



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

*Distributed  
Computing*



# YEET - Yield Embedding, Eliminate Topology

Semester Thesis

Sebastian Brunner

brunns@ethz.ch

Distributed Computing Group  
Computer Engineering and Networks Laboratory  
ETH Zürich

**Supervisors:**

Saku Peltonen, Joël Mathys  
Prof. Dr. Roger Wattenhofer

March 19, 2026

## Abstract

Graph Neural Networks (GNNs) that use message-passing are fundamentally limited to local neighbourhood information, hindering their ability to capture complex structural motifs and long-range dependencies. To address this, we propose YEET (Yield Embedding, Eliminate Topology), a framework that integrates contrastive learning with generative Transformer pre-training. YEET employs a two-stage architecture featuring a GNN tokenizer and a Transformer-based Masked Autoencoder. One stage encodes local topological structure, while the other captures global graph information. While experimental results on synthetic and real-world datasets demonstrate that YEET successfully captures local connectivity and node-level features, the study highlights the persistent challenge of encoding high-level global properties in molecular tasks.

## 1. Introduction

Graph Neural Networks (GNNs) have traditionally relied on the message-passing paradigm to learn node representations through local neighbourhood aggregation. Although this iterative process is computationally efficient, it is fundamentally limited in its ability to capture global graph information. To overcome these limitations, recent research has shifted toward architectures that bypass traditional message-passing and apply generative Transformer pre-training. These approaches aim to enhance both the expressivity and scalability of graph-based models. Furthermore, to mitigate the heavy reliance on labelled data, self-supervised learning (SSL) has become a vital frontier. By leveraging contrastive frameworks and masked autoencoding (MAE) strategies, it is possible to learn robust representations of the underlying graph.

In this paper, we propose **YEET** (Yield Embedding, Eliminate Topology), a framework that leverages the strengths of contrastive learning and generative pre-training in a streamlined fashion. YEET employs a two-stage pre-training architecture consisting of a GNN tokenizer and a Transformer-based Masked Autoencoder. Our approach is designed to decouple local message-passing from global self-attention, learning expressive global representations while relying on the GNN to encode local topology.

The methodology of YEET is structured as follows:

- **Stage 1: GNN Tokenizer and Contrastive Learning.** A message-passing GNN encodes node features and neighbourhood information to generate dense, node-level latent tokens. To ensure these tokens capture

distinct features and do not collapse, we apply a node-level contrastive objective on augmented input graphs generated by perturbing, concatenating, or substituting initial node features with random noise.

- **Stage 2: Transformer-Based Masked Autoencoder.** The resulting latent tokens are sequentially sorted and augmented with positional encodings to inject structural context. Following the MAE paradigm, a large proportion of these tokens are randomly masked. A Transformer-based encoder processes the visible tokens, while a decoder is optimised to reconstruct the masked targets.

Through several experiments with synthetic and real-world graphs, we evaluate the model’s ability to learn structural and node-level representations. Our results demonstrate that while YEET captures local connectivity in tasks like node degree recovery, the sequence-based Stage 2 highlights the ongoing challenge of encoding high-level properties for complex molecular tasks without more sophisticated positional encodings.

## 2. Background & Related Work

### 2.1. Limitations of Message-Passing GNNs

Message-passing GNNs, where node representations are iteratively updated by aggregating local neighbourhood information, suffer from fundamental limitations. Their expressive power is notoriously constrained by the bounds of the 1-dimensional Weisfeiler-Leman (1-WL) test (Morris et al., 2021; Xu et al., 2019). Consequently, standard GNNs struggle to explicitly capture foundational structural motifs and possess a limited capacity to model long-range dependencies. To address these structural bottlenecks, recent frameworks like the Neural Graph Pattern Machine (GPM) (Wang et al., 2025a) and the Generative Graph Pattern Machine ( $G^2PM$ ) (Wang et al., 2025b) propose bypassing message passing altogether. These approaches extract graph substructures directly and employ generative Transformer pre-training over sequences of these patterns, enhancing model expressivity and scalability.

### 2.2. Contrastive Self-Supervised Learning on Graphs

To alleviate the reliance on abundant labelled data, self-supervised learning (SSL)—particularly contrastive learning—has emerged as a dominant approach for graph representation. Typical contrastive frameworks, such as BGRL (Thakoor et al., 2023) or GRACE (Zhu et al., 2020), aim to maximise the agreement of node representations across multiple augmented graph views. These views are typically generated by corrupting the graph at both the structural and attribute levels to provide diverse node contexts. However,

contrastive approaches heavily depend on complicated training strategies, meticulous structural data augmentations, and negative sampling. The first stage of our proposed YEET framework leverages the strengths of contrastive learning in a more streamlined fashion. By applying a node-level contrastive objective on augmented input graphs—generated by perturbing, concatenating, or substituting initial node features with random noise—YEET ensures that the GNN-generated tokens capture distinct topological features.

### 2.3. Generative Masked Graph Autoencoders

Inspired by the widespread success of masked autoencoders (MAEs) in natural language processing and computer vision, generative SSL on graphs has gained considerable traction. GraphMAE (Hou et al., 2022) focused on feature reconstruction, utilising a masking strategy coupled with a scaled cosine error to robustly train the autoencoder. To further improve cross-task generalisability, GiGaMAE (Shi et al., 2023) introduced a collaborative latent space reconstruction paradigm. Rather than explicitly reconstructing raw graph components like features or edges, GiGaMAE reconstructs latent embeddings that capture both topology and attribute information simultaneously. YEET extends this trajectory of generative pre-training in its second stage. By feeding dense, node-level latent tokens from a GNN tokenizer into a Transformer-based MAE, YEET successfully separates local message-passing from global self-attention, learning expressive global representations while relying on the GNN to encode local topology.

## 3. Methodology

Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$  denote a graph, where  $\mathcal{V}$  is the set of nodes with  $|\mathcal{V}| = N$ ,  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  is the set of edges with  $|\mathcal{E}| = M$ , and  $\mathbf{X}$  and  $\mathbf{E}$  are node and edge features. Each node  $v \in \mathcal{V}$  is associated with a feature vector  $\mathbf{x}_v \in \mathbb{R}^{d_n}$ , and each edge  $e \in \mathcal{E}$  is associated with a feature vector  $\mathbf{e}_e \in \mathbb{R}^{d_m}$  (if applicable).

The proposed YEET framework employs a two-stage pre-training architecture: a GNN tokenizer and a Transformer-based Masked Autoencoder (MAE). This approach leverages both local message passing and global self-attention to learn expressive node representations.

### 3.1. Stage 1: GNN Tokenizer and Contrastive Learning

In the first stage, a message-passing GNN—such as the Graph Isomorphism Network (GIN) (Xu et al., 2019)—serves as a tokenizer. This network encodes node features and neighbourhood information to generate dense, node-level latent representations (tokens):

$$T = \text{GNN}(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}) \in \mathbb{R}^{N \times d_t} \quad (1)$$

To ensure these tokens capture distinct features and do not collapse, we apply a node-level contrastive learning objective. We generate an augmented input graph by perturbing (Eq. 2), concatenating (Eq. 3), or substituting (Eq. 4) the initial node features with random noise.

$$\mathbf{x}'_v = \mathbf{x}_v + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_n}) \quad (2)$$

$$\mathbf{x}'_v = [\mathbf{x}_v \parallel \epsilon], \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_\epsilon}) \quad (3)$$

$$\mathbf{x}'_v = \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_n}) \quad (4)$$

Both the unaltered graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$  and the augmented graph  $\mathcal{G}' = (\mathcal{V}, \mathcal{E}, \mathbf{X}', \mathbf{E})$  are passed through the tokenizer GNN. The contrastive loss (Eq. 5) is computed by defining positive pairs as the embeddings of the same underlying node across two distinct augmented views. Negative samples are drawn randomly from other nodes.

$$\mathcal{L}_{CL} = \frac{1}{N} \sum_{v \in \mathcal{V}} \left[ \overbrace{1 - \cos(\mathbf{t}_v, \mathbf{t}'_v)}^{\text{positive samples}} + \overbrace{\frac{1}{k} \sum_{u \in \mathcal{V}_{-v}} \max(0, \cos(\mathbf{t}_v, \mathbf{t}'_u))}^{\text{negative samples}} \right] \quad (5)$$

where  $\mathcal{V}_{-v} \subset \mathcal{V} \setminus \{v\}$ ,  $|\mathcal{V}_{-v}| = k$

### 3.2. Stage 2: Transformer-Based Masked Autoencoder

**Token Sequence and Encoding.** Following tokenization, the resulting latent tokens  $T \in \mathbb{R}^{N \times d_t}$  are sequentially sorted relative to their scalar product with a random but fixed anchor  $t_{\text{anchor}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_t})$  to establish a deterministic ordering. This deterministic ordering establishes a consistent spatial mapping, enabling the decoder to align latent representations with their corresponding reconstruction targets. Sinusoidal positional encodings (*PE*) are then added to inject structural context, yielding a token sequence  $T_{\text{sort}}$ :

$$T_{\text{sort}} = \text{sort}(T, t_{\text{anchor}}) + PE. \quad (6)$$

Following the MAE paradigm, a large proportion of these tokens is randomly masked and discarded. Let  $M$  denote the set of masked token indices, and  $T_{-M}$  denote the remaining subset of visible tokens. Only the visible tokens, prepended with a learnable classification token ( $t_{\text{CLS}}$ ), are processed by a Transformer-based encoder to yield node embeddings  $Z$ :

$$Z = \text{Encoder}([t_{\text{CLS}} \parallel T_{-M}]) \quad (7)$$

**Decoding and Reconstruction.** Before decoding, the original sequence length  $N$  is restored. The embeddings from

the encoder,  $Z$ , are returned to their original sorted positions. The omitted nodes are replaced with a shared, learnable mask token,  $z_{\text{mask}}$ . Sinusoidal positional encodings are added to this full sequence to provide the decoder with the spatial context of the masked elements. Let this restored sequence be denoted as  $Z'$ :

$$Z' = \text{restore}([Z \parallel z_{\text{mask}}, T_{\text{sort}}] + PE). \quad (8)$$

The complete sequence is then passed through a Transformer-based decoder to generate predictions  $\hat{Y}$ :

$$\hat{Y} = \text{Decoder}(Z') \quad (9)$$

### 3.3. Optimization Objective

The reconstruction quality of the MAE is optimised using a Mean Squared Error (MSE) loss. This loss, denoted as  $\mathcal{L}_{\text{recon}}$ , is computed exclusively on the difference between the reconstructed outputs  $\hat{Y}_i$  and the specific targets  $Y_i$  of the masked tokens ( $i \in M$ ):

$$\mathcal{L}_{\text{recon}} = \frac{1}{|M|} \sum_{i \in M} \|\hat{Y}_i - Y_i\|^2, \quad Y_i \in \left\{ \mathbf{x}_i, \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2}, \mathbf{t}_i, \frac{\mathbf{t}_i}{\|\mathbf{t}_i\|_2} \right\} \quad (10)$$

The overall model is thus optimised by jointly minimising this reconstruction MSE alongside the contrastive loss ( $\mathcal{L}_{\text{CL}}$ ) from Stage 1. The final total loss can be formalised as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{CL}} \quad (11)$$

Figure 1 visualises the architecture.

## 4. Experiments

To evaluate the model’s ability to learn structural and node-level representations, we conduct several experiments with synthetic and real-world graphs, as well as different downstream and reconstruction tasks.

### 4.1. Datasets

We utilise four datasets to assess distinct facets of the framework, ranging from structural recovery to real-world classification.

**Synthetic chain graphs:** To train and evaluate the model to capture both global and local structural dependencies, we employ a hierarchical dynamic generation process. Each graph instance is initialised by constructing a backbone path graph  $\mathcal{P}_n$ , where the length  $n$  is sampled uniformly from  $\{2, 3, 4, 5\}$ . Then, each node  $v \in \mathcal{V}(\mathcal{P}_n)$  is augmented by attaching a stochastically

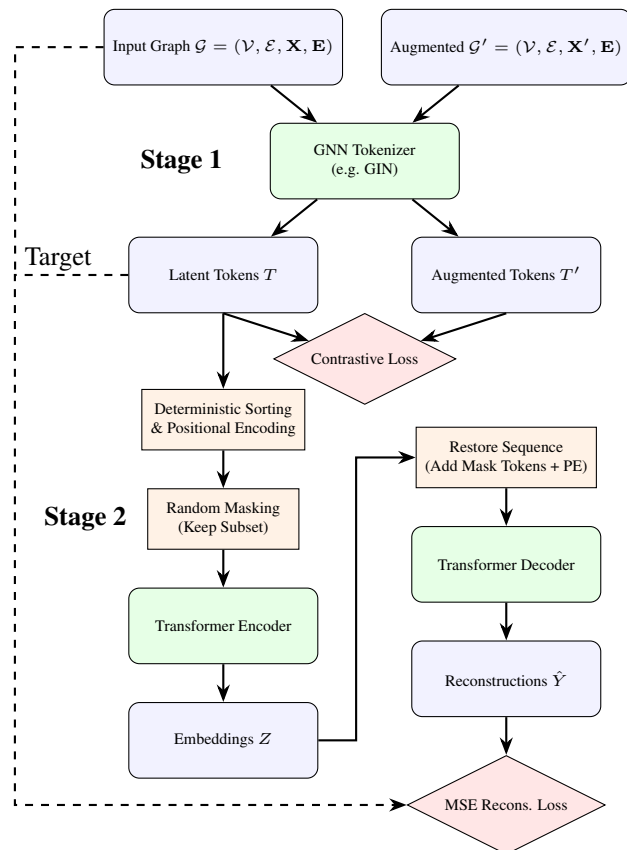


Figure 1. The 2-staged YEET architecture. Stage 1 aims to encode local information while Stage 2 captures global information.

selected subgraph  $\mathcal{G}_{\text{sub}}$ . These subgraphs are sampled from a predefined library of primitive topologies  $\mathcal{L} = \{\text{empty, path, tree, triangle, cycle, star, complete}\}$ . This hierarchical construction ensures that while all graphs share a common underlying skeleton, the local structure varies significantly. Since these synthetic graphs lack inherent node features, random features (see Equation 4) are used.

**PCBA and HIV:** These datasets, frequently used in graph benchmarks, exhibit node features and provide labels for graph classification. We used trained embeddings for the integer-based node and edge features as in Hu et al. (2020). These datasets are used to ablate the Stage 2 reconstruction ability as well as evaluate the model’s expressiveness in downstream tasks.

**CIFAR-10:** To verify the effectiveness of Stage 2, we applied the MAE to images from the CIFAR-10 dataset.

### 4.2. Ablations

To isolate the contribution of each component, we evaluate the model in several modified configurations.

**Hyperparameters:** To test the impact of the different com-

ponents, we ablated the GNN tokenizer and Transformer encoder, both with 1 and 6 layers each. To investigate the impact of information density, we evaluated masking rates across a broad spectrum: 0.1, 0.5, and 0.9.

**Reconstruction ability:** To verify that Stage 2 is able to reconstruct features, we tested the same architecture used in Stage 2 to reconstruct images from the CIFAR-10 dataset. We also conducted a comparative analysis of 'norm-first' (pre-block) versus 'norm-last' (post-block) normalisation to identify the configuration that best ensures training stability.

For graphs, we tried reconstructing different targets: raw and standardised features ( $\mathbf{x}_i$ ), and raw and normalised latent tokens ( $\mathbf{t}_i$ ), as described in Equation 10.

### 4.3. Pretraining Metrics

During pretraining, the model is evaluated based on the Mean Squared Error (MSE) between the reconstructed outputs and the targets, as well as the Contrastive Loss (CL) on the tokens. Beyond quantitative metrics, we perform qualitative analysis by visualising the evolution of latent tokens and embeddings to monitor for representation collapse or convergence patterns.

### 4.4. Downstream Tasks

To assess the structural expressiveness of the pre-trained embeddings, the model was evaluated on several downstream tasks. For evaluation, the pretrained model is frozen, and only a linear head is applied to the embeddings. For datasets with incompatible feature dimensions, a trainable linear projection is used to align the input features with the model's latent dimension.

**Node degree recovery:** This task involves predicting a node's degree based solely on its individual embedding to test local structural awareness.

**Graph classification:** Given the learnt classification token  $t_{[CLS]}$ , predict the given graph label in the HIV dataset.

**Pairwise distance:** Given two node embeddings and the learnt classification token  $t_{[CLS]}$ , predict the distance (number of hops) between these two nodes.

## 5. Results

This section presents the empirical data gathered from pre-training and downstream evaluation.

### 5.1. Reconstruction and Embedding Quality

While the reconstruction of raw features  $\mathbf{x}_i$  was successful without a Stage 1 contrastive loss  $\mathcal{L}_{CL}$  (see Figure 2), the model was less successful when both losses were applied

simultaneously. Reconstructing raw latent tokens  $\mathbf{t}_i$  also proved problematic due to their dynamic nature. The latent tokens  $\mathbf{t}_i$  exhibited a collapse toward a minimal  $\ell^\infty$ -norm. This resulted in an artificially low Stage 2 loss  $\mathcal{L}_{recon}$  that did not reflect meaningful feature recovery. When the target latent tokens were normalised, the training stabilised and a better reconstruction was achieved, as shown in Figure 3.

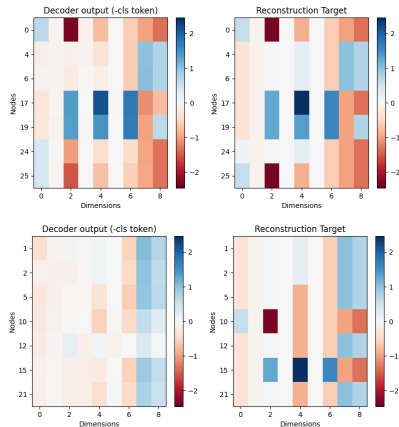


Figure 2. *Top:* Model can reconstruct most features well. *Bottom:* Model fails to reconstruct high-variance or outlier features. PCBA dataset, no Stage 1 contrastive loss  $\mathcal{L}_{CL}$ .

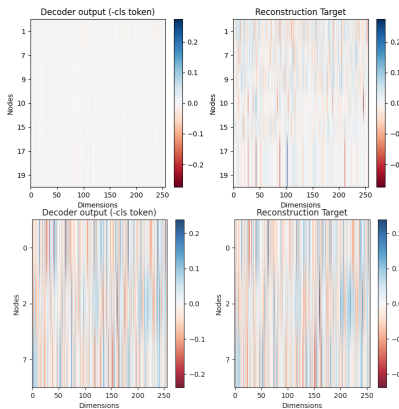


Figure 3. *Top:* The model struggles to reconstruct unnormalised latent tokens  $\mathbf{t}_i$ . *Bottom:* Normalised latent tokens  $\frac{\mathbf{t}_i}{\|\mathbf{t}_i\|_2}$  can be reconstructed well.

Furthermore, node embeddings  $\mathbf{z}_i$  regressed to a mean for raw latent targets, with representations being nearly equivalent despite the tokens  $\mathbf{t}_i$  being distinct. For normalised targets, however, the model yields distinct embeddings, as shown in Figure 4.

### 5.2. Tokenizer, Encoder and Masking Ablations

Ablation studies conducted on the chain dataset focused on the impact of the number of GNN tokenizer layers, encoder depth, and masking rate.

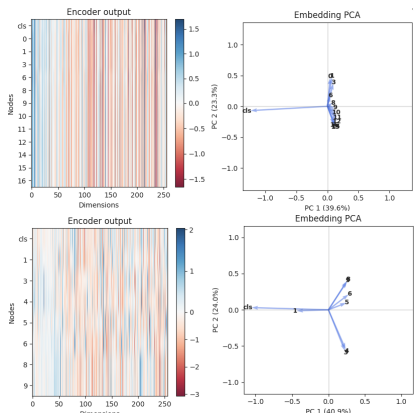


Figure 4. *Top*: Reconstructing unnormalised tokens  $t_i$  leads to a collapse in embeddings. *Bottom*: Normalising tokens stabilises training, leading to distinct embeddings.

**Masking Rate:** Varying the masking rate influences the reconstruction loss  $\mathcal{L}_{\text{recon}}$ , with a rate of 0.1 leading to the smallest loss. The contrastive loss  $\mathcal{L}_{\text{CL}}$  remains unaffected. This behaves as expected since reconstructing fewer targets is deemed easier.

**Tokenizer depth:** We observe an inverse relationship between tokenizer depth and loss types: reducing the number of layers increases  $\mathcal{L}_{\text{CL}}$  while simultaneously lowering  $\mathcal{L}_{\text{recon}}$ . This phenomenon likely stems from the GNN’s restricted receptive field, which prevents it from distinguishing between nodes with different orbits, resulting in highly similar (higher  $\mathcal{L}_{\text{CL}}$ ) but more easily reconstructible (lower  $\mathcal{L}_{\text{recon}}$ ) tokens.

**Encoder depth:** Varying the encoder depth affects both losses the least. Although a deeper encoder reconstructs the target better, the effects are negligible compared to those of the tokenizer.

Table 1 shows how each ablation affects the losses.

Table 1. Ablations.

Masking Rate	Tokenizer Layers	Encoder Layers	$\mathcal{L}_{\text{CL}}$	$\mathcal{L}_{\text{recon}}$	$\mathcal{L}_{\text{total}}$
0.9	6	6	0.048	0.003	0.052
0.9	6	1	0.054	0.004	0.058
0.9	1	6	0.897	0.001	0.899
0.9	1	1	0.896	0.002	0.898
0.5	6	6	0.050	0.002	0.053
0.5	6	1	0.050	0.002	0.053
0.5	1	6	0.897	0.002	0.898
0.5	1	1	0.897	0.002	0.900
0.1	6	6	0.051	0.002	0.052
0.1	6	1	0.037	0.002	<b>0.039</b>
0.1	1	6	0.898	0.001	0.889
0.1	1	1	0.895	0.002	0.897

### 5.3. Downstream Tasks

The model was evaluated on three downstream tasks.

**Node Degree Recovery:** While only pretrained on the synthetic chain dataset, linear probing on either the HIV or the chain dataset led to a MSE of 0.40 and 0.28, respectively. For comparison, simply predicting the mean on both datasets leads to an MSE of 2.20 and 3.70.

**Graph Classification (HIV):** The HIV dataset is highly imbalanced, as only a small fraction of molecules exhibit inhibitory activity. A random, uninformed classifier yields an AUROC of 0.5. Out of distribution pretraining on the chain dataset did not produce meaningful embeddings for the HIV task, resulting in an AUROC below 0.5. Pretraining on the HIV dataset was more successful, but the results depend on the reconstruction target and the applied input augmentation. While reconstructing features failed to yield predictive representations (AUROCs below 0.5), both reconstructing latents from features with concatenated (see Equation 3) and added (see Equation 2) randomness led to AUROCs of 0.57 and 0.64. Although the model outperforms the majority-class baseline, it falls short of logistic regression (0.72 AUROC (Wu et al., 2018)) and remains non-competitive with current state-of-the-art models (0.84 AUROC (Wei et al., 2021)).

**Pairwise Distance:** Predicting the distance of two nodes given just their corresponding node embeddings and the learnt classification token  $t_{[\text{CLS}]}$  is a hard task. Our model beat the mean prediction (MSE of 4.18) with an MSE of 3.04, showing that it did not regress to the mean. Noteworthy, however, is that the model performed better the closer the two nodes were. The model correctly predicted almost half of the samples that are just 1 hop away and nearly a third within a 2 or 3 hop neighbourhood.

### 5.4. Stability and Normalisation

We compared ‘norm-first’ and ‘norm-last’ configurations within the Transformer-based MAE. The ‘norm-first’ configuration was able to reconstruct the image features. In contrast, the ‘norm-last’ configuration failed to recover structural details, converging to a uniform, low-variance output (see Figure 5). Furthermore, experiments with a baseline Vision Transformer (ViT) showed that reconstruction fails when the input is partitioned into too few tokens (e.g., 4 patches vs 16). This suggests that the Stage 2 Transformer requires a minimum threshold of token granularity to effectively learn the underlying data distribution.

## 6. Discussion

The YEET architecture demonstrates the feasibility of a two-stage pre-training pipeline that bridges local message-passing with global self-attention mechanisms. By decou-

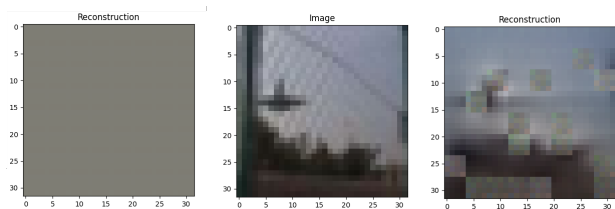


Figure 5. *Left*: Reconstruction with "norm-last". *Middle*: Reconstruction target. *Right*: Reconstruction with "norm-first". Same architecture as Stage 2.

pling these processes, the framework provides a structured approach to learning node representations that capture both immediate neighbourhood structures and broader dependencies.

### 6.1. Stability, Targets, and Normalisation

A critical finding in our experiments was the sensitivity of the Stage 2 Transformer to the target distribution. Although reconstructing features was possible, it did not lead to meaningful embeddings in the HIV task. A possible cause for this could be that the features themselves lack global meaning while also being detailed enough to require a lot of attention to reconstruct, leading the model to focus on the wrong aspects.

Reconstructing raw latent tokens proved problematic, as the representations tended to collapse toward a minimal  $\ell^\infty$ -norm. This collapse, resulting in artificially low loss values that fail to create useful representations, might be due to the model taking the "path of least resistance". This shows that when allowing a model to train its own targets, some form of regularisation is critical.

### 6.2. Structural Awareness vs. Global Downstream Tasks

The model's performance on downstream tasks reveals a discrepancy between local structural awareness and global graph classification. YEET showed strong results in predicting node degrees and promising results for pairwise distances, outperforming simple mean-prediction baselines.

However, for complex tasks like HIV graph classification, the model remains non-competitive with state-of-the-art architectures. While the use of added or concatenated randomness in Stage 1 improved AUROC scores to 0.64, the model still lagged behind logistic regression baselines. This suggests that while the GNN tokenizer captures local connectivity well, the sequence-based Stage 2 struggles to encode the high-level properties necessary for molecular tasks. This suggests that, in its current form, Stage 2 might actively disrupt the local feature signals rather than enhance them.

### 6.3. Tokenizer Depth Matters

Our ablation studies highlighted a trade-off between tokenizer depth and reconstruction quality. Reducing the number of GNN layers lowered  $\mathcal{L}_{recon}$  but increased  $\mathcal{L}_{CL}$ . We attribute this to the restricted receptive field of shallower GNNs, which produces simpler, more easily reconstructible tokens that lack the nuance required to distinguish complex node orbits.

### 6.4. Future Work

Although  $\ell_1$  or  $\ell_2$  regularisation could offer more flexibility than explicit normalisation, superior results may be achieved by guiding Stage 2 with losses that prioritise directionality over exact values, such as NT-Xent contrastive loss (Chen et al., 2020) or Scaled Cosine Error reconstruction (Hou et al., 2022). Future iterations could explore alternatives to our deterministic sorting method to better preserve permutation invariance or investigate more sophisticated positional encodings (PEs). Applying classic graph PEs, such as random walk probabilities or Laplacians, combined with a Hungarian loss, could increase the model's capability to capture meaningful global information by injecting structural information into the attention process of Stage 2. Additionally, reconstructing an exponential moving average (EMA), which has been successfully applied in other foundation models such as DINO (Caron et al., 2021), could further enhance training by allowing latent tokens to adapt while, at the same time, providing a more stable reconstruction target. Moreover, similar to the pretraining of BERT (Devlin et al., 2019), adding a training objective for the classification token  $t_{[CLS]}$ , such as predicting the pairwise distance between nodes, could induce meaning into the classification token and yield improved results.

In summary, while YEET demonstrates that message-passing and self-attention can be decoupled, the current implementation highlights that local topology remains a dominant signal that the global transformer cannot yet fully replace without more sophisticated positional encodings.

## References

- Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers, 2021. URL <https://arxiv.org/abs/2104.14294>.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations, 2020. URL <https://arxiv.org/abs/2002.05709>.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for lan-

- guage understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders, 2022. URL <https://arxiv.org/abs/2205.10803>.
- Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.
- Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks, 2021. URL <https://arxiv.org/abs/1810.02244>.
- Shi, Y., Dong, Y., Tan, Q., Li, J., and Liu, N. Gigamae: Generalizable graph masked autoencoder via collaborative latent space reconstruction, 2023. URL <https://arxiv.org/abs/2308.09663>.
- Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping, 2023. URL <https://arxiv.org/abs/2102.06514>.
- Wang, Z., Zhang, Z., Ma, T., Chawla, N. V., Zhang, C., and Ye, Y. Beyond message passing: Neural graph pattern machine, 2025a. URL <https://arxiv.org/abs/2501.18739>.
- Wang, Z., Zhang, Z., Ma, T., Zhang, C., and Ye, Y. Generative graph pattern machine, 2025b. URL <https://arxiv.org/abs/2505.16130>.
- Wei, L., Zhao, H., Yao, Q., and He, Z. Pooling architecture search for graph classification. In *CIKM*, 2021.
- Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: A benchmark for molecular machine learning, 2018. URL <https://arxiv.org/abs/1703.00564>.
- Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks?, 2019. URL <https://arxiv.org/abs/1810.00826>.
- Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning, 2020. URL <https://arxiv.org/abs/2006.04131>.