real-time trigger)

6.4 Hardware Configurations

- CPU: Intel Xeon (typical HLT1 node)
- GPU: NVIDIA A100 / V100 (if available in HLT)
- Evaluate ONNX Runtime optimizations

7 Experiment 5: Generalization and Robustness

7.1 Objective

Test model robustness to out-of-distribution inputs and edge cases.

7.2 Test Cases

1. **Interpolation test**: Random samples within training domain
2. **Boundary test**: Tracks near edge of acceptance
3. **Extrapolation test**:
    - Momentum slightly outside training range (0.4 GeV, 110 GeV)
    - Extreme slopes ($|t_x|, |t_y| > 0.3$)
    - Unusual starting positions
4. **Field variation**: Different field map versions (if available)
5. **Step size variation**: $\Delta z \neq 8000$ mm

7.3 Metrics

- Error degradation outside training domain
- Failure modes (numerical instability, NaN outputs)
- Comparison with RK extrapolator on same edge cases

8 Experiment 6: Learning Dynamics Analysis

8.1 Objective

Understand training behavior and optimization landscape.

8.2 Analyses

1. **Loss landscape visualization**
   - 1D and 2D loss surface plots
   - Identify local minima, saddle points
2. **Gradient flow analysis**
   - Gradient norms per layer during training
   - Detect vanishing/exploding gradients
   - Compare MLP vs PINN gradient dynamics
3. **Loss component evolution**
   - Plot $\mathcal{L}_{\text{data}}$, $\mathcal{L}_{\text{PDE}}$, $\mathcal{L}_{\text{IC}}$ separately
   - Identify competition between objectives
   - Optimal λ scheduling?
4. **Feature importance**
   - Which inputs most affect outputs?
   - Sensitivity to q/p vs position vs slopes

9 Additional Suggested Experiments

Based on typical challenges in physics-informed machine learning:

9.1 A. Activation Function Study

Compare ReLU, Tanh, SiLU (Swish), GELU. Physics-informed networks often benefit from smooth activations (Tanh, SiLU) due to gradient requirements.

9.2 B. Normalization Strategy

- Input normalization: Z-score vs min-max vs physics-based
- Output scaling: Direct mm/μrad vs normalized residuals
- Batch normalization: May interfere with physics loss

9.3 C. Training Data Volume Study

Train on 1M, 5M, 10M, 25M, 50M samples. Determine data efficiency and whether PINN requires less data than pure MLP.

9.4 D. Multi-Step Extrapolation

Train single model, apply recursively:

- Train on $\Delta z = 1000$ mm
- Apply $8\times$ for full 8000 mm extrapolation
- Compare accuracy vs single-step model
- Error accumulation analysis

9.5 E. Uncertainty Quantification

- Ensemble methods (multiple models)
- MC Dropout for epistemic uncertainty
- Heteroscedastic outputs (predict mean + variance)
- Important for downstream track fitting

9.6 F. Transfer Learning

- Train on simplified field, fine-tune on real field
- Train on one z-region, transfer to another
- Pre-train on simulation, fine-tune on data (if available)

9.7 G. Integration with Track Reconstruction

- Replace RK extrapolator in full reconstruction chain
- Measure impact on track finding efficiency
- Measure impact on momentum resolution
- End-to-end physics performance (mass resolution, etc.)

10 Data Analysis and Visualization

10.1 Standard Plots for Each Experiment

1. **Training curves**: Loss vs epoch (train and validation)
2. **Residual distributions**: Histograms of $(y_{\text{pred}} - y_{\text{true}})$ for each output
3. **2D residual maps**: Residuals vs input variables (identify systematic patterns)
4. **Pull distributions**: $(y_{\text{pred}} - y_{\text{true}})/\sigma$ should be Gaussian
5. **Momentum dependence**: Accuracy metrics binned by momentum
6. **Correlation plots**: Predicted vs true for each output

10.2 Summary Tables

Each experiment produces a row in the master results table with columns:

- Model name, architecture, parameters
- Training time (GPU-hours)
- Final train/val/test loss
- MAE and RMSE for x, y, t_x, t_y
- Max absolute error
- Inference time (various batch sizes)
- Notes and observations

11 Timeline and Resources

11.1 Computational Requirements

Table 7: Estimated computational requirements

Experiment Set	Jobs	Est. GPU-hours/job	Total GPU-hours
Architecture comparison	12	2–8	60
Physics ablation	8	4	32
Momentum studies	9	4	36
Total core experiments	**29**	–	**∼130**
Additional experiments	20–30	varies	∼100

11.2 HTCondor Job Status

Submitted: January 22, 2026

- Core + ablation experiments: Clusters 3880122–3880142 (20 jobs)
- Momentum studies: Clusters 3880158–3880166 (9 jobs)

11.3 Post-Training Workflow

1. Collect results from `trained_models/<exp_name>/`
2. Extract metrics from `history.json` files
3. Run unified analysis notebook
4. Generate paper figures
5. Update model registry with best configurations

12 Model Registry Protocol

After training completes, update the model registry:

1. **Location**: `trained_models/registry.json`
2. **Fields per model**:
   - Unique identifier (experiment name)
   - Path to saved weights
   - Architecture specification
   - Training hyperparameters
   - Performance metrics (test loss, accuracy)
   - Timestamp and git commit
3. **Best model selection**:
   - Mark best overall model
   - Mark best per category (fastest, most accurate, best generalization)

13 Appendix: File Locations

```
experiments/next_generation/
+-- data_generation/
|   +-- data/
|       +-- training_50M.npz      # Main training data (50M tracks)
|       +-- training_low_p.npz    # Low momentum (10M, 0.5-5 GeV)
|       +-- training_mid_p.npz    # Mid momentum (10M, 5-20 GeV)
|       +-- training_high_p.npz   # High momentum (10M, 20-100 GeV)
+-- training/
|   +-- jobs/                     # HTCondor .sub files (29 experiments)
|   +-- logs/                     # Job output/error logs
|   +-- train_wrapper.sh          # HTCondor execution wrapper
+-- trained_models/
|   +-- <exp_name>/
|       +-- model.pt              # PyTorch weights
|       +-- config.json           # Architecture + hyperparameters
|       +-- history.json          # Training loss history
+-- analysis/
|   +-- analyze_results.ipynb     # Post-training analysis
+-- notes/
    +-- experimental_protocol.tex # This document
```