Energy Transformer

Benjamin Hoover[*]
IBM Research
Georgia Tech
benjamin.hoover@ibm.com

Yuchen Liang[*]
Department of CS
RPI
liangy7@rpi.edu

Bao Pham[*]
Department of CS
RPI
phamb@rpi.edu

Rameswar Panda
MIT-IBM Watson AI Lab
IBM Research
rpanda@ibm.com

Hendrik Strobelt
MIT-IBM Watson AI Lab
IBM Research
hendrik.strobelt@ibm.com

Duen Horng Chau
College of Computing
Georgia Tech
polo@gatech.edu

Mohammed J. Zaki
Department of CS
RPI
zaki@cs.rpi.edu

Dmitry Krotov
MIT-IBM Watson AI Lab
IBM Research
krotov@ibm.com

Abstract

Our work combines aspects of three promising paradigms in machine learning, namely, attention mechanism, energy-based models, and associative memory. Attention is the power-house driving modern deep learning successes, but it lacks clear theoretical foundations. Energy-based models allow a principled approach to discriminative and generative tasks, but the design of the energy functional is not straightforward. At the same time, Dense Associative Memory models or Modern Hopfield Networks have a well-established theoretical foundation, and allow an intuitive design of the energy function. We propose a novel architecture, called the Energy Transformer (or ET for short), that uses a sequence of attention layers that are purposely designed to minimize a specifically engineered energy function, which is responsible for representing the relationships between the tokens. In this work, we introduce the theoretical foundations of ET, explore its empirical capabilities using the image completion task, and obtain strong quantitative results on the graph anomaly detection and graph classification tasks.

1 Introduction

Transformers have become pervasive models in various domains of machine learning, including language, vision, and audio processing. Every transformer block uses four fundamental operations: attention, feed-forward multi-layer perceptron (MLP), residual connection, and layer normalization. Different variations of transformers result from combining these four operations in various ways. For instance, [1] proposes to frontload additional attention operations and backload additional MLP layers in a sandwich-like manner instead of interleaving them, [2] prepends an MLP layer before the attention in each transformer block, [3] uses neural architecture search methods to evolve even more sophisticated transformer blocks, and so on. Various methods exist to approximate the attention operation, multiple modifications of the norm operation, and connectivity of the block; see, for example, [4] for a taxonomy of different models. At present, however, the search for new transformer

[*]B.Hoover, Y.Liang, and B.Pham equally contributed to this work.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

https://doi.org/10.52202/075280-1197

architectures is driven mostly by empirical evaluations, and the theoretical principles behind this growing list of architectural variations is missing.

Additionally, the computational role of the four elements remains the subject of discussions. Originally, [5] emphasized attention as the most important part of the transformer block, arguing that the learnable long-range dependencies are more powerful than the local inductive biases of convolutional networks. On the other hand more recent investigations [6] argue that the entire transformer block is important. The "correct" way to combine the four basic operations inside the block remains unclear, as does an understanding of the core computational function of the entire block and each of its four elements.

On the other hand, Associative Memory models, also known as Hopfield Networks [7, 8], have been gaining popularity in the machine learning community thanks to theoretical advancements pertaining to their memory storage capacity and novel architectural modifications [9]. Specifically, it has been shown that increasing the sharpness of the activation functions can lead to super-linear [10] and even exponential [11] memory storage capacity for these models, which is important for machine learning applications. This new class of Hopfield Networks is called Dense Associative Memories or Modern Hopfield Networks. Study [12] additionally describes how the attention mechanism in transformers is closely related to a special model of this family with the $\mathrm{softmax}$ activation function.

There are high-level conceptual similarities between transformers and Dense Associative Memories, since both architectures are designed for some form of denoising of the input. Transformers are typically pre-trained on a masked-token task, e.g., in the domain of Natural Language Processing (NLP), certain tokens in the sentence are masked and the model predicts the masked tokens. Dense Associative Memory models are designed for completing the incomplete patterns. They can be trained in a self-supervised way by predicting the masked parts of the pattern, or denoising the pattern.

There are also high-level differences between the two approaches. Associative Memories are recurrent networks with a global energy function so that the network dynamics converges to a fixed point attractor state corresponding to a local minimum of the energy function. Transformers are typically not described as dynamical systems at all. Rather, they are thought of as feed-forward networks built from the four computational elements discussed above. Even if one thinks about them as dynamical systems with tied weights, e.g., [13], there is no reason to expect that their dynamics converge to a fixed point attractor (see the discussion in [14]).

Additionally, a recent study [15] uses a form of Majorization-Minimization algorithms [16] to interpret the forward path in the transformer block as an optimization process. This interpretation requires imposing certain constraints on the operations inside the block, and attempting to find an energy function that describes the constrained block. We take a complementary approach by using the intuition developed in Associative Memory models to *start* with an energy function that is engineered for the problem of interest. The optimization process and the resulting architecture of the transformer block in our approach is a *consequence* of this specifically chosen energy function.

Concretely, we use the recent theoretical advancements and architectural developments in Dense Associative Memories to design an energy function tailored to route the information between the tokens. The goal of this energy function is to represent the relationships between the semantic contents of tokens describing a given data point (e.g., the relationships between the contents of the image patches in the vision domain, or relationships between the nodes' attributes in the graph domain). The core mathematical idea of our approach is that the sequence of these unusual transformer blocks, which we call the Energy Transformer (ET), minimizes this global energy function. Thus, the sequence of conventional transformer blocks is replaced with a single ET block, which iterates the token representations until they converge to a fixed point attractor state. In the image domain, this fixed point corresponds to the completed image with masked tokens replaced by plausible auto-completions of the occluded image patches. In the graph domain, the fixed point reveals the anomaly status of a node given its neighbors, see Figure 1, or the graph label. The energy function in our ET block is designed with the goal to describe the *relationships between the tokens*. Examples of relationships in the image domain are: straight lines tend to continue through multiple patches, given a face with one eye being masked the network should inpaint the missing eye, etc. In the graph domain, these are the relationships between the features of nodes; or features of nodes and graph label in graph classification.

The core mathematical principle of the ET block – the existence of the global energy function – dictates strong constraints on the possible operations inside the block, the order in which these

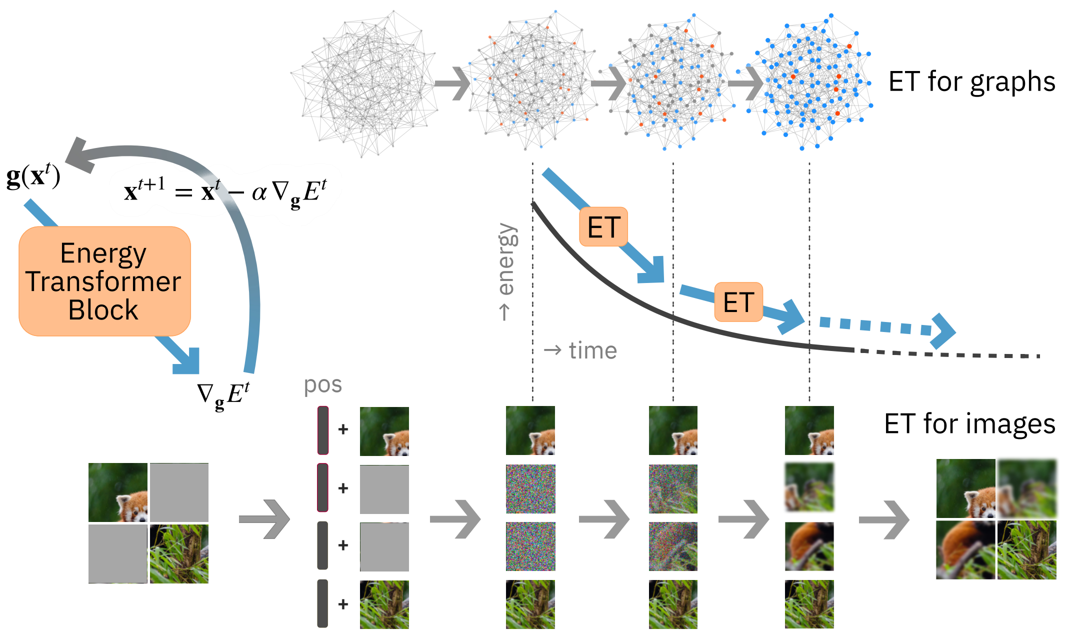

Figure 1: Overview of the Energy Transformer (ET). Instead of a sequence of conventional transformer blocks, a single recurrent ET block is used. The operation of this block is dictated by the global energy function. The token representations are updated according to a continuous time differential equation with the time-discretized update step $\alpha = dt/\tau$. On the image domain, images are split into non-overlapping patches that are linearly encoded into tokens with added learnable positional embeddings (POS). Some patches are randomly masked. These tokens are recurrently passed through ET, and each iteration reduces the energy of the set of tokens. The token representations at or near the fixed point are then decoded using the decoder network to obtain the reconstructed image. The network is trained by minimizing the mean squared error loss between the reconstructed image and the original image. On the graph domain, the same general pipeline is used. Each token represents a node, and each node has its own positional encoding. The token representations at or near the fixed point are used for the prediction of the anomaly status of each node, or the graph label.

operations are executed in the inference pass, and the symmetries of the weights in the network. As a corollary of this theoretical principle, the attention mechanism of ET is different from the attention mechanism commonly used in feed-forward transformers [5]. Lastly, our network may be viewed as an example of a broader class of Energy-Based Models [17] frequently discussed in the AI community. The proposed model is defined through the specific choice of the energy function, which, on the one hand, is suitable for the computational task of interest, and, on the other hand, results in optimization equations closely related to the forward pass in feed-forward transformers.

2 Energy Transformer Block

We now introduce the theoretical framework of the ET network. For clarity of presentation, we use language associated with the image domain. For the graph domain, one should think about "image patches" as nodes on the graph.

The overall pipeline is similar to the Vision Transformer networks (ViTs) [18] and is shown in Figure 1. An input image is split into non-overlapping patches. After passing these patches through the encoder and adding the positional information, the semantic content of each patch and its position is encoded in the token x_{iA}. In the following the indices $i, j, k = 1...D$ are used to denote the token vector's elements, indices $A, B, C = 1...N$ are used to enumerate the patches and their corresponding tokens. It is helpful to think about each image patch as a physical particle, which has a complicated internal state described by a D-dimensional vector $\mathbf{x}_A$. This internal state describes the identity of the particle (representing the pixels of each patch), and the particle's positional embedding (the patch's

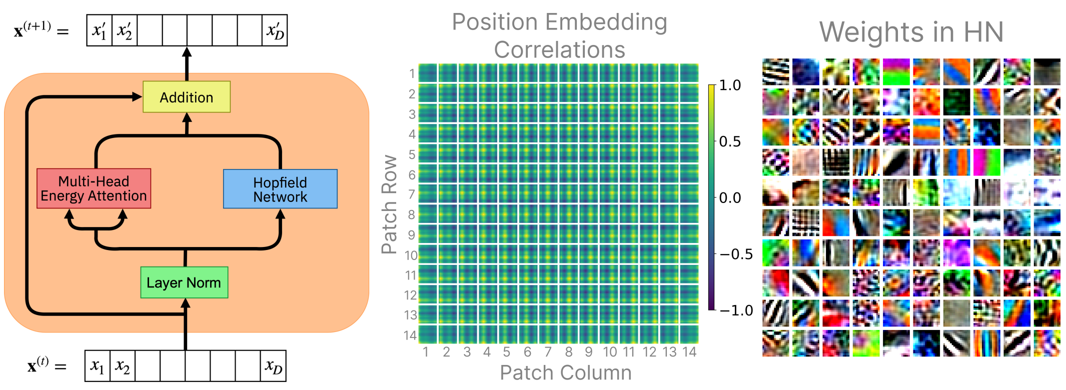

Figure 2: **Left**: Inside the ET block. The input token $\mathbf{x}^{(t)}$ passes through a sequence of operations and gets updated to produce the output token $\mathbf{x}^{(t+1)}$. The operations inside the ET block are carefully engineered so that the entire network has a global energy function, which decreases with time and is bounded from below. In contrast to conventional transformers, the ET-based analogs of the attention module and the feed-forward MLP module are applied in parallel as opposed to consecutively. **Center**: The cosine similarity between the learned position embedding of each patch and every other patch. In each cell, the brightest patch indicates the cell of consideration. **Right**: 100 selected memories stored in the HN memory matrix, visualized by the decoder as 16x16 RGB image patches. This visualization is unique to our model, as traditional Transformers cannot guarantee image representations in the learned weights.

location within the image). The ET block is described by a continuous time differential equation, which describes interactions between these particles. Initially, at $t = 1$ the network is given a set containing two groups of particles corresponding to open and masked patches. The "open" particles know their identity and location in the image. The "masked" particles only know where in the image they are located, but are not provided the information about what image patch they represent. The goal of ET's non-linear dynamics is to allow the masked particles to find an identity consistent with their locations and the identities of open particles. This dynamical evolution is designed so that it minimizes a global energy function, and is guaranteed to arrive at a fixed point attractor state. The identities of the masked particles are considered to be revealed when the dynamical trajectory reaches the fixed point. Thus, the central question is: how can we design the energy function that accurately captures the task that the Energy Transformer needs to solve?

The masked particles' search for identity is guided by two pieces of information: identities of the open particles, and the general knowledge about what patches are in principle possible in the space of all possible images. These two pieces of information are described by two contributions to the ET's energy function: the energy based attention and the Hopfield Network, respectively, for reasons that will become clear in the next sections. Below we define each element of the ET block in the order they appear in Figure 2.

Layer Norm

Each token, or a particle, is represented by a vector $\mathbf{x} \in \mathbb{R}^D$. At the same time, most of the operations inside the ET block are defined using a layer-normalized token representation

$$g_i = \gamma \frac{x_i - \bar{x}}{\sqrt{\frac{1}{D}\sum\limits_j \left(x_j - \bar{x}\right)^2 + \varepsilon}} + \delta_i, \quad \text{where} \quad \bar{x} = \frac{1}{D}\sum_{k=1}^{D} x_k \tag{1}$$

The scalar γ and the vector elements δ_i are learnable parameters, ε is a small regularization constant. Importantly, this operation can be viewed as an activation function for the neurons and can be defined as a partial derivative of the Lagrangian function (see [19, 20, 21] for the discussion of this property)

$$L = D\gamma \sqrt{\frac{1}{D}\sum_j \left(x_j - \bar{x}\right)^2 + \varepsilon} + \sum_j \delta_j x_j, \quad \text{so that} \quad g_i = \frac{\partial L}{\partial x_i} \tag{2}$$

Multi-Head Energy Attention

The first contribution to the ET's energy function is responsible for exchanging information between the particles (tokens). Similarly to the conventional attention mechanism, each token generates a pair of queries and keys (ET does not have a separate value matrix; instead the value matrix is a function of keys and queries). The goal of the energy based attention is to evolve the tokens in such a way that the keys of the open patches are aligned with the queries of the masked patches in the internal space of the attention operation. Below we use index $\alpha = 1...Y$ to denote elements of this internal space, and index $h = 1...H$ to denote different heads of this operation. With these notations the energy-based attention operation is described by the following energy function:

$$E^{\text{ATT}} = -\frac{1}{\beta}\sum_{h=1}^{H}\sum_{C=1}^{N}\log\left(\sum_{B\neq C}\exp\left(\beta A_{hBC}\right)\right) \tag{3}$$

where the attention matrix A_{hBC} is computed from query and key tensors as follows:

$$\begin{aligned} A_{hBC} &= \sum_{\alpha} K_{\alpha hB}\; Q_{\alpha hC}, && \mathbf{A} \in \mathbb{R}^{H\times N\times N} \\ K_{\alpha hB} &= \sum_{j} W^{K}_{\alpha hj}\; g_{jB}, && \mathbf{K} \in \mathbb{R}^{Y\times H\times N} \\ Q_{\alpha hC} &= \sum_{j} W^{Q}_{\alpha hj}\; g_{jC}, && \mathbf{Q} \in \mathbb{R}^{Y\times H\times N} \end{aligned} \tag{4}$$

and the tensors $\mathbf{W}^K \in \mathbb{R}^{Y\times H\times D}$ and $\mathbf{W}^Q \in \mathbb{R}^{Y\times H\times D}$ are learnable parameters.

From the computational perspective each patch generates two representations: query (given the position of the patch and its current content, where in the image should it look for the prompts on how to evolve in time?), and key (given the current content of the patch and its position, what should be the contents of the patches that attend to it?). The log-sum energy function (3) is minimal when for every patch in the image its queries are aligned with the keys of a small number of other patches connected by the attention map. Different heads (index h) contribute to the energy additively.

Hopfield Network Module

The next step of the ET block, which we call the Hopfield Network (HN), is responsible for ensuring that the token representations are consistent with what one expects to see in realistic images. The energy of this sub-block is defined as:

$$E^{\text{HN}} = -\sum_{B=1}^{N}\sum_{\mu=1}^{K} G\Big(\sum_{j=1}^{D}\xi_{\mu j}\; g_{jB}\Big), \qquad \xi \in \mathbb{R}^{K\times D} \tag{5}$$

where $\xi_{\mu j}$ is a set of learnable weights (memories in the Hopfield Network), and $G(\cdot)$ is an integral of the activation function $r(\cdot)$, so that $G(\cdot)' = r(\cdot)$. Depending on the choice of the activation function this step can be viewed either as a classical continuous Hopfield Network [8] if the activation function grows slowly (e.g., $r(\cdot)$ =ReLU), or as a modern continuous Hopfield Network [10, 12, 19] if the activation function is sharply peaked around the memories (e.g., $r(\cdot) =$ power, or softmax). The HN sub-block is analogous to the feed-forward MLP step in the conventional transformer block but requires that the weights of the projection from the token space to the hidden neurons' space to be the same (transposed matrix) as the weights of the subsequent projection from the hidden space to the token space. Thus, the HN module here is an MLP with shared weights that is *applied recurrently*. The energy contribution of this block is low when the token representations are aligned with some rows of the matrix ξ, which represent memories, and high otherwise.

Dynamics of Token Updates

The inference pass of the ET network is described by the continuous time differential equation, which minimizes the sum of the two energies described above

$$\tau\frac{dx_{iA}}{dt} = -\frac{\partial E}{\partial g_{iA}}, \quad \text{where} \quad E = E^{\text{ATT}} + E^{\text{HN}} \tag{6}$$

Here x_{iA} is the token representation (input and output from the ET block), and g_{iA} is its layer-normalized version. The first energy is low when each patch's queries are aligned with the keys of its neighbors. The second energy is low when each patch has content consistent with the general expectations about what an image patch should look like (memory slots of the matrix ξ). The dynamical system (6) finds a trade-off between these two desirable properties of each token's representation. For numerical evaluations, equation (6) is discretized in time.

To demonstrate that the dynamical system (6) minimizes the energy, consider the temporal derivative

$$\frac{dE}{dt} = \sum_{i,j,A} \frac{\partial E}{\partial g_{iA}} \frac{\partial g_{iA}}{\partial x_{jA}} \frac{dx_{jA}}{dt} = -\frac{1}{\tau} \sum_{i,j,A} \frac{\partial E}{\partial g_{iA}} M^A_{ij} \frac{\partial E}{\partial g_{jA}} \leq 0 \tag{7}$$

The last inequality sign holds if the symmetric part of the matrix

$$M^A_{ij} = \frac{\partial g_{iA}}{\partial x_{jA}} = \frac{\partial^2 L}{\partial x_{iA} \partial x_{jA}} \tag{8}$$

is positive semi-definite (for each value of index A). The Lagrangian (2) satisfies this condition.

Relationship to Modern Hopfield Networks and Conventional Attention

One of the theoretical contributions of our work is the design of the energy attention mechanism and the corresponding energy function (3). Although heavily inspired by prior work on Modern Hopfield Networks, our approach is fundamentally different from it. Our energy function (3) may look somewhat similar to the energy function of a continuous Hopfield Network with the $\mathrm{softmax}$ activation function. The main difference, however, is that in order to use Modern Hopfield Networks recurrently (as opposed to applying their update rule only once) the keys must be constant parameters (called memories in the Hopfield language). In contrast, in our energy attention network the keys are *dynamical variables* that evolve in time with the queries.

To emphasize this further, it is instructive to write explicitly the ET attention contribution to the update dynamics (6). It is given by (for clarity, assume only one head of attention):

$$-\frac{\partial E^{\text{ATT}}}{\partial g_{iA}} = \sum_{C \neq A} \sum_{\alpha} W^Q_{\alpha i}\, K_{\alpha C}\, \underset{C}{\text{softmax}}\Big(\beta \sum_{\gamma} K_{\gamma C}\, Q_{\gamma A}\Big) + W^K_{\alpha i}\, Q_{\alpha C}\, \underset{A}{\text{softmax}}\Big(\beta \sum_{\gamma} K_{\gamma A}\, Q_{\gamma C}\Big)$$

In both terms the $\mathrm{softmax}$ normalization is done over the token index of the keys, which is indicated by the subscript in the equation. The first term in this formula is the conventional attention mechanism [5] with the value matrix equal to $\mathbf{V} = (\mathbf{W}^Q)^T \mathbf{K} = \sum_\alpha W^Q_{\alpha i} K_{\alpha C}$. The second term is the brand new contribution that is missing in the original attention mechanism. The presence of this second term is crucial to make sure that the dynamical system (6) minimizes the energy function if applied recurrently. This second term is the main difference of our approach compared to the Modern Hopfield Networks. The same difference applies compared to the other recent proposals [15].

Lastly, we want to emphasize that our ET block contains two different kinds of Hopfield Networks acting in parallel, see Figure 2. The first one is the energy attention module, which is inspired by, but not identical to, Modern Hopfield Networks. The second one is the "Hopfield Network" module, which can be either a classical or Modern Hopfield Network. These two should not be confused.

For completeness, the contribution of the "Hopfield Network" module to the update equation (6) can be written as

$$-\frac{\partial E^{\text{HN}}}{\partial g_{iA}} = \sum_{\mu=1}^{K} \xi_{\mu i} G'\Big(\sum_{j=1}^{D} \xi_{\mu j} g_{jA}\Big) \tag{9}$$

which is applied to every token individually (there no mixing of different tokens).

3 Qualitative Inspection of the ET framework on ImageNet

We trained[*] the ET network on the masked image completion task using ImageNet-1k dataset [22]. Each image was broken into non-overlapping patches of 16x16 RGB pixels, which were projected

[*]The code is available: https://github.com/bhoov/energy-transformer-jax.

with a single affine encoder into the token space. Half of these tokens were "masked", by replacing them with a learnable MASK token. A distinct learnable position encoding vector was added to each token. Our ET block then processes all tokens recurrently for T steps. The token representations after T steps are passed to a simple linear decoder (consisting of a layer norm and an affine transformation). The loss function is the standard MSE loss on the occluded patches. See more details on the implementation and the hyperparameters in Appendix B.

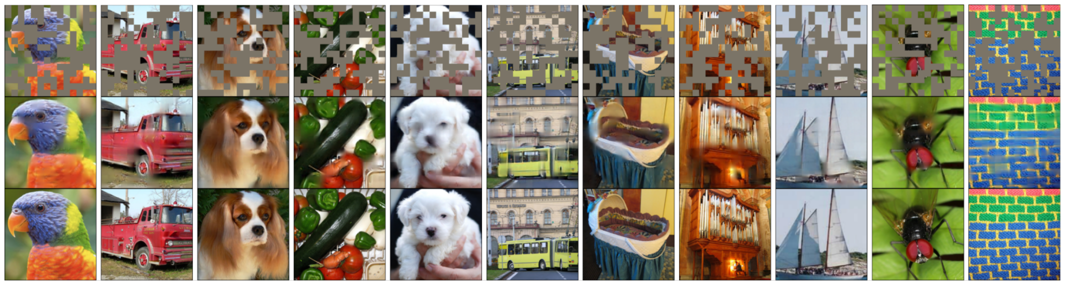

Figure 3: Reconstruction examples of our Energy Transformer using images from the ImageNet-1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: output reconstructions after 12 time steps. *Bottom row*: original images.

Examples of occluded/reconstructed images (unseen during training) are shown in Figure 3. In general, our model learns to perform the task very well, capturing the texture in dog fur (column 3) and understanding meaningful boundaries of objects. However, we observe that our single ET block struggles to understand some global structure, e.g., failing to capture both eyes of the white dog (column 5) and completing irregular brick patterns in the name of extending the un-occluded borders (last column). We additionally inspect the positional encoding vectors associated with every token, Figure 2, where the model learns a locality structure in the image plane that is very similar to the original ViT [18]. The positional embedding of each image patch has learned higher similarity values for neighboring tokens than for distant tokens.

Our network is unique compared to standard ViTs in that the iterative dynamics only *move* tokens around in the same space from which the final fixed point representation can be decoded back into the image plane. This functionality makes it possible to visualize essentially any *token representation*, *weight*, or *gradient of the energy* directly in the image plane. This feature is highly desirable from the perspective of interpretability, since it makes it possible to track the updates performed by the network directly in the image plane as the computation unfolds in time. In Figure 2 this functionality is used for inspecting the learned weights of the HN module directly in the image plane. According to our theory, these weights should represent basis vectors in the space of all possible image patches. These learned representations look qualitatively similar to the representations typically found in networks trained on image datasets, e.g., [23].

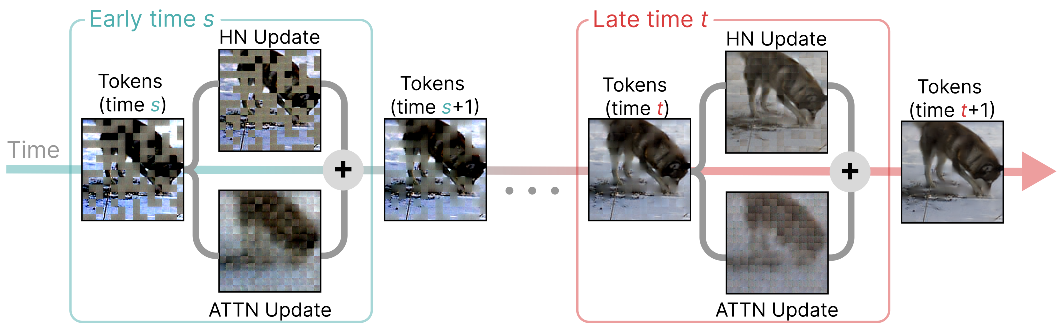

Figure 4: Token representations and gradients are visualized using the decoder at different times during the dynamics. The Energy Attention (ATTN) block contributes general structure information to the masked patches at *earlier* time steps, whereas the Hopfield Network (HN) significantly sharpens the quality of the masked patches at *later* time steps.

Table 1: Performance on Yelp, Amazon, T-Finance, and T-Social datasets with different training ratios. Following [24], mean and standard deviation over 5 runs with different train/dev/test split are reported (standard deviations are only included if they are available in the prior work). Best results are in **bold**. Our model is state of the art or near state of the art on every category.

	Datasets	Split	GraphConsis	CAREGNN	PC-GNN	BWGNN	MLP	GT	ET (Ours)
Macro-F1	Yelp	1%	$56.8_{\pm 2.8}$	$62.1_{\pm 1.3}$	$59.8_{\pm 1.4}$	$61.1_{\pm 0.4}$	$53.9_{\pm 0.2}$	$61.7_{\pm 0.4}$	$\mathbf{63.0_{\pm 0.6}}$
		40%	$58.7_{\pm 2.0}$	$63.3_{\pm 0.9}$	$63.0_{\pm 2.3}$	$71.0_{\pm 0.9}$	$57.5_{\pm 0.8}$	$68.7_{\pm 0.4}$	$\mathbf{71.5_{\pm 0.1}}$
	Amazon	1%	$68.5_{\pm 3.4}$	$68.7_{\pm 1.6}$	$79.8_{\pm 5.6}$	$\mathbf{90.9_{\pm 0.7}}$	$74.6_{\pm 1.2}$	$88.6_{\pm 0.5}$	$89.3_{\pm 0.7}$
		40%	$75.1_{\pm 3.2}$	$86.3_{\pm 1.7}$	$89.5_{\pm 0.7}$	$92.2_{\pm 0.4}$	$79.1_{\pm 1.2}$	$91.7_{\pm 0.8}$	$\mathbf{92.8_{\pm 0.3}}$
	T-Finance	1%	71.7	73.3	62.0	84.8	61.0	81.5	$\mathbf{85.1_{\pm 1.0}}$
		40%	73.4	77.5	63.1	86.8	70.5	83.6	$\mathbf{88.2_{\pm 1.0}}$
	T-Social	1%	52.4	55.8	51.1	75.9	50.0	64.3	$\mathbf{79.1_{\pm 0.7}}$
		40%	56.5	56.2	52.1	**83.9**	50.3	68.2	$83.5_{\pm 0.4}$
AUC	Yelp	1%	$66.4_{\pm 3.4}$	$75.0_{\pm 3.8}$	$\mathbf{75.4_{\pm 0.9}}$	$72.0_{\pm 0.5}$	$59.8_{\pm 0.4}$	$72.5_{\pm 0.6}$	$73.2_{\pm 0.8}$
		40%	$69.8_{\pm 3.0}$	$76.1_{\pm 2.9}$	$79.8_{\pm 0.1}$	$84.0_{\pm 0.9}$	$66.5_{\pm 1.0}$	$81.9_{\pm 0.5}$	$\mathbf{84.9_{\pm 0.3}}$
	Amazon	1%	$74.1_{\pm 3.5}$	$88.6_{\pm 3.5}$	$90.4_{\pm 2.0}$	$89.4_{\pm 0.3}$	$83.6_{\pm 1.7}$	$89.0_{\pm 1.2}$	$\mathbf{91.9_{\pm 1.0}}$
		40%	$87.4_{\pm 3.3}$	$90.5_{\pm 1.6}$	$95.8_{\pm 0.1}$	$\mathbf{98.0_{\pm 0.4}}$	$89.8_{\pm 1.0}$	$95.4_{\pm 0.6}$	$97.3_{\pm 0.4}$
	T-Finance	1%	90.2	90.5	90.7	91.1	82.9	90.0	$\mathbf{92.8_{\pm 1.1}}$
		40%	91.4	92.1	91.2	94.3	87.1	88.2	$\mathbf{95.0_{\pm 3.0}}$
	T-Social	1%	65.2	71.2	59.8	88.0	56.3	81.4	$\mathbf{91.9_{\pm 0.6}}$
		40%	71.2	71.8	68.4	**95.2**	56.9	82.5	$93.9_{\pm 0.2}$

We additionally visualize the gradients of the energy function (which are equal to the token updates, see Equation 6) of both ATTN block and the HN block, see Figure 4. Early in time, almost all signal to the masked tokens comes from the ATTN block, which routes information from the open patches to the masked ones; no meaningful signal comes from the HN block to the masked patch dynamics. Later in time we observe a different phenomenon: almost all signal to masked tokens comes from the HN module while ATTN contributes a blurry and uninformative signal. Thus, the attention module is crucial early in the network dynamics, feeding signal to masked patches from the visible patches, whereas the HN is crucial later in the dynamics as the model approaches the final reconstruction, sharpening the masked patches. All the qualitative findings presented in this section are in accord with the core computational strategy of the ET block as it was designed theoretically in section 2.

4 Graph Anomaly Detection

Having gained empirical insights in the image domain, we turn to quantitatively evaluating ET's performance on the graph anomaly detection problem[*], a task with plenty of strong and recently published baselines. Graph Convolutional Networks (GCN) [25] have been widely used for this task due to their capability of learning high level representations of graph structures and node attributes [26, 27]. However, vanilla GCNs suffer from the over-smoothing problem [28]. In each layer of the forward pass, the outlier node aggregates information from its neighbors. This averaging makes the features of anomalies less distinguishable from the features of benign or normal nodes. Our approach does not suffer from this problem, since the routing of the information between the nodes is done through the energy based attention, and the information aggregation is based on the attention scores.

For anomaly detection on graphs in the ET framework, consider an undirected graph with N nodes. Every node has a vector of raw attributes $\mathbf{y}_A \in \mathbb{R}^F$, where F is the number of node's features. Every node also has a binary label l_A indicating whether the node is benign or anomalous. We focus on node anomaly and assume that all edges are trusted. The task is to predict the label of a node given the graph structure and the node's features. Since there are far more benign nodes in the graph than anomalous ones, anomaly detection can be regarded as an imbalanced node classification task.

First, similarly to images, the feature vectors for every node are converted to a token representation using a linear embedding $\mathbf{E}$ and adding a learnable positional embedding λ_A

$$\mathbf{x}_A^{t=1} = \mathbf{E}\, \mathbf{y}_A + \lambda_A \tag{10}$$

where the superscript $t = 1$ indicates the time of the update of the ET dynamics. This token representation is iterated through the ET block for T iterations. When the retrieval dynamics becomes

[*]The code is available: https://github.com/zhuergou/Energy-Transformer-for-Graph-Anomaly-Detection/.

Table 2: Graph classification performance on eight datasets of TUDataset. Following [36], mean and standard deviation obtained from 100 runs of 10-fold cross validation are reported. For baselines standard deviations are only included if they are available in prior work. If the entry is unavailable in prior literature it is denoted by '-'; best results are in **bold**. The performance difference between non-baseline approaches (including ours) and the baselines (specified by their gray cell) is indicated by ▼(decrease), ▲(increase), and ▼(no change within the error bars) along with the value.

Method	**Dataset**							
	PROTEINS	NCI1	NCI109	DD	ENZYMES	MUTAG	MUTAGENICITY	FRANKENSTEIN
WKPI (kmeans)	$78.5_{\pm 0.4}$ ▼(6.4)	$87.5_{\pm 0.5}$	$85.9_{\pm 0.4}$ ▼(1.5)	$82.0_{\pm 0.5}$ ▼(13.7)	-	$85.8_{\pm 2.5}$ ▼(14.2)	-	-
WKPI (kcenters)	$75.2_{\pm 0.4}$ ▼(9.7)	$84.5_{\pm 0.5}$ ▼(3.0)	$87.4_{\pm 0.3}$	$80.3_{\pm 0.4}$ ▼(15.4)	-	$88.3_{\pm 2.6}$ ▼(11.7)	-	-
Spec-GN	-	$84.8_{\pm 1.6}$ ▼(2.7)	$83.6_{\pm 0.8}$ ▼(3.8)	-	$72.5_{\pm 5.8}$ ▼(5.9)	-	-	-
Norm-GN	-	$84.9_{\pm 1.7}$ ▼(2.6)	$83.5_{\pm 1.3}$ ▼(3.9)	-	$73.3_{\pm 8.0}$ ▼(5.1)	-	-	-
GWL-WL	$75.8_{\pm 0.6}$ ▼(9.1)	-	-	-	$71.3_{\pm 1.1}$ ▼(7.1)	-	-	$78.9_{\pm 0.3}$
HGP-SL	$84.9_{\pm 1.6}$	$78.5_{\pm 0.8}$ ▼(9.1)	$80.7_{\pm 1.2}$ ▼(6.7)	$81.0_{\pm 1.3}$ ▼(14.7)	$68.8_{\pm 2.1}$ ▼(9.6)	-	$82.2_{\pm 0.6}$	-
DSGCN	$77.3_{\pm 0.4}$ ▼(7.6)	-	-	-	$78.4_{\pm 0.6}$	-	-	-
U2GNN	$80.0_{\pm 3.2}$ ▼(4.9)	-	-	$95.7_{\pm 1.9}$	-	$88.5_{\pm 7.1}$ ▼(11.5)	-	-
NDP	$73.4_{\pm 3.1}$ ▼(11.5)	$74.2_{\pm 1.7}$ ▼(13.3)	-	$72.8_{\pm 5.4}$ ▼(22.9)	$44.5_{\pm 7.4}$ ▼(34.9)	$87.9_{\pm 5.7}$ ▼(12.1)	$77.9_{\pm 1.4}$ ▼(4.3)	-
ASAP	$74.2_{\pm 0.8}$ ▼(10.7)	$71.5_{\pm 0.4}$ ▼(16.0)	$70.1_{\pm 0.6}$ ▼(17.3)	$76.9_{\pm 0.7}$ ▼(18.8)	-	-	-	$66.3_{\pm 0.5}$ ▼(12.6)
EvoG	-	-	-	-	55.7 ▼(22.7)	**100.0**	-	-
ET (Ours)	$\mathbf{90.3}_{\pm \mathbf{0.7}}$ ▲(5.4)	$\mathbf{90.1}_{\pm \mathbf{0.1}}$ ▲(2.6)	$\mathbf{90.5}_{\pm \mathbf{0.1}}$ ▲(3.1)	$\mathbf{95.9}_{\pm \mathbf{0.8}}$ ▲(0.2)	**99.8** ▲(21.4)	$96.6_{\pm 0.2}$ ▼(3.4)	$\mathbf{98.7}_{\pm \mathbf{0.1}}$ ▲(16.5)	$\mathbf{99.8}_{\pm \mathbf{0.1}}$ ▲(20.9)

stable, we have the final representation for each node $\mathbf{x}_A^{t=T}$ (or more precisely $\mathbf{g}_A^{t=T}$, since the outputs are additionally passed through a layer norm operation after the final ET update). This output is concatenated with the initial (layer normalized) token to form the final output

$$\mathbf{g}_A^{\text{final}} = \mathbf{g}_A^{t=1} \mid\mid \mathbf{g}_A^{t=T} \tag{11}$$

where $||$ denotes concatenation. Following [24], the node representation $\mathbf{g}_A^{\text{final}}$ is fed into an MLP with a sigmoid activation function to compute the anomaly probabilities p_A. The weighted cross entropy

$$\text{Loss} = \sum_A \Big[\omega\; l_A \log(p_A) + (1 - l_A) \log(1 - p_A)\Big] \tag{12}$$

is used to train the network. Above, ω is the ratio of the benign labels (l_A = 0) to anomalous ones (l_A = 1). Attention is restricted to 1-hop neighbors of the target node.

4.1 Experimental Evaluation

Four datasets are used for the experiments. YelpChi dataset [29] aims at opinion spam detection in Yelp reviews. Amazon dataset is used to detect anomalous users under the Musical Instrument Category on *amazon.com* [30]. T-Finance and T-Social datasets [24] are used for anomalous account detection in transactions and social networks, respectively. For these four datasets, the graph is treated as a homogeneous graph (i.e., all the edges are of the same type), and a feature vector is associated with each node. The task is to predict the label (anomaly status) of the nodes. For each dataset, either 1% or 40% of the nodes are used for training, and the remaining 99% or 60% are split $1:2$ into validation and testing sets, see Appendix C for details.

We compare with state-of-the-art approaches for graph anomaly detection, which include GraphConsis [31], CAREGNN [32], PC-GNN [33] and BWGNN [24]. Additionally, multi-layer perceptrons (MLP) and Graph Transformer (GT) [34] are included in the baselines for completeness. Following previous work, macro-F1 score (unweighted mean of F1 score) and the Area Under the Curve (AUC) are used as the evaluation metrics on the test datasets [35]. See Appendix C for more details on training protocols and the hyperparameters choices. The results are reported in Table 1. Our ET network demonstrates very strong results across all the datasets.

5 Graph Classification with ET

To fully explore the efficacy of ET, we further evaluate its performance on the graph classification problem[*]. Unlike the anomaly detection task, where we predict a label for every node, in the graph classification task, a single label is predicted for the entire graph.

Consider a graph G, where each node has a raw feature vector $\mathbf{y}_A \in \mathbb{R}^F$ with F feature-dimension. Each vector is projected to the token space yielding $\mathbf{x}_A \in \mathbb{R}^D$, where D is the token dimension. A learnable CLS token $\mathbf{x}_{\text{CLS}}$ is concatenated to the set of tokens resulting in the token matrix $\mathbf{X} \in \mathbb{R}^{(N+1)\times D}$, and a learnable linear projection of the top k smallest eigen-vectors of the normalized Laplacian matrix, following [37], is added to $\mathbf{X}$. Graph structural information is provided to ET within the attention operation. We also use a stacked version of the Energy Transformer consisting of S ET blocks, where each block has the same number of temporal unfolding steps T and its own LayerNorm. The final representation of the CLS token $\mathbf{x}_{\text{CLS}}^{\ell=S,\, t=T}$ is projected via a linear embedding into the predictor space. A softmax over the number of classes is applied to this predictor representation, and the cross-entropy loss is used to train the network. See Appendix D for full details.

5.1 Experimental Evaluation

Eight graph datasets from the TUDataset [36] collection are used for experimentation. NCI1, NCI109, MUTAG, MUTAGENICITY, and FRANKENSTEIN are a common class of graph datasets consisting of small molecules with class labels representing toxicity or biological activity determined in drug discovery projects [36]. Meanwhile, DD, ENZYMES, and PROTEINS represent macromolecules. The task for both DD and PROTEINS is to classify whether a protein is an enzyme. Lastly, for ENZYMES, the task is to assign enzymes to one of the six classes, which reflect the catalyzed chemical reaction [36]. See Table 8 for more details of the datasets.

We compare ET with the current state-of-the-art approaches for the mentioned datasets, which include WKPI-kmeans [38], WKPI-kcenters [38], DSGCN [39], HGP-SL [40], U2GNN [41], and EvoG [42]. Additionally, approaches [39, 43, 44, 45, 46], which are close to the baselines, are included to further contrast the performance of our model. Following the 10-fold cross validation process delineated in [36], accuracy score is used as the evaluation metric and reported in Table 2. In general, we have observed that the modified ET demonstrates strong performance across the eight datasets. With the exception of MUTAG, ET beats other methods on all the baselines by a substantial margin.

6 Discussion and Conclusions

A lot of recent research has been dedicated to understanding the striking analogy between Hopfield Networks and the attention mechanism in transformers. At a high level, the main message of our work is that the *entire* transformer block (including feed-forward MLP, layer normalization, and residual connections) can be viewed as a single large Hopfield Network, not just attention alone. At a deeper level, we use recent advances in the field of Hopfield Networks to design a novel energy function that is tailored for dynamical information routing between the tokens; and representation of a large number of relationships between those tokens. We have tested the ET network qualitatively on the image completion task, and quantitatively on node anomaly detection and graphs classification. The qualitative investigation reveals the perfect alignment between the theoretical design principles of our network and its empirical computation. The quantitative evaluation demonstrates strong results, which stand in line or exceed the methods recently developed specifically for these tasks. We believe that the proposed network will be useful for other tasks and domains (e.g., NLP, audio, and video), which serve as future directions for a comprehensive investigation.

There are two metrics describing computational costs of our architecture: memory footprint of the model and the number of flops. In terms of the memory footprint our model wins (is smaller) compared to feedforward transformers with independent weights and even ALBERT-style shared-weight transformers (see Appendix G and Table 15). Regarding the number of flops, our model has the same parametric scaling as the conventional transformers (quadratic in the number of tokens), but the energy attention has a constant factor of two more flops (due to the second term in the attention-induced updates of the tokens).

Finally, we have developed an auto-grad framework called HAMUX [47] that makes building flexible energy-based architectures, including ET, convenient and efficient (see HAMUX).

[*]The code is available: https://github.com/Lemon-cmd/Energy-Transformer-For-Graph.

References

[1] Ofir Press, Noah A Smith, and Omer Levy. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*, 2019.

[2] Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

[3] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR, 2019.

[4] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021.

[7] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[8] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[9] Dmitry Krotov. A new frontier for hopfield networks. *Nature Reviews Physics*, pages 1–2, 2023.

[10] Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[11] Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Upgang, and Franck Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, May 2017.

[12] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[13] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

[14] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[15] Yongyi Yang, Zengfeng Huang, and David Wipf. Transformers from an optimization perspective. *arXiv preprint arXiv:2205.13891*, 2022.

[16] Ying Sun, Prabhu Babu, and Daniel P Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2016.

[17] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[19] Dmitry Krotov and John J. Hopfield. Large associative memory problem in neurobiology and machine learning. In *International Conference on Learning Representations*, 2021.

[20] Fei Tang and Michael Kopp. A remark on a paper of krotov and hopfield [arxiv: 2008.06996]. *arXiv preprint arXiv:2105.15034*, 2021.

[21] Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[24] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. *arXiv preprint arXiv:2205.15508*, 2022.

[25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[26] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM, 2019.

[27] Zhen Peng, Minnan Luo, Jundong Li, Luguo Xue, and Qinghua Zheng. A deep multi-view framework for anomaly detection on attributed networks. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[28] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[29] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 985–994, 2015.

[30] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908, 2013.

[31] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1569–1572, 2020.

[32] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 315–324, 2020.

[33] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pages 3168–3177, 2021.

[34] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[35] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[36] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020.

[37] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022.

[38] Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[39] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Bridging the gap between spectral and spatial domains in graph neural networks. *CoRR*, abs/2003.11702, 2020.

[40] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Zhao Li, Chengwei Yao, Dai Huifen, Zhi Yu, and Can Wang. Hierarchical multi-view graph pooling with structure learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[41] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 193–196, New York, NY, USA, 2022. Association for Computing Machinery.

[42] Miguel Domingue, Rohan Dhamdhere, Naga Durga Harish Kanamarlapudi, Sunand Raghupathi, and Raymond Ptucha. Evolution of graph classifiers. In *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5, 2019.

[43] Mingqi Yang, Yanming Shen, Rui Li, Heng Qi, Qiang Zhang, and Baocai Yin. A new perspective on the effects of spectrum in graph neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25261–25279. PMLR, 17–23 Jul 2022.

[44] Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3756–3762. AAAI Press, 2015.

[45] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[46] Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. ASAP: Adaptive structure aware pooling for learning hierarchical graph representations. *arXiv preprint arXiv:1911.07979*, 2019.

[47] Benjamin Hoover, Duen Horng Chau, Hendrik Strobelt, and Dmitry Krotov. A universal abstraction for hierarchical hopfield networks. In *The Symbiosis of Deep Learning and Differential Equations II*.

[48] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[49] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

[50] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020.

[51] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. Error-bounded graph anomaly loss for gnns. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1873–1882, 2020.

[52] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. Inductive anomaly detection on attributed networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1288–1294, 2021.

[53] Yuchen Liang, Dmitry Krotov, and Mohammed J Zaki. Modern hopfield networks for graph embedding. *Frontiers in big Data*, 5, 2022.

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[55] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *CoRR*, abs/2106.05234, 2021.

[56] Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 655–665, New York, NY, USA, 2022. Association for Computing Machinery.

[57] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28798–28810. Curran Associates, Inc., 2021.

[58] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. *arXiv preprint arXiv:2303.06147*, 2023.

[59] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[60] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[61] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[62] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[63] Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. Gradient-centralization: A new optimization technique for deep neural networks. 2020.

[64] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[65] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[66] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

A Notations Used in the Main Text and Appendices

Table 3 lists all the notations used in this paper.

Table 3: Notations used in the paper.

Notation	Description
F	dimension of node's feature space
D	dimension of token space
N	number of tokens
Y	number of hidden dimensions in the attention
K	number of hidden dimensions in the Hopfield Network module
T	number of recurrent time steps
H	number of heads
k_h	height of each image patch
k_w	width of each image patch
P	number of pixels per image patch ($3 \times k_h \times k_w$)
$\mathbf{y}_A$	input feature vector of node A
$\mathbf{x}_A$	vector representation of token A
x_{iA}	each element of vector representation of token A
$\mathbf{g}_A$	vector representation of token A after layernorm
g_{iA}	each element of vector representation of token A after layernorm
$\mathbf{K}$	key tensor
$\mathbf{Q}$	query tensor
$K_{\alpha hB}$	each element of the key tensor $\mathbf{K}$
$Q_{\alpha hC}$	each element of the query tensor $\mathbf{Q}$
A_{hBC}	each element of the attention tensor
l_A	label of node A on graph
$r(\cdot)$	activation function of the Hopfield Network module
$G(\cdot)$	energy contribution of the Hopfield Network module

B Details of Training on ImageNet

We trained the ET network on a masked-image completion task on the ImageNet-1k (IN1K) dataset. We treat all images in IN1K as images of shape 224×224 that are normalized according to standard IN1K practices (mean 0, variance 1 on the channel dimension) and use data augmentations provided by the popular `timm` library [48] (See Table 4). Following the conventional ViT pipeline [18], we split these images into non-overlapping patches of 16x16 RGB pixels which are then projected with a single affine encoder into the token dimension D for a total of 196 encoded tokens per image. We proceed to randomly and uniformly assign 100 of these tokens as "occluded" which are the only tokens considered by the loss function. "Occluded" tokens are designated as follows: of the 100 tokens, 90 tokens are replaced with a learnable MASK token of dimension D and 10 we leave untouched (which we find important for the HN to learn meaningful patch representations). To all tokens we then add a distinct learnable position bias.

These tokens are then passed to our Energy Transformer block which we recur for T steps (the "depth" of the model in conventional Transformers). At each step, the feedback signal (the sum of the energy gradients from our attention block and HN block) is subtracted from our original token representation with a scalar step size $\alpha = \frac{dt}{\tau}$ which we treat as a non-learnable hyperparameter in our experiments. The token representations after T steps are passed to a simple linear decoder (consisting of a layer norm and an affine transformation) to project our representations back into the image plane. We then use the standard MSE Loss between the original pixels and reconstructed pixels for only the 100 occluded patches. We allow self attention as in the following formula for the energy of multiheaded attention

$$E^{\text{ATT}} = \sum_h -\frac{1}{\beta} \sum_C \log \left(\sum_B \exp \left(\beta \sum_\alpha K_{\alpha hB}\, Q_{\alpha hC} \right) \right) \tag{13}$$

We give details of our architectural choices in Table 4. In the main paper we present our Energy Transformer with a configuration similar to the standard base Transformer configuration (e.g., token dimension 768, 12 heads each with $Y = 64$, softmax's $\beta = \frac{1}{\sqrt{Y}}, \dots$), with several considerations learned from the qualitative image evaluations:

- The $\frac{dt}{\tau}$ (step size) of 1 implicitly used in the traditional transformer noticeably degrades our ability to smoothly descend the energy function. We find that a step size of 0.1 provides a smoother descent down the energy function and benefits the image reconstruction quality.
- We observe that our MSE loss must include some subset of un-occluded patches in order for the HN to learn meaningful filters.
- Values of β in the energy attention that are too high prevent our model from training. This is possibly due to vanishing gradients in our attention operation from a `softmax` operation that is too spiky.
- Without gradient clipping, our model fails to train at the learning rates we tried higher than 1e-4. We observe that gradient clipping not only helps our model train faster at the trainable learning rates, it also allows us to train at higher learning rates.

Our architecture and experiments for the image reconstruction task were written in JAX [49] using Flax [50]. This engineering choice means that our architecture definitions are quite lightweight, as we can define the desired energy function of the ET and use JAX's autograd to automatically calculate the desired update.

Table 4: Hyperparameter, architecture, and data augmentation choices for ET model during ImageNet-1k masked training experiments. Data augmentations are listed as parameters passed to the equivalent `timm` dataloader functionality.

Training	
batch_size	768
epochs	100
lr	5e-4
warmup_epochs	2
start & end lr	5e-7
b1, b2 (ADAM)	0.9, 0.99
weight_decay	0.05
grad_clipping	1.

Architecture	
token_dim	768
num_heads	12
head_dim	64
β	1/8
train_betas	No
step size α	0.1
depth	12
hidden_dim HN	3072
bias in HN	None
bias in ATT	None
bias in LNORM	Yes

Data Augmentation	
random_erase	None
horizontal_flip	0.5
vertical_flip	0
color_jitter	0.4
scale	(0.08, 1)
ratio	(3/4, 4/3)
auto_augment	None

B.1 Exploring the Hopfield Memories

A distinctive aspect of our network is that any variable that has a vector index i of tokens can be mapped into the data domain by applying the decoder network to this variable. This makes it possible to inspect all the weights in the model. For instance, the concept of "memories" is crucial to understanding how Hopfield networks function. The memories within the HN module represent the building blocks of *all possible image patches* in our data domain, where an encoded image patch is a superposition of a subset of memories. The complete set of memory vectors from the HN module is shown in Figure 5. The same analysis can be applied to the weights of the ET-attention module. In Figure 6, we show all the weights from this module mapped into the image plane.

B.2 Positional Bias Correlations

The relationships between our position embeddings exhibit similar behavior to the position correlations of the original ViT in that they are highly susceptible to choices of the hyperparameters (Figure 10 of [18]). In particular, we consider the effect of weight decay and the β parameter that serves as the inverse temperature of the attention operation (see Equation 3). The lower the temperature (i.e., the higher the value of β), the spikier the softmax distribution. By using a lower β, we encourage the attention energy to distribute its positional embeddings across a wider range of patches in the model.

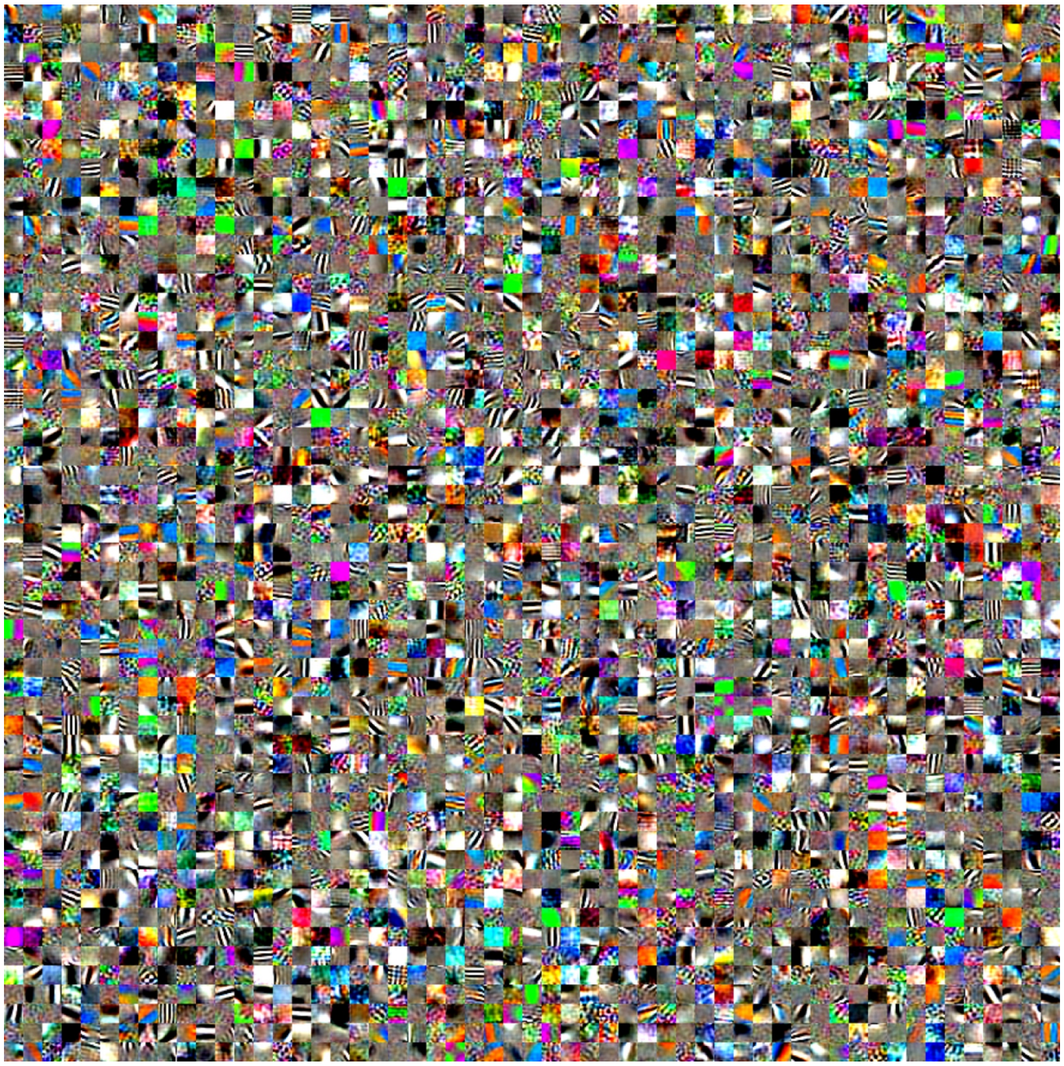

Figure 5: Visualizing a randomly selected 3025 patch memories of the 3072 learned by weight matrix in the Hopfield Network module (HN) of our model. These memories are vectors of the same dimensions D as the patch tokens, stored as rows in the weight matrix ξ. Each image patch is visualized using the model's trained decoder.

C Details of ET training on Anomaly Detection Task

Graph anomaly detection refers to the process of detecting outliers that deviate significantly from the majority of the samples. Neural network based methods are very popular due to their capability of learning sophisticated data representations. DOMINANT [26] utilizes an auto-encoder framework, using a GCN as an encoder and two decoders for structural reconstruction and attribute reconstruction. ALARM [27] aggregates the encoder information from multiple view of the node attributes. Another study [51], propose a novel loss function to train graph neural networks for anomaly-detectable node representations. In [52] generative adversarial learning is used to detect anomaly nodes where a novel layer is designed to learn the anomaly-aware node representation. Recently, [24] pointed out that anomalies can lead to the "rightshift" of the spectral energy distribution – the spectral energy concentrates more on the high frequencies. They designed a filter that can better handle this phenomenon. We propose a new anomaly detection model from the perspective of Associative Memory (pattern matching), which does not have the over-smoothing problem often faced by GCNs, and has better model interpretability (outliers should be far from the common pattern). We also notice

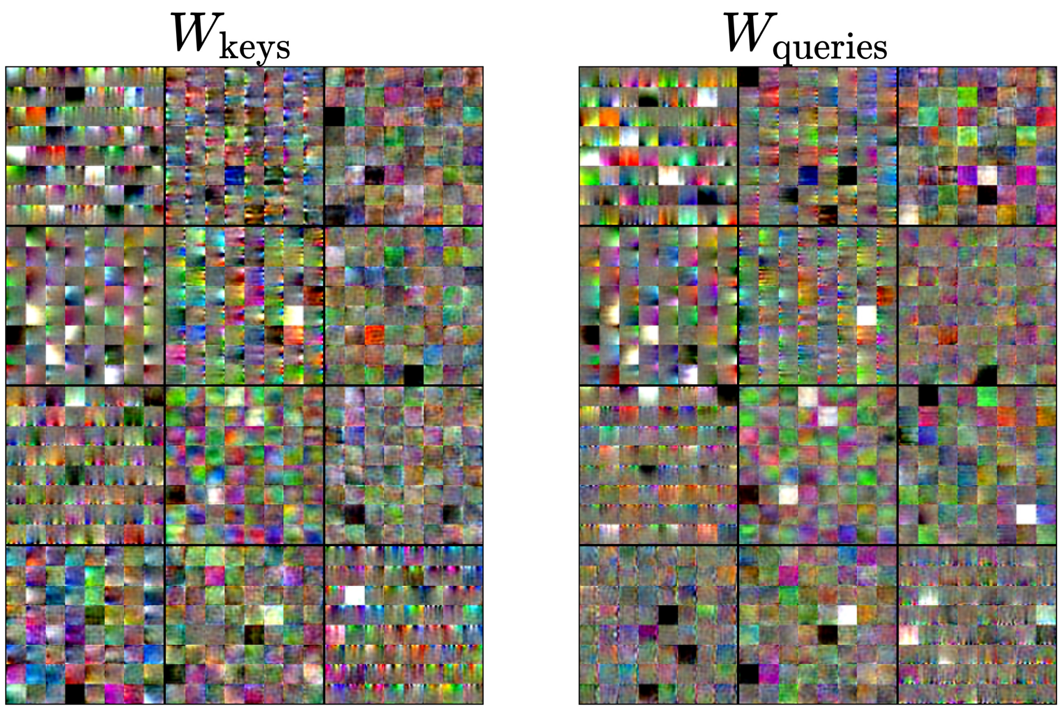

Figure 6: Visualizing the token dimension of the "key" and "query" matrices of the attention as image patches. Each head is represented as a cell on the 4×3 grid above. We use the trained decoder of our model to visualize each weight.

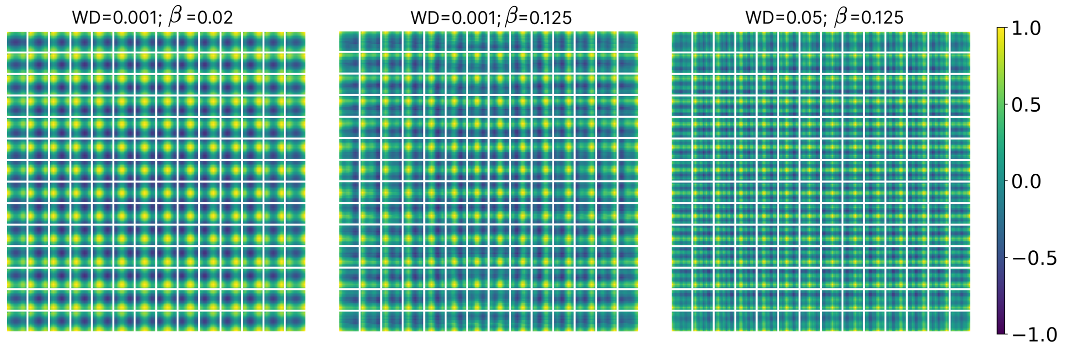

Figure 7: The cosine similarity between position biases of patches when the ET model is trained under different hyperparameter choices for β (inverse temperature of the attention energy) and weight decay. Our ET sees a trend where smoother correlations are observed with smaller β and weight decay.

that Modern Hopfield Networks have been used before for node classification, link prediction, and graph coarsening tasks [53].

C.1 Detailed Model Structure for the Graph Anomaly Detection

First, we compute the features that are passed to our energy-based transformer. Each node's features $\mathbf{y}_A \in R^F$ are mapped into the token space $\mathbf{x}_A \in R^D$, using a linear projection $\mathbf{E}$. Learnable positional embeddings λ_A are added to this token at $t = 1$,

$$\mathbf{x}_A^{t=1} = \mathbf{E}\mathbf{y}_A + \lambda_A \tag{14}$$

At each time step the input to the ET-block is layer normalized:

$$\mathbf{g}_A^t = \text{LayerNorm}(\mathbf{x}_A^t) \tag{15}$$

Let $\mathbf{W}^Q \in R^{Y\times H\times D}$ and $\mathbf{W}^K \in R^{Y\times H\times D}$ be the query and key weight matrices, respectively. Here Y is the projection dimension in the attention operation, H is the number of heads. We define

$$\begin{aligned} K_{\alpha hB} &= \sum_j W^K_{\alpha hj}\, g_{jB} \\ Q_{\alpha hC} &= \sum_j W^Q_{\alpha hj}\, g_{jC} \end{aligned} \tag{16}$$

If we let h indicate the index of the head, we have

$$\Delta x^t_{iA} = \sum_{C\in\mathcal{N}_A}\sum_{h,\alpha}\Big[W^Q_{\alpha hi}\, K_{\alpha hC}\, \omega_{CA} + W^K_{\alpha hi}\, Q_{\alpha hC}\, \omega_{AC}\Big] + \sum_\mu \xi_{\mu i}\, r\Big(\sum_j \xi_{\mu j} g_{jA}\Big) \tag{17}$$

where

$$\omega_{CA} = \underset{C}{\text{softmax}}\Big(\beta\sum_\gamma K_{\gamma hC}\, Q_{\gamma hA}\Big) \tag{18}$$

Here β controls the temperature of the softmax, $\mathcal{N}_A$ stands for the neighbors of node A —a set of all the nodes connected to node A, r is the ReLU function. Restriction of the attention operation to the neighborhood of a given node is similar to that used in the Graph Attention Networks (GAT), see [54]. Finally, we have residual connection (which is a natural consequence of the discretized differential equation dynamics)

$$\mathbf{x}^{t+1}_A = \mathbf{x}^t_A + \mathbf{\Delta x}^t_A \tag{19}$$

Intuitively, the first term in Equation 17 describes the influence (attention score) of the neighbor nodes with respect to the target node, the second term describes the influence of the target node with respect to each of its neighbor, and the third term is the contribution of the Hopfield Network module. It can be shown that the forward pass of our energy-based transformer layer minimizes the following energy function:

$$E = -\frac{1}{\beta}\sum_C\sum_h \log\left(\sum_{B\in\mathcal{N}_C}\exp\left(\beta\sum_\alpha K_{\alpha hB}\, Q_{\alpha hC}\right)\right) - \frac{1}{2}\sum_{C,\mu} G\Big(\sum_j \xi_{\mu j}\, g_{jC}\Big) \tag{20}$$

This energy function will decrease as the forward pass progresses until it reaches a local minimum.

After T iterations when the retrieval is stable, we have the final representation for each node $\mathbf{g}^{\text{final}}_A$ as

$$\mathbf{g}^{\text{final}}_A = \mathbf{g}^{t=1}_A \;||\; \mathbf{g}^{t=T}_A \tag{21}$$

where || is the concatenation sign. Following [24], we treat anomaly detection as semi-supervised learning task in this work. The node representation $\mathbf{g}^{\text{final}}_A$ is fed to another MLP with the sigmoid function to compute the abnormal probability p_A, weighted log-likelihood is then used to train the network. The loss function is as follow:

$$\text{Loss} = \sum_A \Big[\omega\, l_A \log(p_A) + (1 - l_A)\log(1 - p_A)\Big] \tag{22}$$

where ω is the ratio of normal labels (l_A = 0) to anomaly labels (l_A = 1).

C.2 Experimental Details

We train all models for 100 epochs using the Adam optimizer with a learning rate of 0.001, and use the model with the best Macro-F1 on the validation set for reporting the final results on the test set. Following [24], we use training ratios 1% and 40% respectively (randomly select 1% and 40% nodes of the dataset to train the model, and use the remaining nodes for the validation and testing). These remaining nodes are split 1:2 for validation:testing. The statistics of the datasets are listed in Table 5. For the four datasets used in the experiments, Amazon and Yelp datasets can be obtained from the DGL library, T-Finance and T-Social can be obtained from [24]. We report the average performance of 5 runs on the test datasets. The hyperparameters of our model are tuned based on the validation set, selecting the best parameters within 100 epochs. To speedup the training process, for the large graph datasets T-Finance and T-Social, we sample a different subgraph to train for each epoch (subgraphs have 5% of the nodes with respect to the whole training data). The hyperparameters

Dataset	$\|V\|$	$\|E\|$	Anomaly(%)	Features
Amazon	11944	4398392	6.87%	25
Yelp	45954	3846979	14.53%	32
T-Finance	39357	21222543	4.58%	10
T-Social	5781065	73105508	3.01%	10

Table 5: Summary of all the datasets.

include the number of hidden dimensions in ET-attention Y, the number of neurons K in the hidden layer within the Hopfield Network Module, the number of time iterations T, and the number of heads H. The weights are learned via backpropagation, which includes embedding projection $\mathbf{E}$, positional embedding λ_A, softmax inverse temperature parameter β, ET-attention weight tensors $\mathbf{W}^Q$ and $\mathbf{W}^K$. The optimal hyperparameters used in Table 1 are reported in Table 6. The last row in that table summarizes the range of the hyperparameter search that was performed in our experiments. In general, we have observed that for small datasets (Yelp, Amazon, T-Finance) a 1 or 2 applications of our network is sufficient for achieving strong results, for larger datasets (T-Social) more iterations (3) are necessary. For even bigger dataset (ImageNet) our network needs about 12 iterations.

Dataset	Y	K	T	H
Amazon (40%)	128	640	1	2
Amazon (1%)	64	128	1	1
Yelp (40%)	128	256	1	1
Yelp (1%)	128	256	1	1
T-Finance (40%)	128	256	1	3
T-Finance (1%)	128	256	1	1
T-Social (40%)	128	256	3	3
T-Social (1%)	128	256	3	3
Range of hyperparameters	{64, 128, 256}	{2Y, 3Y, 4Y, 5Y}	{1,2,3}	{1,2,3}

Table 6: Hyperparameters choice of our method on all the datasets.

D Graph Classification with ET

Recently, there have been attempts of leveraging Transformer into the graph domain, but only certain key modules, such as feature aggregation, are replaced in GNN variants by the softmax attention [55]. However, there remains an interesting and open question about suitability of the Transformer architecture to model graphs and how to apply it in the graph classification domain. In this section we explain how ET can be used for graph classification.

D.1 Details of Graph Classification ET Model

Given a graph $G = (V, \mathbf{A}, \mathcal{E})$, we have the set of nodes $V = \{v_1, v_2, \ldots, v_N\}$, the adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$, and if it exists, the edge feature matrix $\mathcal{E} \in \mathbb{R}^{N\times N\times P'}$ where each edge has P' raw features. Each feature vector $\mathbf{y}_A \in \mathbb{R}^F$ corresponding to a node v_A is first projected to the token space $\mathbf{x_A} \in \mathbb{R}^D$ via a linear embedding. Then, the CLS token $\mathbf{x}_{\text{CLS}}$ is concatenated to the set of tokens resulting in $\mathbf{X} \in \mathbb{R}^{(N+1)\times D}$ and the positional embedding $\lambda \in \mathbb{R}^{(N+1)\times D}$ is added to it afterwards.

To obtain the positional embedding $\lambda \in \mathbb{R}^{(N+1)\times D}$, the adjacency matrix $\mathbf{A}$ is first padded in the upper left corner with ones resulting in $\tilde{\mathbf{A}} \in \{0,1\}^{(N+1)\times(N+1)}$ as a means to provide the CLS token full connectivity with all of the nodes in a given graph. Following [37], the top k smallest eigen-vectors $\tilde{\lambda} \in \mathbb{R}^{(N+1)\times k}$ are extracted from the normalized Laplacian matrix $\tilde{\mathbf{L}}$ obtained from $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{23}$$

and projected to the token space via a linear embedding to form λ. Note, $\mathbf{I}$ and $\tilde{\mathbf{D}}$ are the identity matrix and the degree matrix of $\tilde{\mathbf{A}}$.

Meanwhile, the attention in ET is modified to take in $\hat{\mathbf{A}} \in \mathbb{R}^{(N+1)\times(N+1)\times H}$ the parameterized adjacency tensor, which acts as the weighted 'attention mask' that enables the model to consider the graph structural information. To obtain $\hat{\mathbf{A}}$, a 2D-convolutional layer with H filters equals to the number of heads in the attention block, 'SAME' padding, and a stride of one is performed on the outer product of $\mathbf{X}$ to itself. The result is then multiplied with the tensor $\mathbf{A}' \in \mathbb{R}^{(N+1)\times(N+1)\times H}$ element-wise (denoted by $\odot$) via broadcasting

$$\hat{\mathbf{A}} = \text{Conv2D}(\mathbf{X} \otimes \mathbf{X}) \odot \mathbf{A}'. \tag{24}$$

The tensor $\mathbf{A}'$ is a linear projection, via a linear embedding layer, of the edge feature matrix $\mathcal{E}$ if it exists, where P' is projected to H dimension; or the padded adjacency matrix $\tilde{\mathbf{A}}$ where $P' = 1$. Altogether, the resulting energy equation is

$$E^{\text{ATT}} = -\frac{1}{\beta} \sum_{h} \sum_{C} \log \left(\sum_{B \neq C} \exp \left(\beta \sum_{\alpha} K_{\alpha h B}\, Q_{\alpha h C} \odot \hat{A}_{hC} \right) \right). \tag{25}$$

Moreover, to only consider the energy of edges in G, we set non-connected entries in $\hat{\mathbf{A}}$ to be $-\infty$, such that the gradient of such entries would be zero following the dynamics. Note, if non-connected entries are not ignored, the gradient of such entries would be non-zero and hence, the resultant attention energy might violate some of the graph structural information.

Meanwhile, in this implementation, the overall model consists of S vertically stacked ET blocks, where each block shares the same number of T depth and has its own LayerNorm. Similarly, the token representation $\mathbf{X}^{\ell,\, t}$ at dynamic step t corresponding to a block ℓ is layer-normalized,

$$\mathbf{g}^{\ell,\, t} = \text{LayerNorm}(\mathbf{X}^{\ell,\, t}). \tag{26}$$

Keep in mind, $\mathbf{X} = \mathbf{X}^{\ell=1,\, t=1}$ is the initial token representation.

Following the dynamic equations 6, we inject a small amount of noise $\epsilon^{\ell,\, t} \in (0, 1)$, generated from a normal distribution using a standard deviation σ_ϵ and zero mean, into the gradient of energy function to produce $\mathbf{X}^{\ell,\, t+1}$, the new token representation of block ℓ. The premise of this noise injection is to 'robustify' the model and help it escape saddle points of the energy function.

$$\mathbf{X}^{\ell,\, t+1} = \mathbf{X}^{\ell,\, t} - \alpha \nabla_{\mathbf{g}} E^{\ell,\, t} + \sqrt{\alpha}\, \epsilon^{\ell,\, t}, \qquad \epsilon^{\ell,\, t} \sim \mathcal{N}(0, \sigma_\epsilon^2). \tag{27}$$

Once stability is reached in the retrieval dynamics of a block ℓ, the final representation $\mathbf{X}^{\ell,\, t=T}$ is then passed on to the next block $\ell + 1$ and the whole process is repeated again. When the final token representation $\mathbf{X}^{\ell=S,\, t=T}$ is computed by the last block S, the resultant CLS token $\mathbf{x}_{\text{CLS}}^{\ell=S,\, t=T} \in \mathbb{R}^{D'}$, extracted from $\mathbf{X}^{\ell=S,\, t=T}$, is utilized as the predictor of the current graph G.

D.2 Experimental Evaluation

As mentioned prior, eight datasets from TUDataset [36] are used for experimentation. Based on the collected results in Table 2, we observed that the modified ET demonstrates the best performance across all datasets with the exception of MUTAG. Additionally, as a means to contrast ET with other graph transformers, ET is trained and evaluated with five datasets from GNNBenchmark [37]. Specifically, PATTERN and CLUSTER for node classification; CIFAR10 and MNIST for graph classification; and lastly, ZINC for graph-regression. We compare ET to the recently introduced graph transformers – namely, EGT [56], Graphormer [57], Exphormer [58], GT [59], and GAT [54]. The results are depicted in table 7.

Based on the collected results, ET outperforms its baselines on PATTERN and ZINC datasets, and has competitive performance on MNIST and CLUSTER datasets. Meanwhile, ET performs the worst on the CIFAR10 dataset, in which its performance is relatively close to GAT [54] and Graphformer [57]. The reason for such a bad performance can be attributed to the overfitting problem seen during the training of the model, which is also experienced by [56, 59]. Hence, while ET performs very well on TUDataset, further explorations are required to enable ET to work better on CIFAR10, MNIST, and CLUSTER datasets. However, ET has overall demonstrated great performance on node binary-classification and graph-regression tasks.

Table 7: Graph classification performance on five datasets of GNNBenchmark [37]. The reported mean and standard deviation are obtained from 100 runs of evaluation on the test set of each dataset. For baselines standard deviations are only included if they are available in prior work. If the entry is unavailable in prior literature it is denoted by '-'; best results are in **bold**. The performance difference between non-baseline approaches (including ours) and the baseline (specified by the gray cell in each column) is indicated by ▼(drop in performance with respect to the baseline), ▲(increase in performance with respect to the baseline), and ▼(no change). Additionally, $\downarrow$ indicates lower is better and $\uparrow$ indicates higher is better.

Method	**Dataset**				
	CIFAR10 (Accuracy) $\uparrow$	MNIST (Accuracy) $\uparrow$	PATTERN (Accuracy) $\uparrow$	CLUSTER (Accuracy) $\uparrow$	ZINC (MAE) $\downarrow$
GT	-	-	$84.808_{\pm 0.068}$ ▼(2.013)	$73.169_{\pm 0.622}$ ▼(6.063)	$0.226_{\pm 0.014}$ ▼(0.112)
GAT	$64.223_{\pm 0.455}$ ▼(10.531)	$95.535_{\pm 0.205}$ ▼(2.879)	$78.271_{\pm 0.186}$ ▼(8.550)	$70.587_{\pm 0.447}$ ▼(8.645)	$0.384_{\pm 0.007}$ ▼(0.276)
EGT	$68.702_{\pm 0.409}$ ▼(6.052)	$98.173_{\pm 0.087}$ ▼(0.241)	$86.821_{\pm 0.020}$	$\mathbf{79.232_{\pm 0.348}}$	$0.108_{\pm 0.009}$
Graphormer	$65.978_{\pm 0.579}$ ▼(8.776)	$97.905_{\pm 0.176}$ ▼(0.509)	$86.650_{\pm 0.033}$ ▼(0.171)	$74.660_{\pm 0.236}$ ▼(4.572)	$0.122_{\pm 0.006}$ ▼(0.014)
Exphormer	$\mathbf{74.754_{\pm 0.194}}$	$\mathbf{98.414_{\pm 0.038}}$	$86.734_{\pm 0.008}$ ▼(0.087)	$78.220_{\pm 0.045}$ ▼(1.012)	-
ET (Ours)	$63.920_{\pm 0.173}$ ▼(10.834)	$97.011_{\pm 0.045}$ ▼(1.403)	$\mathbf{90.690_{\pm 0.013}}$ ▲(3.869)	$77.169_{\pm 0.044}$ ▼(2.063)	$\mathbf{0.096_{\pm 0.000}}$ ▲(0.012)

D.3 Experimental Details

In the graph domain, it is common to concatenate all of the feature vectors of all graphs in a batch together. However, in order for ET to work, we form the batch dimension by separating the feature vectors of all graphs in a given batch and utilize the largest node count to pad all graphs such that they all share the same number of nodes. Additionally, we set a limit on the number of nodes equal to 500, to prevent out-of-memory error. Specifically, if a graph has a node count exceeding the limit, the number of utilized nodes is equal to the limit. Hence, a portion of the graph structural information is ignored as a result. However, it is worth mentioning such a graph is rare in the experimental datasets. Additionally, instead of ignoring the padded entries, they are altered to be sink nodes in a graph, such that they are connected to the actual nodes of the graph but not vice versa. We found this approach to be helpful when it comes to computing the positional embedding and training the model. Additionally, following [37, 56], the sign of the top k eigen-vectors is flipped randomly during training and fixed during evaluation. The whole experiment is implemented using JAX[49], Flax [50], Optax [60], and PyTorch Geometric [61] frameworks.

Table 8: The statistics and properties of the eight datasets of TUDataset (additional node attributes are indicated by '+' if exist).

Dataset	**Graphs**	**Avg. Nodes**	**Avg. Edges**	**Node Attr**	**Classes**
MUTAG	188	17.93	19.79	7	2
ENZYMES	600	32.63	62.14	18 + 3	6
PROTEINS	1113	39.06	72.82	0 + 4	2
DD	1178	284.32	715.66	89	2
NCI1	4110	29.87	32.30	37	2
NCI109	4127	29.68	32.13	38	2
MUTAGENICITY	4337	30.32	30.77	14	2
FRANKENSTEIN	4337	16.90	17.88	780	2

For the eight datasets of TUDataset, we train all models for 300 epochs using AdamW [62] and Gradient Centralization [63]. The best model is selected based on its performance obtained from the 10-fold cross validation process delineated in [36]. Since the task is classification, all models are trained with the cross-entropy loss function and label smoothing [64], where the smoothing parameter is set to 0.05. Additionally, the cosine-annealing with warm-up learning rate scheduler [65] is utilized, where the initial and end learning rates are both set as $5e-6$ while the peak learning rate is 0.001. The number of warm-up steps is set to 50 epochs while the batch size is 32 for all datasets. We report the average performance of 100 runs on the 10-fold cross validation process with random seeding. The hyperparameters of our model are tuned based on the performance of the cross validation, selecting within 100 epochs. The optimal hyper-parameters are reported in Table 9 and the statistics of the used datasets are reported in Table 8. We also include the number of gpu-devices used for training ET.

Table 9: Hyperparameter and architecture choices for ET during TUDataset experiments.

Training	
batch_size	32
epochs	300
peak lr	1e-3
warmup_epochs	50
initial and ending lr	5e-6
b1, b2 (ADAM)	0.9, 0.99
weight_decay	0.05
grad_clipping	None
num. of gpu devices	2

Architecture	
token_dim	128
num_heads	12
head_dim	64
β	$\frac{1}{\sqrt{64}}$
train_betas	Yes
step size α	0.01
k eigenvalues	15
noise σ_ϵ	0.02
depth	1
block_size	4
kernel_size	[3, 3]
dilation_size	[1, 1]
hidden_dim HN	512
bias in HN	None
bias in ATT	None
bias in LNORM	Yes
num. of params per ET block	262,929
avg. total num. of params	1071294

Similarly, for the five datasets of GNNBenchmark, we train all models using using AdamW [62] and Gradient Centralization [63]. The best model is selected based on its performance obtained from the validation set across 10 runs. For the CIFAR10 and MNIST datasets, the additional node positional features are concatenated to the node features (i.e., pixel values coordinates). All models, with the exception of ZINC, CIFAR10, and MNIST models, are trained using the cross-entropy loss function with label smoothing, where the smoothing parameter is set to 0.05. The ZINC model is trained with the mean absolute error function. Meanwhile, as a means to reduce overfitting, for the CIFAR10 and MNIST models, the mean squared error loss function is utilized to reconstruct the edge feature matrix $\mathcal{E}$ in addition to the cross entropy loss function. Specifically, the parameterized adjacency tensor $\hat{\mathbf{A}}$ is projected to match P' raw feature dimension of $\mathcal{E}$, where both tensors are as part of the mean squared error function. In addition, all models are trained with the cosine-annealing with warm-up learning rate scheduler, where the number of warm-up steps is 50. We report the average performance of 100 evaluation runs on the testing set with random seeding. The hyperparameters of each model are tuned based on the performance of cross validation, selecting within 50 epochs. The optimal hyper-parameters are reported in Tables 10 and 11 and the statistics of the used datasets are reported in [37].

E ET for heterogeneous Graph

In this section, we show the performance of our ET model in the heterogeneous graph case. For the heterogeneous graph case (where more than one type of edges exist in the graph), we first run our ET model for different subgraphs (corresponding to different types of edges), and then aggregate the final representations using two methods. We have tried max pooling and concatenation for the aggregation step. Max pooling means to pick the largest value across all the subgraph representations, and concatenation means to concatenate the representations obtained from different subgraphs. Table 12 shows the comparison between these two variants of our model and BWGNN (heterogeous case). ET performs better than heterogeneous BWGNN on Amazon, but loses to heterogeneous BWGNN on Yelp. Interestingly, BWGNN in the heterogeneous setting performs worse than BWGNN in the homogeneous setting on Amazon.

Table 10: Hyperparameter choice for ET training on GNNBenchmark datasets.

Parameter	Dataset				
	CIFAR10	MNIST	PATTERN	CLUSTER	ZINC
batch_size	128	128	128	128	128
epochs	150	150	100	300	500
initial lr	1e-3	1e-3	1e-3	1e-3	5e-7
ending lr	5e-5	5e-5	5e-7	5e-7	5e-7
peak lr	1e-3	1e-3	1e-3	1e-3	1e-3
warmup_epochs	50	50	50	50	50
b1, b2 (ADAM)	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)
weight_decay	0.05	0.05	0.05	0.05	0.05
grad_clipping	None	None	None	None	None
num. of gpu devices	4	4	1	1	8

Table 11: Architecture choice for ET on GNNBenchmark datasets.

Parameter	Dataset				
	CIFAR10	MNIST	PATTERN	CLUSTER	ZINC
token_dim	128	128	128	128	128
num_heads	12	12	12	12	12
head_dim	64	64	64	64	64
β	$\frac{1}{\sqrt{64}}$	$\frac{1}{\sqrt{64}}$	$\frac{1}{\sqrt{64}}$	$\frac{1}{\sqrt{64}}$	$\frac{1}{\sqrt{64}}$
train_betas	Yes	Yes	Yes	Yes	Yes
step size α	0.25	0.25	0.1	0.1	0.1
k eigenvalues	15	15	15	15	15
noise σ_ϵ	0.02	0.02	0.005	0.005	0.005
depth	4	4	4	4	1
block_size	2	2	2	2	4
kernel_size	[3, 3]	[3, 3]	[3, 3]	[3, 3]	[3, 3]
dilation_size	[1, 1]	[1, 1]	[1, 1]	[1, 1]	[1, 1]
hidden_dim HN	512	512	512	512	512
bias in HN	None	None	None	None	None
bias in ATT	None	None	None	None	None
bias in LNORM	Yes	Yes	Yes	Yes	Yes
total num. of params	529224	528968	527936	528964	1052409

F Ablation Study for Attention and Hopfield Network Modules

As we described in the main text the the ET network consists of two modules processing the tokens in parallel: the attention module (ATT) and the Hopfield Network module (HN). The ATT module is responsible for routing the information between the tokens, while the HN module is responsible for reinforcing the token representation to be consistent with the general expectation about the particular data domain. It is interesting to explore the contribution that these two subnetworks produce on the task performed by the network. In this section we ablate the ET architecture by dropping each of the two subnetworks and measuring the impact of the ablation on the performance.

Table 12: ET for anomaly detection in heterogeneous graph setting. Best results are in **bold**.

	Datasets	Split	MaxPool	Concatenation	BWGNN (Heterogenous)
Macro-F1	Yelp	1%	$61.5_{\pm 0.4}$	$61.7_{\pm 0.2}$	$\mathbf{67.02}$
		40%	$70.7_{\pm 0.6}$	$71.1_{\pm 0.1}$	$\mathbf{76.96}$
	Amazon	1%	$\mathbf{88.3_{\pm 1.6}}$	$87.4_{\pm 1.4}$	83.83
		40%	$\mathbf{92.1_{\pm 0.2}}$	$91.8_{\pm 0.2}$	91.72
AUC	Yelp	1%	$72.2_{\pm 0.5}$	$72.8_{\pm 0.2}$	$\mathbf{76.95}$
		40%	$84.3_{\pm 0.4}$	$85.1_{\pm 0.1}$	$\mathbf{90.54}$
	Amazon	1%	$\mathbf{90.8_{\pm 1.4}}$	$90.6_{\pm 1.0}$	86.59
		40%	$\mathbf{97.5_{\pm 0.1}}$	$97.2_{\pm 0.6}$	97.42

F.1 On Graphs

The results on graphs are reported in Table 13. From this table it is clear that most of the computation is performed by the ATT block on this task, which pools the information about other tokens to the token of interest. When ATT block is kept, but HN block is removed the network looses 1% or less relative to the full ET (occasional improvements of the ablated model compared to the full ET are within the statistical error bars). In contrast, removing ATT module and keeping only the HN, the ET network effectively turns into an MLP with shared weights that is recurrently applied. In this regime the network can only use the features of a given node for that node's anomalous status prediction. This results in a more significant drop in performance, which is about 5% on average.

Table 13: Ablation study with respect to ATT block and HN Block. Best results are in **bold**.

	Datasets	Split	ATT✓|HN✗	ATT✗|HN✓	full model (Ours)
Macro-F1	Yelp	1%	$62.5_{\pm 0.3}$ ▼	$57.4_{\pm 0.5}$ ▼(-5.6)	$\mathbf{63.0_{\pm 0.6}}$
		40%	$70.6_{\pm 0.5}$ ▼	$71.2_{\pm 0.7}$ ▼	$\mathbf{71.5_{\pm 0.1}}$
	Amazon	1%	$\mathbf{89.5_{\pm 0.9}}$ ▲	$87.4_{\pm 1.0}$ ▼	$89.3_{\pm 0.7}$
		40%	$91.7_{\pm 0.5}$ ▼(-1.1)	$88.7_{\pm 0.3}$ ▼(-4.1)	$\mathbf{92.8_{\pm 0.3}}$
	T-Finance	1%	$84.7_{\pm 1.0}$ ▼	$80.3_{\pm 0.6}$ ▼(-4.8)	$\mathbf{85.1_{\pm 1.0}}$
		40%	$87.4_{\pm 0.7}$ ▼	$82.3_{\pm 0.8}$ ▼(-5.9)	$\mathbf{88.2_{\pm 1.0}}$
	T-Social	1%	$\mathbf{79.8_{\pm 0.6}}$ ▲	$72.7_{\pm 1.0}$ ▼(-6.4)	$79.1_{\pm 0.7}$
		40%	$82.9_{\pm 1.0}$ ▼	$78.6_{\pm 1.2}$ ▼(-4.9)	$\mathbf{83.5_{\pm 0.4}}$
AUC	Yelp	1%	$72.9_{\pm 0.3}$ ▼	$67.4_{\pm 0.7}$ ▼(-5.8)	$\mathbf{73.2_{\pm 0.8}}$
		40%	$83.5_{\pm 0.4}$ ▼(-1.4)	$83.1_{\pm 0.6}$ ▼(-1.8)	$\mathbf{84.9_{\pm 0.3}}$
	Amazon	1%	$90.7_{\pm 0.8}$ ▼	$89.8_{\pm 1.2}$ ▼	$\mathbf{91.9_{\pm 1.0}}$
		40%	$96.8_{\pm 0.6}$ ▼	$95.7_{\pm 0.5}$ ▼(-1.6)	$\mathbf{97.3_{\pm 0.4}}$
	T-Finance	1%	$91.7_{\pm 1.2}$ ▼	$90.2_{\pm 0.8}$ ▼(-2.6)	$\mathbf{92.8_{\pm 1.1}}$
		40%	$94.3_{\pm 2.6}$ ▼	$90.2_{\pm 2.1}$ ▼	$\mathbf{95.0_{\pm 3.0}}$
	T-Social	1%	$\mathbf{92.2_{\pm 0.8}}$ ▲	$86.4_{\pm 0.7}$ ▼(-5.5)	$91.9_{\pm 0.6}$
		40%	$93.1_{\pm 0.8}$ ▼	$88.3_{\pm 1.3}$ ▼(-5.6)	$\mathbf{93.9_{\pm 0.2}}$

F.2 On Images

The ablation results for image reconstruction are shown in Table 14. Each experiment was trained using the same hyperparameter settings as shown in Table 4. After training the model on IN1K, we calculate the average MSE on the reconstructed masked tokens for the validation set (using the same 50% masking ratio used for training) across 10 different random seeds for the masking.

We make several conclusions from these ablation studies.

- We gain several insights regarding the use of "self-attention" in our ET (when a token patch query is allowed to consider itself as a key in the attention weights). When both self-attention and HN are present (ET-Full+Self), there is no noticeable benefit over ET-Full for a token to attend to itself. In fact, preventing the ATTN energy module from attending to itself slightly improves the performance. However, when the HN is removed (ET-NoHN*), we notice that allowing self-attention (ET-NoHN+Self) outperforms the version that prevents self-attention (ET-NoHN).
- On its own, allowing self-attention (ET-NoHN+Self) in the ATTN module performs nearly as well as the full ET at a fraction of the total parameters. However, MSE is a forgiving metric for blurry reconstructions. While ATTN can capture the global structure of the image quite well, it does so at the expense of image sharpness (Figure 4).
- As expected, removal of the ATTN energy module performs the worst, because the HN operates tokenwise and has no way to aggregate token information across the global image without ATTN.

Figure 8 shows our best performing model (ET-Full) on the qualitative image reconstructions corresponding to the *largest* errors across IN1K validation images, averaged across all masking seeds. Likewise, Figure 9 shows the *lowest* errors across IN1K validation images and masking seeds. In general, image reconstructions that require ET to produce sharp, high frequency, and high contrast lines negatively impact MSE performance.

Table 14: Module ablation tests for image reconstruction task, reporting average IN1K validation MSE on masked tokens after 100 epochs. Reported number of parameters excludes the constant number of parameters in the affine encoder and decoder.

Model	Has ATTN?	Allow self-attn?	Has HN?	**NParams** (in ET block)	**MSE**
ET-Full **(Ours)**	✓	✗	✓	3.7M	$\mathbf{0.306_{\pm 0.10}}$
ET-Full+Self	✓	✓	✓	3.7M	$0.312_{\pm 0.10}$
ET-NoHN+Self	✓	✓	✗	1.3M	$0.343_{\pm 0.10}$
ET-NoHN	✓	✗	✗	1.3M	$0.403_{\pm 0.11}$
ET-NoATT	✗	✗	✓	2.5M	$0.825_{\pm 0.20}$

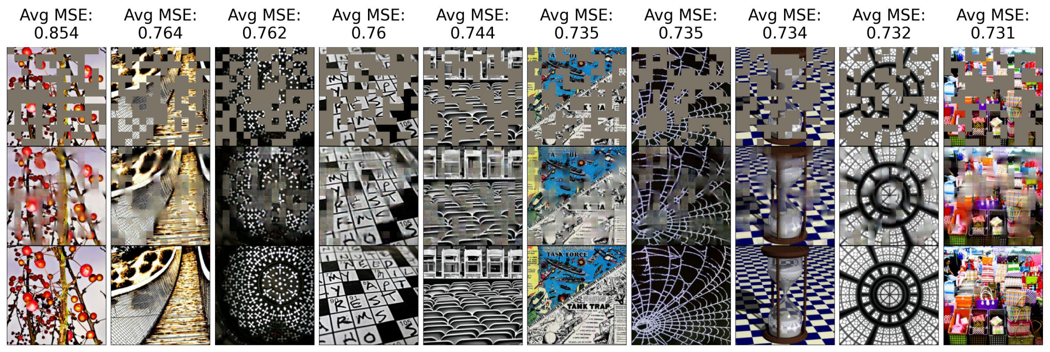

Figure 8: Reconstruction examples of images with the worst MSE from the IN1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: all tokens reconstructed after 12 time steps. *Bottom row*: original images.

G Parameter comparison

The energy function enforces symmetries in our model, which means ET has fewer parameters than its ViT counterparts. In particular, ET has no "Value Matrix" $\mathbf{W}^V$ in the attention mechanism, and the HN module has only one of the two matrices in the standard MLP of the traditional Transformer

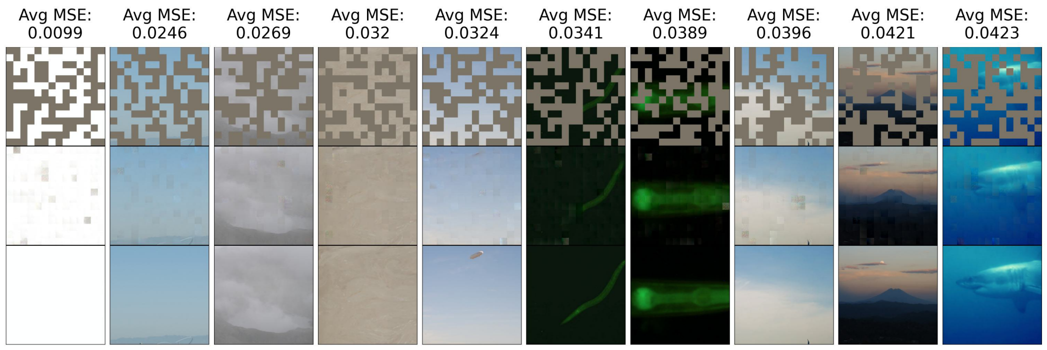

Figure 9: Reconstruction examples of images with the best (lowest) MSE from the IN1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: all tokens reconstructed after 12 time steps. *Bottom row*: original images.

block. We report these differences in Table 15. We take the ET configuration used in this paper, which has an architecture fully comparable to the original ViT-base [18] with `patch_size=16`, and report the number of parameters against ViT-base and an "ALBERT" version of ViT [14] where a single ViT block is shared across layers. We saw no benefit when including biases in ET, so we also exclude the biases from the total parameter count in the configuration of ViT and ALBERT-ViT. We report both the total number of parameters and the number of parameters per Transformer block. Furthermore, as a means for a fairer comparison, we reduce the parameters of ViT-Base such that the parameter count is similar to that of ET. We contrast ET and the reduced ViT on the image reconstruction task using PSNR (Peak-Signal-to-Noise-Ratio) and SSIM (Structural-SIMilarity) [66] as the evaluation metrics of the reconstructions. The results are demonstrated in Table 16, where the performance of ET is very close to that of the reduced ViT, which has a slight advantage in parameter count.

Table 15: Comparison between the number of parameters in a standard ViT, an ALBERT version of ViT where standard Transformer blocks are shared across layers, and our ET. Comparison is done assuming no biases in any operation.

Model	**NParams**		**NParams** (per block)	
ViT-Base	86.28M	▼0.00%	7.08M	▼0.00%
ALBERT ViT-Base	8.41M	▼90.25%	7.08M	▼0.00%
ET	**4.87M**	**▼94.36%**	**3.54M**	**▼50.02%**

Table 16: Comparison between ET and 'comparable-size' ViT on image reconstruction task. Given ViT-Base, we reduce its parameter count down to a number similar to that of ET for the image domain. The metrics, PSNR (Peak-Signal-to-Noise-Ratio) and SSIM (Structural-SIMilarity), are recorded for the image reconstruction evaluations.

Model	**NParams**		**PSNR**		**SSIM**	
ViT	5.52M	▼0.00%	**22.11**	▼0.00	**0.715**	▼0.00
ET	**4.28M**	**▼22.46%**	21.51	▼0.59	0.681	▼0.03

H Formal Algorithm for Training and Inference of ET

Algorithm 1: Training and inference pseudocode of ET for image reconstruction task

```
HyperParameters
    $\alpha$: Energy descent stepsize
    $\epsilon$: Learning rate
    $p$: Token mask probability
    $b$: batch size
Parameters
    $\mathbf{W}^K \in \mathbb{R}^{Y \times H \times D}$, $\mathbf{W}^Q \in \mathbb{R}^{Y \times H \times D}$: Key, Query kernels of the Energy Attention
    $\xi \in \mathbb{R}^{M \times D}$: Kernel of Hopfield Network
    $\gamma_{\text{norm}} \in \mathbb{R}$, $\delta_{\text{norm}} \in \mathbb{R}^D$: Scale, bias of LayerNorm
    $\text{MASK} \in \mathbb{R}^D$: Mask token
    $\delta_{\text{pos}} \in \mathbb{R}^{N \times D}$: Position bias, added to each token
    $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{P \times D}$, $\delta_{\text{enc}} \in \mathbb{R}^D$: Kernel, bias of affine Encoder
    $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{D \times P}$, $\delta_{\text{dec}} \in \mathbb{R}^D$: Kernel, bias of affine Decoder
Infer
    Inputs
        Corrupted image tokens $\tilde{X} \in \mathbb{R}^{N \times D}$
    Add position biases: $\tilde{X} \leftarrow \tilde{X} + \delta_{\text{pos}}$;
    for timesteps $t = 1, \ldots, T$ do
        Normalize each token:
            $\tilde{g} \leftarrow \text{LayerNorm}(\tilde{X}; \gamma_{\text{norm}}, \delta_{\text{norm}})$;                $\tilde{g} \in \mathbb{R}^{N \times D}$
        Calculate Energy of tokens:
            $E \leftarrow \text{EnergyTransformer}(\tilde{g}; \mathbf{W}^K, \mathbf{W}^Q, \xi)$;        $E \in \mathbb{R}$
        $\tilde{X} \leftarrow \tilde{X} - \alpha \nabla_{\tilde{g}} E$;
    return $\tilde{X}$
Train
    Inputs
        Dataset $S_{\text{train}}$ with elements $X \in \mathbb{R}^{\text{channels} \times \text{height} \times \text{width}}$
    Initialize
        Randomly initialize from $\mathcal{N}(0, 0.02)$:
            $\mathbf{W}^K, \mathbf{W}^Q, \xi, \text{MASK}, \mathbf{W}_{\text{enc}}, \mathbf{W}_{\text{dec}}, \delta_{\text{pos}} \sim \mathcal{N}(0, 0.02)$
        Set other biases to zero: $\delta_{\text{enc}}, \delta_{\text{dec}}, \delta_{\text{norm}} \leftarrow 0$
        Set LayerNorm scale to one: $\gamma_{\text{norm}} \leftarrow 1$
    for epoch $n = 1, \ldots, N_{\text{epoch}}$ do
        $S_{\text{epoch}} \leftarrow S_{\text{train}}$
        for batch $B \subset S_{\text{epoch}}$                                  $B \in \mathbb{R}^{b \times \text{channels} \times \text{height} \times \text{width}}$
        do
            Convert image into non-overlapping patches:
                $B_{\text{patch}} \leftarrow \text{Patchify}(B)$;                              $B_{\text{patch}} \in \mathbb{R}^{b \times N \times P}$
            Embed image patches into tokens:
                $X \leftarrow \text{Encode}(B_{\text{patch}}; \mathbf{W}_{\text{enc}}, \delta_{\text{enc}})$;              $X \in \mathbb{R}^{b \times N \times D}$
            Replace image tokens randomly by MASK:
                $\tilde{X}, I_{\text{mask}} \leftarrow \text{Mask}(X; \text{MASK}, p)$               $\tilde{X} \in \mathbb{R}^{b \times N \times D}$, $I_{\text{mask}} \in \{0, 1\}^{b \times N}$
            Reconstruct tokens with ET:
                $\tilde{X} \leftarrow$ Infer$(\tilde{X})$
            Decode tokens:
                $\hat{B}_{\text{patch}} \leftarrow \text{Decode}(\tilde{X}[I_{\text{mask}}]; \mathbf{W}_{\text{dec}}, \delta_{\text{dec}})$;       $\hat{B}_{\text{patch}} \in \mathbb{R}^{b \times N \times P}$
            Calculate MSE loss on corrupted tokens:
                $L \leftarrow \text{Mean}(|\hat{B}_{\text{patch}}[I_{\text{mask}}] - B_{\text{patch}}[I_{\text{mask}}]|^2)$      $L \in \mathbb{R}$
            params $\leftarrow$ params $- \epsilon \nabla_{\text{params}} L$
            $S_{\text{epoch}} \leftarrow S_{\text{epoch}} \setminus B$
    return params
```