Appendix

A. Relativistic Heavy-Ion Collision Schematic

To complement the main text, we include a schematic overview of the stages of a relativistic heavy-ion collision. While the ML-JET dataset focuses on final-state particle distributions represented as 32×32 images, this figure provides broader context by illustrating the physical processes that lead to these detector-level patterns. It highlights how two ultra-relativistic nuclei collide, form a short-lived quark–gluon plasma (QGP), and subsequently hadronize into final-state particles. These final distributions are the basis of the images used in our classification benchmark (see Fig. 1 in the main paper).

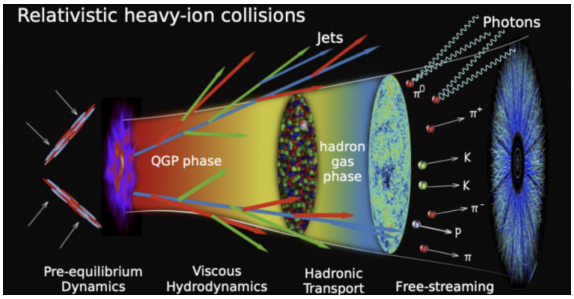


Figure 7. Schematic overview of relativistic heavy-ion collisions. Two heavy nuclei (e.g., Pb–Pb at the LHC or Au–Au at RHIC) collide at ultra-relativistic energies, creating extreme temperature and density conditions. The collision produces a quark–gluon plasma (QGP), which expands, cools, and hadronizes into final-state particles. These particles are detected as jets and soft hadrons, forming the inputs to the ML-JET dataset [15].

B. Virtual Image Aggregation Algorithm

For reproducibility, we provide pseudocode for the proposed momentum-based aggregation method. This algorithm groups raw event images with identical $(\alpha_s, Q_0, \text{module})$ labels and averages them into a single "virtual" profile, producing a CSV index of aggregated samples. This implementation underlies the moment-based aggregation method described in Section 3.3.

Algorithm 1 Virtual Image Aggregation for Dataset Construction

Require: Dataset root directory $\mathcal{D}$ with labeled event images as .npy files
Require: Label parser `parse_labels(dir)` $\rightarrow (e, \alpha, Q_0)$
Require: Aggregation group size k
Ensure: Aggregated file-label CSV with N_{agg} entries

Initialize $\mathcal{G} \leftarrow \emptyset$ ▷ Dictionary mapping label tuples to file paths
for all directory $d \in$ `os.listdir`$(\mathcal{D})$ **do**
  $label \leftarrow$ `parse_labels`(d)
  for all file $f \in$ `os.listdir`(d) such that f ends with '.npy' **do**
    $\mathcal{G}[label]$.`append`(f)
  end for
end for
Initialize empty list $\mathcal{A}$ for aggregated samples
$id \leftarrow 0$
for all $label \in$ `keys`$(\mathcal{G})$ **do**
  Shuffle $\mathcal{G}[label]$
  for $i = 0$ to $|\mathcal{G}[label]| - k$ step k **do**
    $group \leftarrow \mathcal{G}[label][i : i + k]$
    $agg\_id \leftarrow$ "agg_" $\|$ `zfill`(id)
    $\mathcal{A}$.`append`$((agg\_id, group, label))$
    $id \leftarrow id + 1$
  end for
end for
Save $\mathcal{A}$ as a CSV file with columns: `agg_id`, `file_paths`, e, α, Q_0

C. Visualization of Aggregated Event Samples

To illustrate the variability of the ML-JET dataset, we visualize aggregated two-dimensional histograms of Pb–Pb collision events in the pseudorapidity–azimuthal plane (η, ϕ), constructed using the momentum-based aggregation procedure introduced in Section 3.3. Figure 8 shows a 12×10 grid of averaged samples, where each row corresponds to a distinct combination of the energy loss module (MATTER or MATTER–LBT), strong coupling constant $\alpha_s \in \{0.2, 0.3, 0.4\}$, and virtuality separation scale $Q_0 \in \{1.0, 1.5, 2.0, 2.5\}$. For each (α_s, Q_0, E) setting, ten representative aggregated events are displayed. The color scale encodes normalized transverse momentum p_T, highlighting the jet energy deposition pattern across different parameter regimes. This visualization demonstrates the diversity of event structures across the design space. Differences in intensity and spread of the distributions reflect the underlying parton energy loss mechanisms and medium response. These aggregated samples provide intuition for the learning task: predicting the physics parameters (E, α_s, Q_0) directly from sparse event images is non-trivial, yet crucial for advancing data-driven modeling in heavy-ion physics.

D. Task-Weighting Schemes

In Section 3, we introduced the composite objective

$$\mathcal{L}_{\text{total}} = \lambda_{\text{energy}}\mathcal{L}_{\text{energy}} + \lambda_{\alpha_s}\mathcal{L}_{\alpha_s} + \lambda_{Q_0}\mathcal{L}_{Q_0}.$$

To explore how emphasizing different heads influences performance, we designed a set of interpretable weighting schemes. Each scheme is identified by a shorthand name

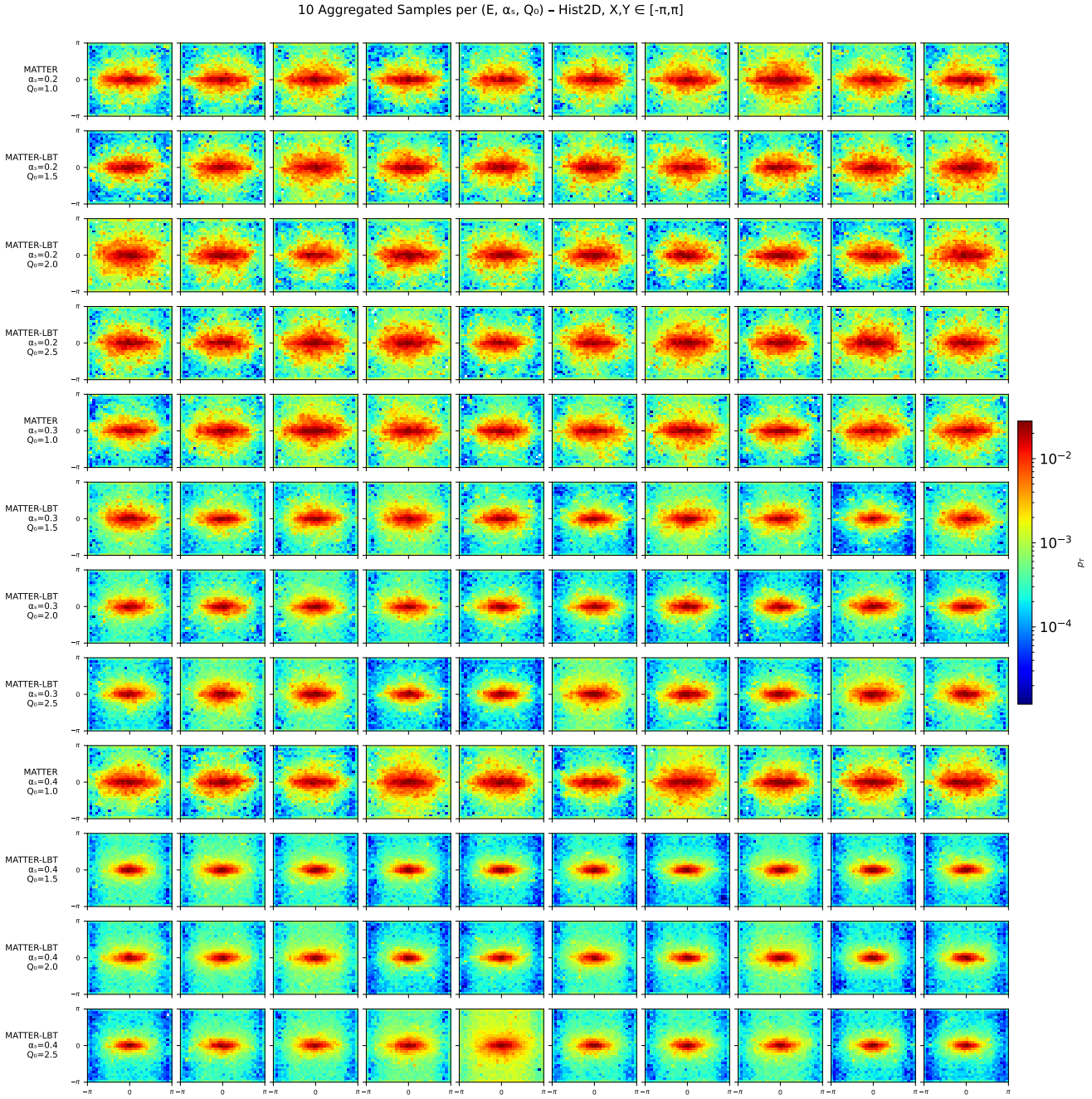


Figure 8. Grid of aggregated Pb–Pb collision events from the ML-JET dataset. Each row corresponds to a unique combination of energy loss module (MATTER or MATTER–LBT), strong coupling constant $\alpha_s \in \{0.2, 0.3, 0.4\}$, and virtuality separation scale $Q_0 \in \{1.0, 1.5, 2.0, 2.5\}$. For each (E, α_s, Q_0) setting, ten representative aggregated event images are shown as 2D histograms in the (η, ϕ) plane, with color intensity indicating normalized transverse momentum p_T.

(S1–S10), the triplet of coefficients $(\lambda_{\text{energy}}, \lambda_{\alpha_s}, \lambda_{Q_0})$, and a short rationale.

These schemes provide structured ablation points that make it possible to compare trends across backbones. The full sweep results for each scheme are reported in Table 3 of the main paper.

Table 4. Loss-weighting schemes used in our ablations.

ID	λ_{energy}	λ_{α_s}	λ_{Q_0}	Description
S1	1.0	1.0	1.0	Balanced (equal weights)
S2	0.8	0.8	1.4	Q_0 mild emphasis
S3	0.6	0.8	1.6	Q_0 mid emphasis
S4	0.5	0.7	1.8	Q_0 strong emphasis
S5	0.4	0.6	2.0	Q_0 max emphasis
S6	0.8	1.2	1.0	α_s mild emphasis
S7	0.6	1.6	0.8	α_s strong emphasis
S8	1.2	0.9	0.9	Energy bump (binary head stress test)
S9	0.9	1.0	1.1	Q_0 very mild emphasis
S10	0.7	1.0	1.3	$Q_0 + \alpha_s$ paired emphasis

E. Additional Metrics: Precision and Recall

F. Initialization Strategies

We evaluated three initialization settings during backbone training:

(i) **None:** training from scratch with framework default weight initialization. For fully connected or convolutional layers, these are typically variance-preserving schemes such as Xavier [6] or Kaiming [8], designed so that the variance of activations remains stable across layers:

$$\text{Var}[W_{ij}] = \begin{cases} \frac{2}{n_{\text{in}}+n_{\text{out}}}, & \text{(Xavier)} \\ \frac{2}{n_{\text{in}}}, & \text{(Kaiming ReLU)}, \end{cases} \tag{7}$$

where n_{in} and n_{out} are the input and output dimensions.

(ii) **Gaussian:** a custom scheme in which each weight matrix W of a linear or convolutional layer is sampled i.i.d. from a zero-mean Gaussian:

$$W_{ij} \sim \mathcal{N}(0, \sigma^2), \quad \sigma = 0.02, \tag{8}$$

with all bias terms initialized to zero. This corresponds to the PyTorch routine:

$$W \leftarrow \texttt{Normal}(0, 0.02), \quad b \leftarrow 0, \tag{9}$$

applied layer-wise. Such initialization stabilizes gradients in the early epochs by bounding the variance of activations.

(iii) **Tiny backbone:** when available (e.g., ViT-Tiny), we trained the 32×32 variant directly to avoid interpolation overhead. For models without native 32×32 support, input images were upsampled to 224×224 using bilinear interpolation:

$$I_{224}(x, y) = \sum_{i,j} I_{32}(i, j)\, \phi(x/7 - i)\, \phi(y/7 - j), \tag{10}$$

where $\phi(\cdot)$ is the bilinear kernel.

These design choices are reflected in Table 2.

For completeness, we report per-task Precision and Recall values in addition to Accuracy and Macro-F1. Table 5 presents results for all backbones, while Table 6 extends this analysis to the loss weight ablation study. Across both backbones and weighting schemes, the results corroborate our main findings: Precision and Recall follow the same trends as Accuracy and F1, with Q_0 consistently emerging as the most challenging head.

Table 5. Per-task Precision and Recall for each backbone. Metrics are reported separately for Energy loss, α_s, and Q_0. Accuracy and F1 are reported in the main paper (Table 2).

	Metric	EfficientNet	ConvNeXt (Tiny, Gaussian)	**ViT-CoMer (Tiny, Gaussian)**	Swin V2	Mamba (No Init)
E	Prec (%)	100.0	100.0	**100.0**	99.6	100.0
	Rec (%)	100.0	100.0	**100.0**	99.9	100.0
α_s	Prec (%)	94.9	93.5	**95.9**	88.6	93.9
	Rec (%)	94.7	93.5	**95.8**	88.2	93.9
Q_0	Prec (%)	76.9	73.4	**77.4**	61.8	75.1
	Rec (%)	70.2	74.1	**78.2**	63.1	75.2

Table 6. Per-task Precision and Recall for the loss weight ablation study. All experiments use the same scheduler (RLRP), learning rate, and batch size as in Table 3. We vary the task weights $(\lambda_{\text{energy}}, \lambda_{\alpha_s}, \lambda_{Q_0})$ (schemes S2–S7).

	Metric	EfficientNet	ConvNeXt	ViT-CoMer	Swin V2	Mamba
L_w	λ_{energy}	0.6	0.4	0.8	0.8	0.6
	λ_{α_s}	1.6	0.6	0.8	1.2	0.8
	λ_{Q_0}	0.8	2.0	1.4	1.0	1.6
E	Prec (%)	100.0	100.0	100.0	100.0	100.0
	Rec (%)	100.0	100.0	100.0	100.0	100.0
α_s	Prec (%)	95.8	93.8	**96.2**	89.6	94.5
	Rec (%)	95.5	93.8	**96.2**	89.6	94.4
Q_0	Prec (%)	**79.7**	75.2	77.2	67.5	75.9
	Rec (%)	**78.5**	75.3	76.7	67.7	74.7

G. Learning Curves

To complement the main text (Section 5), we provide complete learning curves for all backbones. Each figure shows training and validation accuracy as a function of epochs under the `ReduceLROnPlateau` scheduler. These curves illustrate convergence dynamics across architectures: CNNs converge quickly but plateau early, Transformers benefit from smoother training, and Mamba exhibits slower yet steady convergence.

An additional observation is that EfficientNetV2 reaches a joint accuracy of $\sim$74% but triggers early stopping once

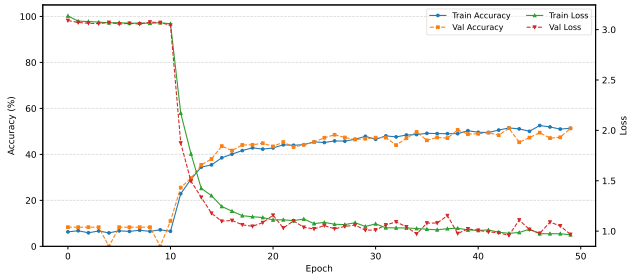
validation accuracy plateaus, whereas ViT-CoMer continues to improve throughout training. This highlights differences in optimization dynamics: CNNs can saturate rapidly, while Transformers retain capacity for further gains with extended training.

H. Use of LLMs

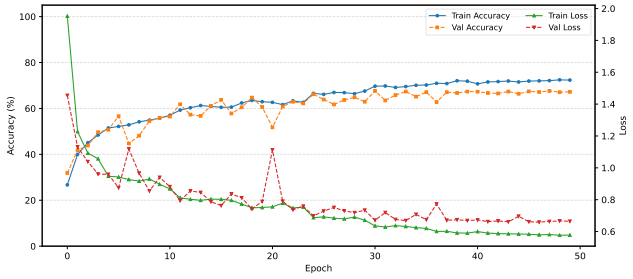
We used ChatGPT as a general-purpose assistive tool in limited parts of this work. Specifically, ChatGPT was employed to:

- **Polish writing**, including improving clarity, conciseness, and readability of the manuscript.
- **Reformat technical content**, such as adjusting figure/table captions and ensuring consistent referencing style.
- **Assist with anonymization**, helping to identify and rephrase text that could reveal author identity.

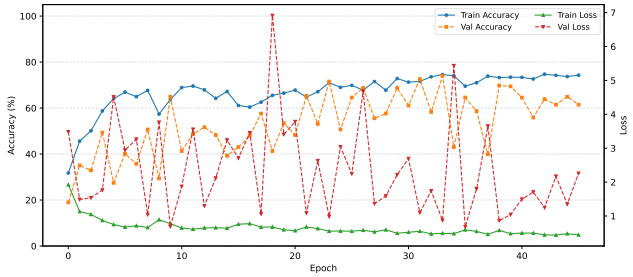
ChatGPT was not involved in research ideation, dataset design, experimental setup, or analysis. All scientific ideas, methods, experiments, and conclusions were developed independently by the authors.



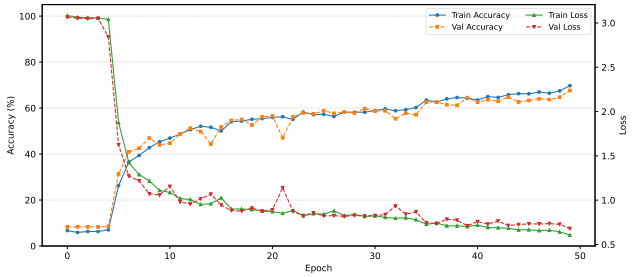
(a) Swin Transformer V2



(b) ConvNeXt V2



(c) EfficientNet V2



(d) Mamba

Figure 9. Learning curves (training/validation accuracy vs. epochs) for four backbones on ML-JET under `ReduceLROnPlateau`. CNNs (EfficientNetV2, ConvNeXt) converge quickly but plateau earlier; ViT-CoMer shows smoother improvements; Mamba converges more slowly yet steadily.