FAST AND PRECISE LEARNED CHARGED-PARTICLE TRAJECTORY REGRESSION AT THE LARGE HADRON COLLIDER

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a training recipe that treats charged-particle trajectory parameter regression on high-energy physics detector data as a sequence-modeling task. Kalman filters and linearized least-squares fits have been the classical standard approach for this task: they are optimal estimators for sparsely sampled linear-Gaussian data and are commonly used for trajectory parameter regression (fitting). The classical fitting techniques implemented for this domain reach a final precision of one part in 10^5 through detailed modeling of detector geometry, material, detection effects and precise numerical integration of the equations of motion through the detector's inhomogeneous magnetic field. With this study, we demonstrate that using a bidirectional gated linear recurrent encoder, one is able to reproduce the full precision of classical track fitting techniques. Using a custom kernel, we also achieve significantly higher throughput during GPU inference, compared to classical fitting software running on similarly priced multi-core CPU servers representing typically employed hardware. Such a speedup would lead to considerable cost savings for the pattern recognition at the Large Hadron Collider. To our knowledge, this is the first end-to-end learned track fit to reach the full precision and, at the same time, offer the opportunity to reduce the computing costs.

1 INTRODUCTION

Machine learning approaches to data processing problems in high-energy physics are an active field of R&D. Detecting the patterns of high-energy charged particles produced in proton–proton collisions at experiments at CERN such as ATLAS and CMS [1, 2] and regressing their trajectory parameters is one of the most computationally complex tasks required to extract physics information from the massive datasets acquired by the experiments (illustrated in Fig. 1).

This charged particle "tracking" task is a "connect-the-dots" game: each particle traverses layers of detector material; it scatters and deposits energy through ionization in particle detectors. Each such energy deposit in a detector we call a "hit". A pattern recognition algorithm has to recover which hits belong to which particle and what its trajectory was. The challenge grows in compute complexity when many particles arrive at once (Fig. 1). At the High-Luminosity Large Hadron Collider (HL-LHC), which is scheduled to start taking data in 2030, every $25\,\mathrm{ns}$ a crossing of two proton bunches will give rise to on average 200 concurrent proton–proton interactions and in turn yield of the order of half a million hits across the detectors of the tracking sub-system [3–6]. The pattern recognition algorithms reconstruct between 1000 and 2000 trajectories for such an event. Regressing the trajectory parameters with the highest precision for each particle trajectory is fundamental to achieving the physics goals of the experiments.

The Kalman filter is a classical, optimal estimator for this regression task, which linearizes the problem using numerical field integration [8] for the particle transport in the detector's magnetic field. A Kalman filter is itself a state-space model [9, 10, 8]. This motivated our choice of encoder: its state update has the same structure as the Kalman filter update, as detailed in Section 3.2. We use this correspondence as a guide for our design, not as a claim that other encoders or further optimization could produce architectures achieving higher throughput. We tested four encoder families at matched parameter count under an identical training recipe (Fig. A.2). Three of them, a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

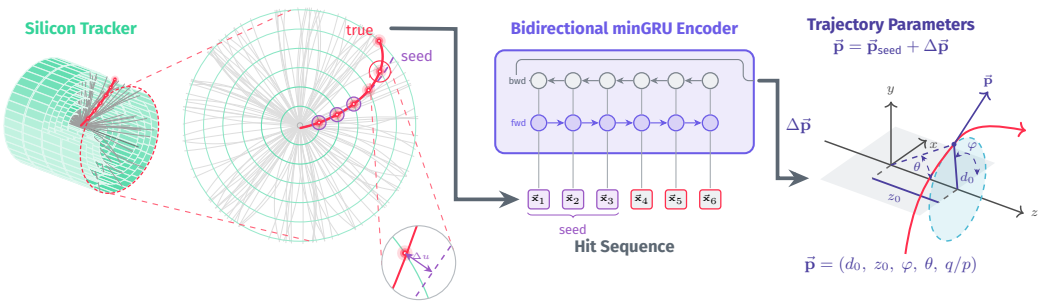

Figure 1: Charged-particle track fitting as sequence regression. A charged particle's hits across the silicon layers of the tracking detector [7] form an ordered sequence of feature vectors $\vec{\mathbf{x}}_N$. An analytic helix through the first three hits gives a seed estimate $\vec{\mathbf{p}}_{\text{seed}}$; each hit carries measured coordinates plus three seed residuals: Δu, Δv, compressed with asinh, and s_{helix}. A bidirectional minGRU encoder reads the sequence and predicts a correction $\Delta\vec{\mathbf{p}}$ to the seed, giving the perigee parameters $\vec{\mathbf{p}} = (d_0, z_0, \varphi, \theta, q/p)$ at closest approach to the collision axis. Schematic, not to scale.

bidirectional minGRU [11], a bidirectional Mamba-2 [12] and a Transformer [13], reach the reference precision. A parameter-matched diagonal state-space model without an input-dependent gate fell short under the same protocol, most clearly in the momentum estimate. Of the three, the bidirectional minGRU encoder gave the highest throughput in our implementation (Section 3.5), and we use it for all main results.

To our knowledge, this is the first learned trajectory fit to match classical Kalman-filter precision, offering the potential for significant cost savings through higher throughput at comparable hardware cost, and for replacing, within two days of training directly on simulated data, months of detector-specific software development usually required for a new detector design study.

1.1 Related work

Learned approaches to charged-particle reconstruction at the LHC have so far mostly focused on the "trajectory finding" problem, the combinatorial assignment of detector hits to candidate trajectories, rather than on "trajectory fitting", the regression of trajectory parameters at the precision downstream physics requires. The most-developed line uses graph neural networks. The ATLAS GNN4ITk pipeline [14] hands track candidates to the classical Kalman filter for parameter estimation. The complementary transformer line is exemplified by the MaskFormer-style segmentation plus reconstruction architecture of Van Stroud et al. [15]. It similarly targets joint hit assignment and per-trajectory property estimation, but reports its main metrics on assignment efficiency and fake rate. The closest prior attempt at learned "trajectory fitting" on the same detector is the transformer-based study of Couthures et al. [16], which targets parameter regression on the Open Data Detector [7] with full ACTS [17] simulation; the reported per-parameter resolutions approach the Kalman filter results without matching them.

Kalman filter fitting also runs on GPUs in production where the propagation step admits a detector-specific shortcut, e.g. at LHCb and ALICE experiments [18, 19]. For general-purpose silicon trackers, where material effects and field integration cannot be simplified this way, GPU-based Kalman fitting remains a topic of R&D with only modest gains in computing costs so far [20]. The obstacle is structural: the propagation between hits is data-dependent (which surface comes next, how far to step, which material to cross) and is, by itself, an iterative numerical field integration task. It does not reduce to one dense kernel shared by every trajectory, and GPU threads diverge instead. Our learned fit has no propagation step at inference: geometry, material and field are absorbed into the weights during training, so every trajectory regression results in the same dense computation task and parallelizes trivially, at Kalman filter precision.

2 DATA

We use the COLLIDERML dataset [6, 21], publicly available at huggingface.co/datasets/CERN, an open-source dataset of fully simulated $\sqrt{s} = 14$ TeV proton–proton physics (collisions with a 14 TeV center-of-mass energy between the two colliding proton bunches) on the "Open Data Detector" (ODD) [7], an open community silicon-tracker geometry to facilitate algorithmic R&D (cutaway in Fig. 1). The detector sits inside a solenoid magnet that produces an almost uniform $3\,\mathrm{T}$ magnetic field along the collision axis, which forces charged particles onto curved trajectories and in turn allows their momentum to be inferred from that curvature. As this work focuses only on the regression problem within the track reconstruction chain, we assume perfect hit-to-track matching, so the learned minGRU and the baseline Kalman filter both receive exactly the same sequence of hits.

Each silicon hit contributes 12 measured features (position, derived angles, and detector identifiers); three further per-hit features derived from an analytic helix seed are described in Section 3. The positions measured at the sensor were overlaid with Gaussian smearing, in order to simulate the actual detection precision. Hits are ordered by the time at which they were measured, which gives the learned minGRU a meaningful inductive bias about positional information, letting it retrace the original trajectory the particle took through the detector.

Training uses a mixture of 381 M tracks: single muons uniform in transverse momentum $p_{\mathrm{T}} \in [1, 110]$ GeV (191.5 M), and single muons log-uniform in $p_{\mathrm{T}} \in [0.9, 110]$ GeV ($\sim$190 M) supplying low-momentum statistics, each required to have $|\eta| \leq 3$ and 6–20 hits in the tracking system[1]. Muons are elementary particles that serve as the field's standard calibration candle for any new charged-particle reconstruction algorithm.

Evaluation uses four held-out muon samples disjoint from all training data: single muons at fixed p_{T} of 2, 10, and 50 GeV (10^5 tracks each) and a uniform 1–70 GeV muon spectrum (3 M tracks). The fixed-p_{T} samples cover the range from the low momentum regime (2 GeV), in which multiple scattering (deflection of the particle by the detector material) is the dominating effect, to the high momentum regime (50 GeV) for which the precision is only limited by the intrinsic resolution of the detector measurements.

The standard metric for evaluating a track fitter in particle physics is an iteratively 3σ-clipped RMS of the residuals, the spread of the central body of the residual distribution after a fixed-rule outlier rejection. It is calculated by iteratively clipping away residuals lying outside of a 3σ range until the distribution stabilizes. This is not the plain RMS the wider machine-learning literature usually reports; on heavy-tailed data the plain RMS is dominated by a small number of outliers and obscures the behavior of the model on the bulk of the sample. Resolution-curve uncertainties are the analytic RMS standard error per bin.

3 METHOD: A SEED-GUIDED BIDIRECTIONAL MINGRU TRAJECTORY PARAMETER ESTIMATOR

The model maps the input hit sequence of a single track, ordered by hit measurement time, to the five perigee parameters of that track, $(d_0, z_0, \varphi, \theta, q/p)$ that describe the trajectory at its closest approach to the nominal collision axis (Fig. 1). The regression target is defined as the deviation from an initial "seed" perigee estimate, which is analytically derived from the first three hits of the trajectory. The seed perigee together with the deviations per hit allows the model to predict a correction to the seed trajectory, in a similar fashion to how a Taylor-expanded Kalman filter implementation fits for the linear correction term to the "seed" trajectory [8]. Fig. 2 summarizes the architecture of the seed-guided bidirectional minGRU model.

3.1 ANALYTIC SEED

A seeding trajectory estimate constructed from at least three hits is commonly used in classical trajectory finding and reconstruction techniques [17]. We use the seed trajectory for this study

[1] $\eta = -\ln\tan\frac{\theta}{2}$ is the "pseudorapidity" calculated from polar angle θ, a standard collider-physics feature that measures the particle's direction relative to the collision axis ($\eta = 0$ is perpendicular to the beam, larger $|\eta|$ means closer to the collision direction).

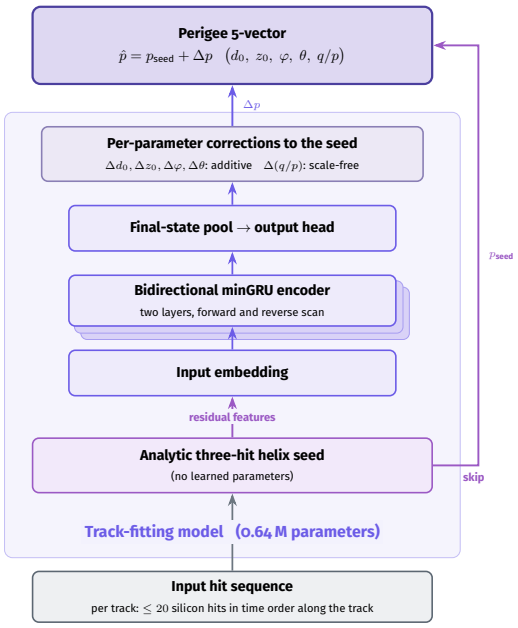

Figure 2: Architecture of the seed-guided bidirectional minGRU model. An analytic three-hit helix seed (top, no learned parameters) supplies residual features to the encoder input and, via the skip connection, anchors the output. The encoder (0.64 M parameters) predicts per-parameter quantile corrections Δp (7 quantiles each, median as the point estimate); the final estimate is $\hat{p} = p_{\text{seed}} + \Delta p$, additive for $d_0, z_0, \varphi, \theta$ and a scale-free residual for q/p.

because anchoring the regression targets to the seed collapses their wide dynamic range into a residual of comparable scale at every momentum, keeping the model's internal representations well conditioned. Consequently, the same precision of the regression results can be achieved using far fewer parameters, in turn resulting in faster inference, compared to a model that learns the absolute value directly. Concretely, a model that learns the absolute value of the perigee parameters without a seed requires roughly $8\times$ more parameters to achieve the same precision as the model incorporating the seed.

A trajectory of a charged particle in an (approximately) homogeneous 3 T magnetic field of the Open Data Detector is described by a helix only in the ideal case. The three hits of a trajectory seed are sufficient to analytically calculate all five seed perigee parameters using a conformal (inversion) map [22] that turns "find the circle through three points" into "draw the straight line through two points" via elementwise closed-form arithmetic, with no fit or iteration involved.[2] The circle's curvature and turning direction give p_{T} and the charge, and the direction w.r.t. the collision axis extracted from the same hits gives $\cot\theta$. Extrapolating the result geometrically to the collision axis defines the perigee (closest point of approach in the plane perpendicular to the collision axis) at which all five parameters $(d_0, z_0, \varphi, \theta, q/p)$ are given. The only inputs to this procedure are the measured hit positions and the detector-volume identifiers: no learned parameters, no geometry or material description. The identical closed-form computation runs inside the model's own forward pass on the GPU. Technically the seed perigee is computed in float64 rather than the surrounding float32 because of the need to avoid a cancellation error in the beamline transport that becomes significant at high momentum.

The seed perigee then enters the model twice. On the input side every hit gains three features: its two signed distances from the seed helix, decomposed in the curvilinear frame the conventional

[2]The seed is a vectorized port of the ACTS track-parameter estimator [17] (`estimateTrackParams FromSeed`, the conformal-map estimate of the ATLAS experiment's [1] silicon seed maker), combined with the triplet rule of the ACTS truth-seeding algorithm. It is closed-form and computed independently per track from (x, y, z), the detector-volume identifier and the constant $B_z = 3\,\mathrm{T}$, nothing else.

linearized Kalman filter [8] operates in as well and compressed with asinh to tame their tails ($\mathrm{asinh}\,\delta u$, $\mathrm{asinh}\,\delta v$), plus its path length along the helix, s_{helix}. On the output side every head predicts a correction to the seed rather than an absolute value, a target-side skip connection through which no gradient flows.

3.2 Encoder

The 15 per-hit features are min–max normalized to $[0, 1]$ and expanded by a multi-scale Fourier featurization with 16 scales [23], giving a 480-d per-hit vector projected through a dense network to the encoder width $d_{\mathrm{model}} = 128$. The backbone stacks two bidirectional minGRU layers [11] with a hidden width of 192 (0.64 M parameters in total). Each layer linearly projects its input to the forward and reverse gates and candidate states, then runs a forward and a reverse scan. The two directions' states are concatenated (384-d) into the next layer.

The Kalman filter is efficient because it processes the hits one at a time with a small state, so its cost grows only linearly with the number of hits [8, 9]. The minGRU shares this structure: its recurrence mirrors the state-update logic of the Kalman filter applied along the trajectory. At every hit, the Kalman filter first propagates a predicted state $\vec{x}_{\,t-1}$ from the previous estimate, then corrects it with the gain $\mathbf{K}_t$ applied to the difference between the measurement $\vec{m}_t$ and its projection $\mathbf{H}_t\vec{x}_{\,t-1}$ into measurement space. Its core assumption is that both process noise (material effects) and measurement noise can be described as Gaussians. The per-hit minGRU update propagates a hidden state $\vec{h}_{t-1}$ corrected by an input-dependent gate $\vec{z}_t$ applied to the difference between a candidate state $\vec{n}_t$, computed from the embedded measurement $\vec{m}_t$, and the previous state. The state updates in a Kalman filter and the minGRU model are

$$\vec{x}_t = \vec{x}_{\,t-1} \;+\; \mathbf{K}_t\big(\vec{m}_t - \mathbf{H}_t\,\vec{x}_{\,t-1}\big), \tag{1}$$

$$\vec{h}_t = \vec{h}_{t-1} \;+\; \vec{z}_t \odot \big(\vec{n}_t - \vec{h}_{t-1}\big). \tag{2}$$

Here $\odot$ denotes an elementwise (Hadamard) product, following the minGRU formulation [11]. Concretely, $\vec{z}_t = \sigma(\mathrm{Linear}(\vec{m}_t))$ is a per-channel, input-dependent gate and $\vec{n}_t = \mathrm{Linear}(\vec{m}_t)$ a candidate state. This gate plays the role of the Kalman gain $\mathbf{K}_t$ in the classical Kalman filter track fit, and, more loosely, of the explicit field integration and material-effect corrections that feed $\mathbf{K}_t$ there, but it is learned from data rather than derived from a complex model. The correspondence is not exact. The Kalman gain is computed from the uncertainty of the current estimate, and therefore depends on all previous hits along the trajectory. The minGRU gate depends only on the current hit, which is what allows all gates to be precomputed.

One default minGRU layer scan is forward-only while the Kalman filter runs a forward as well as reverse scan [17]. This is why we make the minGRU bidirectional per layer: so the hidden state at each hit is informed by both its past and its future along the trajectory.

The final state of the forward scan (at the outermost hit) and the final state of the reverse scan (at the innermost hit) are concatenated to a 384-d representation, normalized [24] and read by a two-layer head (hidden width 128) producing 7 quantiles for each of the five parameters, 35 outputs in total.

3.3 Losses

All five heads are 7-quantile pinball regressions [25] with equal weight, with the median quantile taken as the point estimate, applied to the seed-anchored targets: Δd_0 and Δz_0 within ± 0.4 and $\pm 3.5\,\mathrm{mm}$, $\Delta\varphi$ (wrapped) and $\Delta\theta$ within ± 15 and $\pm 10\,\mathrm{mrad}$, all four simple additive offsets to the seed, and for q/p a "scale-free" residual $(q/p - q/p_{\mathrm{seed}})\,/\,(|q/p_{\mathrm{seed}}| + \epsilon)$ with $\epsilon = 0.02\ \mathrm{GeV}^{-1}$. The scale-free form makes a 1 GeV and a 100 GeV trajectory place the same relative demand on output precision; with an absolute q/p head the residual is $\sim 10^{-4}$ of the head's range and optimizer noise dominates only very late in training.

3.4 Training

Training runs end-to-end in strict fp32, giving comfortable headroom in the mantissa to the 10^{-5} precision this problem demands. The batch size is deliberately small at 2048: a 36 000 batch loses up to a factor of two on the azimuthal and curvature resolutions, with most of the small-batch gain

arriving during the learning-rate anneal. We assume that this is the generalization benefit of gradient noise: at a given learning rate, smaller batches settle in flatter minima and test better, also at a matched number of steps [26–28], and the same preference is reported for continuous targets such as interatomic energies and forces or time-series forecasts [29, 30]. The model is trained in two phases: a first training phase with Lion [31] under a 25-epoch one-cycle schedule on the 381 M-trajectory mixture ($\sim$ 31 h on one H100 [32]), followed by a Muon-AdamW hybrid optimizer annealing phase [33] under a Warmup–Stable–Decay schedule [34] at large batch ($2 \times 20\,000$ across two H100s, 25 epochs, $\sim$15 h). The annealing phase leaves the clipped core resolutions unchanged and cleans the residual tails, closing a final resolution gap of $\approx 3\%$ on q/p.

3.5 Kernel adaptation

Standard GPU kernels for sequence models target sequences of 10^3 to 10^5 tokens. A charged-particle trajectory has at most 20 hits and 13 on average. A kernel built for the longer regime wastes arithmetic and pays launch overhead a track never needs.

We give every encoder we tested the same treatment. Tracks run in a packed, unpadded layout. For every encoder the token mixing of a layer runs as one fused Triton kernel. For the recurrent encoders that kernel scans along the track and updates each channel independently. For the Transformer the bias, activation, normalization and residual addition around the Transformer's matrix multiplications are folded into those multiplications. The per-hit Fourier encoding ahead of the encoder is compiled into a few fused kernels. The GEMMs of the encoder layers, the dense matrix multiplications that map each hit's feature vector to the next layer, run in fp16 for all three encoders. The recurrence itself accumulates in fp32 inside the kernel (Section A.1), and the seed stays float64. This treatment improves throughput by $6.1\times$ for the minGRU, $6.6\times$ for the Transformer, and $3.6\times$ for Mamba-2 (in tracks/s; Table A.1, Section A.1), measured on an H100 GPU. For the non-selective state-space model we trained, no kernel optimization is attempted, since it fell short of the reference precision under our training recipe (Fig. A.2).

All encoders are compared at matched parameter count ($\sim$0.63 M), with an identical recipe and a shared learning rate. We did not tune depth, width or learning rate per encoder, so the throughput ordering reflects our implementations and may change under further per-architecture optimization.

4 Results

This work advances the field along three main directions. First, the minGRU matches the mathematically optimal classical baseline [9, 10, 8] (a Kalman filter) measured within an established validation framework [17] in absolute precision. This demonstrates that the model is able to learn the full complexity of the detailed modeling of classical trajectory fitting software in terms of detector geometry, material effect corrections and magnetic-field integration to achieve the 10^{-5} relative precision the problem demands from data. Second, the fit is embarrassingly parallel across tracks: with a GPU kernel adapted to the special structure of tracking data, our fastest model fits 1.51 M tracks per second on an NVIDIA RTX 5000 Ada GPU. This demonstrates that the minGRU trajectory parameter estimator is able to achieve a significantly higher throughput compared to a classical Kalman filter trajectory fit run on a Threadripper 3970X 32-core CPU machine, which costs about half as much and fits roughly $\sim$170 k tracks per second. Third, the model learns a detector from its simulated data alone in about two days of training on two H100 GPUs: no hand-coded geometry description, material map, or field integration enters the parameter estimation model. For detector design studies for the Future Circular Collider [35], the training of a seed-guided bidirectional minGRU trajectory parameter estimator may replace the complex implementation work the classical trajectory-fitting approach requires for every candidate detector geometry, allowing for a faster turnaround in detector performance studies.

The result is enabled by the following three findings: small batch sizes as an implicit regularizer for high-precision training in IEEE fp32; a physics-seeded, quantile-loss design; and a custom Triton kernel that improves the throughput compared to the standard kernel implementation on our tracking data.

Table 1: Resolution ratio minGRU / truth-seeded KF per perigee parameter and test sample ($|\eta| \leq 2$): iteratively 3σ-clipped RMS (left) and un-clipped RMS including all tails (right). Values < 1 mean the minGRU achieves better resolutions than the reference. The reference is the truth-seeded KF shipped with the dataset in every row. Brackets are 1σ bootstrap uncertainties on the last digit, from 400 paired replicas per sample.

	clipped core (iter-3σ)					un-clipped (with tails)				
sample	d_0	z_0	φ	θ	q/p	d_0	z_0	φ	θ	q/p
μ, 2 GeV	0.992(2)	0.991(3)	0.989(2)	0.993(2)	1.006(4)	0.989(2)	0.981(4)	0.993(3)	0.984(6)	0.959(4)
μ, 10 GeV	0.993(1)	0.995(2)	0.992(2)	0.991(2)	1.003(3)	0.968(5)	0.950(5)	0.969(5)	0.986(2)	0.987(3)
μ, 50 GeV	0.998(1)	0.999(1)	0.977(2)	0.994(3)	1.013(2)	0.993(1)	0.957(4)	0.977(1)	0.9968(8)	1.018(4)
μ, 1–70 GeV	0.9955(3)	0.9981(3)	0.9815(6)	0.9958(5)	1.0154(7)	0.980(3)	0.960(5)	0.975(7)	0.98(1)	0.96(2)

Table 2: The throughput and throughput per device dollar. GPU rows: this work's model on the deployment path, at the saturating batch size (Fig. 5). Prices are the device-only list prices (GPU board / CPU chip [36]; hosts excluded on every row). The gain of the kernel treatment for each encoder is given in Table A.1.

device	configuration	tracks / s	approximate cost [$]	ratio
32-core CPU	ACTS Kalman filter fit (reference)	∼170 k	2 000	86
RTX 5000 Ada	this work, deployment path	1.51 M	4 000	378
H100 NVL	this work, deployment path	5.41 M	30 000	180

4.1 Physics performance in precision

Table 1 reports the ratio of the minGRU's resolution to the truth-seeded KF's for all five perigee parameters on the four muon test samples, for the clipped core and for the un-clipped RMS that includes every tail. The minGRU is at or below the reference in every entry of both halves of the table within $1.5\,\%$. The un-clipped results include non-Gaussian tails in the data which the classical linear-Gaussian estimator is not handling fully appropriately.

Fig. 3 shows the core resolution versus pseudorapidity on the $10\,\mathrm{GeV}$ muon sample: the two curves track each other, the minGRU at or marginally below the KF in every parameter across the full range studied. Integrated over the sample, the minGRU reaches $17.1\,\mu\mathrm{m}$ on d_0, $22.2\,\mu\mathrm{m}$ on z_0, $0.28\,\mathrm{mrad}$ on φ, $0.15\,\mathrm{mrad}$ on θ and 5.2×10^{-4} GeV^{-1} on q/p over the $46\,284$ tracks estimated by both, indistinguishable from or marginally below the classical fit. At $50\,\mathrm{GeV}$ and $2\,\mathrm{GeV}$ we see similar results, and the resolution holds evenly across the full momentum range as well: Fig. 4 shows the same clipped core resolution versus transverse momentum on the uniform 1–70 GeV muon sample, where the minGRU again tracks the truth-seeded KF in every p_T bin of every parameter. The $2\,\mathrm{GeV}$ and $50\,\mathrm{GeV}$ resolution-versus-η curves and the residual distributions are in Section A.4.

The trained minGRU encoder also recovers a covariance-like structure of perigee parameter estimates without any supervision on it, consistent with the physical correlations we would expect: the per-parameter loss gradients on the shared trunk align within the geometric parameters $(d_0, z_0, \varphi, \theta)$ and are nearly orthogonal to q/p; see Section A.2.

4.2 Throughput

Each of the three encoders we gave a fused kernel improves throughput by between $3.6\times$ and $6.6\times$ over its default kernel, and in throughput the minGRU leads the Transformer by $1.7\times$ and Mamba-2 by $2.6\times$ (Table A.1), all numbers measured on an H100 NVL GPU.

Table 2 gives a comparison of the throughput of the learned trajectory parameter estimator on a workstation RTX 5000 Ada GPU, a datacenter H100 GPU, and a Threadripper 3970X 32-core CPU. The H100 outperforms the workstation GPU in throughput. The model achieves a significantly higher throughput than running the classical ACTS Kalman filter trajectory fitting using all cores of a Threadripper 3970X CPU, even if normalized to the approximate cost of the compute device. The enormous data volumes produced by high-energy physics experiments require the collaborations to

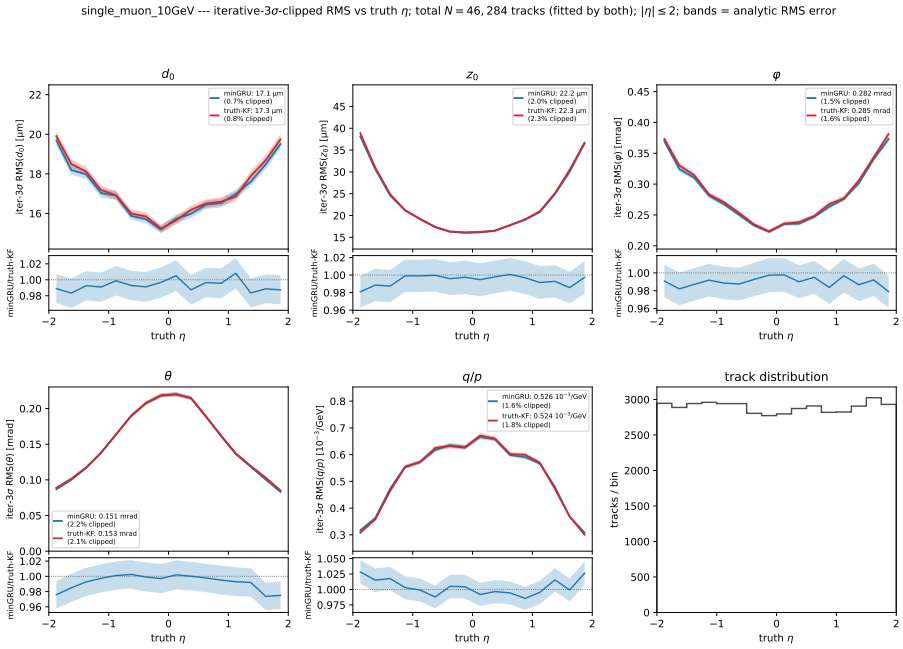

Figure 3: Core resolution versus pseudorapidity on the 10 GeV muon sample ($|\eta| \leq 2$): iterative-3σ-clipped RMS of each perigee parameter for the minGRU (blue) and the truth KF (red), with the minGRU/KF ratio beneath each panel. Each legend gives the unbinned RMS; the title states the total number of trajectories fitted by both estimators. Bands are the analytic RMS standard error.

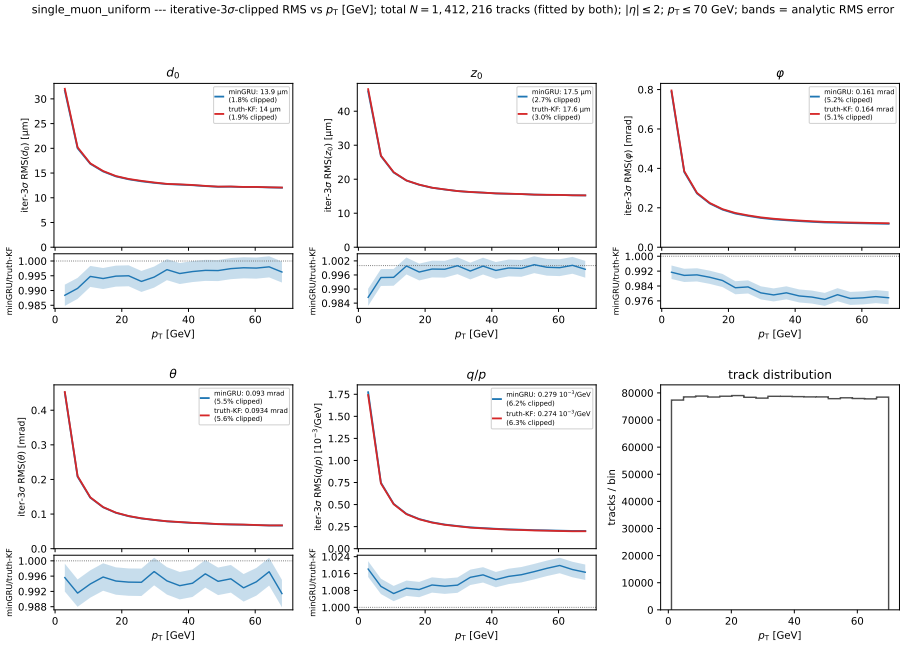

Figure 4: Core resolution versus transverse momentum on the uniform 1–70 GeV muon sample ($|\eta| \leq 2$, equal-width p_{T} bins): iterative-3σ-clipped RMS of each perigee parameter for the minGRU (blue) and the truth KF (red), with the minGRU/KF ratio beneath each panel. Each legend gives the unbinned RMS; the title states the total number of trajectories fitted by both estimators. Bands are the analytic RMS standard error.

use the most cost-effective compute for processing the data. Taking into account the approximate price of the GPUs, the H100 loses its advantage to the RTX 5000 Ada GPU.

The analytic seed is part of the deployed forward pass, so we time it explicitly. At the saturating batch size, the full minGRU forward costs $0.19\,\mu s$ per track on the H100. The on-GPU seed, computed in float64, takes $0.016\,\mu s$ per track, $9\,\%$ of the forward pass on the H100. For the RTX 5000 Ada, the full forward pass takes $0.72\,\mu s$ and the on-GPU seed takes $0.089\,\mu s$ per track, $12\,\%$. The Fourier encoding and the quantile heads take about $50\,\%$, and the encoder itself $40\,\%$ of the full forward pass.

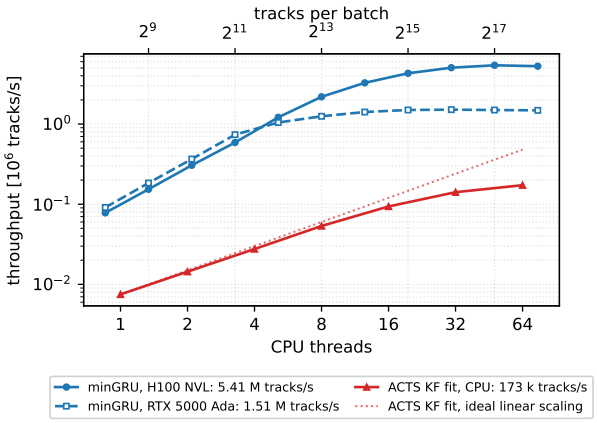

Figure 5: Inference throughput versus batch size (blue, top axis) on a datacenter H100 NVL, for the seed-guided bidirectional minGRU trajectory parameter estimator at fp16 inference. The on-GPU seed calculation is allowed for in every throughput estimate. The red curve is the ACTS KF fit on an AMD Threadripper 3970X versus the number of threads it was given (bottom axis), reaching $\sim$170 k tracks per second at 64 threads; the dotted line is ideal linear scaling from the single-thread measurement. The dashed blue curve is the same model and benchmark on a workstation RTX 5000 Ada.

The batch-size dependence underlying the numbers below, on both a datacenter H100 NVL and a workstation RTX 5000 Ada, is shown in Fig. 5 (Section A.1). Throughput saturates from $\sim$16–64 k tracks per batch on both GPUs. For comparison, the throughput on the Threadripper 3970X 32-core CPU is also shown as a function of the number of threads.

5 Conclusions

We demonstrate that a seed-guided bidirectional minGRU, a gated linear recurrent model, matches the precision of a mathematically optimal estimator, the classical Kalman filter trajectory fitter, for both the Gaussian core and the un-clipped tail distributions. Furthermore, a minGRU run on a workstation GPU yields significantly higher throughput compared to a classical Kalman filter implementation run on a cost-comparable multi-core CPU. The seed-guided bidirectional minGRU trajectory parameter estimator is capable of learning the detector geometry, material effect corrections and the charged-particle transport in the magnetic field from the simulated data itself and does not require complex hand-coded implementations, which for detector-design studies such as a Future Circular Collider replaces months of software engineering and may therefore accelerate detector design studies.

It seems plausible that variants of the seed-guided bidirectional minGRU trajectory parameter estimator presented in this paper may be optimized for electron trajectory fitting, which is complicated by the effects of Bremsstrahlung in the detector material. Learning the joint distribution of non-Gaussian physical effects, which are particularly prominent for electrons, might enable significant precision gains over classical methods. Classical combinatorial Kalman-filter-based trajectory finding strategies use mathematical models similar to the minGRU version presented here. Further adapting the minGRU model may allow it to emulate the combined classical track finding and fitting chain.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, in the form of writing and editing code. We have not used generative AI tools to generate data sets, formulate mathematical claims or write proofs, and the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for brainstorming and for drafting and refining the text of this paper. We have reviewed all AI-assisted work: AI-assisted code was reviewed and tested by the authors, and all AI-assisted text was checked and edited by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments use the public COLLIDERML dataset, available at huggingface.co/datasets/CERN (Section 2). The seed construction, encoder, losses and training schedule are described in Section 3, and the kernel adaptation in Section 3.5 and Section A.1. The code for training, evaluation and the fused inference kernels is provided as anonymized supplementary material and will be released publicly. Training the model takes about two days on two H100 GPUs.

REFERENCES

[1] ATLAS Collaboration. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation*, 3(08):S08003, 2008. doi: 10.1088/1748-0221/3/08/S08003. URL https://cds.cern.ch/record/1129811.

[2] CMS Collaboration. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3 (08):S08004, 2008. doi: 10.1088/1748-0221/3/08/S08004. URL https://cds.cern.ch/record/1129810.

[3] I. Béjar Alonso, O. Brüning, P. Fessia, L. Rossi, L. Tavian, and M. Zerlauth, editors. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*, volume 10/2020 of *CERN Yellow Reports: Monographs*. CERN, 2020. doi: 10.23731/CYRM-2020-0010. URL https://cds.cern.ch/record/2749422.

[4] ATLAS Collaboration. Technical design report for the ATLAS inner tracker pixel detector. Technical Report CERN-LHCC-2017-021; ATLAS-TDR-030, CERN, 2017. URL https://cds.cern.ch/record/2285585.

[5] CMS Collaboration. The Phase-2 upgrade of the CMS tracker. Technical Report CERN-LHCC-2017-009; CMS-TDR-014, CERN, 2017. URL https://cds.cern.ch/record/2272264.

[6] Doğa Elitez, Paul Gessinger, Daniel Murnane, Marcus Selchou Raaholt, Andreas Salzburger, Stine Kofoed Skov, Andreas Stefl, and Anna Zaborowska. ColliderML: The first release of an OpenDataDetector high-luminosity physics benchmark dataset. *arXiv preprint arXiv:2512.15230*, 2025. URL https://arxiv.org/abs/2512.15230.

[7] Paul Gessinger-Befurt, Andreas Salzburger, and Joana Niermann. The Open Data Detector tracking system. *Journal of Physics: Conference Series*, 2438(1):012110, 2023. doi: 10.1088/1742-6596/2438/1/012110. URL https://doi.org/10.1088/1742-6596/2438/1/012110.

[8] Rudolf Frühwirth and Are Strandlie. *Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors*. Particle Acceleration and Detection. Springer, 2021. doi: 10.1007/978-3-030-65771-0. URL https://doi.org/10.1007/978-3-030-65771-0.

[9] R. Frühwirth. Application of Kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research A*, 262(2–3):444–450, 1987. doi: 10.1016/0168-9002(87)90887-4. URL https://doi.org/10.1016/0168-9002(87)90887-4.

[10] Are Strandlie and Rudolf Frühwirth. Track and vertex reconstruction: From classical to adaptive methods. *Reviews of Modern Physics*, 82(2):1419–1458, 2010. doi: 10.1103/RevModPhys.82.1419. URL https://doi.org/10.1103/RevModPhys.82.1419.

[11] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs all we needed? *arXiv preprint arXiv:2410.01201*, 2024. URL https://arxiv.org/abs/2410.01201.

[12] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

[14] S. Caillou, P. Calafiura, X. Ju, D. Murnane, T. Pham, C. Rougier, J. Stark, A. Vallier, and ATLAS Collaboration. Physics performance of the ATLAS GNN4ITk track reconstruction chain. In *26th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2023)*, volume 295 of *EPJ Web of Conferences*, page 03030, 2024. doi: 10.1051/epjconf/202429503030. URL https://cds.cern.ch/record/2871986.

[15] Samuel Van Stroud, Philippa Duckett, Max Hart, Nikita Pond, Sébastien Rettie, Gabriel Facini, and Tim Scanlon. Transformers for charged particle track reconstruction in high-energy physics. *Physical Review X*, 15(4):041046, 2025. doi: 10.1103/md46-yqgd. URL https://arxiv.org/abs/2411.07149.

[16] Jeremy Couthures, Corentin Allaire, Marco Delmastro, David Rousseau, and Alexis Vallier. Transformer-based track fitting for HL-LHC. Poster, Connecting the Dots 2025, 2025. URL https://indico.cern.ch/event/1499357/contributions/6628634/.

[17] Xiaocong Ai, Corentin Allaire, Noemi Calace, Angéla Czirkos, Markus Elsing, Irina Ene, et al. A common tracking software project. *Computing and Software for Big Science*, 6(1):8, 2022. doi: 10.1007/s41781-021-00078-8. URL https://arxiv.org/abs/2106.13593.

[18] Pierre Billoir, Thomas Boettcher, Michel De Cian, and Lennart H. Uecker. Track fitting at the full LHC collision rate. *arXiv preprint*, 2026. URL https://arxiv.org/abs/2607.14793.

[19] David Rohr, Sergey Gorbunov, Marten Ole Schmidt, and Ruben Shahoyan. GPU-based online track reconstruction for the ALICE TPC in Run 3 with continuous read-out. *EPJ Web of Conferences*, 214:01050, 2019. doi: 10.1051/epjconf/201921401050. URL https://arxiv.org/abs/1905.05515. Proceedings of CHEP 2018.

[20] Paul Gessinger, Heather M. Gray, Attila Krasznahorkay, Charles Leggett, Joana Niermann, Andreas Salzburger, Stephen Nicholas Swatman, and Beomki Yeo. traccc: GPU track reconstruction library for HEP experiments. *EPJ Web of Conferences*, 337:01187, 2025. doi: 10.1051/epjconf/202533701187. URL https://arxiv.org/abs/2505.22822. Proceedings of CHEP 2024.

[21] Daniel Murnane, Paul Gessinger, Doğa Elitez, Andreas Salzburger, Andreas Stefl, Anna Zaborowska, Marcus Selchou Raaholt, and Stine Kofoed Skov. ColliderML-Release-1. Hugging Face Datasets, 2025. URL https://huggingface.co/datasets/CERN/ColliderML-Release-1.

[22] V. Karimäki. Effective circle fitting for particle trajectories. *Nuclear Instruments and Methods in Physics Research A*, 305(1):187–191, 1991. doi: 10.1016/0168-9002(91)90533-V. URL https://doi.org/10.1016/0168-9002(91)90533-V.

[23] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 7537–7547, 2020. URL https://arxiv.org/abs/2006.10739.

[24] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. URL https://arxiv.org/abs/1910.07467.

[25] Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643. URL https://doi.org/10.2307/1913643.

[26] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1609.04836.

[27] Samuel L. Smith and Quoc V. Le. A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.06451.

[28] Samuel L. Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 9058–9067, 2020. URL https://proceedings.mlr.press/v119/smith20a.html.

[29] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13: 2453, 2022. doi: 10.1038/s41467-022-29939-5. URL https://arxiv.org/abs/2101.03164.

[30] Anastasia Borovykh, Cornelis W. Oosterlee, and Sander M. Bohté. Generalization in fully-connected neural networks for time series forecasting. *Journal of Computational Science*, 36: 101020, 2019. doi: 10.1016/j.jocs.2019.07.007. URL https://arxiv.org/abs/1902.05312.

[31] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2302.06675.

[32] NVIDIA Corporation. NVIDIA H100 tensor core GPU datasheet. Datasheet, 2024. URL https://resources.nvidia.com/en-us-gpu-resources/h100-datasheet-24306.

[33] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. Blog post, 2024. URL https://kellerjordan.github.io/posts/muon/.

[34] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.06395.

[35] FCC Collaboration, A. Abada, et al. FCC physics opportunities. *European Physical Journal C*, 79(6):474, 2019. doi: 10.1140/epjc/s10052-019-6904-3. URL https://cds.cern.ch/record/2652776.

[36] Device list prices used for the cost normalisation. NVIDIA RTX 5000 Ada Generation: $4,000 MSRP, Tom's Hardware, "Nvidia's RTX 5000 Ada now available" (2023), https://www.tomshardware.com/news/nvidias-rtx-5000-ada-now-available-ad102-with-32gb-of-gddr6. AMD Ryzen Threadripper 3970X (32 cores, 64 threads): $1,999 official launch price (November 2019), Wccftech, "AMD Ryzen Threadripper 3970X 32 Core $1999 & 3960X 24 Core $1399 CPUs Official" (2019), https://wccftech.com/amd-ryzen-threadripper-3970x-3960x-hedt-trx40-cpu-official/. NVIDIA H100 NVL: no published MSRP, market price approximately $30,000 per board in 2026, IntuitionLabs, https://intuitionlabs.ai/articles/nvidia-ai-gpu-pricing-guide. Accessed 2026-09-26.

A TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A.1 KERNEL ADAPTATION FOR SHORT SEQUENCES

Table A.1 reports single-H100 (NVL) inference throughput for the three encoders of Fig. A.2 (trunk-matched at ~ 0.63 M parameters) that admit a fused short-sequence kernel, before and after the treatment of Section 3.5, which packs the layout, fuses the token mixing into one Triton kernel per encoder, compiles the Fourier encoding and runs the projections in fp16.

Table A.1: Single-H100 (NVL) inference throughput, in tracks/s, at the saturating batch size (131 000 tracks/batch). The kernel's default configuration is: strict IEEE fp32, standard recurrent and SSM kernels, full-attention for the Transformer (flash attention does not give any gains for short sequences). *Deployed* is the fastest physics-gated path: packed batches, a fused Triton kernel for the token mixing, a fused Fourier encoding, and fp16 projections for all three encoders. All three encoders have the same parameter count within 3 %. *RTX 5000 Ada* reports the same deployed path on a single RTX 5000 Ada GPU.

encoder	default kernel H100	optimized H100	gain H100	optimized RTX 5000 Ada
minGRU	0.88 M	5.41 M	6.1×	1.51 M
Transformer	0.48 M	3.18 M	6.6×	0.855 M
Mamba-2	0.58 M	2.06 M	3.6×	0.649 M

A.2 LEARNED STRUCTURE, TRUNK-GRADIENT COSINE PROBE

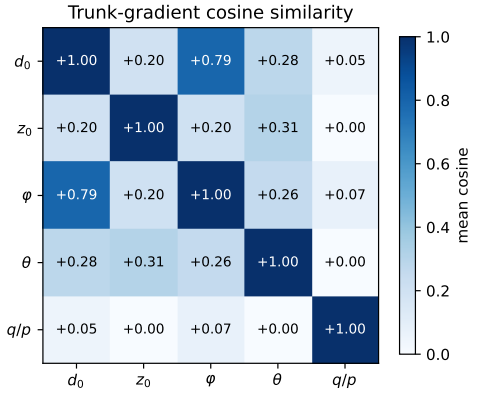

Figure A.1: Trunk-gradient cosine similarity of the paper model. Mean cosine between the per-parameter loss gradients on the shared trunk (0.63 M parameters, output head excluded), averaged over 450 minibatches of 2048 tracks (0.92 M tracks); each cell shows the mean $\pm$ its standard error. The four geometry parameters $(d_0, z_0, \varphi, \theta)$ are mutually aligned, most strongly on the transverse perigee pair (d_0, φ), while q/p is nearly orthogonal to all of them.

A trunk-gradient cosine probe (Fig. A.1) asks whether the trained encoder respects the geometric structure of the problem: for each parameter we backpropagate only that parameter's loss and measure the alignment of the resulting trunk gradients. The matrix separates into a geometry block and a curvature block. Every pairing among $(d_0, z_0, \varphi, \theta)$ is positive and many standard errors away from zero, led by the transverse perigee pair (d_0, φ) at $+0.807 \pm 0.008$ and the longitudinal lever-arm pair (z_0, θ) at $+0.315 \pm 0.011$; the remaining four lie between $+0.20$ and $+0.25$. Curvature is nearly orthogonal to all of them: $|C_{\cdot,q/p}| \leq 0.048$, with $C_{z_0,q/p} = +0.002 \pm 0.003$ and $C_{\theta,q/p} = +0.003 \pm 0.003$ indistinguishable from zero. The trunk therefore couples exactly the four parameters that classical perigee geometry couples, and learns the curvature along a nearly orthogonal direction. This is physically sensible, since the sagitta is a distinct information channel from the pointing

constraints, and it is consistent with the scale-free q/p head, whose gradient scale is decoupled by construction. None of this arises from any supervision on the classical fit, on the perigee covariance, or on residuals.

A.3 Encoder comparison in physics performance

Fig. A.2 compares the four encoders we tested, all matched to the minGRU encoder's parameter count to within $3\,\%$ and trained with the identical recipe, on every muon test sample. The four cover both ways of mixing tokens, recurrence and attention, and both kinds of state update, input-dependent and fixed: the Transformer as the general-purpose encoder [13], Mamba-2 as the selective state-space model we started our R&D on [12], the minGRU as a simpler gated form [11], and a non-selective diagonal state-space model as a similar recurrence without selectivity.

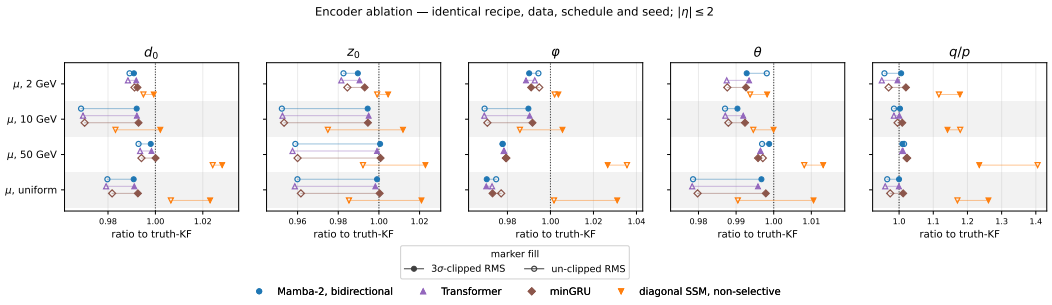

Figure A.2: Encoder comparison. Resolution ratio to the truth-seeded KF for each perigee parameter (panels) and each muon test sample (rows) at $|\eta| \leq 2$, for four encoders trained with the same recipe, data, schedule, random seed and parameter budget ($\sim$0.63 M) and read out through the same seed, features and quantile heads. These are first-stage models, trained for 25 epochs at one shared learning rate and without the second-stage training of the deployed model. Filled markers are the iterative-3σ-clipped core, open markers the un-clipped RMS including all tails. The dotted line marks parity with the reference. Three of the four reach the same precision. The non-selective diagonal state-space model falls behind, most clearly in q/p. Single seed per encoder; shared learning rate.

A.4 Results supplement, per-sample figures

Every resolution figure below is the iterative-3σ-clipped RMS computed from the tracks estimated by both the minGRU and the truth-seeded KF shipped with the dataset, one sample per figure, with the minGRU/KF ratio beneath each panel, inside the acceptance $|\eta| \leq 2$ of Section 4. Each legend states the unbinned RMS and the tracks removed by the clip; the title states the total tracks; bands are the analytic RMS standard error. The residual histograms give, per legend line, the iterative-3σ RMS and the clipped fraction. The fourth parameter is the polar angle θ; η is a binning axis only.

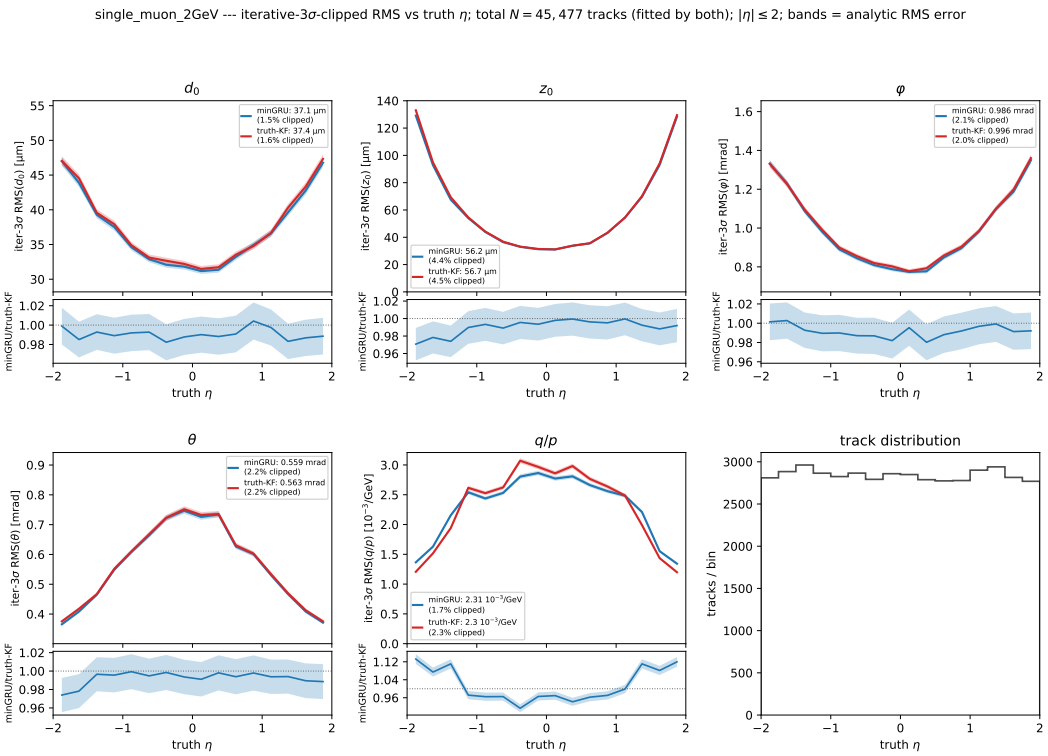

Figure A.3: **single muons,** 2 **GeV**, iterative-3σ-clipped RMS versus η (minGRU/KF ratio beneath each panel; clip fractions in the legends, total in the title).

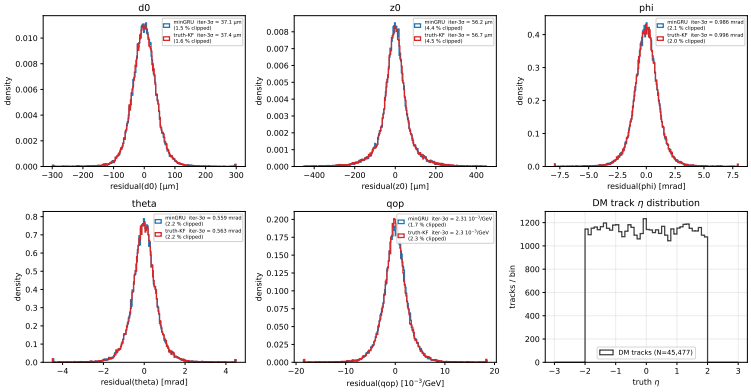

Figure A.4: **single muons,** 2 **GeV**, residual distributions; each legend gives the iterative-3σ RMS and the clipped fraction.

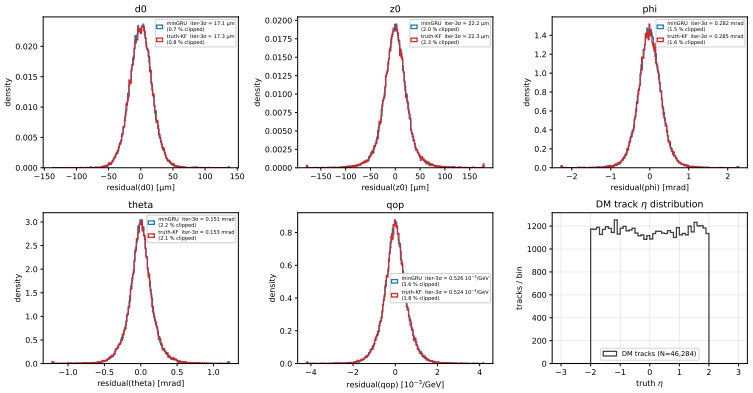

Figure A.5: **single muons,** 10 **GeV**, residual distributions; each legend gives the iterative-3σ RMS and the clipped fraction. (The 10 GeV resolution-versus-η curves are the main-text Fig. 3.)

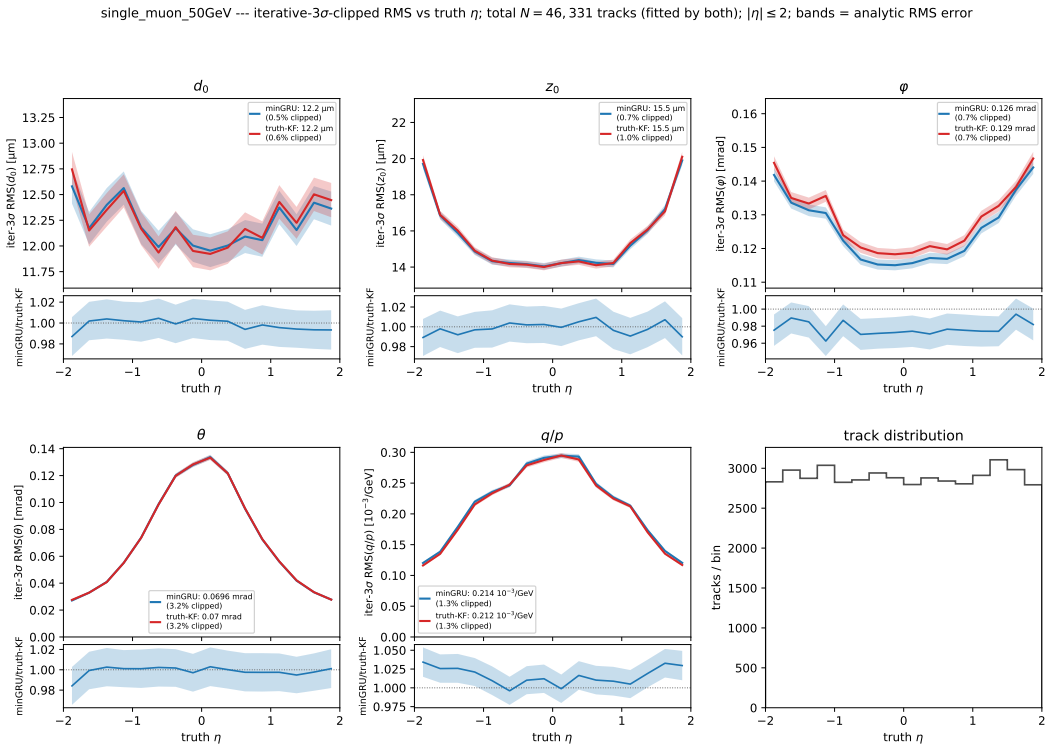

Figure A.6: **single muons,** 50 **GeV**, iterative-3σ-clipped RMS versus η (minGRU/KF ratio beneath each panel; clip fractions in the legends, total in the title).

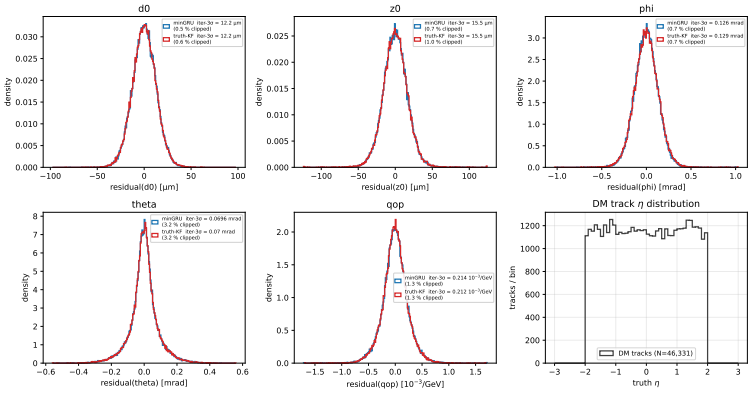

Figure A.7: **single muons,** 50 **GeV**, residual distributions; each legend gives the iterative-3σ RMS and the clipped fraction.