LAPIN AMK
Lapland University of Applied Sciences

Leveraging Physics Foundation Models to Predict the Evolution of Particle Jets

Cordruwisch Tom

Bachelor's Thesis

Machine Learning and Data Engineering
Bachelor of Engineering

2025

LAPIN AMK
Lapland University of Applied Sciences

Thesis Abstract

Machine Learning and Data Engineering
Bachelor of Engineering

Author(s)	Tom Cordruwisch	**Year**	2025
Supervisor(s)	Kenneth Karlsson Dr. Anna Hallin		
Commissioned by	University of Hamburg		
Title	Leveraging Physics Foundation Models to Predict the Evolution of Particle Jets		
Number of pages	55		

In light of the rapid advancement of machine learning and its increasing uptake in high energy physics, foundation models present an encouraging outlook for responding to concerns about data scarcity and computational cost limiting the scale of contemporary and future research.

This thesis builds on OmniJet-α, a foundation model released in 2024 using a transformer-based backbone architecture. By leveraging the OmniJet-α codebase, including the use of a Vector Quantised-Variational Autoencoder (VQ-VAE) to generate tokens representing the original data, this work aims to extend the model's functionality to support the prediction of particle jet evolution.

For that purpose, the feasibility of adapting its tokenisation strategy and customising the model architecture was investigated, supported by a revised data preprocessing pipeline to transform raw jet data into an optimal machine learning-ready format.

While jet evolution predictability remains uncertain, the study identified critical aspects for future work and successfully demonstrated model inference functionality. The findings highlight the major role of feature engineering in machine learning applications and contribute to the development of foundation models in particle physics.

Keywords: Machine Learning, Foundation Models, Transformer Architecture, VQ-VAE, Particle Collisions, Jet Clustering Algorithms, High Energy Physics

CONTENTS

FOREWORD

This thesis would not have been possible without the support and contributions of others. Access to a high-performance computing cluster at Lapland University of Applied Sciences was instrumental for full-scale machine learning training and significantly extended the scope of what could be achieved. Additionally, the project depended heavily on available open-source libraries and prior research in particle physics and machine learning. These resources were essential to the success of the work and include:

- FastJet – a C++ library for efficient jet clustering with support for Python (Cacciari, Salam & Soyez 2011)
- Vqtorch – a PyTorch library enabling vector quantisation (Huh 2022)
- JetClass – a simulated dataset of over 100 million jets for deep learning applications (Qu, Li & Qian 2022)
- OmniJet-α – a physics foundation model utilising a transformer-based architecture for jet tagging and jet generation (Birk, Hallin & Kasieczka 2024)

The author expresses their sincere gratitude to the people behind these projects. Special recognition is given to Dr. Anna Hallin for the support and supervision provided during the thesis process. Extended thanks are given to the entire research team for the development efforts on OmniJet-α, including co-authors Joschka Birk and Prof. Dr. Gregor Kasieczka.

Their joint work forms the base for this thesis and is exemplary for the importance of collaboration in the scientific research community and the commitment to open science.

Lastly, in line with the principles of ethics and reliability, the author affirms that all the resources used in this project were publicly available. No confidential or personal data was handled at any stage. Throughout the project, the author adhered to the standards of research integrity, transparency, and academic rigour.

SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
CMS	Compact Muon Solenoid
FM	Foundation Model
GPT	Generative Pretrained Transformer
HEP	High Energy Physics
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
QCD	Quantum Chromodynamics
VQ-VAE	Vector Quantised-Variational Autoencoder
WTA	Winner-Take-All

1 INTRODUCTION

Machine learning is a sub-area of research within Artificial Intelligence. With many of its fundamental mathematical concepts originating in the 1950s and 1960s, the beginning of machine learning can be traced back as early as the 1940s, with the McCulloch-Pitts neuron (1943) and the definition of neuronal weights forming links between neurons that can either weaken or strengthen, as described in Donald O. Hebb's book "Organization of Behaviour" (1949). The 1950s saw multiple first forays into creating artificial brains. The term "Artificial Intelligence" was conceived in 1956 by computer scientist John McCarthy. Two years later Frank Rosenblatt created the Perceptron, the progenitor of neural networks, and in 1959, the term "Machine Learning" was coined by Arthur Samuel. Samuel greatly contributed to advancements in computer science research through his efforts to create a program for playing checkers by relying on past experiences to improve (IBM 2025). The 1960s saw other advancements, including multi-layer neural networks and research which laid groundwork for the k-nearest neighbour algorithm that is widely used in classification problems today (Cover & Hart 1967).

Over the following decades, machine learning research slowed down. Referred to as the "AI winter", it represents a period of low interest and research in artificial intelligence between 1970 and the 1990s. Most recently, breakthroughs in modern computing hardware and machine learning have sparked renewed interest in the field and its applications. Starting in the 2010s, this interest has accelerated even further with the introduction of the transformer architecture, described in the landmark 2017 paper "Attention Is All You Need" by a team of Google researchers. Public awareness of AI surged to an all-time high in late 2022, when OpenAI's ChatGPT achieved record-breaking success, reaching over 1 million users within its first week (Forbes 2023).

Since then, Artificial Intelligence and machine learning have remained in the public consciousness, defining an industry-transforming shift that impacts businesses, research and society at large.

These recent developments, relying heavily on ML algorithms and ongoing computational advancements, are often described as a new era of AI, and considered

by some to be the most significant technological advancement in human history (BBC 2022).

The business impact of machine learning is projected to grow exponentially over the next decade. Various reports estimate its market size to reach hundreds of billions of dollars by 2030, with common figures ranging from $200 billion (O'Donnellan 2024) to $400 billion (Law 2023).

Adding to the impact of machine learning on businesses and industries, research focusing on novel machine learning algorithms, architectures and approaches is also progressing at a high pace. A term coined by Stanford researchers in 2021 (Bommasani et al. 2021), foundation models are machine learning models that combine multiple purposes into a single architecture, aiming to create machine learning models with a high degree of versatility. Trained on massive datasets of increasing volume and complexity, they can reach unprecedented performance levels and extract immense amounts of information from data. Considered to be one of the more significant types of machine learning models, they are expected to see major advancements in the years ahead (Lorica 2025). In addition to being powerful, multi-purpose tools, they are seen as a potential precursor to Artificial General Intelligence. Unlike traditional machine learning models, foundation models are designed with fine-tuning in mind—a method for improving performance through additional training on task-specific data (Zheng et al. 2025).

In research, foundation models have attracted considerable interest for their ability to capitalise on extensive datasets through techniques enabling effective unsupervised learning. Their capacity to generalise knowledge to a degree far beyond that of conventional models opens up new possibilities across many domains. OmniJet-α (Birk, Hallin & Kasieczka 2024) is such a foundation model, designed for applications in particle physics, aiming to address challenges with data availability in high energy physics (HEP), where the creation of labelled data is associated with prohibitive costs. On top of that, datasets meeting the specific criteria of machine learning tasks are limited. By pretraining on more readily available unlabelled data and fine-tuning with smaller labelled datasets, OmniJet-α aims improve the achievable model performance and enable research that could otherwise remain infeasible.

OmniJet-α focuses on data from particle collisions, particularly the particle jets produced and recorded in experiments such as the CMS at CERN's Large Hadron Collider. As one of the first foundation models developed for physics, it is still actively being developed but has already demonstrated effective transfer learning. Between generative and classification tasks, it achieved promising fine-tuning results, reaching approximately 90% classification accuracy using only 0.1% of the number of jets needed for comparable accuracy when training from scratch. This exemplifies how models like OmniJet-α possess clear potential to significantly accelerate scientific discoveries in the future. (Birk, Hallin & Kasieczka 2024.)

The aim of this thesis is to examine potential strategies for expanding OmniJet-α's feature set to enable predictions that resemble the behaviour of jet evolution: the progressively developing jet structure characterised by successive particle decays occurring during collision events. In this work, particle decays are not modelled directly, instead they are represented by splitting or 'declustering' of intermediary objects obtained by jet clustering.

The particle decays happening inside jets are of significant interest for advancing our understanding of the Standard Model of particle physics, as many of the occurring processes are fundamentally unobservable. Instead, theoretical modelling is used to support many discoveries, significantly contributing to the confirmation of the Higgs boson in 2012 (CERN 2025). Such discoveries emphasise the motivations for adopting new approaches in science and present a strong justification for studying the applicability of machine learning in other fields of research.

The core idea for modelling jet evolution with machine learning involves the use of jet clustering algorithms to merge detected particles into jets by combining their measured features. By reversing the order of clustering steps, the sequence resembles the declustering of a jet, creating an interconnected tree structure in the process. This structure can be interpreted as a chain of particle decays, effectively simulating the evolution of a particle jet from source to detection. The declustering sequence can then serve as training data for a machine learning model that may either predict or be fine-tuned to study particle decays.

While this does not necessarily represent a literal reconstruction of the physical processes inside a particle jet, it offers a promising direction for gaining insights into jet substructure, intermediate particles, and their behaviour. It could prove particularly beneficial in future-proofing foundation models and fine-tuning them to compare predictions with experimental measurements. Finally, this approach would offer a pre-existing pipeline that could serve as a starting point for how to process real-world data related to particle decays, should such data become available in the future.

2 BACKGROUND

This chapter outlines the theoretical and scientific context, introducing the specific foundation model employed in this research, and describing the methodologies used to generate training data. Besides the focus on the machine learning work, this thesis is also strongly based on particle physics, the field concerning the set of rules that dictates the interactions and behaviours of particles, obeying patterns that machine learning models may be capable of learning and predicting. To this end, simulated particle collision data reflecting these basic principles forms a valuable resource for this work and has a great impact on model capabilities.

Continuing with the questions formulated in this thesis, the main research question is as follows:

“How can the evolution of particle jets be predicted using machine learning?”

To answer this question, the generative part of the foundation model equipped with a token prediction head will be adapted to support the jet evolution prediction. A portion of the original training dataset (JetClass) is repurposed by reclustering constituent particle data using FastJet’s Python interface. This reclustering process uncovers additional, intermediary pseudojet information as well as the full clustering history for each jet.

To support the central research question, the following complementary research sub-questions have been formulated:

“What strategies should be used to predict jet evolution with machine learning and how should the model structure its output predictions?”

Given that the generative model predicts tokens autoregressively, this question looks at how the model can be adapted to create outputs that represent the structural properties exhibited by successive particle decays.

“Which data preprocessing and transformation steps are required for model training?”

This research sub-question guides the selection of necessary preprocessing steps and data representations befitting model training, to effectively learn the structure and progression of jet evolution.

“How can the input features enable the model to make accurate predictions?”

Finally, this sub-question examines the influence of input criteria and data characteristics, such as particle-level information and pseudojets, in relation to model performance.

2.1 Particle physics

As a branch of science aiming to explain the behaviour of all matter in the universe, particle physics centres around the eponymous fundamental particles and the mechanisms that explain their behaviour. The framework that best describes our current understanding of particle interactions is called the Standard Model of particle physics. It describes the electromagnetic, weak and strong forces. (ATLAS Open Data 2025.)

Despite its decades-long history and success, the Standard Model is considered incomplete, not accounting for the gravitational force and failing to unify the electromagnetic, weak and strong interactions into a Grand Unified Theory (GUT). This incompleteness drives the ongoing research efforts, aimed at expanding the model and ultimately aspiring to develop a more comprehensive framework, often referred to as a theory of everything (TOE), for accurately describing fundamental physics with a single, unified set of rules.

2.1.1 Particle collisions

Particle collisions usually involve the collisions of particles in opposing particle beams at extremely high energies, for studying physics at otherwise inaccessible energy scales. These events provide opportunities for validating, refining or challenging existing theories. Such experiment conditions can be created by devices known as particle accelerators, using electric fields and highly advanced magnets to accelerate particles. (Schmidt 2023.)

The world's largest particle accelerator is CERN's Large Hadron Collider (LHC), located in Geneva, Switzerland. First proposed in 1994, it became operational in 2010 following more than a decade of construction. The LHC has played a pivotal role in particle physics, with one of its most publicised achievements being the 2012 discovery of the Higgs boson. The mechanism that predicted the Higgs boson was originally proposed in the 1964 PRL symmetry breaking papers by Peter Higgs, alongside two other teams of researchers.

Over its 15 years of operation, the LHC has had a profound impact on particle physics research. As a natural progression for continuing this research, CERN has proposed an even larger successor: the Future Circular Collider (FCC). This next-generation particle accelerator is currently planned as a two-stage project, beginning with the FCC-ee in the 2040s and succeeded by the FCC-hh in the 2070s. The FCC-hh aims to set foot into a high-energy frontier that would enable discoveries unattainable by the LHC (Benedikt et al. 2022).

Among the currently nine experiments at the LHC are ATLAS, CMS, ALICE and LHCb. To analyse particle collisions in experiments, detectors are equipped with arrays of highly sensitive and precise instruments that record measurements, including particle trajectories, momenta, and energies, which can be used to define particle jets.

Jets are initiated by unstable particles in high-energy collisions. Among these particles are partons, which are quarks and gluons that make up subatomic particles like protons and neutrons. As these partons cannot freely exist, they are highly unstable and undergo a series of successive decays, forming jets. Eventually, the decay products hadronise, a process in which they form stable particles that become observable to detector elements.

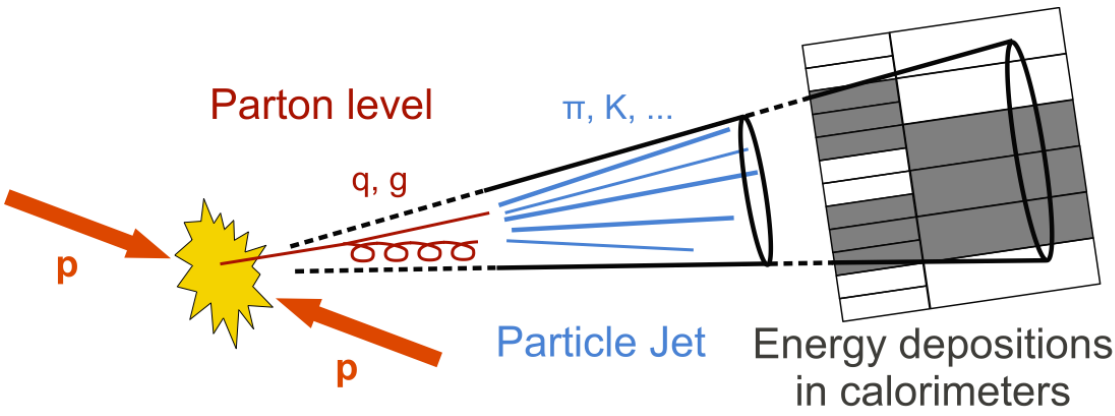

Figure 1: Illustration of a particle jet at the CMS detector (CERN 2012)

Studying the structure and behaviour of particle jets is essential to further develop theoretical physics and to understand the behaviour of quarks and Quantum Chromodynamics (QCD). They reveal important information and properties of the particle they originate from, which can be used to better describe and analyse the collision event. Jets are reconstructed from measured particles using clustering algorithms, which merge particles based on their relative distances, commonly calculated using their momentum. The jet radius parameter R defines the size of the cone in which particles are grouped together.

Jets originating from top quarks in the hadronic top quark decay-channel $t \rightarrow bqq'$ are the focus of this study due to their distinctive substructure, resulting in three clearly discernible subjets. These subjets are created from subsequent decays of the top quark into a bottom quark b and a W^+-boson, with the W^+-boson further decaying into a quark-antiquark pair.

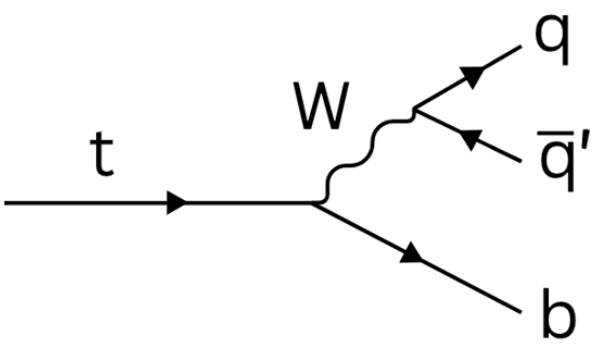

Figure 2: Feynman diagram of top quark decay via hadronic $t \rightarrow bqq'$-channel

The top quark is particularly noteworthy for being the most massive known elementary particle, making it especially relevant for researching the Higgs boson due to the strong coupling to the Higgs field that results from its high mass (CMS Collaboration 2025).

The top quark decay further produces a complex jet substructure which is important in studying QCD and can serve as a means to gauge the capabilities of machine learning models in high energy physics research.

2.1.2 Machine learning applications in particle physics

Recently, machine learning has started to transform many scientific fields and is quickly becoming an indispensable tool for aiding research (Arranz, Bianchini, Di Girolamo & Ravet 2023, 3–4), presenting an ongoing paradigm shift in the applications enabled by data. Despite growing impact, modern ML architectures are still considered to be in their early stages, with many underlying mechanisms not yet fully understood (Kosinski 2024). The major developments in machine learning are unlikely to subside anytime soon and will continue to impact many application areas. In particle physics, machine learning is already well established, with notable models including ParticleNet (Qu & Gouskos 2019) and Particle Transformer (Qu, Li & Qian 2022c). Among its common applications are jet classification, physics anomaly detection (Belis, Odagiu & Aarrestad 2024), superior event reconstruction (Pata et al. 2024) and efficient generation of simulated data (Ratnikov 2020).

Summarising, machine learning continues to shape how experimental data is analysed in particle physics and accelerating research at an increasing pace (HEP ML Living Review 2025). These developments have also strongly influenced workflows at CERN, being incorporated into routine work conducted at the LHC (ATLAS Experiments 2023a; ATLAS Experiments 2023b).

For the future, machine learning is expected to bring greatly enhanced potential for scientific discoveries and presents a viable strategy to analyse and organise the massive amounts of data collected in experiments.

2.2 Foundation models

As an influential development in machine learning, foundation models (FMs) are noted for their architectures that can accommodate large-scale datasets and ability to generalise information. The term foundation model was introduced quite recently in a Stanford Report in 2021 (Bommasani et al.) and as such, its exact definition is still debated, aggravated by the overlapping concepts of transfer-learning, self-supervised learning, NLP and General AI. Still, as the name implies, the foundational aspect forms a common ground in the various definitions, stemming from the aptitude for fine-tuning, adaptability and transfer-learning.

To facilitate pretraining on huge datasets, they often take advantage of self-supervised learning strategies to avoid the need for manually annotated data. Through supplementary training referred to as fine-tuning, they efficiently utilise smaller datasets to adapt to more specialised tasks. While originally designed for natural language processing (NLP), foundation models are now being explored in many scientific and non-scientific areas.

The transformer architecture is a key pillar to enabling many modern foundation model architectures and has demonstrated remarkable capabilities to extend to a broad range of use cases outside of NLP. The merits of knowledge generalisation and formation of contextual relations within data cannot be overstated, being central to positioning it as the current standard for the backbone of foundation models. OmniJet-α is no exception and incorporates a transformer-based architecture to perform its tasks.

2.2.1 Transfer learning

Transfer learning in ML is the ability to learn general knowledge from data that extends beyond the dataset, enabling models trained for their original task to be usable in different, often more specific, tasks. At its core, the idea is to reuse the knowledge gained from pretraining data and apply it to other tasks outside the original scope, which stands against traditional supervised learning methods, where new models are trained from scratch.

These transfer learning capabilities can be further exploited through a process called fine-tuning. Fine-tuning refers to the process of adapting a foundation model to a specific task by supplementing its original training with a more domain-specific dataset. While a foundation model may not always outperform task-specific models, the possibility of fine-tuning makes it highly versatile and high levels of performance can often be achieved without the need for an extensive amount of labelled data, reducing both the cost and labour required to support effective machine learning solutions. (Lalor, Wu & Yu 2017)

In particle physics, for example, OmniJet-α demonstrates transfer learning capabilities for jet tagging by fine-tuning its backbone model pretrained in the generative training. Similarly, in NLP, models such as BERT and GPT are fine-tuned for tasks including support agents or named entity recognition. Fine-tuning is therefore an aspect of machine learning architectures that can increase their value and long-term usefulness through repurposing.

Together, these techniques support the efficient and practical development of new machine learning solutions. Using foundation models, researchers can more easily build high-performing models for specialised applications.

A class of machine learning models that makes FMs more accessible are Generative Pretrained Transformers (GPTs). Also based on the transformer architecture, they are generative models that are available in their pretrained form, primed for fine-tuning. Today, they are commonly used for creating Foundation Models and Large Language Models (LLMs).

Over the years, GPTs have demonstrated exceptional performance across applications, with generalisation potential and scalability contributing to their expansion beyond LLMs, in areas including fraud detection (Zhao, Zhu, Li, Wang & Wang 2023), drug development (Chen, Chu & Yang 2025), and scientific computing (Irwin, Dimitriadis, He & Bjerrum 2022). The success of GPTs is closely tied to their increased efficiency achieved through training parallelisation, transfer learning supported by multi-head attention, and the ability to self-supervise.

2.2.2 Data tokenisation

Data tokenisation is a preprocessing step common in machine learning workflows for converting data into compact data representations called tokens. Tokenisation involves converting input data such as text, images, or physical measurements into discrete units of numerical data. This is useful for translating non-numerical data into a format compatible with transformer models and allows for machine learning approaches that can capitalise on tokenised data.

In the context of NLP, tokens may represent individual words, parts of words, or individual characters, but on top of language-related tasks, tokenisation has been effectively adapted for new data modalities. For example, in computer vision, images can be tokenised into image sections, while in physics, numerical measurements can be embedded into tokens.

The primary objective of tokenisation is to keep essential information in the original data, while transforming it into a format compatible with various machine learning models and techniques. However, tokenisation can present challenges in itself. For models to learn effectively, the process must be well-balanced to compress the amount of information present, while preserving necessary detail. In some cases, heavy customisation of existing implementations can be necessary to maintain efficient representations without losing too much information.

In modern machine learning pipelines, tokenisation is a design decision which can have an enormous impact on the resulting models. For that reason, effective tokenisation strategies are key to achieving competitive, state-of-the-art performance.

2.2.3 Vector Quantised-Variational Autoencoders (VQ-VAEs)

Vector Quantised-Variational Autoencoders are a variant of variational autoencoders (VAEs), a class of deep learning neural network architectures designed to encode data into a latent space, describing its hidden, underlying qualities. Unlike traditional autoencoders, VAEs do not relate inputs with constant vectors, but instead use probability distributions to produce varying outputs for identical inputs. Although this strategy introduces non-deterministic outputs, it allows the

model to better capture hidden variables in the data while reducing overfitting and filtering noise.

VQ-VAEs expand upon VAEs by replacing the continuous variables with quantised representations from a codebook that defines the number of permissible vectors. Each input is encoded into tokens by selecting the nearest vector from this codebook, constraining the space to a finite number of possible tokens. As a result, this quantisation creates more compact representations that can still enable high-fidelity reconstructions. The tokeniser model is trained to conserve the fundamental features of data while aiming for maximum likelihood. (Oord, Vinyals & Kavukcuoglu 2017.)

In the context of OmniJet-α, this is applied by embedding the input features p_T, η_{rel} and φ_{rel} into 4-dimensional latent vectors selected from the codebook. A codebook size of 8192 was found to offer an optimal trade-off between accuracy and efficiency. If the codebook is too small, the model becomes unable to capture information with a sufficient level of detail, which leads to compromised reconstruction performance. In contrast, large codebooks may underutilise the codebook and reach a point of diminishing returns, where further accuracy increases become too computationally expensive as a consequence of increased cost of compression (Mentzer, Minnen, Agustsson & Tschannen 2024).

2.3 OmniJet-α

OmniJet-α is a cross-task foundation model for particle physics introduced in 2024. It features a modular architecture with a backbone composed of a configurable number of GPT-decoder blocks, combined with task-specific heads: a generative head for autoregressive token sequence prediction, and a classifier head for supervised jet classification. The model's design was inspired by the recent success of transformer-based architectures in domains beyond natural language processing. (Birk, Hallin & Kasieczka 2024.)

OmniJet-α incorporates a Vector Quantised-Variational Autoencoder as a data preprocessing step for encoding particle features into discrete tokens. This transforms jets into sequences of constituent particles represented by their respective

tokens and ensures compatibility with its transformer-backbone. This strategy allows it to exhibit clear transfer learning ability in jet classification, impacted by the contextual embeddings that the backbone model learns. The approach marks a significant step toward enabling transfer learning in jet physics and lays important groundwork for foundation models and machine learning research in high energy physics.

2.3.1 Model architecture

The first component of OmniJet-α, the VQ-VAE, serves to encode data into token sequences. In this case, tokens are represented by integers ranging from 0 up to the predefined codebook size. Since the foundation model is trained exclusively on tokenised data, it can also learn to autoregressively predict new tokens from an input token sequence. The tokens represent embeddings of the original data points into the learned 4-dimensional latent vector space. When OmniJet-α generates jets, the resulting token sequences can be decoded back into human-readable format via token reconstruction through the VQ-VAE. (Birk, Hallin & Kasieczka 2024.)

Tokenised datasets are stored in the Apache Parquet format for efficient storage and loading. The model's transformer backbone consists of multiple GPT decoder blocks, each containing multi-head self-attention, feed forward layers, and layer normalisation.

```
(0): GPT_DecoderBlock(
  (mha_block): MultiHeadAttention(
    (key): Linear(in_features=256, out_features=256, bias=False)
    (query): Linear(in_features=256, out_features=256, bias=False)
    (value): Linear(in_features=256, out_features=256, bias=False)
    (dropout): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=256, out_features=256, bias=True)
  )
  (ff_block): FeedForward(
    (net): Sequential(
      (0): Linear(in_features=256, out_features=1024, bias=True)
      (1): ReLU()
      (2): Linear(in_features=1024, out_features=256, bias=True)
    )
  )
  (layernorm_1): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
  (layernorm_2): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
)
```

Figure 3: Standard GPT decoder block configuration

After processing the input data, the backbone's output embeddings are passed to task-specific heads. Although the inputs to the model are token sequences, the particle constituents of jets lack an inherent natural order. Therefore, positional encoding has been omitted.

For jet generation, the backbone embeddings are transformed into logits by the generative head, producing a probability distribution over all possible tokens that can follow a given sequence.

Jet tagging is performed with the classification head, sampling over the available jet classes from a probability distribution, after being trained on the input embeddings.

The model has been pretrained on the JetClass dataset, with additional experiments conducted using real-world data from the CMS experiment, made available through the CERN Open Data portal, leading to the public release of a dataset named Aspen Open Jets for use by the broader research community (Amram et al. 2024).

2.3.2 JetClass dataset

JetClass is a dataset of over 100 million simulated particle jets intended for deep learning applications. It was released alongside the Particle Transformer (ParT) model for jet tagging by Qu, Li and Qian (2022). Recognising the need for openly available ML datasets in particle physics, the researchers used MadGraph, Pythia and Delphes to simulate jets of 10 distinct classes. JetClass played a pivotal role in enabling ParT to outperform contemporary models in jet tagging.

The dataset is distributed in ROOT file format, with each file containing 100,000 jets. Available information includes jet labels, constituent particle four-momenta $(p_x,\ p_y,\ p_z,\ E)$, pseudorapidity (η), azimuthal angle (φ), particle labels, and auxiliary information.

For OmniJet-α, the selected features include the transverse momentum (p_T), pseudorapidity (η_{rel}) and azimuthal angle (φ_{rel}) relative to the jet. Particles with $|\eta_{rel}|$ or $|\varphi_{rel}| > 0.8$ were excluded.

2.4 Jet clustering and jet evolution

When particles are detected in high-energy collisions, they are measured as individual, independent entities. To reconstruct jets, these particles are grouped using clustering algorithms that determine which particle pairs to merge at each step of the clustering sequence. As they do not yet necessarily represent a jet, intermediaries of clustered particles are also called pseudojets. Only once the clustering sequence terminates, is the final pseudojet considered a proper jet. (Cacciari, Salam & Soyez 2011.)

Since jets are not fundamental objects, but emergent structures formed by the hadronisation of high-momentum quarks and gluons into stable particles, there is no single definite way to interpret them. The final composition of a jet depends on the specific clustering algorithm and jet definition parameters used.

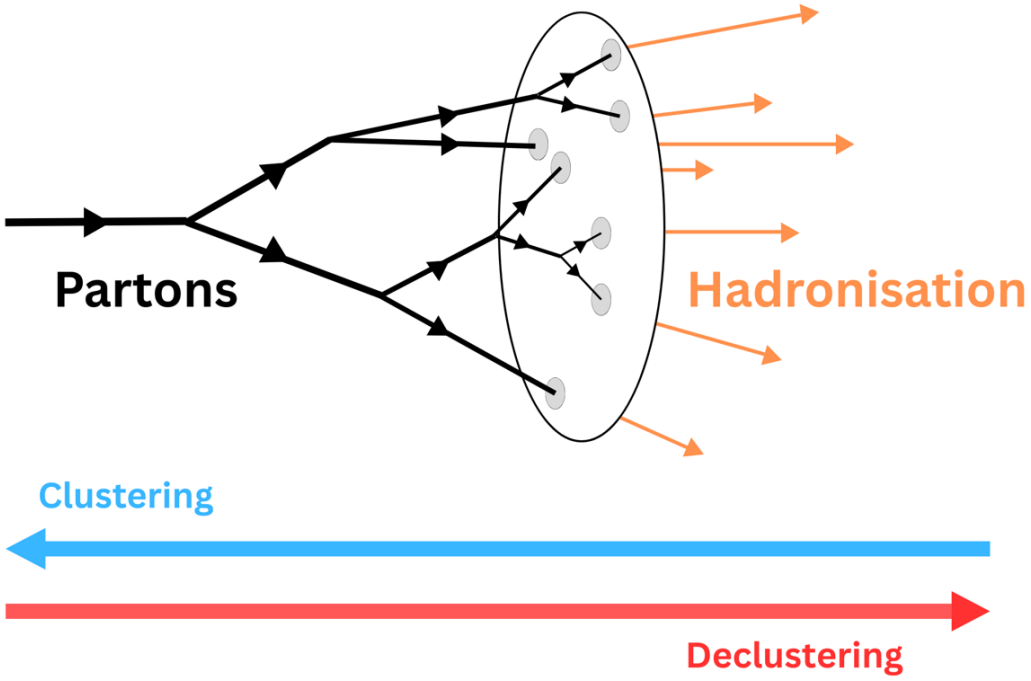

Figure 4: Jet clustering and declustering in a jet. Adapted from ALICE Collaboration (2023).

For jet evolution, declustering can serve as a proxy to describe the particle decays happening inside a jet. By reversing a given clustering order, declustering can produce a sequence of splitting particles, mimicking how high-energy partons fragment into lower energy partons, before eventually turning into the stable particles detected in experiments.

2.4.1 Sequential clustering algorithms

In high-energy particle collisions, stable particles can be grouped into jets according to the set of instructions provided by algorithms. For that purpose, sequential clustering algorithms are widely used, iteratively merging particles or pseudojets based on two measures: the pairwise distance d_{ij} and the beam distance d_{iB} relative to the beam axis. These distances determine the order and structure of a clustering process.

Three commonly used sequential clustering algorithms in particle physics are the k_t, anti-k_t and Cambridge/Aachen algorithms. These algorithms share a common

general form for calculating the pairwise distance and beam distance respectively:

$$d_{ij} = \min\left(p_{ti}^{a}, p_{tj}^{a}\right) \cdot \frac{\Delta_{ij}^{2}}{R^{2}}, \quad d_{iB} = p_{ti}^{a}$$

Here, $\Delta_{ij}^{2} = \left(\eta_i - \eta_j\right)^2 + (\varphi_i - \varphi_j)^2$ represents the squared angular distance in the eta-phi plane, with R stating the jet radius. The exponent a determines the behaviour of the algorithm:

- k_t algorithm ($a = 2$)
- Cambridge/Aachen ($a = 0$)
- anti-k_t algorithm ($a = -2$)

At each clustering step, all pairwise distances and beam distances are evaluated. The smallest distance measure determines which objects are merged into a pseudojet first. If the smallest pairwise distance is the beam distance d_{iB}, the object is considered an inclusive jet and removed from the list of objects. This process repeats until all particles have either been clustered into or are designated as jets.

For example, the k_t algorithm ($a = 2$) defines the two distances as:

$$d_{ij} = \min\left(p_{ti}^{2}, p_{tj}^{2}\right) \cdot \frac{\Delta_{ij}^{2}}{R^{2}}, \qquad d_{iB} = p_{ti}^{2}$$

Due to the squared transverse momentum which exaggerates higher values, the k_t algorithm tends to cluster soft (low-p_T) particles first, leaving clustering of hard particles and pseudojets until the final steps. This makes it well-suited for studying jet substructure and is the reason it was chosen for this research.

2.4.2 Awkward Arrays

Data collected in particle physics is often organised in irregular arrays of non-rectangular shape (Sauerburger 2020). Jet constituents are an example of this

because of their variable number of constituents. In JetClass, the number of constituents per jet typically ranges from 30 to 100 particle constituents. In effort to make efficient use of computing resources and due to large amounts of data processed and gathered in many scientific experiments, it is common to use ragged, “Awkward Arrays” in favour of rectangular arrays as with NumPy or pandas.

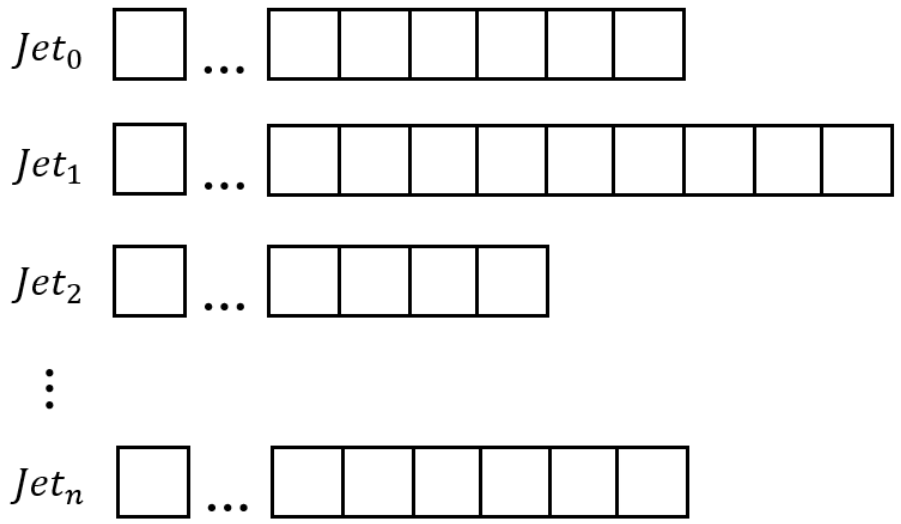

Figure 5: Illustration of jets in ragged arrays

Awkward Array is a library that provides methods designed for computing operations on variable-length arrays, while maintaining efficiency. As the name implies, the inconsistent length and structure of such arrays make them inconvenient to deal with, but as an inherent quality to the data, it must be preserved for jet clustering.

Essentially, Awkward Array offers an alternative to NumPy for handling jagged arrays common for particle physics data, mitigating the risk of information loss from array flattening and avoiding some of the inefficiencies introduced by padding to rectangular shapes.

Making use of Awkward Array to carry out efficient jet clustering on physics data such as proton–proton (pp) and electron–positron (e−e+) collisions, FastJet is a widely used software library in high energy physics. Originally developed for C++, it has since gained support for interfacing with Python. It is a powerful tool that allows for clustering jets using custom jet definitions by selecting the clustering

algorithm, jet radius and the recombination scheme that controls the rules for momentum addition (Catani & Zeppenfeld 2000). The WTA-scheme is a recombination scheme that dictates how the transverse momenta p_T are combined at each clustering step, aligning it with the pseudojet that has the higher momentum (Larkoski & Thaler 2014).

FastJet's high efficiency and compatibility with Awkward Array make it an indispensable tool for clustering jet data as is produced by the LHC. As the name suggests, special measures were taken to optimise computational speed of operations, making heavy use of nearest-neighbour algorithms in most of its jet-finding strategies and achieving a clustering performance of $O(N \ln N)$ instead of the originally expected complexity of $O(N^3)$ as realised by previous approaches (Cacciari & Salam 2005).

3 IMPLEMENTATION

To ensure compatibility with required libraries and dependencies, including the FastJet library, most of the development work for this thesis was conducted on a Windows 10 system running a virtual Ubuntu environment via the Windows Subsystem for Linux (WSL). Programming was done in Python using Visual Studio Code, with the WSL extension installed, for interfacing with the Ubuntu development environment.

The latest version (v1.0.2) of the OmniJet-α GitHub repository was cloned into the WSL environment using the `git clone` command. Subsequently, the JetClass dataset was manually downloaded and moved into the project directory. Only dataset files corresponding to top-quark jets of the $t \to bqq'$-decay channel were used, identified by the filename pattern `TTBar_XXX.root`. Files representing the other nine jet classes were excluded. The resulting subset of the dataset totalled 22.2GB (uncompressed), distributed over 125 ROOT files of approximately 180MB each, with 100,000 jets per file. The files were organised into three separate subfolders for training, testing, and validation, with a split of 100, 20 and 5 files respectively.

Project dependencies were mainly installed via the Docker `requirements.txt` file, with the exception of vqtorch, which was manually installed, as it is required for vector quantisation in the VQ-VAE tokeniser.

Full-scale training of both the VQ-VAE tokeniser and the generative transformer model was conducted on a JupyterHub server equipped with an NVIDIA A100 80GB GPU. Experiment performance metrics were monitored using the Comet ML platform.

3.1 Data preparation

As the JetClass dataset is stored in the ROOT file format, the Uproot library is used to access the data in Python. Uproot enables loading of ROOT files without requiring a C++ environment, allowing for easy integration into Python-based data pipelines.

```
# Define filepath and name
filename = "TTBar_100.root"
filepath = "jetclass/test_20M/" + filename

# Open the file
file = uproot.open(filepath)
```

Figure 6: Loading a ROOT file in Python with Uproot

ROOT is a scientific data analysis framework maintained by CERN, specifically designed to handle large volumes of physics data. It provides efficient storage, fast Input/Output performance, and hierarchical data access through its `.root` file format (Antcheva et al. 2009).

Each JetClass file contains keys representing particle-level and jet-level information. These include:

- Particle kinematics $(p_x,\ p_y,\ p_z,\ E)$
- Particle identification labels
- Jet class labels
- Jet substructure properties such as pseudorapidity η, azimuthal angle ϕ and the number of constituents

```
[part_px],                [part_py],                [part_pz]
[part_energy],            [part_deta],              [part_dphi]
[part_d0val],             [part_d0err],             [part_dzval]
[part_dzerr],             [part_charge],            [part_isChargedHadron]
[part_isNeutralHadron],   [part_isPhoton],          [part_isElectron]
[part_isMuon],            [label_QCD],              [label_Hbb]
[label_Hcc],              [label_Hgg],              [label_H4q]
[label_Hqql],             [label_Zqq],              [label_Wqq]
[label_Tbqq],             [label_Tbl],              [jet_pt]
[jet_eta],                [jet_phi],                [jet_energy]
[jet_nparticles],         [jet_sdmass],             [jet_tau1]
[jet_tau2],               [jet_tau3],               [jet_tau4]
[aux_genpart_eta],        [aux_genpart_phi],        [aux_genpart_pid]
[aux_genpart_pt],         [aux_truth_match]
```

Figure 7: Available keys in JetClass files

After loading, the four-momenta of particles in all jets are extracted and stored as vector arrays to allow for efficient vectorised operations. Each jet is then truncated to a maximum of 128 particles. Preprocessing cuts are applied with a preprocessing dictionary `pp_dict_cuts`, removing particles not sufficiently aligned with their corresponding jet axis. The resulting particle data is then fully prepared for clustering via FastJet.

3.1.1 Jet clustering with FastJet

The particle data is reclustered into new jets using the k_t-algorithm, with a jet-radius of $R = 0.8$ and using the Winner-Take-All (WTA) recombination scheme. The resulting output includes:

- Inclusive jets
- Jet constituents and the number of constituent particles
- Jet structure array: p_T, η_{rel} and φ_{rel} for each constituent, intermediary pseudojet and inclusive jet
- Unique history order: the sequence of constituent and pseudojet mergers leading to the proper, inclusive jet

This output is returned as Awkward Arrays, which can be utilised to analyse the jet substructure and jet clustering.

In contrast to inclusive jets, exclusive jets represent non-final pseudojets prior to the final clustering step. For instance, when setting $n\_jets = 3$, the last three pseudojets before their final mergers are returned.

FastJet's `_jets()` array, hereafter referred to as the "jet structure array", contains four-momentum information for the structural elements of each jet: constituent particles, intermediary pseudojets and the final inclusive jet. It is one of two major components of the data preprocessing pipeline established in this work, used for validating clustering sequences and extracting the features used for tokenisation.

One more especially relevant piece of data obtained through FastJet is the clustering history or “unique history order”. It contains a complete sequence of clustering steps that describes the order that needs to be followed to obtain the inclusive jet. By parsing it, it can be used reconstruct the full clustering process and details which particles or pseudojets are merged at each step. As a concrete example, Figure 8 shows the unique history order output for a jet in the file TTBar_029.root, clustered using the k_t algorithm with a radius of $R = 0.8$ and the WTA recombination scheme.

```
[0, 10, 25, 33, 23, 34, 14, 45, 24, 30, 40, 49, 13, 28, 44, 27, 46, 50, 11, 15, 35, 56, 60,
1, 31, 41, 3, 8, 48, 51, 2, 21, 26, 32, 42, 16, 20, 37, 29, 38, 55, 58, 61, 4, 22, 36, 9, 47,
18, 19, 43, 53, 6, 7, 12, 39, 52, 17, 54, 57, 5, 59, 62, 63]
```

Figure 8: Clustering history of a jet

Each jet consists of $2N - 1$ elements and $N - 1$ pseudojets, where N is the number of constituent particles and the final pseudojet represents the fully clustered jet. In this example, the jet consists of 32 constituents (indices 0-31) and 32 pseudojets (indices 32-62), with index 62 corresponding to the final jet. The last element 63 indicates the termination of the clustering.

By referencing the jet structure array, the features for each constituent and pseudojet can be obtained, as shown in Figure 9.

```
print("Particle four-momentum:\n")

print(cluster.jets()[0,0])
print("pT:", cluster.jets()[0,0].pt)
print("η:", cluster.jets()[0,0].eta)
print("ϕ:", cluster.jets()[0,0].phi)
```

```
Particle four-momentum:

{px: 139, py: 58.9, pz: -409, E: 436}
pT: 150.56389627495253
η: -1.7248798045144453
ϕ: 0.4019544204658763
```

Figure 9: Particle features: four-momentum $(p_x,\ p_y,\ p_z,\ E)$, p_T, η and φ

In the sequence in Figure 8, the first pseudojet (index 33) resulted from merging particles 10 and 25, with $p_T \approx 11.15$ and $p_T \approx 1.91$ respectively. Their combined transverse momentum $p_T \approx 13.06$ matches the values for pseudojet 33. This ap-

proach makes it possible to cross-reference the clustering sequence with the particle and pseudojet features from the jet structure array, providing a way to validate the clustering behaviour and track the evolution of jet structure.

3.1.2 Simulating jet evolution

For the simulation of jet evolution, binary trees are an important concept, since they sufficiently describe the structure of binary particle decays, as represented by their declustering steps. A binary tree is a hierarchical structure composed of elements called nodes. Intermediate nodes are characterised by their connections to a higher-level node, referred to as parent, and up to two lower-level nodes, referred to as children. The starting point of the tree is called the root, and end points of a tree are called leaf nodes. Each level of hierarchy is defined by its depth, representing the number of links from the root to any given node.

In the type of binary tree present in this work, each node apart from the root has exactly one parent node and either zero or two children, forming a structure consisting of branches that represent the paths between nodes. To traverse a binary tree, common strategies are algorithms such as the depth-first search (DFS) and breadth-first search (BFS) algorithm. DFS moves down each branch up to its leaves where it moves back until it gets to a new branch, whereas BFS explores all the nodes of the current depth before proceeding to the next. (Verma 2022.)

This binary tree formulation is especially relevant in the context of jet physics, where it creates a consistent pattern for how consecutive (binary) particle decays can be modelled. Jets that have been reclustered can be transformed into binary trees by referencing the previously introduced unique history order of FastJet. Particle constituents form the tree leaves and pseudojets constitute the intermediate parent nodes, with the final clustered jet embodied by the root.

By interpreting clustering steps in the unique history order as the inverse of a particle decay, the declustering steps are obtained through the reversal of this order. Rather than merging two particles into one parent, declustering is the act of splitting pseudojets into children, conceptually mimicking how a particle decays into two decay products.

However, when read in reverse as is, the unique history order loses the hierarchical information required to reconstruct the full binary tree structure. To solve this ambiguity, the clustering history is parsed in its intended forward direction while tracking both, constituents and pseudojets by their indices. Each merging step can then be recorded as an array of three indices, consisting of a parent and its two children. This triplet structure preserves the tree hierarchy and provides clear instructions for reconstructing the full tree structure, starting from the root. As stable constituent particles do not split further, they are assigned None as their children.

```
[[62, 59, 61],
 [59, 5, 57],
 [5, None, None],
 [57, 54, 53],
 [54, 17, 52],
 [17, None, None],
 [52, 39, 6],
 [39, 12, 7],
 [12, None, None],
 [7, None, None],
 [6, None, None],
```

Figure 10: Section of an array with declustering steps

To simplify model training and reduce the number of necessary modifications to the model architecture, the nested structure is flattened into one-dimensional arrays per jet, preserving the same information and maintaining better compatibility with the architecture of the generative model.

3.1.3 VQ-VAE tokenisation

For data tokenisation, the Vector Quantised-Variational Autoencoder (VQ-VAE) was retrained and modified from its original implementation for several key reasons.

First, although the number of jet constituents was limited to 128, the clustering process introduced additional pseudojets, increasing the number of elements per

jet by up to $N_{particles} - 1$ (127 for jets with 128 particles). Consequently, the maximum sequence length increased to 255, necessitating a higher padding length parameter of 256.

In addition, the original VQ-VAE was trained exclusively on jet constituents, not pseudojets. Since pseudojets result from combining particle momenta, their typical transverse momentum is considerably higher than that of average constituents. This change caused a considerable drop in reconstruction performance of pseudojets in the baseline. By relying on FastJet's jet structure array, pseudojets could be included alongside constituents in the training data, boosting the overall reconstruction accuracy substantially. To accommodate the added pseudojets and the resulting wider range of information needing to be encoded, the codebook size was increased by 50% to allow for more granular token representations.

The effect of the replace frequency parameter, that controls how easily the VQ-VAE assigns new vectors to infrequently used tokens, was also studied. The tested values (5 and 20), however, showed no notable improvements over the default value of 10.

The modified VQ-VAE was trained on a computing cluster equipped with a high-performance A100 GPU and 80GB of memory. Using input data of 40,000 top-quark jets per file, the training completed after approximately 8 hours. A best model checkpoint callback tracked performance based on the absolute p_T reconstruction error metric and saved the 5 best-scoring model checkpoints.

Compared to the baseline VQ-VAE, the modified version displayed significant improvements in reconstruction accuracy with notably reduced p_T discrepancies in reconstructed pseudojets.

```
        -- pT of last 5 pseudojet elements --
Original jet:           pT: [187, 226, 175, 362, 588]
New VQ-VAE:             pT: [179, 250, 167, 370, 586]
Baseline VQ-VAE:        pT: [162, 201, 170, 361, 456]

Original jet:           pT: [89.6, 146, 235, 292, 595]
New VQ-VAE:             pT: [82.6, 149, 239, 272, 595]
Baseline VQ-VAE:        pT: [83.4, 124, 205, 233, 446]

Original jet:           pT: [195, 243, 216, 296, 540]
New VQ-VAE:             pT: [177, 251, 210, 285, 497]
Baseline VQ-VAE:        pT: [247, 247, 215, 274, 562]

Original jet:           pT: [300, 392, 179, 440, 619]
New VQ-VAE:             pT: [288, 398, 186, 427, 588]
Baseline VQ-VAE:        pT: [314, 377, 238, 419, 551]

Original jet:           pT: [101, 181, 383, 232, 615]
New VQ-VAE:             pT: [94.1, 165, 371, 230, 588]
Baseline VQ-VAE:        pT: [91.9, 159, 315, 195, 447]
```

Figure 11: Comparison between reconstructed and original p_T values

Over the whole jet structure, the mean percentage error was lowered to less than 3.5%. In contrast, the original VQ-VAE had notably lower accuracy of close to 6%, with inclusive jets specifically displaying a mean error of around 13%.

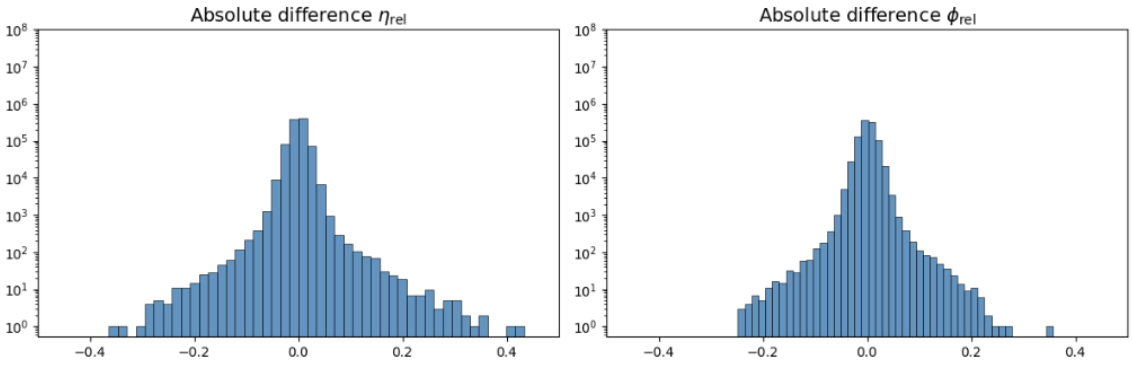

Figure 12: Distribution of absolute errors for η_{rel} and φ_{rel}

Additionally, the revised model demonstrated clear improvements in reconstructing the relative pseudorapidity and azimuthal angle, with the absolute reconstruction errors of η_{rel} and φ_{rel} remaining low overall and outperforming the baseline model. These results suggest that the new VQ-VAE is more effective at tokenising data and generating high-quality inputs for the jet evolution task.

Following the tokenisation, each sequence of constituents and pseudojets consists of tokens in the form of integer values, ranging from 0 up to the VQ-VAE's codebook size - 1. However, these tokens are still in the standard order given by FastJet, not containing any information pertaining to the order of clustering steps. Therefore, before storing the data in the Apache Parquet file format, the tokens must be arranged according to the jet declustering order. For that purpose, in place of constituent and pseudojet indices, the tokens generated by the VQ-VAE are inserted into the arrays containing the declustering information based on their indices.

For constituent parents that do not decluster further, indicated by `None` objects, values are replaced with a stop token equivalent to the VQ-VAE's codebook size+1 to avoid being reconstructed into actual values, but still allow being learned by the generative model. The resulting arrays represent each declustering step as a triplet of tokens: the parent, the left child, and the right child.

```
File: TTBar_113_tokenised.parquet
Jet 8:

Parent = left + right: [4101]   =       [21] + [11472]
Parent = left + right: [21]     =       [9084] + [2218]
Parent = left + right: [9084]   =       [7305] + [9084]
Parent = left + right: [7305]   =       [12289] + [12289]
Parent = left + right: [9084]   =       [756] + [7262]
Parent = left + right: [756]    =       [5310] + [3062]
Parent = left + right: [5310]   =       [12289] + [12289]
Parent = left + right: [3062]   =       [7118] + [9557]
Parent = left + right: [7118]   =       [12289] + [12289]
Parent = left + right: [9557]   =       [12289] + [12289]
```

Figure 13: Interpretation of encoded token sequences

Whereas Figure 13 illustrates how the token sequences should be interpreted, Figure 14 shows the format of the pure numerical sequences that the tokenised data is stored as.

```
[    0.  4101.    21. 11472.    21.  9084.  2218.  9084.  7305.  9084.
  7305. 12289. 12289.  9084.   756.  7262.   756.  5310.  3062.  5310.
 12289. 12289.  3062.  7118.  9557.  7118. 12289. 12289.  9557. 12289.
 12289.  7262.  3446.  8196.  3446.  3679.  4930.  3679.  3693.  7119.
  3693. 12289. 12289.  7119. 12289. 12289.  4930. 12289. 12289.  8196.
```

Figure 14: Raw data as NumPy array

Due to the unambiguous representation of each declustering step and the consequently introduced repetition of parent nodes, along with the inclusion of stop tokens, the length of the resulting arrays increases by a factor of 3, extending the maximum sequence length to 765.

Finally, the processed Awkward Arrays are stored as `.parquet` files, finalising the data preprocessing and producing the final dataset used for training the machine learning model.

3.2 Generative model

The generative model is composed of a multi-layered GPT backbone with a configurable number of encoder blocks and a generative head. For the jet evolution task, the backbone was left mostly unchanged, but the model head was modified more substantially. Specifically, instead of a single generative head processing the full backbone output, the output is split into left and right outputs, and the architecture extended to two heads: a left and a right head. Both heads have separate sub-tasks, predicting the tokens representing the left and right children of a parent.

For that purpose, after initial loading of the tokenised dataset, the input sequence is sliced into three subarrays: input, left targets and right targets. The input contains the whole sequence of tokenised particles/pseudojets with parents, left and right children, whereas the targets contain the tokens of the respective children that the two heads are trained to predict.

First, the input token sequences are transformed by the backbone model into embeddings via an embedding table. Due to the now sequential nature of the

training data (successive particle decays, instead of unordered collections of constituents), positional encoding was incorporated into the backbone. The positional embeddings encode information about the order of tokens within each jet by their index. The positional embeddings are then simply added to the inputs and processed by a stack of GPT decoder blocks, applying multi-head attention and feed forward transformations to produce contextualised embeddings.

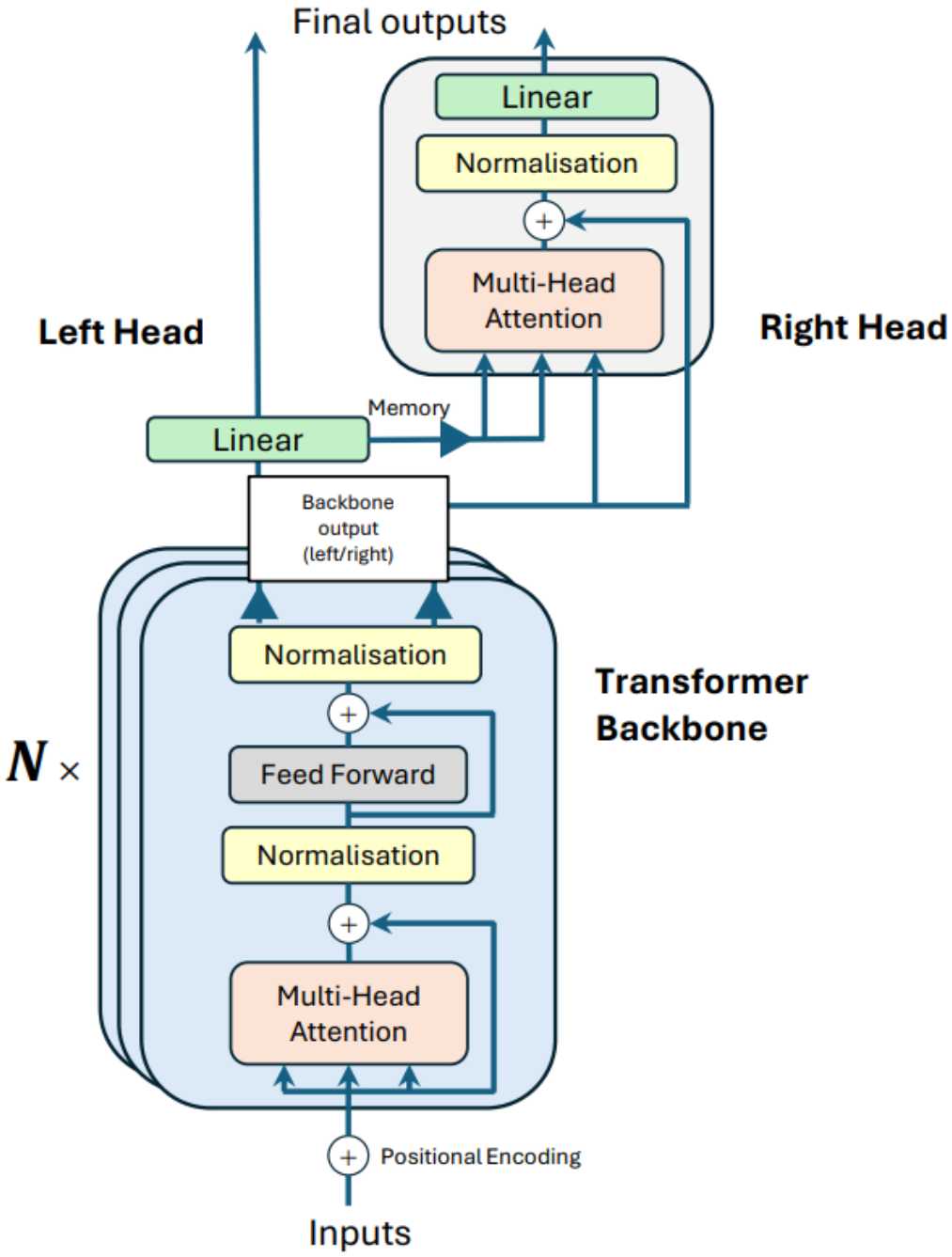

Figure 15: Revised model architecture with dual head configuration

Afterwards, the backbone output, which still contains the embeddings of the whole input, is divided into embeddings matching with the left and right children. These are then passed to the matching heads.

Given the dependency between decay products in a particle decay, e.g. momentum conservation, a cross-attention layer was added to allow conditioning the right head's predictions on the left. While left head retains the original design, its keys and values are also passed to the right head as memory for cross-attention. The output of the cross-attention layer is then added to the right inputs and normalised. Finally, like with the left head, it is passed through a linear transformation for producing logits.

Since it would be insufficient to just calculate the loss of the left and right head independently, due to the two possible permutations of outputs and targets, two losses are computed, one matching the logits from left and right head with the left and right targets accordingly, and another one with a swapped order, instead matching the left head's logits with the right targets and vice versa. The minimum loss values were then used to calculate the final mean loss.

3.2.1 Model training

The jet evolution model was trained using a slightly modified set of hyperparameters from the baseline generative model. Although the original configuration of eight attention heads and three GPT blocks was left unchanged, the padding length and maximum sequence length for the autoregressive token sequence generation were raised to 512 to account for the longer input sequences. To prevent crashes during training, the embedding dimension needed to be scaled up accordingly. The vocabulary size was set to 12,290 to reflect the VQ-VAE's codebook size of 12,288, plus the additional start and stop tokens.

Training was performed by loading 30,000 jets from each file in the tokenised dataset, bringing the total to 3 million jets. The best model checkpoint-callback was activated, as with the VQ-VAE, to save the five best models as measured by their absolute p_t reconstruction error during validation. The training concluded after 7.5 hours and 27 epochs, using a single A100 80GB GPU machine.

3.2.2 Batch generation

In addition to predicting the products of particle decays from a token input sequence, it is also possible for the model to generate token sequences while assembling them in a binary tree structure. This is accomplished by reusing each pair of predicted children to act as parents for future predictions. This strategy is similar to the original jet generation with OmniJet-α, but it introduces the need for several additional considerations. For one, generating binary trees in parallel is a non-trivial task, particularly when accounting for leaves. Furthermore, a simple programming stack approach is not viable as the number of elements is dynamic and would introduce variation during generation, requiring additional work.

The favoured approach, although suboptimal, was to generate predictions for all nodes, including those of stop tokens. As a result, the number of total nodes in the tree doubles at each depth, producing $2^{N+1} - 1$ nodes for a tree of depth N. For instance, trees of depth 8 already consist of 511 tokens, severely limiting achievable tree complexity. Analysing tree depths present in the dataset revealed a typical depth ranging between 8 and 14. Despite this limitation, the approach was considered to be viable enough for implementation within the project scope.

In the final implementation, the model uses a loop to generate two tokens at each step, basing its predictions on the current running sequence: one prediction made by the left head and the other made by the right. The left head's predicted token is appended first, after which the right head predicts on the updated sequence and the right token is appended. To preserve the [parent, left child, right child] triplet structure that was used during training and to generate a parent for the next prediction, a third token must also be appended before the start of the next loop. Since each future parent token is derived from a previously generated child, the third appended token is selected from the already generated sequence. This generative loop repeats until the maximum sequence length is reached.

3.2.3 Trained model inference

To utilise the fully trained model for inference and generate token predictions, the `model.predict()` method offers a straightforward and flexible interface. It is used passing the token sequences and then predicts the final two tokens by

masking the last two token positions. The model's left and right heads then output the most probable child tokens based on the preceding context.

```
--- Predicting both children for the following sequence: [27]

Left head prediction: 8086
Right head prediction: 4942

Ground truth:
Parent: 27 => left: 5379, right: 2218

Predicted sequence: [27, 8086, 4942]

Actual sequence: [27, 5379, 2218]
```

Figure 16: Model inference

Both the input and the predicted token sequences can be reconstructed back into physical space by reconstructing them with the VQ-VAE, which makes the predictions immediately interpretable and quantifiable.

```
Predicted sequence: [27, 5379, 2218, 5379, 5185, 1546]

Actual sequence: [27, 5379, 2218, 5379, 1427, 7168]
100%|██████████| 1/1 [00:00<00:00, 152.03it/s]

Reconstructed actual tokens:
Parent: {pt: 188, etarel: -0.24, phirel: -0.443}
Left child: {pt: 67.2, etarel: -0.457, phirel: -0.121}
Right child: {pt: 131, etarel: -0.248, phirel: -0.439}

Reconstructed predicted tokens:
Parent: {pt: 190, etarel: -0.2, phirel: -0.446}
Left child: {pt: 142, etarel: 0.0213, phirel: -0.196}
Right child: {pt: 33.6, etarel: 0.138, phirel: 0.384}
```

Figure 17: Model inference and reconstructed values

In addition to longer sequences, the model supports inference on single tokens by passing an arbitrary integer value within the permitted vocabulary range via the `model.predict()` method. The output again consists of two predicted tokens,

produced by the respective left and right heads, which can also be called independently. This provides a practical way for evaluating the model's ability to predict the decay products of a particle without additional context.

```
--- Predicting both children of a single token: [56] ---

Left head prediction: 718
Right head prediction: 262

Parent: 56 => left: 718, right: 262
```

Figure 18: Single token inference

Furthermore, the model can generate entire token sequences as outlined in the batch generation sub-chapter. This functionality is implemented through the `.generate_batch()` method, constructing complete batches of generated sequences up to a desired length.

```
[[0, 9856, 6961, 9856, 11558, 263, ..., 7409, 3094, 1242, 1242, 1242, 2441],
 [0, 9041, 6245, 9041, 5008, 266, 6245, ..., 3930, 3930, 3930, 74, 1768, 8572],
 [0, 3658, 11047, 3658, 1866, 1866, ..., 4716, 1965, 9967, 1509, 11472, 6556],
 [0, 5399, 5399, 5399, 8213, 8213, ..., 9640, 5010, 1852, 1781, 1781, 10588],
 [0, 10054, 11060, 10054, 1850, 8441, ..., 10487, 10487, 8259, 269, 269, 9835],
 [0, 9165, 4210, 9165, 8252, 8252, ..., 11975, 11975, 1560, 6847, 6847, 4172],
 [0, 11729, 8303, 11729, 11543, 11543, ..., 4960, 4539, 11020, 11176, 4243],
 [0, 2491, 9691, 2491, 11481, 11481, ..., 10507, 6133, 10160, 7582, 6891, 1946],
 [0, 3315, 10241, 3315, 2922, 9665, ..., 2278, 7633, 1460, 1460, 1460, 1460],
 [0, 9183, 9425, 9183, 9717, 9717, ..., 4564, 4564, 4564, 4564, 4564, 8387],
 [0, 3354, 3354, 3354, 3868, 3800, ..., 10237, 10237, 6131, 9853, 9386, 3215],
 [0, 8042, 8042, 8042, 10367, 11384, ..., 1294, 1294, 1294, 1294, 1294, 10994]]
--------------------------------------------------------------------------------
```

Figure 19: Batch generation of jets with encoded hierarchical structure

These capabilities form the model's feature set for inference, enabling experimentation, evaluation and the generation of jet-like structures made up of tokens. This enables a variety of potential usages, like the synthetic replication of jet substructures, the study of particle jet behaviour and the simulation of jet evolution.

4 RESULTS

This work produced a range of results throughout its different development phases. Early on, a method was implemented to reliably parse and interpret the unique history order generated by FastJet in Python. While a small portion of samples was later excluded due to low transverse momentum, the method proved effective in handling cases where FastJet reclustered jets into multiple inclusive jets, which accounted for less than 3% of the total dataset.

Individual clustering steps were successfully extracted from the clustering history, enabling the construction of complete declustering sequences. These sequences included both jet constituents and pseudojets, represented as discrete tokens.

Preprocessing of all 125 files in the dataset, containing jets of the hadronic top-quark decay channel, was fully automated. The data pipeline reads data from ROOT files, performs reclustering using FastJet, extracts the declustering sequences, substitutes tokens, and finally stores the results in Apache Parquet format. Initial implementations of the extraction and token insertion processes were less computationally efficient, requiring approximately 30 minutes per file. Through optimisations (e.g. inserting elements into arrays instead of appending to the end), this time was reduced to under 3 minutes per file on a standard desktop PC.

The VQ-VAE's architecture was modified to improve its ability to reconstruct both constituent particles and pseudojets. As anticipated, the revised model demonstrates significantly improved reconstruction performance relative to the baseline, which is expected since the latter had been trained on data lacking pseudojets. Since pseudojets often exhibit high transverse momentum and different typical value ranges for pseudorapidity and azimuthal angle compared to constituents, their absence in the original training data limited the baseline model's capabilities.

Retrained VQ-VAE models showed clear improvements in reconstruction accuracy. However, persisting limitations later became clear. Most notably, noise in the upper p_T range suggested that the current tokenisation strategy does not encode pseudojets with sufficient granularity. Rectifying this shortcoming is essential for improving both the VQ-VAE and any of its downstream applications.

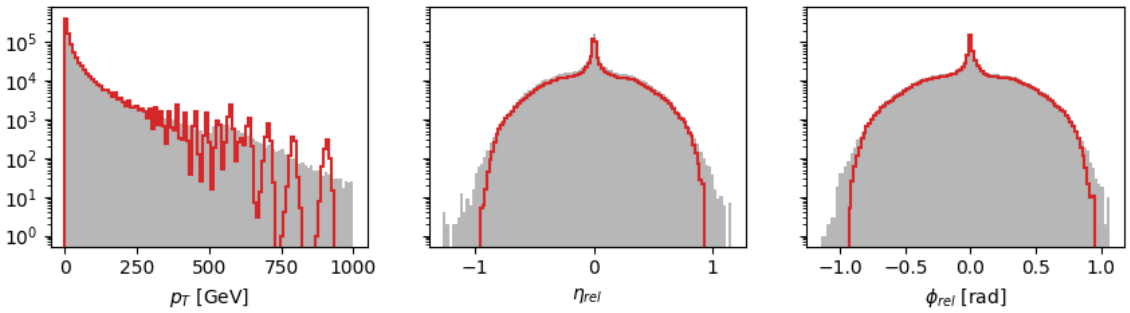

Figure 20: Reconstructed features compared to actual distributions

The generative model, which was expanded with the newly designed architecture, initially showed promising behaviour during training, with rapidly decreasing validation loss. However, this metric was soon found to be unreliable for indicating actual predictive performance.

An attempt to remedy the class imbalance between tokens was made, as it was observed that certain tokens occurred at a much higher frequency. This was done by computing token weights based on the overall token frequency in the dataset and incorporating them into the loss function. Despite this change, the inclusion of token weights did not appear to make an impact. When training the final model using this new loss calculation, after approximately two hours of training the validation loss had abruptly increased before gradually decreasing again, indicating training instability.

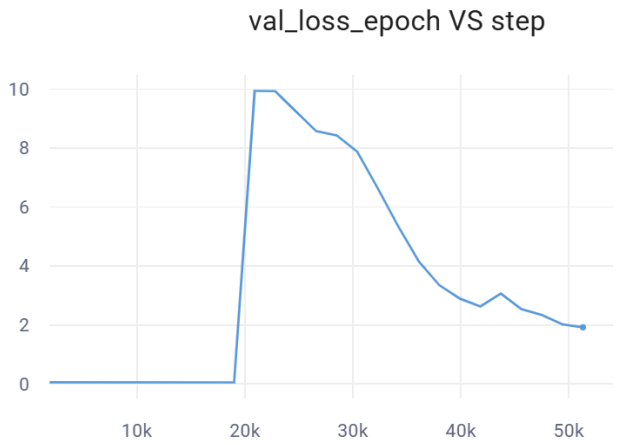

Figure 21: Validation loss during training

While the fully trained model is capable of generating predictions during both standard inference and batchwise jet generation, predictions are poor, as confirmed in testing.

Despite these shortcomings, the model exhibits non-uniform probability distributions and variable confidence scores, rather than uniformly low-confidence that would be expected from models without learning capabilities, suggesting some level of pattern learning ability. Additionally, the model's right head produces (slightly) differing outputs when conditioned on predictions from the left head, compared to isolated inference, indicating that cross-attention may be a viable strategy for creating the needed dependency between the two predictions.

Nevertheless, token predictions are largely incorrect, and both heads frequently output identical tokens. This tendency becomes especially apparent during batch generation, where token sequences show little diversity, even after accounting for expected repetition of parent tokens.

Reconstruction of these sequences using the VQ-VAE confirmed that the generated tokens did not correspond to expected particle features in the physical space. This indicates that, even though the model architecture may be capable of learning meaningful representations in principle, the training data did not contain sufficiently learnable patterns, likely impacted by the shortcomings in the tokenisation process.

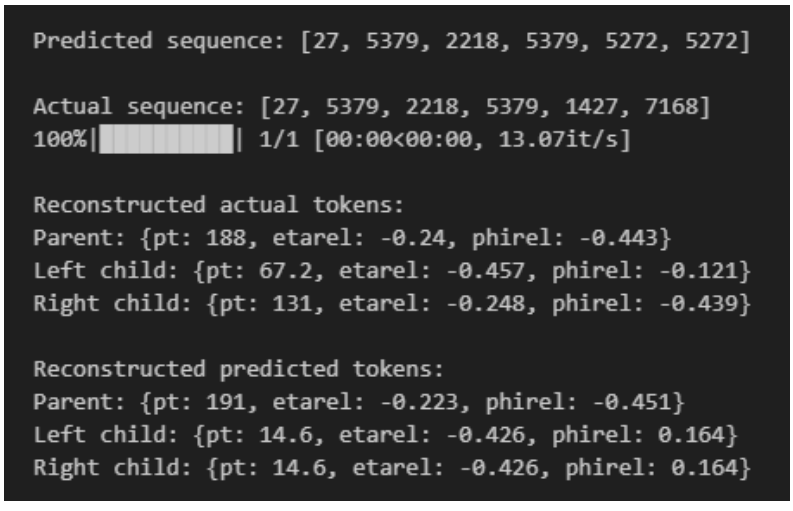

Figure 22: Reconstruction of predicted tokens

To explore the impact of jet definitions for jet clustering, a final version of the VQ-VAE was trained using data reclustered with FastJet's E-scheme to provide a direct comparison point to the WTA-pt scheme that was used earlier throughout the study. Although limitations remained, this model produced the most accurate results so far, reducing reconstruction noise and demonstrating clear potential for future work.

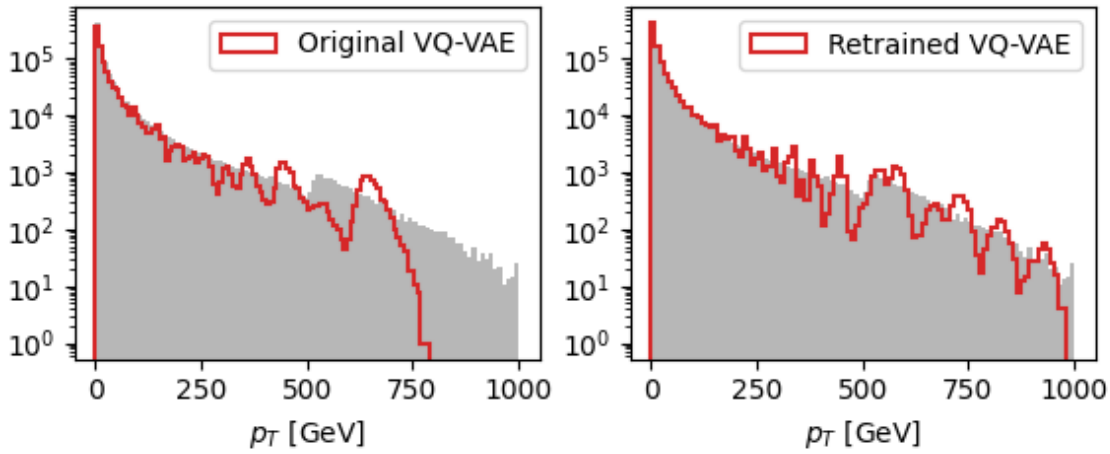

Figure 23: Distributions of reconstructed p_T (red line) of particles and pseudojets for the baseline VQ-VAE and the final VQ-VAE model. The true p_T values are represented by the grey distribution.

4.1 Limitations

The main limitation was the time frame of the project. Having only a couple of months to form the knowledge base, prepare and learn the tools and resources, and engineer the required solutions presented a steep challenge and gave rise to most of the following technical limitations.

At the conclusion of the research, tokenisation and feature engineering for the VQ-VAE were identified as critical limiting factors, particularly in their impact on the generative model's learning capacity. The modified VQ-VAE demonstrated improvements over the baseline, especially in the p_T range above $500\ GeV$. This was expected however, given that the original model was trained exclusively using particles which are infrequent in this range. For the modified version, the upper p_T range was better covered by the pseudojets present in its training data.

Despite the improvements, a central issue lies in the fact that the relative pseudorapidity and azimuthal angle are computed with respect to the jet axis. Pseudojets near the final jet tend to have values closely approaching zero, causing most of them to exhibit very similar values. As a result, the VQ-VAE tends to approximate many pseudojets using only a small subset of tokens, reducing the effective resolution of their representation.

Transverse momentum, while better preserved in the modified setup, also suffered from insufficient detail, likely compounded by the aforementioned limitations. This lack of detail in their representations poses a significant challenge for the jet evolution modelling task. Since pseudojets represent the earlier stages of the jet formation process, errors in their reconstruction severely compromise the latter stages of the jet evolution. After all, if the initial stages are modelled unreliably, accurate prediction of subsequent particle decays becomes increasingly unlikely, as the model is trained on and informed by the preceding sequences.

Another complication arose from how the clustering sequences were encoded using FastJet's clustering history. Specifically, due to varying jet definitions, a single jet could be split into multiple inclusive jets after reclustering. This introduced a mismatch between the clustering histories and the jet structure array, complicating the accurate substitution of tokens during preprocessing. A method for correctly adjusting the indices of the clustering histories was eventually implemented, but these cases were ultimately excluded from training. This was due to affected samples accounting for less than 3% of the dataset, and because their combined transverse momentum, originally in the $500 - 1000\ GeV$ range, was shared across multiple lower-energy jets. This made them less suitable for model training as it would introduce unwanted outliers in the data.

Further limitations emerged during the tokenisation process due to how particles are filtered and transformed prior to clustering. A preprocessing dictionary was used to apply cuts, discarding certain particles and applying feature transformations for tokenisation. However, applying these cuts after clustering introduced inconsistencies in the clustering history, rendering it unusable. Although applying cuts before clustering resolved this, the use of the WTA-pt scheme caused another complication: the altered alignment of the jet axis. Consequently, these

changes led to additional unintended cuts, again resulting in mismatches between reclustered jets and token sequences. This problem was addressed through the creation of a separate dictionary that only applies feature transformation, allowing cuts and feature transformation to be applied independently.

Several issues were caused by stop tokens, as the VQ-VAE is unable to decode tokens outside its defined token space. This necessitated the removal of stop tokens from sequences before reconstruction, but results in a significant loss of hierarchical information. As the stop tokens are essential for preserving the tree structure, their exclusion severely affects the explanatory power of the generated sequences in the current setup.

Regarding the format and structure of the generated training data, while storing the parent-child relationships as triplets simplified parsing, it also significantly increased sequence length, tripling it when accounting for the additional padding required. This contributes to inefficient storage and increases computational overhead during model training, particularly in memory usage and batch processing.

During batch generation, assembling full binary trees without pruning the leaves, i.e. skipping stop tokens for prediction, quickly becomes infeasible with increasing tree depth due to the exponential growth in the number of nodes. However, dynamic generation of trees in a vectorised and parallelisable manner is non-trivial and was out of the scope of the work.

Finally, the current implementation uses a token value of 0 as the start token for each sequence in the batch generation. Instead of generating predictions using this token value, future iterations should aim to sample from a more representative token distribution instead.

4.2 Recommendations

Building on the limitations outlined in the previous section, the author established several key opportunities for improvements to enhance the performance and robustness of the VQ-VAE model, the generative model, and the accompanying data preprocessing pipeline.

First, exploring alternative methods for parsing and encoding of declustering sequences, while preserving their hierarchical structure could lead to a substantial reduction in sequence length by up to 66%. This would reduce storage and training overhead and facilitate the implementation of various tree traversal strategies for training and inference. It may also eliminate the need for complex indexing schemes and support better optimisation of batch generation.

Unlike its Python interface, the FastJet C++ package offers its complete feature set and may provide more effective means of retracing the clustering history and further streamline the data preprocessing pipeline.

To improve tokenisation, modifications in the feature engineering are essential. Transforming the relative pseudorapidity and azimuthal angle in a way that balances out the dataset and magnifies the differences between pseudojets should allow the VQ-VAE to resolve their features better. In addition, identifying other particle- and jet-level features with high predictive quality for jet evolution could be instrumental in achieving satisfactory downstream model performance. High tokenisation accuracy and resolution should remain one of the core objectives of future work in this direction.

In its current form, the VQ-VAE does not support the input of stop tokens, and it is recommended to either adapt it to explicitly support handling of stop tokens, or to employ masking strategies to preserve structural information when reconstructing data that depends on these intermediary stop tokens for structural information. Alternatively, developing an improved representation of the binary tree structure could remove the need for intermediary stop tokens entirely and alleviate the issues related to high sequence length.

It may also be of interest to calculate the tree depths of pseudojets and particles and include them in the input features for training the VQ-VAE, as it could allow it to better capture differences between data points than relying solely on p_T, η_{rel} and φ_{rel}.

Although the modified generative model addresses the dependencies present in particle decays, further experimentation with the model architecture is advised.

In the long term, a robust model for jet evolution may be integrated with the current generative and classification tasks to unlock even greater transfer learning potential.

Finally, avoiding the generation of new tokens from stop tokens in batch generation is crucial for good performance. A solution for this would enable longer, more expressive tree structures without excessive computational requirements. A better way to initiate these sequences would be to replace the currently fixed start token with one sampled from a token distribution that reflects the desired properties, such as tokens that represent partons initiating a jet or subjet.

These recommendations present a basis for extending this study and support future endeavours to model and understand jet evolution, ultimately motivated by the goal of exploring the boundaries of the Standard Model and advancing the understanding of fundamental physics.

4.3 Conclusions

The author presents a fully functional end-to-end pipeline for processing and storing reclustered particle jet data suitable for training machine learning models to predict particle jet evolution. A proposed model architecture was implemented and trained, possessing the required functionality to make simultaneous predictions of token pairs and autoregressively generating batches of hierarchically structured data, but has yet to demonstrate successful predictions.

Although the model is unsuccessful at making accurate predictions in its current form, the findings of this study demonstrate that the prediction of jet evolution can, in principle, be implemented using a transformer-based architecture, provided that learnable patterns exist in the training data, thus addressing the main research question of: “How can the evolution of particle jets be predicted using machine learning?”

While no definitive claims can be made with respect to the predictability of jet evolution, a data preprocessing strategy has been established and tested. The strategy encompasses the entire transformation process from raw ROOT files through specific steps, to files storing the resulting tokenised representations.

The work further addresses the sub-questions by demonstrating changes to the model architecture, adapting it to the hierarchical data format that was created for representing jets. It takes necessary steps to address its goal by implementing positional embeddings in the backbone, dividing backbone output between the two heads, one of which was enhanced with cross-attention, and finally, an adapted loss-calculation. Two core functionalities were realised in accordance with the new task: token predictions and batch generation.

Nevertheless, refinements are needed to improve the data format, particularly regarding the efficient conversion of binary trees into one-dimensional arrays. This may be supported by more flexible parsing strategies, like enabling both depth-first or breadth-first orders for training input, or by directly encoding the structural information into the token representations.

The final sub-question, concerning the extent to which input features support accurate predictions, can be answered in part. Although the current model failed to produce meaningful outputs, it is suspected that lacking expressiveness of tokens was a key reason for this. Thus, this study highlights the importance of preprocessing and selection of input features, especially in designing a robust tokenisation strategy for both, constituents and pseudojets.

5 DISCUSSION

The primary objective of this thesis was to assess the practicality of predicting jet evolution using modern machine learning methods. Emphasis was placed on leveraging the capabilities of OmniJet-α, a foundation model that was motivated by recent, transformative advances in ML.

Foundation models represent an emerging and developing field. The creation of capable models pretrained on unlabelled datasets that can be augmented with small amounts of data via fine-tuning and transfer learning is especially valuable in particle physics, where labelled data is often resource-intensive to obtain.

In this work, a way for transforming jet data into suitable training data for OmniJet-α was conceived and developed. The model itself was adapted to the task, and modifications to the data representation, particularly through revised tokenisation, were introduced for that purpose. Although the model was unable to demonstrate reliable predictions, it serves as a proof of concept for providing a concrete implementation, a realistic strategy, and an architecture that may guide future research efforts.

The lack of predictive accuracy can be in part attributed to differences between features of pseudojets and their constituents. This led to an uneven resolution of token representations, particularly those of pseudojets. This was especially disruptive, as this data likely has a dominant impact on model performance and prevented the model from learning the intended patterns in the data.

The work carried out in this thesis demanded a lot of personal development, as not only was the codebase complex and required a great amount of additional learning, but it was further accompanied by the multitude of tools and skillsets essential in creating and effectively modifying all the different interconnecting parts.

For one, libraries that are mainly used in scientific fields, like FastJet and Awkward, but also other tools and interfaces employed, including the VQ-VAE and

Comet ML, were mostly unfamiliar. Previous experience with virtual Ubuntu environments, .ROOT files, JupyterHub and, to a limited extent, transformer architectures, played a key role for becoming more easily accustomed to the project.

On top of that, it was a challenge to absorb such a large quantity of new information about particle physics in such a short timeframe. Over the course of this research, the continuously increasing level of comprehension of the subject matter became perceptible and was a hugely motivating factor for tackling the various intermediate challenges. In hindsight, a better grasp of the domain, more deliberate data analysis combined with a more structural approach, and a less condensed time schedule would probably have yielded greater results. It became especially apparent, just how important extensive domain knowledge is for making informed decisions in data preparation. Despite that, the many skills that were developed and what was ultimately achieved, exceeded the author's expectations set in the beginning and represent a significant personal milestone.

Finally, to give some ideas for plausible future applications based on the knowledge that was gained, and assuming at least partial success in predicting jet evolution with machine learning, such a model may be applied to enhance and optimise computer simulations. Generating data for rare types of particle decays could be achievable with fine-tuning and would help complement existing datasets with data imbalances. Unusual or anomalous jets in experiments may be more readily identified to support discovering data of high interest.

A generalisable model may further enable the learning transfer to experiments at different energy scales, enabling an alternative source of predictions for outcomes of future experiments. As another actively evolving form of ML, reinforcement learning could pose an interesting approach for training physics foundation models by relying on policy instead of data.

These constitute just a small range of prospects for future research, and while physics foundation models are still in their preliminary stages, they possess undeniable potential to vastly accelerate scientific progress and remove barriers in the exploration of new frontiers of research.

REFERENCES

ALICE Collaboration. 2023. Measurement of the angle between jet axes in Pb—Pb collisions at $\sqrt{S_{NN}} = 5.02$ TeV. arXiv preprint arXiv:2211.08928. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.2303.13347

Antcheva, I., Ballintijn, M., Bellenot, B., Biskup, M., Brun, R., Buncic, N., Canal, P.H., Casadei, D., Couet, O., Fine, V., Franco, L., Ganis, G., Gheata, A., Maline, G. G., Goto, M., Iwaszkiewicz, J., Kreshuk, A., Segura, D. M., Maunder, R., Moneta, L., Naumann, A., Offermann, E., Onuchin, V., Panacek, S., Rademakers, F., Russo, P. & Tadel, M. 2009. ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization. Accessed on 16 May 2025 https://doi.org/10.48550/arXiv.1508.07749

Amram, O., Anzalone, L., Birk, J., Faroughy, D. A., Hallin, A., Kasieczka, G., Krämer, M., Pang, I., Reyes-Gonzales, H. & Shih, D. 2024. Aspen Open Jets: Unlocking LHC Data for Foundation Models in Particle Physics. Accessed on 20 May 2025 https://doi.org/10.48550/arXiv.2412.10504

Arranz, D., Bianchini, S., Di Girolamo, V. & Ravet, J. 2023. Trends in the use of AI in science – A bibliometric analysis, Publications Office of the European Union. European Commission: Directorate-General for Research and Innovation, 3–4. Accessed on 29 May 2025 https://data.europa.eu/doi/10.2777/418191

ATLAS Experiments 2023a. ATLAS searches for new phenomena using unsupervised machine learning for anomaly detection. Accessed on 12 May 2025 https://atlas.cern/Updates/Briefing/Anomaly-Detection

ATLAS Experiments 2023b. Machine learning is revolutionising our understanding of particle “jets”. Accessed on 12 May 2025 https://atlas.cern/Updates/Briefing/Boosting-Jets-AI

ATLAS Open Data 2025. The Standard Model of Particle Physics and Beyond. Accessed on 20 May 2025 https://opendata.atlas.cern/docs/documentation/introduction/SM_and_beyond

BBC. 2022. The Biggest Event in Human History. Accessed on 12 May 2025 https://www.bbc.co.uk/programmes/m001216j

Belis, V., Odagiu, P. & Aarrestad, T. K. 2024. Machine learning for anomaly detection in particle physics. Reviews in Physics, 12. 100091. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.2312.14190)

Benedikt, M., Chance, A., Dalena, B., Denisov, D., Giovannozzi, M., Gutleber, J., Losito, R., Mangano, M., Raubenheimer, T., Riegler, W., Shiltsev, V., Schulte, D., Tommasini, D. & Zimmermann, F. 2022. Future Circular Hadron Collider FCC-hh: Overview and Status. Accessed on 19 May 2025. https://doi.org/10.48550/arXiv.2203.07804

Birk, J., Hallin, A & Kasieczka, G. 2024. OmniJet-α: the first cross-task foundation model for particle physics. Machine Learning: Science and Technology. 5(3):035031. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.2403.05618

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, A., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang , W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K. & Liang, P. 2021. On the Opportunities and Risks of Foundation Models. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.2108.07258

Cacciari, M. & Salam, G. P. 2005. Dispelling the N3 myth for the kt jet-finder. arXiv:hep-ph/0512210. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.hep-ph/0512210

Cacciari, M., Salam, G. P. & Soyez G. 2011. FastJet user manual: (for version 3.0. 2). Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.1111.6097

Catani, S. & Zeppenfeld, D. 2000. Jet Algorithms. The QCD and standard model working group: Summary report, 132-140. Accessed on 12 May 2025 https://inspirehep.net/literature/1747734

CERN 2025. How did we discover the Higgs boson? Accessed on 12 May 2025 https://home.cern/science/physics/higgs-boson/how

Chen, T. S., Chu, J. W. & Yang, J. M. 2025. Expanding Chemical Space: Developing a Compound Generative Pre-trained Transformer for De Novo Drug Design bioRxiv 2025.01.24.634665. Accessed on 17 May 2025 https://doi.org/10.1101/2025.01.24.634665

CMS Collaboration 2025. The two most massive quarks put the spotlight on the Higgs boson. Accessed 29 May 2025 https://cms.cern/news/two-most-massive-quarks-put-spotlight-higgs-boson

Cover, T. & Hart, P. 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory. Vol. 13 No. 1, 21-27. Accessed on 12 May 2025 https://doi.org/10.1109/TIT.1967.1053964

HEP ML Living Review 2025. A Living Review of Machine Learning for Particle Physics. Accessed on 12 May 2025 https://iml-wg.github.io/HEPML-LivingReview/

Huh, M. 2022. Vqtorch: PyTorch Package for Vector Quantization. Accessed on 12 May 2025 https://github.com/minyoungg/vqtorch

IBM 2025. The games that helped AI evolve. Accessed on 12 May 2025 https://www.ibm.com/history/early-games

Irwin, R., Dimitriadis, S., He, J. & Bjerrum, E. J. 2022. Chemformer: a pre-trained transformer for computational chemistry. Mach. Learn.: Sci. Technol. 3 015022. Accessed on 17 May 2025 https://doi.org/10.1088/2632-2153/ac3ffb

Kosinski, M. 2024. What is black box artificial intelligence (AI)? IBM 29 October 2024. Accessed on 16 May 2025 https://www.ibm.com/think/topics/black-box-ai

Larkoski, A. J. & Thaler, J. 2014. Aspects of jets at 100 TeV. Physical Review D, American Physical Society, Vol. 90. No. 3, 034010. Accessed on 12 May 2025 https://doi.org/10.1103/PhysRevD.90.034010

Law, M. 2023. Big data market be worth $400bn by 2030, driven by AI and ML. Technology Magazine 22 August 2023. Accessed on 12 May 2025 https://technologymagazine.com/articles/big-data-market-be-worth-400bn-by-2030-driven-by-ai-and-ml

Lalor, J. P., Wu, H. & Yu, H. 2017. Improving Machine Learning Ability with Fine-Tuning. Accessed on 17 May 2025 https://doi.org/10.48550/arXiv.1702.08563

Lorica, B. 2025. Foundation Models: What's Next for 2025 and Beyond. Gradient Flow 5 February 2025. Accessed on 12 May 2025 https://gradientflow.com/foundation-models-whats-next-for-2025-and-beyond/

Marr, B. 2023. A Short History Of ChatGPT: How We Got To Where We Are Today. Forbes 19 May 2023. Accessed on 20 May 2025 https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/

Mentzer, F., Minnen, D., Agustsson, E.& Tschannen, M. 2023. Finite Scalar Quantization: VQ-VAE Made Simple. Accessed on 19 May 2025 https://doi.org/10.48550/arXiv.2309.15505

O'Donnellan, R. 2024. Machine learning by the numbers: Its impact on business. Intuition 21 September 2024. Accessed on 12 May 2025 https://www.intuition.com/machine-learnings-business-impact-by-the-numbers/

Pata, J., Wulff, E., Mokhtar, F., Southwick, D., Zhang, M., Girone, M. & Duarte J. 2024. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Communication Physics, 7, 124. Accessed on 12 May 2025 https://doi.org/10.1038/s42005-024-01599-5

Qu, H. & Gouskos, L. 2019. Jet tagging via particle clouds (V1). Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.1902.08570

Qu, H., Li, C. & Qian, S. 2022. JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics (1.0.0) [Data set]. Zenodo. Accessed on 16 May 2025 https://doi.org/10.5281/zenodo.6619768

Qu, H., Li, C. & Qian, S. 2022. Particle Transformer for Jet Tagging. Accessed on 12 May 2025 https://doi.org/10.48550/arXiv.2202.03772

Ratnikov, F. 2020. Generative Adversarial Networks for LHCb Fast Simulation. EDP Sciences: EPJ Web of Conferences, Vol. 245, 02026. Accessed 12 May 2025 https://doi.org/10.48550/arXiv.2003.09762

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Accessed on 19 May 2025. https://doi.org/10.1037/h0042519

Sauerburger, F. 2020. Awkward arrays and numba. Frank Sauerburger 11 March 2020. Accessed on 12 May 2025 https://frank.sauerburger.io/2020/03/11/awkward-and-numba.html

Schmidt, P. W. 2023. Collision (physics). AccessScience. Accessed on 20 May 2025 https://doi.org/10.1036/1097-8542.149000

Van den Oord, A., Vinyals, O. & Kavukcuoglu, K. 2017. Neural Discrete Representation Learning. 2 November 2017. Accessed on 17 May 2025 https://doi.org/10.48550/arXiv.1711.00937

Verma, J. 2022. Breadth-First Search (BFS) and Depth-First Search (DFS) for Binary Trees in Java. DigitalOcean 4 August 2022. Accessed on 20 May 2025 https://www.digitalocean.com/community/tutorials/breadth-first-search-depth-first-search-bfs-dfs

Zhao, Z. Y., Zhu, Z., Li, G., Wang, W. & Wang, B. 2023. Generative Pretraining at Scale: Transformer-Based Encoding of Transactional Behavior for Fraud Detection Accessed on 19 May 2025. https://doi.org/10.48550/arXiv.2312.14406

Zheng, H., Shen, L., Tang, A., Luo, Y., Hu, H., Du, B., Wen, Y. & Tao D. 2025. Learning from models beyond fine-tuning. Nat Mach Intell, 7, 6–17. Accessed on 12 May 2025 https://doi.org/10.1038/s42256-024-00961-0