Binary Patch Encoding for Numerically Intensive Scientific Computing with Large Language Models

Anonymous ACL submission

Abstract

Despite the revolutionary success of large language models (LLMs) in natural language processing and task agents, their application to real-world scientific problems—particularly in domains involving large-scale numerical data—remains challenging. This limitation stems primarily from the inefficiency of the Byte-Pair-Encoding (BPE) method when handling numerical data. To address this gap, we propose a binary patch encoding method and integrate it into an LLM architecture named BigBang-Neutron, which can efficiently process mixed textual and large-scale numerical datasets. We demonstrate the efficacy of our method on Jet Origin Identification (JoI), a critical categorization task in high-energy physics that distinguishes jets originating from different quarks or gluons, which is one of the number-intensive classification problems in particle physics. Experimental results show that BigBang-Neutron achieves performance comparable to state-of-the-art task-specific JoI models. Furthermore, we investigate the scaling behavior of BigBang-Neutron's performance with increasing data volume, indicating its potential to serve as a foundational model for particle physics data analysis and holds promise for extension to a broad range of scientific computing applications. The project code is publicly available on GitHub, which will be provided after the manuscript is accepted.

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across a wide range of natural language tasks and are increasingly applied in scientific domains. As model scales grow, emergent capabilities—such as in-context learning or compositional generalization—begin to appear, enabling flexible downstream adaptation with minimal supervision (Gage, 1994; Bommasani et al., 2021; Wei et al., 2022; Henighan et al., 2020). However, this success remains largely confined to symbolic inputs such as natural language, mathematical expressions, or DNA sequences. A notable limitation arises when LLMs encounter numerical data: their predictions often exhibit poor arithmetic consistency, unit confusion, and a lack of quantitative precision, posing a critical barrier to scientific applications that demand reliable numerical reasoning (Lewkowycz and et al., 2022).

Recent studies (Xue and et al., 2022; Mueller and et al., 2022) suggest that these failures are not simply a consequence of limited model capacity or training data, but are fundamentally tied to subword encoding strategies such as Byte Pair Encoding (BPE)(Radford et al., 2019; Brown, 2020). In practice, BPE can tokenize the same number inconsistently, for example, 12345 can be segmented as '12', '34', and '5' in one context or as '1', '23', and '45' in another, thereby obscuring its original numerical semantics. It also produces fragmented, non-contiguous token IDs (e.g., '7' and '8' mapped to 4779 and 5014), which complicates numerical processing when ordered or structured token IDs would be advantageous. A related limitation appears in single-digit encoding (Dubey et al., 2024), where multi-digit values such as 15 are decomposed into '1' and '5', each assigned an independent token ID, further disrupting numerical continuity. These issues are especially problematic in scientific applications, where measurements are continuous, high-dimensional, and represented in floating-point form (Nat, 2021).

To address the structural limitations of the subword-based encoding method for numerical data, we have developed a decoder-only transformer architecture called Big Bang Transformer–Neutron (BigBang-Neutron), featuring an innovative number encoding method called **binary patch encoding method**. This binary-native method encodes input data as byte sequences, preserving the intrinsic structure and quantitative integrity of numerical data and avoiding ambiguities

caused by segmenting or merging numeric and textual information. Binary patch encoding method demonstrates robust capabilities in unifying the representation of diverse data modalities, including text, numerical values, and images. A significant proportion of scientific data generated in large-scale experiments is stored in binary formats. This characteristic enables direct input into the binary patch encoding framework without requiring additional preprocessing steps, thereby streamlining the process and facilitating fully end-to-end model training. The performance of BigBang-Neutron is then benchmarked with the task of Jet Origin Identification (JoI), which is one of the hardest categorization problems in particle physics, with number-intensive inputs, and can act as a stress test for the proposed encoding method.

On the task of JoI, we observe that BigBang-Neutron achieves performance comparable to that of specialized models like ParticleNet (Qu and Gouskos, 2020) and Particle Transformer (Qu et al., 2022). As a general-purpose LLM architecture capable of pretraining on a multimodal mixture of textual and large-scale numerical data, BigBang-Neutron thus represents a significant step toward building foundational models for scientific research, with the capacity to handle diverse data types and tasks.

2 Related Work

Recent work has explored byte-level modeling as an alternative to tokenizer-dependent pipelines for sequence and multimodal data. BGPT (Wu et al., 2024) treats byte-level autoregressive modeling as a unified framework operating directly on raw bytes, enabling consistent processing across heterogeneous formats and modalities. MEGABYTE (Yu et al., 2023) advances this direction with a hierarchical architecture that partitions long sequences into blocks and separates intra- and inter-block dependencies, improving scalability while retaining fine-grained byte representations. SpaceByte (Slagle, 2024) extends byte modeling to multimodal content by injecting layout-aware positional structure into linearized byte sequences, and BLT (Pagnoni et al., 2024) instead projects byte inputs into a compact latent space to enable more efficient sequence modeling.

Parallel work represents heterogeneous signals as patch token sequences. Vision Transformer (ViT)(Dosovitskiy, 2020) applies this paradigm to images by dividing them into fixed-size patches mapped to token embeddings, allowing Transformers to process visual content as sequences. Google's foundation time-series model(Das et al., 2024) similarly discretizes continuous signals and groups them into patches to capture long-range temporal structure through sequence-level attention. Both patch-based and byte-level approaches pursue unified cross-modal modeling via sequence abstraction, differing mainly in encoding granularity and inductive bias.

3 BigBang-Neutron

The BigBang-Neutron architecture, as a multimodal binary model, adheres to a general transformer decoder-only framework but incorporates specialized components designed to understand text, scientific symbols, numbers, and images. The critical architectural components of BigBang-Neutron are described in Section 3.1, and the encoding method of BigBang-Neutron is described in Section 3.2.

3.1 Model architecture

Overview BigBang-Neutron converts input sequences into high-dimensional byte-level embeddings via encoding and patch embedding, which are then processed by stacked Transformer decoder layers. Rotary Position Embeddings (RoPE) (Su et al., 2024) inject relative positional information into self-attention, enabling the model to capture numerical relationships across tokens. Each decoder layer uses causal self-attention to maintain a left-to-right generation order, while the feed-forward network provides nonlinear feature transformation and contextual abstraction. A final linear-softmax layer produces next-token predictions over the byte vocabulary.

BigBang-Neutron includes two versions, patch and non-patch, which can be switched based on specific requirements.

- Patch version is a patch-based byte-level Model. It incorporates patches to enhance the overall computational speed of the model. As shown in Figure 1, patches allow the model to process input sequences more efficiently by breaking them into smaller, manageable segments. The patch version model is designed to be computationally efficient, making it suitable for tasks where speed is a critical factor. With an emphasis on computational efficiency

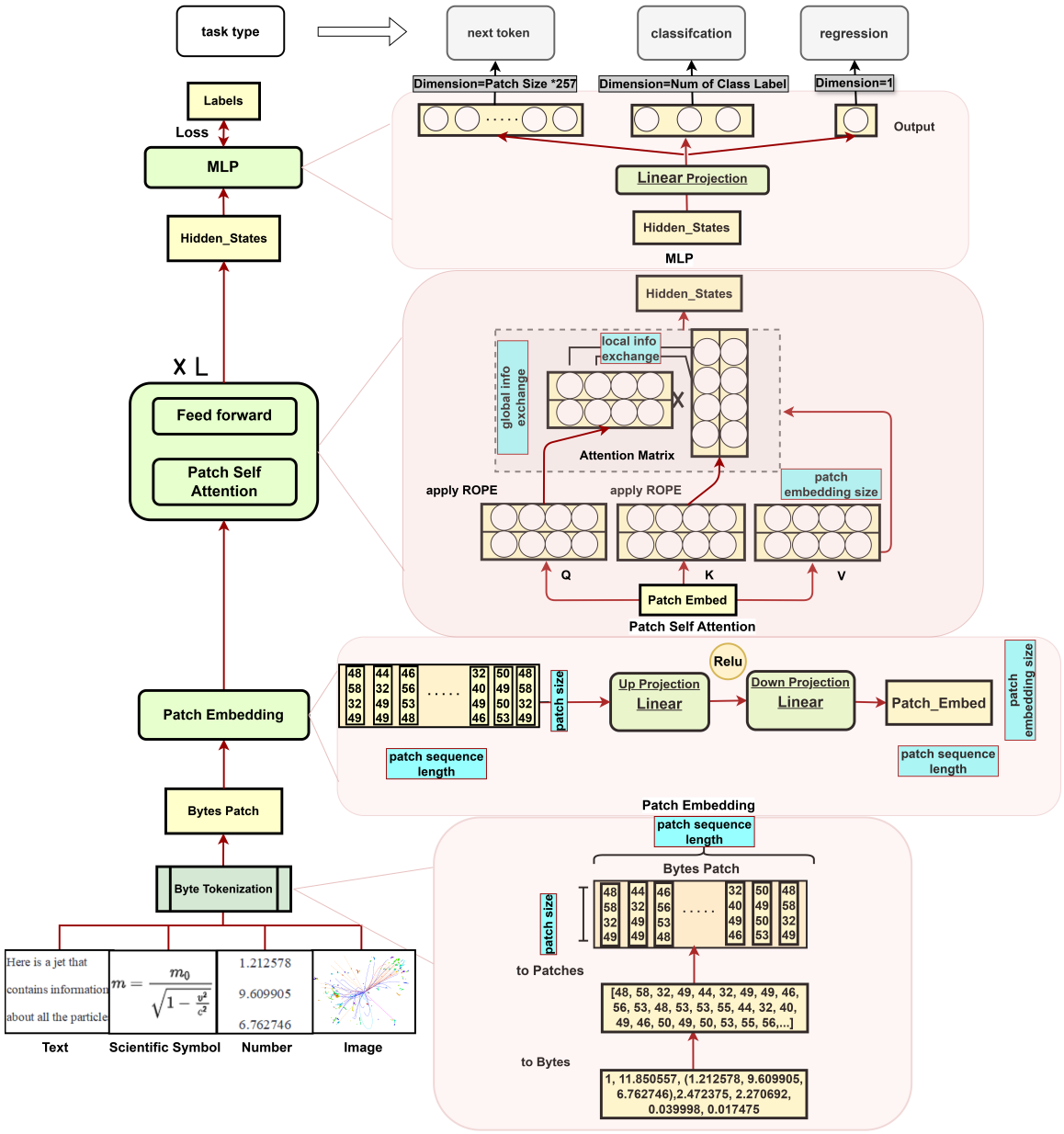

Figure 1: The overall architecture of BigBang-Neutron. The green sections represent the model architecture modules, while the yellow sections indicate the inputs and outputs of these model modules.

and the categorical nature of the JoI task's training objectives, employing this version of the model is deemed appropriate.

- Non-patch version is a byte-level model. It operates at the byte level, providing a more granular and detailed representation of the input data.

Patch embedding The embedding module consists of two linear layers with ReLU activation, mapping each patch (or individual byte in the non-patch setting) to a dense vector representation.

Patch Self-Attention Multi-head self-attention operates at the patch level. Attention masks are reshaped to align with patch structure; when a patch contains mixed valid and padded elements, the entire patch is masked to preserve patch-level consistency and enforce meaningful intra- and inter-patch dependencies.

Language Model Head The output dimension of the language model head in the patch-based model is defined as Patch Size $\times$ 257. Here, 257 represents the total number of byte IDs, which include the byte values from 0 to 255, plus the padding ID represented by 256, and Patch Size is the number of patches into which the text sequence is divided. This design allows the model to generate predictions for each patch independently, maintaining the efficiency and effectiveness of the patch-based approach.

3.2 Encoding

Our encoding method supports multiple data modalities—including text, scientific symbols, numerical data, and images—through a unified byte-level representation. Since all data are stored in binary form, we adopt a byte-based encoding (8 bits per token), which enables consistent processing across modalities and simplifies preprocessing by removing the need for modality-specific encoding. For textual data, we use UTF-8 encoding to transform characters into byte sequences. When it comes to numerical data, we offer a dual strategy: one involves treating numbers as strings and encoding them in UTF-8, while the other involves converting numbers to their native numerical types before transforming them into byte sequences. This approach offers several advantages:

- Preservation of Numerical Meaning: By converting numbers into byte sequences, the inherent meaning of the original numeric value is preserved.
- Consistent encoding: The byte encoding method ensures that the same number is always encoded in the same way, regardless of context, avoiding the inconsistencies seen in BPE.
- Continuous Token IDs: Byte encoding results in a continuous and systematic mapping of numerical values to token IDs. By employing specific computer encoding techniques, numbers like 7 and 8 can be converted into sequential token IDs, thereby streamlining the management and processing of numerical data.

Our byte encoding method operates on the principle of encoding all input data into byte arrays, which serve as a common ground for further processing. The general workflow consists of four main steps:

1. Data Collection and Preparation: Gather the raw data, including text, numbers, and scientific formulas.
2. Byte Array Conversion:
   - Text: Characters are translated into their corresponding Unicode byte values.
   - Numbers: Numbers can be processed in two distinct ways. First, a number can be treated as a string and subsequently encoded using UTF-8. For instance, the number 12345 would be represented as the string '12345' and then encoded into a byte array[49, 50, 51, 52, 53]. This method is particularly useful for preserving the exact format and any leading zeros that might be significant. Second, a number can be converted into its numerical form (e.g., an integer) and then transformed into a byte array. The integer 43125 can be directly converted into a byte array representation[0, 0, 0, 0, 0, 0, 168, 117]. This approach is more efficient for arithmetic operations and when the numerical value itself is crucial.
   - Scientific Formulas or Symbols: Complex expressions are parsed and serialized into byte sequences that capture both the structure and content of the formula. In the instance of the formula $E = mc^2$, it would be encoded as a byte array $[69, 61, 109, 99, 94, 50]$ representing the structure and variables.

- Image: Images are typically represented as pixel values, where each pixel can have one or more bytes depending on the color depth. For grayscale images, each pixel is represented by a single byte (8-bit), while for color images (e.g., RGB), each pixel is represented by three bytes (24-bit). The conversion process involves reading the image file and extracting the pixel values into a byte array. Consider a small 2x2 grayscale image, its pixel values, represented as $[128, 64, 192, 255]$, would correspond to the byte array $[128, 64, 192, 255]$ upon conversion.

3. Encoding: Concatenate the byte arrays from different data types into a single sequence, formatted as follows:

$$\begin{aligned}&\text{bos}\{\text{text bytearray}\}\{\text{num bytearray}\}\\&\quad\{\text{formula bytearray}\}\{\text{img bytearray}\}\text{eos}\end{aligned} \tag{1}$$

Here, bos and eos denote the beginning and end of the sequence, respectively, facilitating the identification of boundaries within the processed data.

4. Byte Patch Formation: The byte sequences are systematically divided into fixed-length byte patches based on the Patch Size parameter. In the context of Patch Version encoding, setting the Patch Size to 16 bytes leads to the formation of each patch containing exactly 16 bytes. For Non-Patch Version encoding, setting the Patch Size to 1 byte results in the formation of each byte as a standalone token.

4 Experiments

Quarks and gluons are the fundamental constituents of matter that participate in the strong interaction. At high energies—such as in collider experiments—they hadronize into collimated sprays of particles known as jets (see Fig. 2). Accurately identifying the origin of each jet is crucial for interpreting collider events, as jets dominate most final states and encode key information about the underlying hard scattering. With advances in detector and accelerator technologies, jet origin identification (JoI) has emerged as a central problem in modern collider physics. JoI is a number-intensive classification task and therefore provides a strong

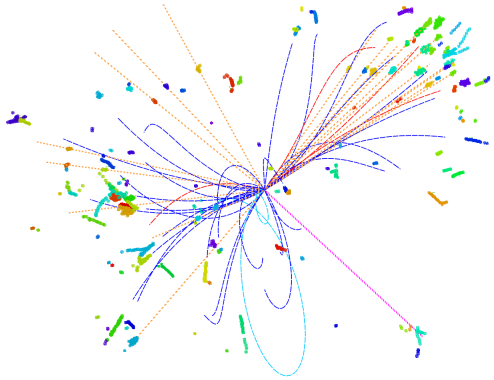

Figure 2: Event display of an $e^+e^- \to \nu\bar{\nu}H \to \nu\bar{\nu}gg$ ($\sqrt{s}$ = 240 GeV) event simulated and reconstructed with the CEPC baseline detector (The CEPC Study Group, 2018). Different particles are depicted with colored curves and straight lines: red for $e^{\pm}$, cyan for $\mu^{\pm}$, blue for $\pi^{\pm}$, orange for photons, and magenta for neutral hadrons.

stress test for our proposed method. Each jet is represented by on the order of 10^3 floating-point values describing up to $\sim$ 50 final-state particles, and the model must infer scores for 11 jet categories. Because the differences among jet types are often subtle—particularly for light-flavor (u, d, s) and gluon jets—the task poses a significant discrimination challenge.

4.1 Data set

The dataset used in this study consists of simulated jet samples derived from $H \to$ di-jet events at a center-of-mass energy of 240 GeV, corresponding to the design conditions of a future electron-positron Higgs factory. Event generation was performed using Whizard 1.95 (Kilian et al., 2011) for the hard process and Pythia 6 (Sjostrand et al., 2006) for parton showering and hadronization. Detector effects and reconstruction performance were modeled using Delphes (de Favereau et al., 2014) fast simulation. Each simulated event contains two jets, which are individually extracted and treated as separate samples.

The dataset includes 11 jet categories, corresponding to five quarks (u, d, s, c, b), their associated anti-quarks, and gluons. An equal number of jets from each category were used in training to ensure class balance. For each jet, all final-state particles within the clustering radius are included, and their kinematic and species-level information are used as input to the model. In particular, charged particles are supplemented with track impact pa-

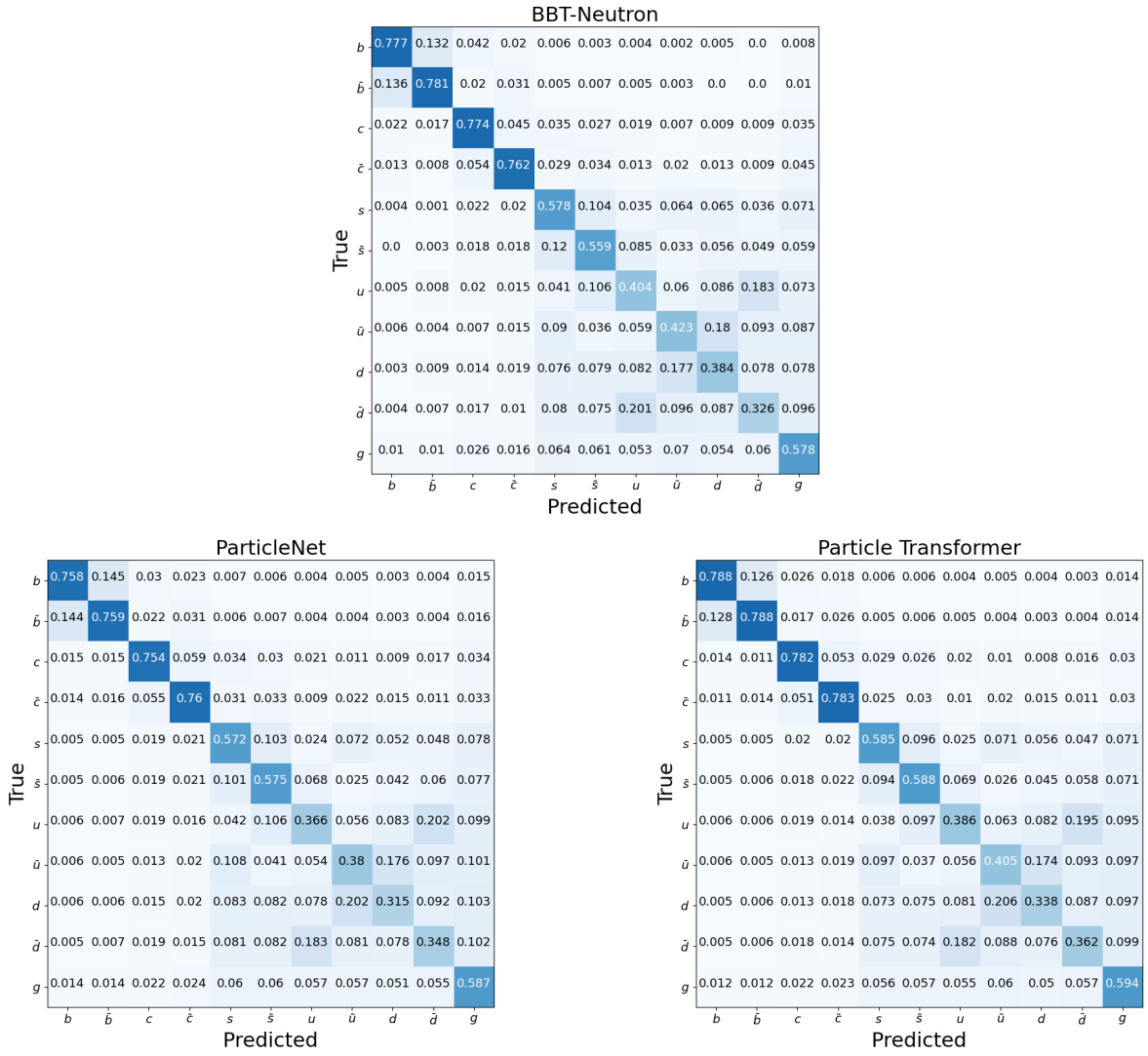

Figure 3: With the statistics of each jet one million, 60% of them used for training, 20% for validation, and another 20% for testing, the confusion matrix M_{11} obtained by BigBang-Neutron, ParticleNet, and Particle Transformer for $\nu\bar{\nu}H, H \to jj$ events at 240 GeV center-of-mass energy. Each matrix is normalized to unity for each truth label (row).

rameter variables to enhance discriminating power. The full set of input features is detailed in Ref. (Zhu et al., 2024).

4.2 Training

Data Preparation Following the encoding procedure in Section 3.2, the jet dataset is preprocessed into byte patches for model training. Each jet is first reformatted into a unified textual representation in which every particle is assigned an index and a fixed-order list of attributes separated by a colon. The attribute values are then converted into byte sequences and segmented into fixed-length patches according to the Patch Size. This produces a sequence of byte patches of length Patch Sequence Length, where each patch has dimensionality equal to the patch size and serves as the model input.

Training Objective The JoI task is formulated as a multi-class jet classification problem, where the model predicts the jet type based on its features. The label set consists of eleven categories: b-jet, $\bar{b}$-jet, c-jet, $\bar{c}$-jet, s-jet, $\bar{s}$-jet, u-jet, $\bar{u}$-jet, d-jet, $\bar{d}$-jet, and gluon-jet.

Training Details Training is performed using cross-entropy loss with the AdamW optimizer. The model contains approximately 160M parameters and is trained for 30 epochs with a batch size of 512. The initial learning rate is 1×10^{-4} and is scheduled via cosine annealing to improve optimization stability and convergence.

5 Results

JoI, a number-intensive classification task, serves as a rigorous test for our encoding method. We compare BigBang-Neutron with two specialized models, ParticleNet (PN) (Qu and Gouskos, 2020) and Particle Transformer (ParT) (Qu et al., 2022), using the 11-dimensional confusion matrix M_{11} shown in Fig. 3, which assigns each jet to the category with the highest predicted score. M_{11} is approximately symmetric and block-diagonalizable into 2×2 blocks for each quark species, providing a clear overview of correct and incorrect predictions. BigBang-Neutron exhibits performance comparable to that of PN and ParT. To further evaluate JoI, we analyze two metrics: flavor-tagging efficiency (half the sum of values within each block, ignoring quark/antiquark distinction) and charge-flip rate (the ratio of off-diagonal elements to the total block sum, reflecting quark–antiquark misidentification). These metrics are subsequently used to assess BigBang-Neutron's scaling behavior.

We compare the scaling behavior of BigBang-Neutron with two state-of-the-art specialist models, PN and ParT, across training datasets ranging from 100 to 10 million jet samples. As shown in Figs.4, all models exhibit S-shaped scaling curves in both flavor-tagging efficiency and jet charge–flip rates, with uncertainties reflecting model variation and binomial statistical fluctuations [1]. Distinct differences emerge between the generalist and specialist architectures. PN and ParT display similar trends and achieve meaningful performance even with small datasets, improving smoothly before saturating at larger data scales. In contrast, BigBang-Neutron performs near-random at small data sizes and requires substantially more data—up to an order of magnitude more for b, c, and s jets—to reach comparable accuracy. Once it passes a critical data threshold, however, its performance rises sharply and eventually matches or exceeds the specialist models, particularly for b, s, and u jets.

A similar pattern appears in the jet-charge task, a binary classification problem distinguishing quark and antiquark jets of the same flavor. BigBang-Neutron remains near-random below $\sim 10^4$ samples, then undergoes a rapid transition to above-random performance, reminiscent of emergent behavior in large language models (Wei et al., 2022). Beyond $\sim 3\times10^6$ samples, its charge-flip rates converge to those of PN and ParT, after which further gains scale smoothly with data size.

These sharp transitions suggest an emergent capability specific to the generalist architecture. Specialist models incorporate strong domain-specific inductive biases, which enable efficient learning but also lead to earlier saturation. BigBang-Neutron, by contrast, processes inputs as left-to-right byte sequences without explicit symmetry constraints; it therefore requires larger datasets to internalize the relevant geometric and relational structure, but displays a pronounced performance jump once this structure is learned. The results indicate that BigBang-Neutron can recover spatial symmetries from data, even without explicitly encoding them.

6 Discussion and Future Work

Contemporary scientific research relies on vast experimental infrastructures, such as particle colliders in high-energy physics that generate enormous volumes of data. A substantial portion of researchers' efforts is focused on developing specialized models tailored for specific analytical tasks within these datasets. In contrast, general-purpose architectures offer the potential to support diverse analytical tasks within a single foundational model, increasing efficiency and enabling performance gains through transfer learning and multi-dimensional data integration.

In this study, we apply BigBang-Neutron, an LLM architecture incorporating a binary-native encoding method, to the complex task of JoI in high-energy physics, demonstrating that it achieves performance comparable to specialized AI tools like PN and ParT. Although BigBang-Neutron requires larger datasets to match the accuracy of these specialized models, it scales effectively and delivers competitive results. This work underscores the potential of large language models for scientific domains that demand precise numerical representation and processing. BigBang-Neutron lays the groundwork for further exploration of LLM applications in high-energy physics and other fields requiring sophisticated numerical data handling. Its

[1] The error bars shown in the performance plots account for both model-related uncertainties and statistical fluctuations from inference. For each training dataset size (as indicated on the X-axis), a separate model is trained, and its performance is evaluated on a large, fixed-size test sample containing 10^5 jets. The dominant contribution to the uncertainty arises from the binomial statistical error associated with jet flavor tagging efficiency, which is estimated as $\sqrt{\epsilon(1-\epsilon)/N_{\text{train}} + \epsilon(1-\epsilon)/N_{\text{test}}}$, where ϵ denotes the tagging efficiency, and N_{train} and N_{test} denote the numbers of jets in the training and test samples, respectively.

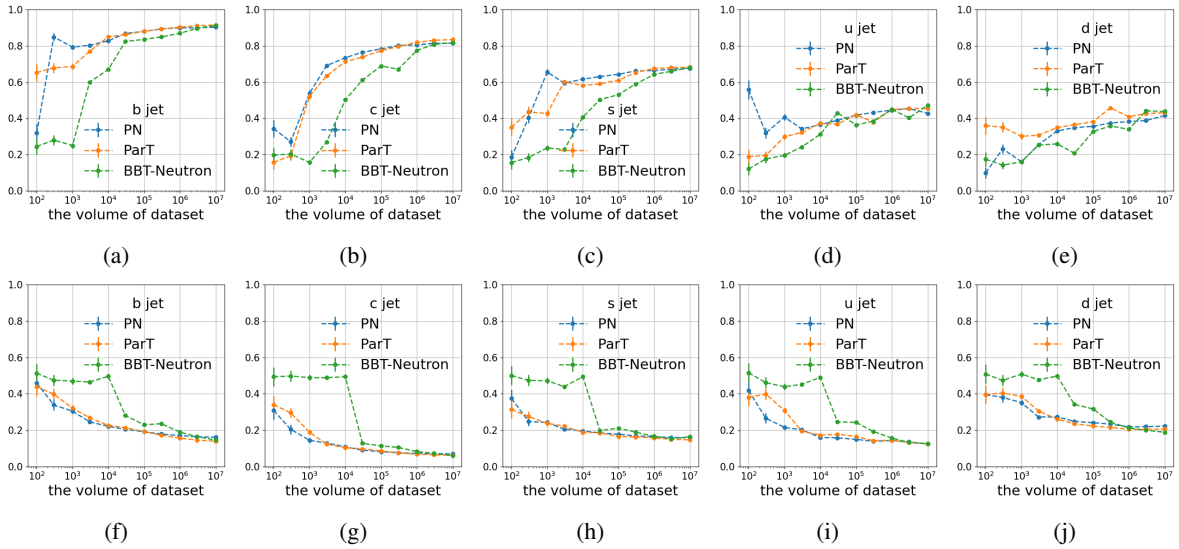

Figure 4: The scaling behavior of BigBang-Neutron, ParticleNet (PN), and Particle Transformer (ParT) as a function of training data volume is illustrated. The top panels and bottom panels correspond to jet flavor tagging efficiency and jet charge flip rate, respectively. Error bars represent statistical uncertainties.

scalability suggests that such models may eventually support more general-purpose or task-agnostic learning across scientific domains.

Moreover, akin to the phase transitions observed in the capabilities of LLMs with increasing model size and computational scale, it is expected that generalist architectures like BigBang-Neutron will exhibit further performance enhancements with continued training and scaling. Rather than replacing specialized models outright, generalist LLMs may complement existing approaches by offering greater flexibility and broader applicability. BigBang-Neutron illustrates the potential for LLM-based architectures to handle diverse scientific tasks such as classification and clustering through natural language inputs with minimal structural modifications to its output layer.

In future work, we will expand BigBang-Neutron's pretraining to encompass a wider range of tasks related to particle collision experiments, aiming to develop a foundational model for high-energy physics data analysis. Additionally, we plan to pretrain BigBang-Neutron on a combination of high-quality textual and experimental data, enabling simultaneous learning of physical theories and experimental methodologies.

7 Conclusion

We present BigBang-Neutron, a large language model architecture tailored for scientific computing tasks, particularly those arising in Big Science experiment infrastructures. A key feature of BigBang-Neutron is its binary patch encoding method, which enables efficient training on large-scale numerical data. Applied to the JoI task—a number-intensive classification challenge in particle physics—BigBang-Neutron achieves performance comparable to state-of-the-art models. Through comparative analysis, we evaluated the scaling behavior of BigBang-Neutron against specialized models, including Particle Transformer and ParticleNet. Interestingly, emergent phenomena observed in BigBang-Neutron, absent in Particle Transformer and ParticleNet, suggest that domain-specific structural constraints in specialized models may hinder the performance phase transitions typically seen during model scaling. We release BigBang-Neutron as open-source for the scientific community to facilitate further exploration across diverse tasks.

Limitations

Although the proposed BBT–Neutron binary patch encoding framework provides a simple and unified sequence modeling pipeline, several limitations remain. First, the current design adopts a fixed-length patching scheme, which may segment byte sequences that correspond to a semantically coherent unit (e.g., a full word or symbol), potentially weakening the alignment between patch boundaries and linguistic structure. This constraint suggests that future extensions could incorporate adaptive

or dynamically sized patches, as in latent-space approaches such as BLT (Pagnoni et al., 2024), to better preserve semantic continuity. Second, the present model intentionally maintains a minimalistic next-token prediction architecture and does not introduce additional local information extractors within each patch. In contrast, prior work such as BGPT (Wu et al., 2024) and MEGABYTE (Yu et al., 2023) leverages intra-patch modeling modules (e.g., MLPs or lightweight Transformers) to capture finer-grained local structure, which may further enhance representational capacity at the cost of architectural simplicity. Finally, the current system employs task-specific LM heads for next-word generation, classification, and regression, rather than a unified output interface; developing a more generalizable and task-agnostic head remains an interesting direction for improving extensibility and cross-task consistency.

References

2021. Why data-hungry ai needs astronomy. *Nature Astronomy*, 5:725.

Rishi Bommasani, Drew A. Hudson, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258v2*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. 2014. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Tom Henighan, Tom B. Brown, and 1 others. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361v1*.

Wolfgang Kilian, Thorsten Ohl, and Jurgen Reuter. 2011. WHIZARD: Simulating Multi-Particle Processes at LHC and ILC. *Eur. Phys. J. C*, 71:1742.

Aitor Lewkowycz and et al. 2022. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*.

Jonas Geiping Mueller and et al. 2022. Investigating numerical reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, and 1 others. 2024. Byte latent transformer: Patches scale better than tokens, 2024. *URL https://arxiv.org/abs/2412.09871*, 2412.

Huilin Qu and Loukas Gouskos. 2020. ParticleNet: jet tagging via particle clouds. *Phys. Rev. D*, 101(5):056019.

Huilin Qu, Congqiao Li, and Sitian Qian. 2022. Particle Transformer for jet tagging. In *Proceedings of the 39th International Conference on Machine Learning*, pages 18281–18292.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. 2006. PYTHIA 6.4 Physics and Manual. *JHEP*, 05:026.

Kevin Slagle. 2024. Spacebyte: Towards deleting tokenization from large language modeling. *Advances in Neural Information Processing Systems*, 37:124925–124950.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

The CEPC Study Group. 2018. CEPC conceptual design report: Volume 2 - physics & detector. *Preprint*, arXiv:1811.10545.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Shangda Wu, Xu Tan, Zili Wang, Rui Wang, Xiaobing Li, and Maosong Sun. 2024. Beyond language models: Byte models are digital world simulators. *arXiv preprint arXiv:2402.19155*.

Linting Xue and et al. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823.

Yongfeng Zhu, Hao Liang, Yuexin Wang, Huilin Qu, Chen Zhou, and Manqi Ruan. 2024. ParticleNet and its application on CEPC jet flavor tagging. *Eur. Phys. J. C*, 84(2):152.