Chapter 1

Foundation Models in High Energy Physics

One of the main driving forces behind the successes and widespread adoption of machine learning in NLP and CV is the utilization of large pre-trained models, often referred to as foundation models (FMs) [1]. Especially for language tasks, most modern applications are built on top of publicly available *pre-trained* FMs, such as BERT [2] or GPT [3, 4], which already possess a basic understanding of language. FMs are large models typically trained on a large corpus of domain-related data to learn expressive but generic representations of the subject. This pre-training step is designed to be non-task-specific and is usually self-supervised (SSL), after which FMs may be *fine-tuned* on specific *downstream* tasks.

This system of SSL pre-training followed by supervised fine-tuning is prevalent in the many FMs in NLP [2–6] and CV [7–12]. Current research in these fields is focused on developing new SSL strategies and improving pre-training efficiency. Pre-training tasks in NLP typically involve predicting the next token in a sequence, as with GPT, or randomly masked tokens, as with BERT. Common downstream tasks include sentiment analysis and machine translation. The FM in these tasks is also referred to as the *backbone* as it contains most of the parameters, though additional learnable layers may still be required.

As discussed in this thesis, the field of HEP has increasingly integrated ML methods to tackle diverse challenges, including event reconstruction, anomaly detection, and data generation. These developments have largely mirrored the trends of the wider ML community: model sizes across all fields have grown exponentially, and transformer-based neural networks have become the dominant architecture for many tasks. However, HEP has yet to fully embrace the FM paradigm, which could provide a powerful tool for the field, though studies are now emerging that explore this potential [13–20].

This section describes work done on developing SSL strategies for particles physics jets expressed as sets of constituents, as used in **????**. Our initial paper on the topic, Golling et al. [15], introduced the masked particle modelling (MPM) method and demonstrated that it could be a viable pre-training strategy for jet data. Our follow-up study, Leigh et al. [16], further refined this method into MPMv2, demonstrating its effectiveness in various downstream tasks. Code for both studies is available is publicly available [21, 22] and additional results for this section can be found in Appendix A.

1.1 Masked Particle Modelling

Developing a good SSL strategy for jet data would be particularly advantageous as experimental data is unlabelled. Many ML models are trained in a supervised manner on simulated datasets, in order to have access to truth labels. However, running high-quality physics simulations is a time-consuming process, and they do perfectly model real physics. This casues a domain shift between the synthetic samples the model was trained on and the real data to which it is then applied. A good SSL strategy would allow training on the billions of real collision events recorded by the LHC experiments over the past few years.

The most popular SSL strategies can be broadly grouped into invariance-based methods and generative methods [11]. Invariance-based methods train an encoder to produce similar embeddings given multiple augmented views of the same sample. These methods typically require hand-crafted and meaningful augmentations that are specific to the data modality. In image data, these might correspond to random scaling, cropping, masking, or color jittering; augmentations we know *a priori* that do not change the content of the sample. For jet data it is unclear what augmentations would be optimal. JetCLR [13] used approximate but physically inspired augmentations, such as rotations of the constituents about the jet axis and the smearing of soft constituents to estimate soft gluon radiation. This works well on simplified datasets, where each jet constituent is expressed as a massless kinematic object, but it is unclear how well it would work on sets of reconstructed tracks as used in **??**. R3SL [17] is a framework where the multiple views are generated by reusing the same synthetic underlying event duplicated at some point in the simulation pipeline. The simulation is then completed with different seeds or settings. While this method results in more physically accurate augmentations, it explicitly requires synthetic data.

Current theory on biological systems suggests that representation learning is driven by the need for an internal model that can predict sensory input changes [23]. This has led to the development of many models which fall into the second category of SSL, known as generative methods. These methods involve corrupting or removing input parts of the input sample and training the model to predict the missing content. Such methods can be framed within the context of denoising autoencoders (DAE) [24]. In a DAE, a lossy augmentation is first applied to the inputs, which are then projected via an encoder into a latent space. Subsequently, a decoder maps this latent representation to the original, uncorrupted signal. Only the encoder is retained post-training for further applications, while the decoder is usually discarded.

For a DAE, one hyperparameter is the level and type of corruption applied to the input. Masking or simply removing a fraction of elements from the input set is a simple, efficient, and effective corruption technique that underpins several notable models in NLP and CV. Unlike designing specific augmentations, it requires minimal prior knowledge of the domain and applies to many fields. This is the approach taken by our method, MPM.

1.1.1 Overview of Methods

The objective of the MPM is to develop a model capable of inferring the attributes of the original particles within the masked set, utilizing information from all other particles in the jet. Given that particles form unordered sets, unlike the sequential nature of sentences, we have created a masked prediction scheme suited for unordered set-based data.

A notable challenge arises from the continuous nature of particle features, which contrasts with the discrete dictionaries in language models but is similar to the challenges encountered when masking image patches in computer vision. We addressed this challenge by developing an additional model to discretize or *tokenize* our data using methods similar to those used in BEiT [10].

Masked Particle Modelling

The representation of a jet as set of particles, each characterized by continuous features, fits well within the masked training paradigm similar to masked language models such as BERT [2]. In this analogy, each particle in the jet is akin to a token in a text block, despite the unordered nature of particles versus the sequential structure of text.

Jets with N constituent particles are described as a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, where each $\mathbf{x}_i$ represents the attributes of a single particle. In MPM, M particles out of the N that constitute the jet are selected using some predefined masking strategy. All selected particles are replaced with a learnable vector called the mask token $\mathbf{m}$. This new set of surviving particles and mask tokens $\mathcal{S} = \{x_i\}_{i=1}^{M} \cup \{\mathbf{m}\}_1^{N-M}$ is fed into a model to recover the original set. An encoder is used to map the set $\mathcal{S}$ into a latent space, and a prediction head is used to predict the original attributes of the masked particles. We use a per-particle loss which is only calculated over the masked elements of the set. A rough outline of the intended pre-training process is shown in Figure 1.1.

Unlike language models, which operate on a finite and discrete vocabulary, many features describing particles are continuous, such as momentum. This distinction influences two key aspects of the model. Firstly, in terms of model input, continuous features can be used directly, as seen in CV, or they can be discretized by binning each feature dimension and using the bin index as an input token, as shown in Finke et al. [25]. Secondly, as the model targets, predicting missing particle features can be done by directly predicting continuous features or, if discretized, using discrete tokens as prediction targets. This choice impacts the pre-training loss function. Predicting continuous features typically involves a regression-type loss while predicting discrete indices frames this task as a classification problem, enabling the model to learn a categorical posterior distribution over indices. Modelling the full posterior distribution can prevent the model from focusing on irrelevant details. In our analysis, we compare discrete and continuous representations for both input and output.

Particles forming a jet are permutation invariant, making it natural to use a permutation equivariant backbone model. However, using the same learnable vector $\mathbf{m}$ for all particles in the masked set makes the model's output identical at every masked position, resulting in degeneracy that can only be eliminated by inducing ordering or adding attributed inter-particle connections. This redundancy prevents zero-error prediction unless all masked particle values are identical. Despite this, MPM remains valuable as learning the marginal density over the masked set is challenging.

Two strategies address this degeneracy. One is to define an ordering of the particles based and use absolute positional encoding as defined in **??**. This may compromise the permutation invariance of the backbone, potentially affecting the pre-training.

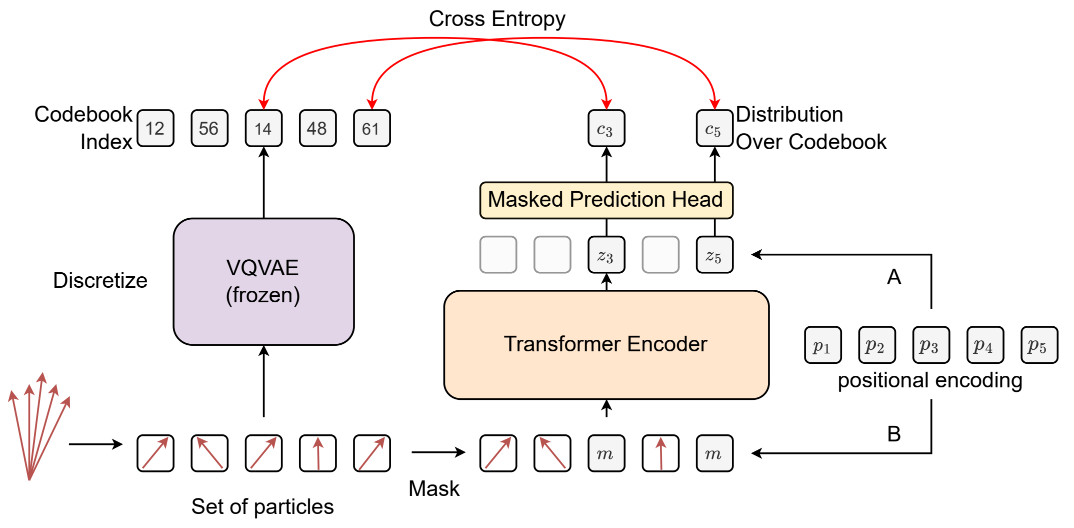


FIGURE 1.1: The proposed model and training scheme for MPM. Jets are represented as a set of particles, and some are selected to be replaced by masked tokens and passed through a transformer encoder. Training aims to predict the discrete token identity, defined by the output of a pre-trained VQVAE, of the masked particles. Positional encoding can be used in the prediction head (A), or in the backbone inputs (B).

This ordering can be based on any particle attribute; transverse momentum is a common choice. However, since the task is to recover the particle's dropped features, including the momentum, the model could achieve this by simply examining the particle's location within the sequence. This is not an issue in images or text, where the position itself defined the dropped attribute.

Another strategy is to only provide this positional encoding at the input of the masked prediction head. This breaks the degeneracy in the loss while preserving the backbone's permutation invariance for downstream tasks. The masked prediction head is a simple MLP with no interactions between elements in the set, so basic comparisons between order are not possible. We compare the approaches of allowing this degeneracy, adding positional encoding to the backbone, and adding positional encoding to the prediction head.

Making Tokens From Particles

An effective tokenization scheme is essential to evaluating the impact of using discretized particles for model input or recovery. The most straightforward approach involves binning each feature into a finite set of ranges [25]. We find that discretization via the K-Means algorithm [26], whereby fitted clusters are assigned indices used as targets for the backbone, proved to be more performant than binning. Both of these methods however lack contextual information regarding particle features relative to the rest of the jet.

For a context-dependent tokenization scheme, we employed a Vector Quantized Variational Auto-Encoder (VQVAE) [27]. A VQVAE is defined with a set number of codebook vectors. An encoder maps inputs to latent vectors, which are projected onto the nearest codebook element and then decoded. We use transformer as the VQVAE encoder which maps each particle to a single codebook element, conditioned on all other particles in the jet.

Fine-Tuning

Evaluating the utility of a pre-trained backbone requires suitable downstream tasks for benchmarking. This work focuses on jet classification in various contexts, using a weighted pooling on the backbone output followed by a linear layer to match the number of jet classes. Three strategies are examined for training a model for a downstream task:

- **Fixed backbone**: The pre-trained backbone is frozen during the supervised task, with only the linear classification head updated. This tests the linear separability of fixed representations.
- **Fine-tuned**: Both the linear classification head and backbone model are fit to the downstream task, evaluating the utility of pre-training for initializing the model's function.
- **From scratch**: The backbone model is reinitialized with new weights and fit directly on the downstream task, providing a performance benchmark without pre-training influence.

1.1.2 Data

MPM does not require labels and can be directly applied to experimental data. However, due to the lack of readily accessible large open datasets of real jets (at the time of writing), we use MC simulations to develop the framework. We ignore all truth labels during pre-training but use them to evaluate the model's performance on downstream tasks.

We use the publicly available JetClass dataset [28] with 125 million jets across 10 classes (100M for training, 5M for validation, 20M for testing). Jets in this dataset are fat jets, **??**, comprising multiple quark or gluon jets overlapping due to highly boosted decays. Each class represents jets captured when simulating specific decay chains, such as $H \to b\bar{b}$, and $t \to b\ell\nu$. Events in JetClass are generated using PYTHIA [29], with top, W, Z, and Higgs decays modelled with MADGRAPH5 [30]. Detector simulation is performed using DELPHES [31] and is configured to match the CMS experiment. Jets are reconstructed using calorimeter energy deposits with the anti-k_T algorithm [32] with the radius parameter set to $R = 0.8$. Jets are selected to have roughly the same transverse momentum p_T distribution across all classes, ranging from 500 to 1000 GeV. On average, each jet contains 50 particles, with a maximum of 120.

For our experiments, each particle is virtually massless and described by its transverse momentum p_T, pseudo-rapidity to jet axis $\Delta\eta$, and azimuthal angle to jet axis $\Delta\phi$.

1.1.3 Model Architecture

The same transformer encoder architecture is used as the backbone in all experiments. Eight TE Blocks based on the Normformer architecture [33] are used with a model dimension 1024, comprising a total of 40 million parameters. Input particles are linearly embedded into this space.

After passing through the backbone, the output nodes are used in dedicated layers called *heads*. The pre-training head consists of a single MLP applied per node, followed by a softmax for classification or a linear layer for regression. The fine-tuning classification head is a weighted average over all output dimensions, followed by another linear layer and softmax.

All models are trained with the AdamW [34] optimizer using a learning rate of 10^{-4} and a weight decay of 0.01. Pre-training was performed over five epochs on the full JetClass dataset. For supervised fine-tuning, all models were trained for up to 50 epochs with early-stopping enabled. In both settings the learning rate was ramped up linearly from zero over 1000 steps and the norm of the gradients was clipped to 1.0.

In the tokenized settings, a codebook of size 512 was used for the VQVAE and 512 centroids for the K-Means. For the masking strategy we set the mask rate $D_f = {}^{M}/_{N}$ to 0.3, meaning 30% randomly selected particles in the jet were masked.

VQVAE Training

A VQVAE consists of an encoder F and decoder G, a quantization layer h, and a codebook $C = \{c_i\}_{i=1}^{m}$ for vectors $c_i \in \mathbb{R}^n$ where n is the latent space dimension.

The quantization layer $h : \mathbb{R}^n \times C \to c \in C$ acts on a latent vector by assigning it to the closest codebook element. We use the euclidean norm as the distance measure. The codebook C will always be omitted from the arguments of the function h in the following. The output of a VQVAE is defined as,

$$\begin{aligned} \hat{x} &= G(h(F(\mathbf{x}))) && (1.1)\\ &= G(h(\mathbf{z}_e)) && (1.2)\\ &= G(\mathbf{z}_q), && (1.3)\end{aligned}$$

for a given input $\mathbf{x}$. The objective of a VQVAE is typically to minimize the reconstruction measure in the input space. Gradients for this loss however can not be backpropagated to the encoder as the operation h is not differentiable and have to be estimated using the straight-through estimator [35]. An extra commitment loss term is included to create an attractive force between the encodings $(z_e = F(x))$ and their corresponding codebook vectors $(z_q = h(z_e))$,

$$\mathcal{L}_{cmt} = (1-\beta)d(z_e, \mathrm{sg}(z_q)) + \beta d(\mathrm{sg}(z_e), z_q), \tag{1.4}$$

where sg is the stop gradient operator and d is some distance measure which we take to be the euclidean norm. We add this to the reconstruction loss with a relative weight of 10, and use a β of 0.9.

Training a VQVAE [27] can be challenging. One of the main issues is the collapse of the codebook, where all latent vectors are assigned to a single codebook element. We employ several best practices to mitigate this issue [36] via the library vqorch [37]. We use a shared parameterization for the codebook elements, update the commitment loss only every four steps, a synchronized update rule with $\nu = 2$, and a latent dimension of $n = 16$. The codebook elements are initialized using the K-Means clustering algorithm on the latent space of a randomly initialized model. Specific codebook elements which have not been used for 100 consecutive steps are reinitialised. Post-training, the codebook achieves approximately 80% utilization.

The input nodes are each quantized and decoded separately, with the distance calculated on a per-node basis. Therefore, a jet with N constituent particles is assigned to N codebook elements, each of dimension 16. Each of these codebook elements are decoded to N vectors of the same dimension as the input nodes. Using a transformer for the encoder and decoder networks ensures that the model is encoded and decoded conditional on all particles in the jet. The encoder and decoder follow a similar structure to the backbone but with only four layers and a model dimension of 256, and linear layers resize the inputs and outputs.

1.1.4 Experiments

The first set of experiments are conducted to test the model design strategies. These experiments aim to understand the impact of tokenization on the inputs and targets features, examine the quality of tokens learned by the VQVAE, and test the effect of the positional encoding. These are followed by experiments to evaluate the performance of the models on downstream classification tasks. The tasks include:

- In-distribution prediction, where predictions are made using the same classes seen in pre-training
- Out-of-distribution (OOD) prediction, where predictions are made using the classes not seen in pre-training
- Weakly supervised prediction, where a CWoLa style classifier is trained using noisy labels as described in **??**

All experiments are repeated five times with different seeds, and the mean and standard deviation of the results are reported.

Token Creation with a VQVAE

To ensure that latent space codes sufficiently capture essential information about particles in a jet, we examine the quality of decoded jet properties after encoding and quantization in the VQVAE. Figure 1.2 shows that the distributions of decoded jet transverse momentum and mass align well with the input distributions. Despite the apparent overlap in goals between VQVAE and MPM models, VQVAEs primarily focus on data compression, while masked particle models infer missing information to understand a jet and thus face a more challenging task.

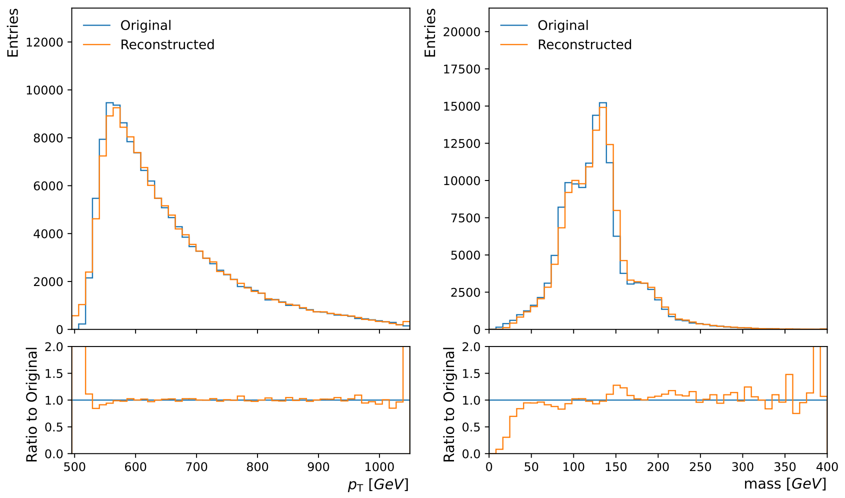


FIGURE 1.2: The reconstruction of the jet mass and transverse momentum using the VQVAE.

1.1.5 Discretization and Positional Encoding

This section examines the pre-training configuration, focusing on the impact of positional encoding, input quantization, and target quantization.

The ordering settings considered include no ordering within the model, ordering at the backbone input, and ordering at the pre-training prediction head input. Ordering is defined by decreasing particle transverse momentum, with the index in the sequence embedded using learned absolute positional encoding (APE). We also test the impact of replacing the continuous features to the backbone with discrete tokens defined by either the VQVAE or K-Means clustering. This is used for the input features, the target features, or both. When a tokenized representation is used for the target, the model is trained using cross-entropy over the codebook indices. Otherwise, we use regression with mean squared error.

After pre-training we fix the backbone and train a linear classifier on the JetClass dataset. The accuracies are quoted in Table 1.1. Ordering solely in the model head demonstrates a clear advantage over ordering the backbone inputs or not ordering at all. Input quantization negatively impacts model performance due to the information loss. Using tokens from either the VQVAE or K-Means as classification targets is substantially better than regressing particle features as a prediction task, since regression offers only a point prediction of average particle properties under the posterior, which has limited predictive power for multi-modal posteriors. Classification loss is more flexible, allowing for the prediction of the full posterior over tokens. VQVAE tokens outperform K-Means tokens for classification, likely due to the VQVAE's complexity in capturing subtle details. However, K-Means tokens perform reasonably well and may be a useful avenue for additional tests due to the ease of implementation compared to the VQVAE.

Following these results, all subsequent tests use a backbone with original continuous inputs, ordering in the prediction head for pre-training, and a classification loss using VQVAE tokens as targets in the pre-training.

TABLE 1.1: Linear probe accuracy on the ten classes of JetClass using backbones with different pre-training strategies. The accuracy is averaged over five different runs with errors on the order of 0.01%.

Inputs	Targets	APE	Accuracy
raw	**VQVAE**	**head**	**56.8**%
original	K-Means	head	56.2%
original	VQVAE	none	54.1%
original	VQVAE	backbone	53.4%
VQVAE	VQVAE	head	51.1%
VQVAE	K-Means	head	49.3%
original	original	head	48.9%
original	original	backbone	46.3%

In-Distribution Classification

The utility of the backbone model for downstream tasks is first tested on in-context data by examining the accuracy of ten-class classification on a JetClass test sample. We perform fine-tuning on the same dataset as with pre-training and then measure

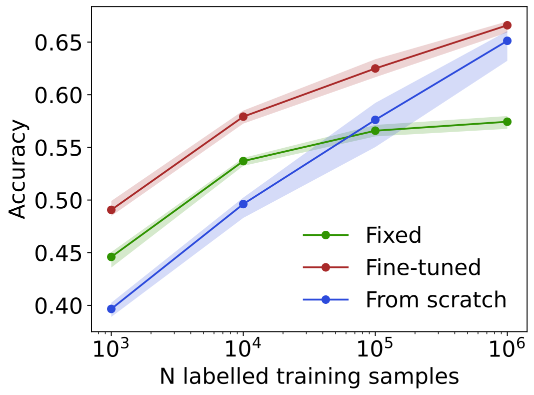


FIGURE 1.3: Accuracy of training strategies as a function of the number of labelled samples. Uncertainty bands depict variability in five runs with different seeds. “Fixed” denotes models with frozen pre-trained weights; “Fine-tuned” denotes models with updated weights; “From scratch” denotes models with randomly initialized weights.

the results with respect to the holdout test set. We modify the number of labelled samples used for fine-tuning to assess the data efficiency.

Figure 1.3 shows that with labelled dataset sizes under 10k jets, pre-training offers a significant performance advantage over the supervised model trained from scratch. This demonstrates the utility of the representation learned during pre-training for downstream tasks. With a sufficiently large labelled dataset, the supervised model surpasses the fixed backbone model and matches the performance of the fine-tuned backbone model. This is expected, as supervised learning on a large enough labelled dataset optimizes the model without pre-training. Overall, pre-training provides a strong set of initial weights for fine-tuning on downstream tasks.

1.1.6 Out-of-Distribution Classification

To test whether the pre-trained model learns features useful for tasks on data beyond those seen during pre-training, we perform an out-of-context test. We pre-train on six of the classes in the dataset, those pertaining to Higgs, light quarks, and gluons, and fine-tune on four new classes containing top quarks and W, Z decays.

The results in Fig. 1.4 show that both pre-trained models, fixed and fine-tuned, outperform training from scratch. This demonstrates that the representations learned during pre-training are useful for out-of-context classes. With sufficient labelled data, the fully supervised model surpases both pre-trained models.

1.1.7 Weakly supervised classification

Fine-tuning tasks are typically supervised and require labels. Physics knowledge can enrich data in certain classes, providing training data with noisy labels. These noisy labels can replace or complement fine-tuning with simulated data, leveraging pure labels in simulation and noisy labels in data. This approach is useful for data-driven weakly supervised search strategies which use these noisy labels to construct a CWoLa classifier as described in **??**. However, CWoLa classification struggles with high-dimensional data when using large models like transformers on low-level set data. Being able to improve their sensitivity via pre-training will make these methods viable for new physics searches.

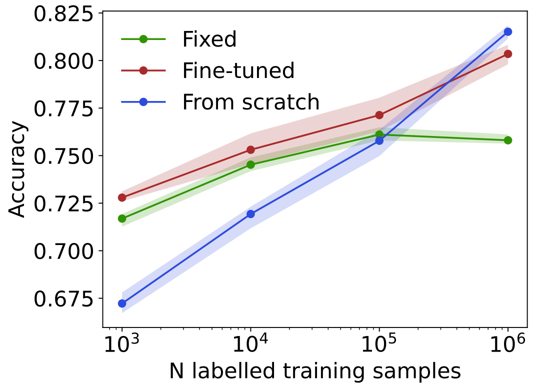


FIGURE 1.4: Accuracy of training strategies as a function of the number of labelled samples as measured on classes not seen during pre-training. Uncertainty bands depict variability in five runs with different seeds. "Fixed" denotes models with frozen pre-trained weights; "Fine-tuned" denotes models with updated weights; "From scratch" denotes models with randomly initialized weights.

To emulate the CWoLa setting, we take two samples of one class (QCD jets), each with one million events, and add N samples of another class (top-quark initiated jets, or top jets) to one dataset. A supervised classifier is then trained to discriminate between these datasets and evaluated on separating pure datasets of each class (QCD vs top jets). In Figure 1.5, significance improvement (SIC) is shown as the ratio of significance before and after applying a 0.5 threshold on the classifier output. Significance is defined as the number of signal class events divided by the square root of the number of background class events passing the threshold. The pre-trained backbone significantly improves the model's performance, even when only the linear head is fine-tuned.

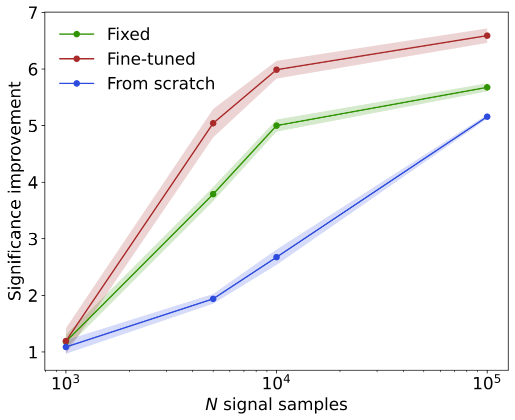


FIGURE 1.5: Models trained with weak supervision to classify datasets of different label proportions. A dataset of one million QCD jets is compared to a dataset with one million QCD jets plus N top jets. Uncertainty bands depict variability in five runs with different seeds. "Fixed" denotes models with frozen pre-trained weights; "Fine-tuned" denotes models with updated weights; "From scratch" denotes models with randomly initialized weights.

1.1.8 Conclusion

We propose the masked particle modelling strategy for pre-training models on unordered sets of inputs, demonstrating its utility in the context of classifying jets. This approach adapts masking strategies from NLP and CV to continuous, unordered particle features. Pre-training with masked particle modelling allows fine-tuned models to perform well on downstream tasks, even with small datasets. Initial results are promising, indicating that larger pre-training datasets and models could address domain adaptation challenges. However, the model trained from scratch can surpass the pre-trained model with sufficient labelled data, indicating an inefficiency in the fine-tuning process.

1.2 Improved Masked Particle Modelling

In the previous section we found that a VQVAE leads to a more performant FM than direct regression. We argued that this was primarily due to two reasons:

- The VQVAE latent space is semantically rich, containing high-level abstractions; giving the MPM encoder a more informative target to learn from (this is also the justification used in BEiT [10]).
- By changing from a regression to a classification task, the backbone is taught the full conditional posterior distribution rather than just seeking the mean.

However, producing the VQVAE requires an additional training step in the pipeline. VQVAEs are notoriously unstable and hard to train. Furthermore, the aforementioned quantization leads to a loss of information.

In this follow-up study we make the following advancements:

- We propose an improved MPM training paradigm, named MPMv2, by enhancing model architecture and addressing existing inefficiencies. We also expand the particle attributes to provide a more detailed representation.
- We provide a detailed study of alternative reconstruction tasks for MPMv2 pre-training, ones that replace the costly VQVAE-derived targets.
- We provide a new test bed for pre-trained models that include a wider set of downstream tasks commonly encountered in jet physics.

From henceforth, we refer to the original MPM method as MPMv1.

1.2.1 Data

We reuse the JetClass dataset [28] but also introduce a new dataset, which we label BTag [38]. The BTag [38] dataset contains 3 million jets differentiated by their flavour: light, charm, or bottom.

Events are generated using PYTHIA [29], which also models parton showering and hadronization, and detector response is simulated using DELPHES [31]. The BTag dataset is different from JetClass in several ways. They are not large-radius or fat jets but are reconstructed using the anti-k_T algorithm with a radius parameter of $R = 0.4$. The jets are only required to have $p_{\mathrm{T}} \geq 20$ GeV, and detector simulation is configured to match the ATLAS experiment instead of CMS. The final significant difference is that BTag only stores charged particles; thus, they have a much lower cardinality, with a maximum of 15 constituents per jet.

We utilize JetClass exclusively for model pre-training, followed by fine-tuning and evaluation on both datasets. The discrepancies between these datasets reflect the realistic variations in particle physics jet definitions across different experimental settings. Targeted kinematic ranges, reconstruction parameters (e.g., anti-kt radius), and object selection vary significantly based on specific physics analyses and are meticulously tuned by experts. These differences provide an opportunity to assess the backbone's generalizability to new downstream tasks and an OOD dataset that is further removed than the class separation we saw in MPMv1.

We also expand the input features to commonly reconstructed attributes beyond p_T, $\Delta\eta$, and $\Delta\phi$. Charged constituents leave tracks in the detector, so we include the lifetime signed longitudinal and transverse impact parameters (d_0, z_0) as well as their reconstruction uncertainties ($\sigma(d_0)$, $\sigma(z_0)$), matching the features used in **??** Neutral particles do not leave tracks, so these features are zero-padded. These 7 variables form the continuous features of the particle, $\mathbf{x}^{c}$. Also included is the particle identity (ID) $\mathbf{x}^{id}$, a one-hot encoded vector that categorizes both the particle type and charge into 8 independent classes. Each particle is therefore represented by a vector of 8 features, 7 continuous and one categorical, $\mathcal{X} = \{\mathbf{x}_i = (\mathbf{x}_i^{c}, \mathbf{x}_i^{id})\}_{i=1}^{N}$, as shown in Table 1.2. The distributions for these features in the two datasets are shown in Figure 1.6.

TABLE 1.2: The features used to describe each jet constituent.

Continuous features $\mathbf{x}^{c}$	
transverse momentum	p_T
pseudorapidity to jet axis	$\Delta\eta$
azimuthal angle to jet axis	$\Delta\phi$
transverse impact parameter	d_0
longitudinal impact parameter	z_0
uncertainty on d_0	$\sigma(d_0)$
uncertainty on z_0	$\sigma(Z_0)$
Particle type $\mathbf{x}^{id}$	
photon	0
negative hadron	1
neutral hadron	2
positive hadron	3
electron	4
positron	5
muon	6
antimuon	7

1.2.2 Overview of Methods

In framing MPMv1 as a DAE, the input jet $\mathcal{X}$, its latent projection $\mathcal{Z}$, and the decoder output $\mathcal{D}$ are all sets; thus, all mappings between them must be permutation equivariant. A transformer acts as the encoder, and a multi-layer perceptron (MLP) acts as the decoder, applied separately per set element. A consequence of having no PE is that the encoder's outputs corresponding to masked inputs are duplicates,

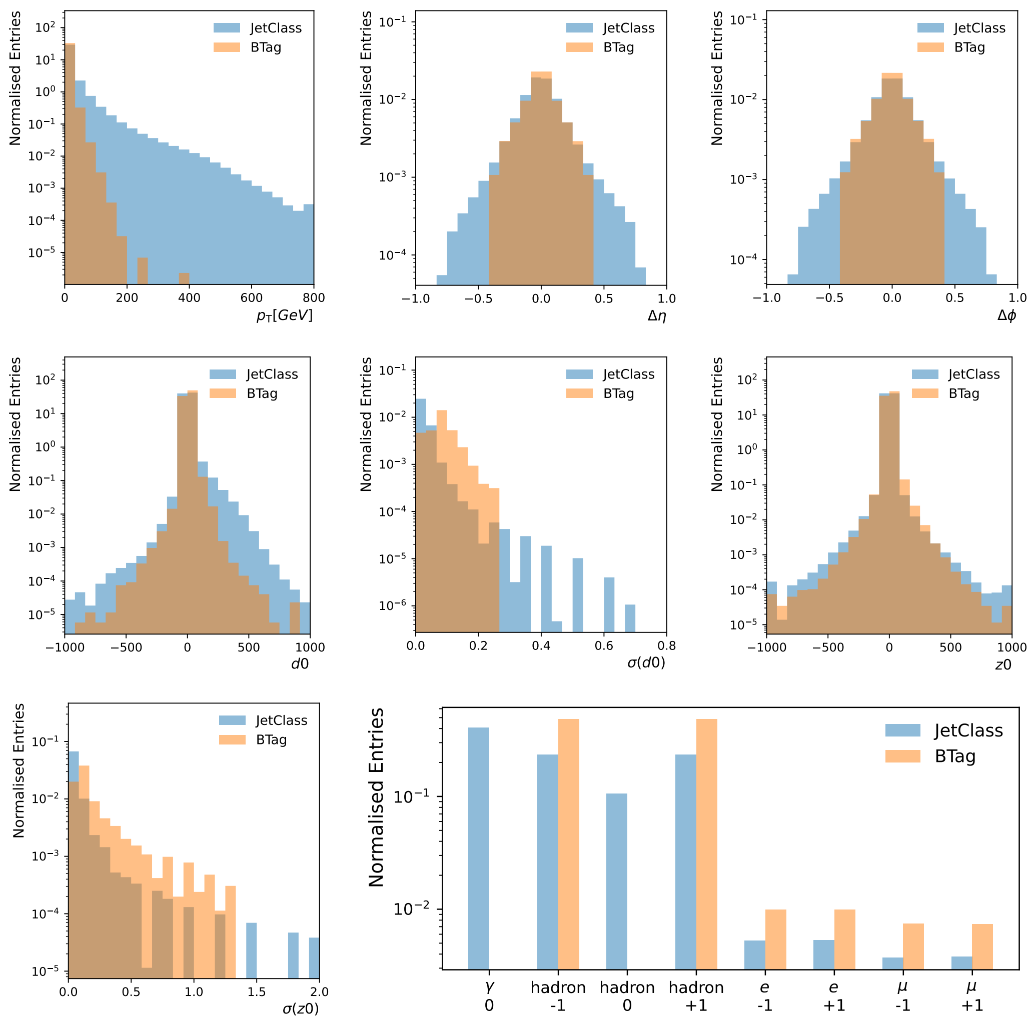


FIGURE 1.6: The distributions of the particle features for the two datasets. The final plot shows the distributions of the particle types $\mathbf{x}^{\text{id}}$ for the two datasets.

so we are forced to add APE to $\mathcal{Z}$ to break this degeneracy for reconstruction while keeping the encoder equivariant. Each element in $\mathcal{D}$ is then used in a tokenized reconstruction task, where it is compared to the corresponding element of the same jet passed through the encoder of a VQVAE.

For MPMv2, we propose removing the masked tokens from the input set, reintroducing them only during decoding, resulting in $\mathcal{Z}$ having a lower cardinality than both $\mathcal{X}$ and $\mathcal{D}$. The masked tokens are identical in each sublayer of the decoder, effectively adding nothing to the message-passing operations. This shifts from a model akin to BERT [2] to one similar to MAE [8]. As such, we also experiment with expanding the decoder to a transformer, designed similarly to the backbone, albeit much smaller.

With a message-passing decoder, APE in the latent space provides too much information, trivializing the reconstruction task. We only provide APE between the masked elements, not the full jet. This can be parameterized by a unique mask token based on the p_T order of the dropped constituents with respect to each other only.

The loss function is derived by comparing $\mathcal{X}$ and $\mathcal{D}$ across various *reconstruction*

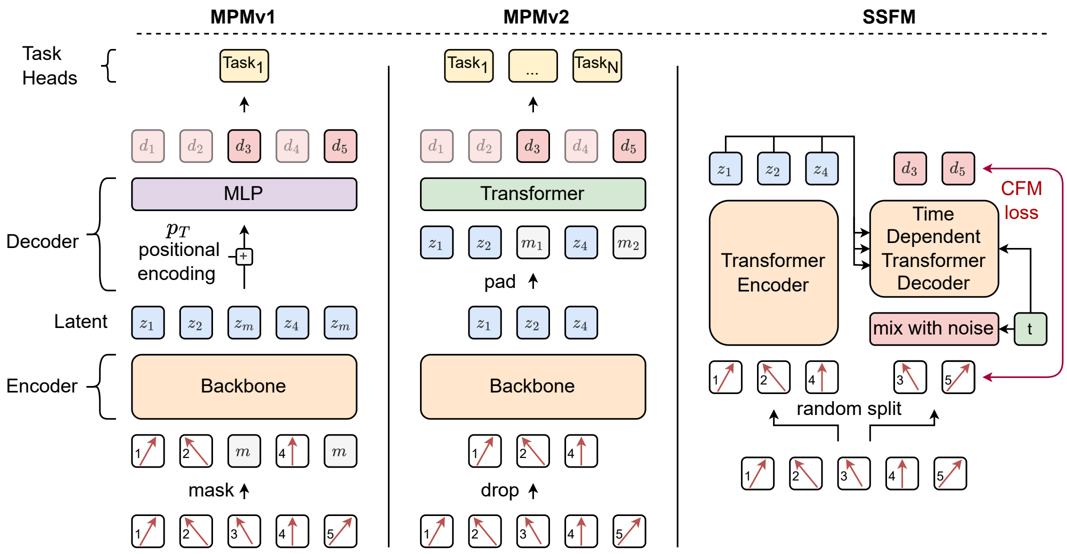


FIGURE 1.7: (left) The original MPMv1 encoder-decoder setup compared to the new MPMv2 model (middle). The new model includes multiple reconstruction tasks, swaps the MLP decoder for a transformer, and only encodes the reduced set. (right) The Set-to-set Flow-Matching which jointly generates the dropped elements of the set using the surviving elements.

tasks. The key differences between MPMv2 and a direct MAE re-implementation include using multiple reconstruction tasks for different feature groups and positional encoding only for the dropped set.

1.2.3 Reconstruction Tasks

MPMv1 only uses a single VQVAE-derived reconstruction task, whereas MPMv2 combines multiple tasks to recover the continuous and categorical features separately. Each task adds extra learnable layers (task head) and contributes to the total pre-training loss. Each of these losses is applied per particle and only calculated for those which were masked/dropped. We explore one task specifically for the categorical feature ($\mathbf{x}^{\text{id}}$) and five options for the continuous features ($\mathbf{x}^{\text{c}}$) and

- Particle Identification: This is to recover the particle type $\mathbf{x}^{\text{id}}$ of the dropped constituent framed as a standard classification problem, so we use a linear layer for the task head and the cross-entropy loss function.
- Direct Regression: While we previously found direct regression insufficient for pre-training, we believe it is worth revisiting owing to the much more powerful decoder. We use a linear layer and find the best results using the L1-loss to recover the particles' continuous features.
- VQVAE Tokenized Classification: This is the method described in the previous section, whereby a frozen VQVAE is used to construct the target indices. We retrain the VQVAE for the expanded set of continuous.
- K-Means Tokenized Classification: We achieved similar results with the K-Means centroids as with the VQVAE tokens in the previous section. However, we can further optimize pre-training on K-Means targets by increasing the number of centroids to 16384. Like the other tasks, we use a single linear layer to map to this space and cross-entropy loss function.

- Conditional Normalizing Flow (CNF): If the strength of the tokenized form of reconstruction over regression is in learning the full posterior distribution $p(\mathbf{x}_i^c|d_i)$, we can reproduce this using a generative model without needing to reframe the task as classification. The loss is the standard negative log-likelihood one uses for NFs.
- Conditional Flow Matching (CFM): Continuing with the idea of using PGM to conditionally generate the dropped continuous features given the surviving ones, we use a diffusion model via the CFM framework described in **??**. We use the loss from **??**.

1.2.4 Set-to-Set Flow Matching

We investigate the set-to-set flow-matching model (SSFM), shown in Figure 1.7, which uses a time-dependent transformer decoder to generate constituents from latent nodes. The approach is similar to the diffusion-masked autoencoder from CV [39]. During the masking the input set $\mathcal{X}$ is split into a reduced set $\mathcal{S}$ of size M and its complement $\mathcal{T}$. The reduced set passes through the encoder to obtain the latent set $\mathcal{Z}$, which is then used in the decoder's cross-attention layers, which trained as a set-CFM model to generate $\mathcal{T}$.

The training task of denoising prevents any degeneracy, eliminating the need for APE or mask tokens. Furthermore, adjusting the masking rate $D_f = \frac{M}{N}$ controls fraction of the jet which is generated by the diffusion model. When $D_f = 1.0$, the decoder functions as a full generative model. Our pre-training setup facilitates the creation of a backbone for embedding, as with MPM, as well as a decoder which could be used as a generative model, similar to those described in **??**. During training, sampling $D_f \sim \mathcal{U}(0.2, 1.0)$ balances these objectives.

1.2.5 Model Architecture

We propose several changes to the MPMv1 model. We switched from the Normformer architecture to a more standard pre-norm encoder with 8 layers, each with a 512-embedding dimension. We employ eight heads for multi-headed self-attention layers, feedforward network dimension multipliers of $\times 2$, and SwiGLU activations [40]. LayerScale [41] is used for attention and dense residual updates. The decoder used in the MPMv2 models uses the same layer types but is considerably smaller, with only four layers and a model dimension of 256. In several of the experiments we use eight register tokens [42] to store global information in the encoder.

Models are trained with the AdamW optimizer, a maximum learning rate of 1×10^{-3}, and weight decay of 1×10^{-5}. The learning rate increases linearly for the first 50k steps and then decays exponentially with a 100k half-life. Pre-training is conducted on the full JetClass training set with a batch size of 1000.

Each task head introduces new layers. Most of these are simple affine layers to map the latent space to the target space. The exceptions to this are the CNF and CFM tasks. For the flow, we use six coupling layers with rational quadratic spline transformations, each containing a two-layer MLP with 128 hidden units. The transformation is conditioned on the decoder outputs by concatenating them to the input of each MLP. The CFM head uses a three-layer MLP with a hidden dimension of 256. The inputs to this model are the partially noised targets $\mathbf{x}_i^c(1-t) + \mathbf{z}t$, the decoder outputs for the particle d_i, and the diffusion time t.

For the SSFM model, we use a transformer decoder with four layers, eight heads, and a model dimension of 256. The layers have more parameters than the decoder used in MPMv2 due to the additional cross-attention layers. A linear layer maps the latent space to the 256 for these cross-attention operations. We use the method described in Peebles and Xie [43] for time injection.

1.2.6 Experiments

For these experiments we first test the impact of the new model design compared to the original MPMv1. This is then followed by a full comparison of the different reconstruction pre-training strategies on a number of downstream tasks.

For the downstream tasks, we use the JetClass and BTag datasets. Each backbone is pre-trained using one of the continuous feature reconstruction tasks (which it is named after) together with the particle ID task. Pre-training is run for 1M steps, after which specific downstream task layers are appended to the encoder, and the model is fine-tuned. All fine-tuning is run for 200k steps with a batch size of 500, allowing for early-stopping using a validation set. We use a randomly initialized network, trained "From Scratch" as a baseline to highlight the performance provided by pre-training and repeat each experiment 5 times to estimate the run-to-run variance.

Fine-tuning performance is greatly enhanced using a specific scheduler for the backbone learning rate. This scheduler is defined by two parameters: the number of frozen-steps and warmup-steps. At the start of training, the backbone is frozen, which allows the new head to adapt. Then, the backbone learning rate catches up to the head learning rate linearly over the warmup-steps.

Ablation Studies

To evaluate the proposed changes to the model, we use an in-distribution classification test on the JetClass dataset with a frozen backbone. After pre-training for 200k steps, we freeze the encoder and add a classifier-head made from two class-attention layers [41], which is then trained using cross-entropy loss for another 200k steps. The regression, K-Means, and particle ID tasks are investigated, and the results of this ablation study are shown in Table 1.3.

Initially, the original training setup with a masking rate of $D_f = 0.3$ was recreated. We then tested updated transformer layers, added impact parameters to the input features, and included the particle ID feature and associated reconstruction task. All of these changes improved the accuracy of the backbone, from 56.2% to 74.0% when using K-Means for the reconstruction task. Changing the decoder to a transformer significantly increases classification accuracy, narrowing the gap between the K-Means and regression pre-training methods. To verify the impact of the decoder change, we rerun the regression task without the impact parameters or particle ID task and found that it achieved an accuracy of 65.0%, an increase of 9.5%.

One additional advantage of adopting the MAE setup is achieving a 40% reduction in GPU memory usage due to the reduced point cloud size passed to the encoder and the use of non-padded representations of the batch [44]. The total inference time also improves by approximately 15%. Additionally, adding registers into the encoder [42] prevents the transformer from overwriting elements in the set with global information, increasing performance with minimal computational cost.

TABLE 1.3: The effects of the model redesign on the accuracy of a classifier head trained using the encoder outputs. All models except the final iteration are pre-trained using 200k steps, a 2-layer decoder, and $D_f = 30\%$.

	Regression		K-Means	
MPMv1 using $(p_\mathrm{T}, \eta, \phi)$	48.9		56.2	
+ updated transformer layers	55.5	$\uparrow$ 6.6	62.2	$\uparrow$ 6.0
+ impact parameter features	62.2	$\uparrow$ 6.7	70.2	$\uparrow$ 8.0
+ constituent ID feature and ID reconstruction task	63.5	$\uparrow$ 1.3	74.0	$\uparrow$ 3.8
+ transformer as decoder (MAE)	79.2	$\uparrow$ 15.7	81.4	$\uparrow$ 7.4
+ registers	80.4	$\uparrow$ 1.2	83.0	$\uparrow$ 1.6
+ longer train (1M steps) + deeper decoder + $D_f = 40\%$	83.3	$\uparrow$ 2.0	84.0	$\uparrow$ 1.0

Mask rate D_f and decoder depth optimization results are shown in Figure 1.8. They both use the K-Means + ID setup, and the encoder depth sweep uses $D_f = 30\%$ while the D_F sweep uses a depth of 2. The model is relatively robust to the mask rate; with values as high as $D_f = 90\%$, it can still achieve an accuracy of over 80%. The optimal mask rate is determined to be $D_f = 40\%$. Increasing the decoder depth improves performance, but only four layers are tested due to computational constraints.

The final model is trained for 1M steps, with a mask rate of $D_f = 40\%$ and a decoder depth of four.

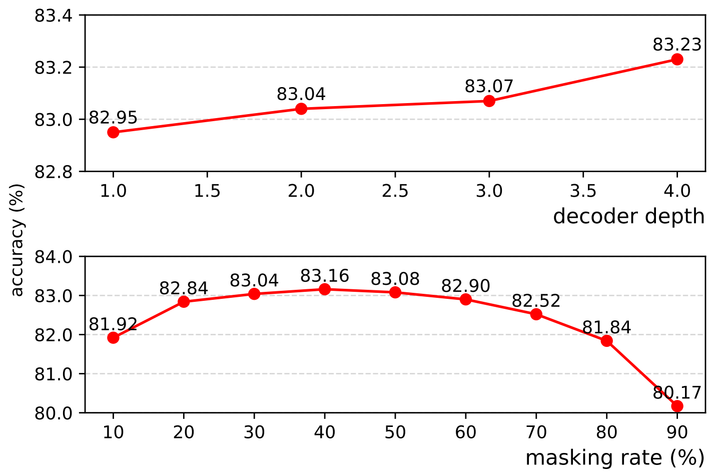


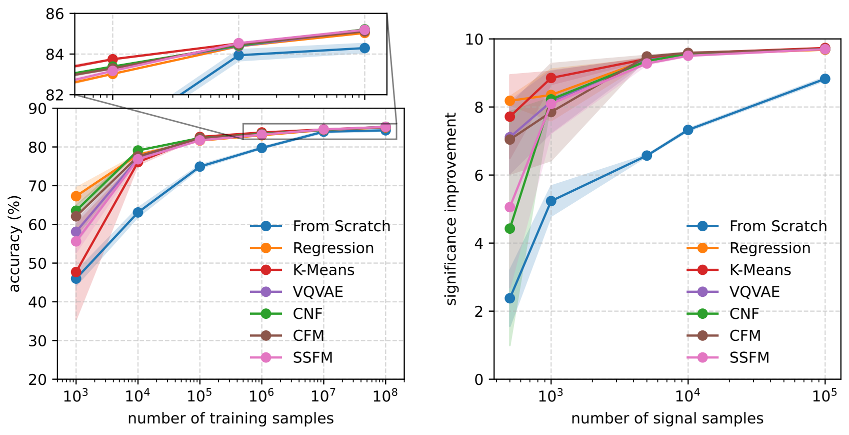
FIGURE 1.8: The effect of the decoder depth (top) and the mask rate (bottom) on the classification accuracy using the outputs produced by an MPMv2 backbone trained with the K-Means and ID tasks.

In-Distribution Classification

Classification is performed on the JetClass dataset using the classifier head as described in the ablation studies. Data efficiency of the backbone is assessed by varying the number of jets used for supervised training from 1k to 100M, as shown in Figure 1.9a. For the 100M sample, all jets in the JetClass training set are used, but 200k steps limits the training to one epoch. At each training set size, pre-trained backbones outperform the randomly initialized network, although the performance boost decreases with the number of jets. At 100M jets, accuracy ranges between 85.0% (regression) and 85.3% (K-Means), compared to 84.3% for random initialization. The K-Means backbone excels with more data, while CNF and Regression backbones are more data-efficient. The Flow backbone matches the performance of the random network with 1M jets using only 10k. The state-of-the-art model ParT [45] employs a similar transformer-based architecture, utilizing additional edge features and training with labels for 1M batches (520M jets), achieving an 86.1% classification accuracy.

Weakly Supervised Classification

We emulate the CWoLa setting using the same methodology as in MPMv1 and use the same classifier head as the previous experiments. In Figure 1.9b, we show the SIC at r_b99% for classifiers on a test set containing pure samples of QCD background and top signal. The pre-trained backbones considerably outperform the benchmark, with the Regression backbone performing the best when only 500 top jets are present in the training set, resulting in a (SIC) of 8.18.



(a) (b)

FIGURE 1.9: The in-distribution performance of the fine-tuned models on the JetClass dataset. (a) shows the accuracy using standard supervised classification as a function of the dataset size. (b) shows the significance-improvement of the models trained in a CWoLa setting as a function of the number of signal samples in the dataset.

Out-of-Distribution Classification

The performance of backbones in classifying the BTag dataset, which comprises lower-energy, narrower jets with a few charged particles, is tested. As shown in Figure 1.10a, the accuracy of the three-class classifier is presented as a function of the number of jets used for training. All pre-trained backbones surpass the benchmark initialization, demonstrating the generalizability of the learned mappings beyond JetClass. The CNF backbone achieves the highest performance in this task, but all pre-trained backbones converge at approximately 70% accuracy with the maximum number of jets.

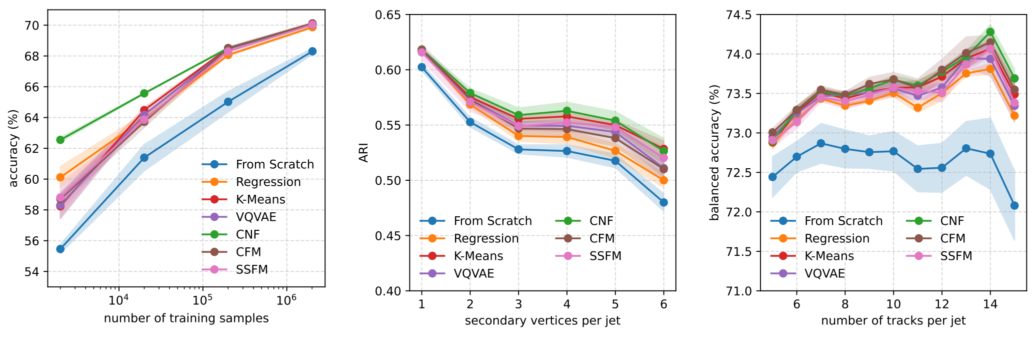
Secondary Vertex Finding

A track vertex denotes a common point from which reconstructed particle tracks originate, indicating an interaction or decay location. Bottom and charm hadrons produced in the collision travel several millimetres beyond the interaction point before decaying, resulting in multiple vertices within the same jet. These vertices are crucial for identifying heavy-flavor jets, as discussed in **??**. Kaon decay also contributes additional vertices. This task is recast as an edge classification problem, where pairs of tracks are classified as positive or negative. This task is similar to instance segmentation in CV, where the goal is to segment the image into unique instances of different objects. There are $N(N-1)/2$ unique pairs to evaluate for a jet with N tracks.

The additional layers for this task follow a twin-network approach [46], where the probability that two tracks x_i and x_j originate from the same vertex is defined by $\sigma\left(G\left[\left|F\left(z_i\right)-F\left(z_j\right)\right|\right]\right)$, with G and F being MLPs, z_i and z_j as backbone outputs, and σ as the sigmoid function. The adjusted Rand index (ARI) [47], as used by Shlomi et al. [48], is the designated performance metric. The ARI is plotted against the number of secondary vertices in Figure 1.10b. The backbone trained with the CNF task performs best, although all backbones surpass the benchmark.

Heavy Track Identification

Beyond grouping tracks by vertex, the task extends to identifying the vertex type associated with each track. It resembles the track origin auxiliary task in **??**. It also draws parallels with the semantic segmentation task in CV. Tracks in the BTag dataset can be linked to a displaced b-quark decay, c-quark decay, or the primary vertex. The task head is a straightforward three-layer MLP appended to the backbone, acting on each constituent independently. Due to the heavily imbalanced class distributions, balanced accuracy is the preferred metric for distinguishing between pre-training methods. As shown in Figure 1.10c, balanced accuracy is plotted as a function of the number of tracks in each jet, with pre-trained backbones consistently outperforming the baselines.



(a) (b) (c)

FIGURE 1.10: The performance of the fine-tuned models on the BTag dataset. (a) shows the supervised jet classifier accuracy versus the number of samples used in fine-tuning. (b) shows the ARI score for the segmentation task versus the number of secondary vertices within each jet. (c) shows the balanced accuracy for the track identification task as a function of the number of tracks in each jet.

1.3 Conclusion

This study aims to build upon MPMv1's findings, in part by assessing the necessity of the costly tokenization step in pre-training. Alternative reconstruction methods were explored, such as simpler tokenization through the K-Means algorithm and conditional generative models. The new models significantly outperform an untrained backbone and the original MPMv1 in various tasks, including those involving out-of-distribution data. However, the differences between these pre-training strategies were minimal. A key improvement was the implementation of a more powerful decoder. Additionally, a novel pre-training method using set-to-set generation was introduced, proving competitive with MPMv2. These findings suggest tokenization is not essential, which may have implications for other SSL models using VQVAE, such as Birk, Hallin, and Kasieczka [18].

Despite the promising results, the self-supervised training strategies for particle physics jets face several limitations. All experiments were conducted using the parametric DELPHES fast simulation package, which does not fully replicate the complexity of real-world data. The next significant step would be transitioning to real data and full simulations. The approach is confined to jets and a few relevant downstream tasks, but another extension would be to move to full event-level data. This would increase the expected set size from $O(100)$ to $O(1000)$, necessitating substantially more computation due to the transformer architecture and addressing these challenges is a key area for future work.

Bibliography

[1] R. Bommasani et al. *On the Opportunities and Risks of Foundation Models*. 2022. DOI: 10.48550/arXiv.2108.07258.

[2] J. Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. DOI: 10.48550/arXiv.1810.04805.

[3] A. Radford et al. *Improving Language Understanding by Generative Pre-Training*. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[4] I. Solaiman et al. *Release Strategies and the Social Impacts of Language Models*. 2019. DOI: 10.48550/arXiv.1908.09203.

[5] M. Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. DOI: 10.48550/arXiv.1910.13461.

[6] T. Brown et al. "Language Models Are Few-Shot Learners". In: Advances in Neural Information Processing Systems. Vol. 33. 2020, pp. 1877–1901. DOI: 10.48550/arXiv.2005.14165.

[7] A. Ramesh et al. *Zero-Shot Text-to-Image Generation*. 2021. DOI: 10.48550/arXiv.2102.12092.

[8] K. He et al. *Masked Autoencoders Are Scalable Vision Learners*. 2021. DOI: 10.48550/arXiv.2111.06377.

[9] J.-B. Alayrac et al. "Flamingo: A Visual Language Model for Few-Shot Learning". In: Advances in Neural Information Processing Systems. 2022. DOI: 10.48550/arXiv.2204.14198.

[10] H. Bao et al. *BEiT: BERT Pre-Training of Image Transformers*. 2022. DOI: 10.48550/arXiv.2106.08254.

[11] M. Assran et al. *Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture*. 2023. DOI: 10.48550/arXiv.2301.08243.

[12] M. Oquab et al. "DINOv2: Learning Robust Visual Features without Supervision". In: *Transactions on Machine Learning Research* (2024). DOI: 10.48550/arXiv.2304.07193.

[13] B. M. Dillon et al. "Symmetries, Safety, and Self-Supervision". In: *SciPost Physics* 12.6 (2022), p. 188. DOI: 10.21468/SciPostPhys.12.6.188.

[14] T. Kishimoto et al. *Pre-Training Strategy Using Real Particle Collision Data for Event Classification in Collider Physics*. 2023. DOI: 10.48550/arXiv.2312.06909.

[15] T. Golling et al. "Masked Particle Modeling on Sets: Towards Self-Supervised High Energy Physics Foundation Models". In: *Machine Learning: Science and Technology* 5.3 (2024), p. 035074.
DOI: 10.1088/2632-2153/ad64a8.

[16] M. Leigh et al. "Is Tokenization Needed for Masked Particle Modelling?" In: Advances in Neural Information Processing Systems. 2024.
DOI: 10.48550/arXiv.2409.12589.

[17] P. Harris et al. *Re-Simulation-based Self-Supervised Learning for Pre-Training Foundation Models*. 2024.
DOI: 10.48550/arXiv.2403.07066.

[18] J. Birk, A. Hallin, and G. Kasieczka. "OmniJet-α: the first cross-task foundation model for particle physics". In: *Machine Learning: Science and Technology* 5.3 (2024), p. 035031.
DOI: 10.1088/2632-2153/ad66ad.

[19] V. Mikuni and B. Nachman. *OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks*. 2024.
DOI: 10.48550/arXiv.2404.16091.

[20] Z. Zhao et al. *Large-Scale Pretraining and Finetuning for Efficient Jet Classification in Particle Physics*. 2024.
DOI: 10.48550/arXiv.2408.09343.

[21] M. Leigh and S. Klein. *MPM*. 2024.
URL: https://github.com/rodem-hep/mpm.

[22] M. Leigh. *JetSSL*. 2024.
URL: https://github.com/mattcleigh/jetssl.

[23] K. Friston. "A Theory of Cortical Responses". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 360.1456 (2005), pp. 815–836.
DOI: 10.1098/rstb.2005.1622.

[24] P. Vincent et al. "Extracting and Composing Robust Features with Denoising Autoencoders". In: International Conference on Machine Learning. 2008, pp. 1096–1103. ISBN: 978-1-60558-205-4.
DOI: 10.1145/1390156.1390294.

[25] T. Finke et al. "Learning the Language of QCD Jets with Transformers". In: *Journal of High Energy Physics* 2023.6 (2023), p. 184.
DOI: 10.1007/JHEP06(2023)184.

[26] S. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.
DOI: 10.1109/TIT.1982.1056489.

[27] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. *Neural Discrete Representation Learning*. 2018.
DOI: 10.48550/arXiv.1711.00937.

[28] H. Qu, C. Li, and S. Qian. *JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics*. Version 1.0.0. 2022.
DOI: 10.5281/ZENODO.6619768.

[29] T. Sjöstrand, S. Mrenna, and P. Skands. "A Brief Introduction to PYTHIA 8.1". In: *Computer Physics Communications* 178.11 (2008), pp. 852–867.
DOI: 10.1016/j.cpc.2008.01.036.

[30] J. Alwall et al. "The Automated Computation of Tree-Level and next-to-Leading Order Differential Cross Sections, and Their Matching to Parton Shower Simulations". In: *Journal of High Energy Physics* 2014.7 (2014), p. 79.
DOI: 10.1007/JHEP07(2014)079.

[31] The DELPHES 3 collaboration et al. "DELPHES 3: A Modular Framework for Fast Simulation of a Generic Collider Experiment". In: *Journal of High Energy Physics* 2014.2 (2014), p. 57.
DOI: 10.1007/JHEP02(2014)057.

[32] M. Cacciari, G. P. Salam, and G. Soyez. "The Anti- k_t Jet Clustering Algorithm". In: *Journal of High Energy Physics* 2008.04 (2008), pp. 063–063.
DOI: 10.1088/1126-6708/2008/04/063.

[33] S. Shleifer, J. Weston, and M. Ott. *NormFormer: Improved Transformer Pretraining with Extra Normalization*. 2021.
DOI: 10.48550/arXiv.2110.09456.

[34] I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. Version 3. 2019.
DOI: 10.48550/arXiv.1711.05101.

[35] Y. Bengio, N. Léonard, and A. Courville. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. 2013.
DOI: 10.48550/arXiv.1308.3432.

[36] M. Huh et al. "Straightening out the Straight-through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks". In: International Conference on Machine Learning. Ed. by A. Krause et al. Vol. 202. 2023, pp. 14096–14113.
URL: https://proceedings.mlr.press/v202/huh23a.html.

[37] M. Huh. *Vqtorch: PyTorch Package for Vector Quantization*. 2022.
URL: https://github.com/minyoungg/vqtorch.

[38] I. Ochoa et al. *Dataset for Flavour Tagging R&D*. Version 1.0.0. Zenodo, 2024.
DOI: 10.5281/zenodo.13350327.

[39] C. Wei et al. *Diffusion Models as Masked Autoencoders*. 2023.
DOI: 10.48550/arXiv.2304.03283.

[40] N. Shazeer. *GLU Variants Improve Transformer*. 2020.
DOI: 10.48550/arXiv.2002.05202.

[41] H. Touvron et al. "Going Deeper With Image Transformers". In: IEEE/CVF International Conference on Computer Vision. 2021, pp. 32–42.
DOI: 10.48550/arXiv.2103.17239.

[42] T. Darcet et al. "Vision Transformers Need Registers". In: International Conference on Learning Representations. 2023.
DOI: 10.48550/arXiv.2309.16588.

[43] W. Peebles and S. Xie. *Scalable Diffusion Models with Transformers*. 2023.
DOI: 10.48550/arXiv.2212.09748.

[44] T. Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. 2022.
DOI: 10.48550/arXiv.2205.14135.

[45] H. Qu, C. Li, and S. Qian. *Particle Transformer for Jet Tagging*. 2024.
DOI: 10.48550/arXiv.2202.03772.

[46] G. Koch, R. Zemel, and R. Salakhutdinov. "Siamese Neural Networks for One-Shot Image Recognition". In: International Conference on Machine Learning. Vol. 37. 2015.
URL: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

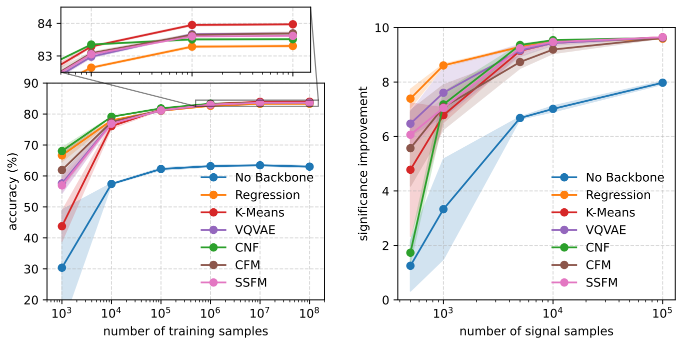
[47] L. Hubert and P. Arabie. "Comparing Partitions". In: *Journal of Classification* 2.1 (1985), pp. 193–218.
DOI: 10.1007/BF01908075.

[48] J. Shlomi et al. “Secondary Vertex Finding in Jets with Neural Networks”. In: *The European Physical Journal C* 81.6 (2021), p. 540.
DOI: 10.1140/epjc/s10052-021-09342-y.

Appendix A

Additional Plots For Foundation Models in High Energy Physics

In addition to fine-tuning, the performance of frozen pre-trained encoders in the same downstream tasks is investigated. The results, presented in Figure A.1 and Figure A.2, demonstrate that these backbones provide a feature-rich latent space.



(a) (b)

FIGURE A.1: The in-distribution performance of the fixed-backbone models on the JetClass dataset. (a) shows the accuracy using standard supervised classification as a function of the dataset size. (b) shows the significance-improvement of the models trained in a CWoLa setting as a function of the number of signal samples in the dataset.

Qualitative results of various continuous reconstructions used by MPMv2 tasks are shown in Figure A.3. Three jets are randomly selected from the JetClass dataset, 40% masking is applied, and each backbone is tasked with reconstructing the dropped constituent features. Zero-padded impact parameters for neutral particles are not plotted. For the regression backbone, direct feature predictions are needed. Reconstruction with the K-Means backbone involves sampling under the discrete distribution of centroid probabilities. The CNF and CFM backbone allow sampling under the normalizing flow, but the CFM does require numerical integration. The Regression predictions frequently collapse towards the centre of the distribution. This is most apparent in the $\Delta\eta$ distribution of Jet-1, which exhibits a bi-modal distribution indicative of a dual-prong jet. While other methods successfully reconstruct this bi-modality, the Regression backbone merely predicts the mean. This lack of diversity might explain why the Regression backbone requires a large decoder to perform well. There is little per-particle information in the space right before the linear layer.

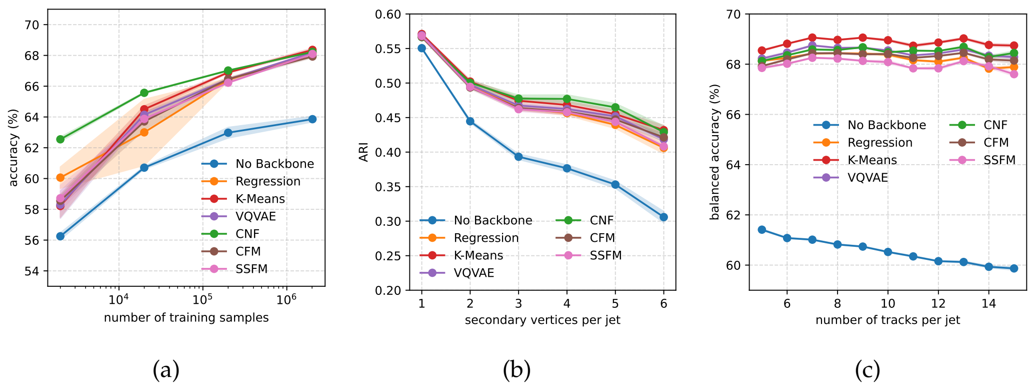


FIGURE A.2: The performance of the fixed backbone models on the BTag dataset. (a) shows the supervised jet classifier accuracy versus the number of samples used in fine-tuning. (b) shows the ARI score for the segmentation task versus the number of secondary vertices within each jet. (c) shows the balanced accuracy for the track identification task as a function of the number of tracks in each jet.

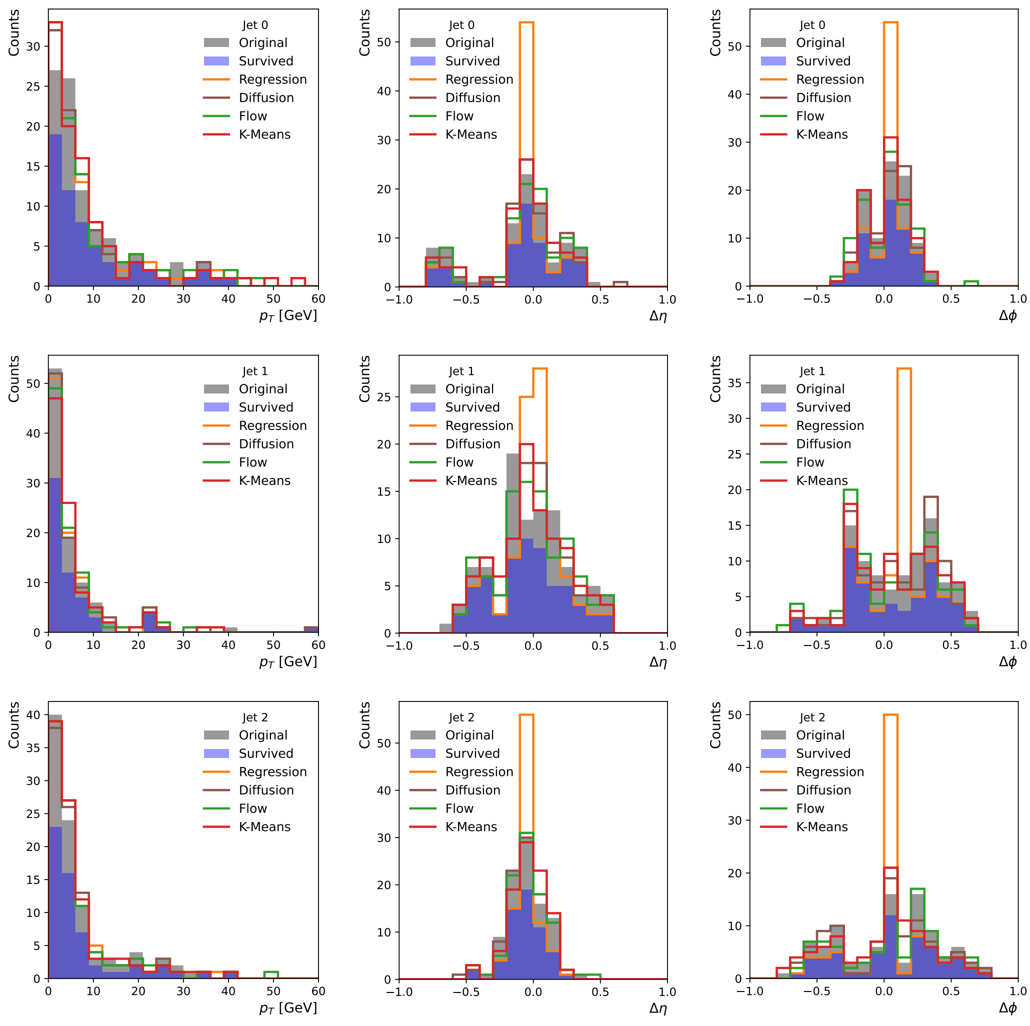


FIGURE A.3: The figure displays three randomly selected jets (rows) from the JetClass dataset, with their $(p_{\mathrm{T}}, \Delta\eta, \Delta\phi)$ distributions plotted (columns). Grey shading indicates the original jet distributions, while blue shading represents the distributions after 40% of the constituents were dropped. Coloured lines illustrate the reconstructed jets using various methods. The optimal reconstruction would ideally align with the original grey distribution.