EMCNet : Graph-Nets for Electron Micrographs Classification

Sagar Srinivas*
sagar.sakhinana@tcs.com
TCS Research
Bangalore, India

Rajat Sarkar*
rajat.sarkar1@tcs.com
TCS Research
Pune, India

Venkataramana Runkana
venkat.runkana@tcs.com
TCS Research
Pune, India

ABSTRACT

Characterization of materials via electron micrographs is an important and challenging task in several materials processing industries. Classification of electron micrographs is complex due to the high intra-class dissimilarity, high inter-class similarity, and multi-spatial scales of patterns. However, existing methods are ineffective in learning complex image patterns. We propose an effective end-to-end electron micrograph representation learning-based framework for nanomaterial identification to overcome the challenges. We demonstrate that our framework outperforms the popular baselines on the open-source datasets in nanomaterials-based identification tasks. The ablation studies are reported in great detail to support the efficacy of our approach.

KEYWORDS

graph neural networks, computer vision, materials characterization

1 INTRODUCTION

Quality control in materials processing is of paramount importance across diverse industrial sectors such as batteries, semiconductors, *etc.* Electron microscopes are used to create high-resolution images of materials known as electron micrographs. They provide the topography, morphology, composition, and crystallographic information to examine the structure of materials at the nanoscale. The nanomaterial recognition tasks[2] are challenging compared to the image-recognition benchmark datasets in degree of detail, complexity of patterns, and information density. The conventional modeling approaches for image-recognition tasks such as ConvNets([35], [32]), Vision transformers(ViTs, [19], [40], [16], [10], [56], [38]) and hybrid architectures([54], [27], [53]) suffer from inherent drawbacks of (a) requirement of large training datasets to introduce appropriate inductive biases and (b) large scale model complexity. The traditional Graph Neural Networks(GNNs, [39], [42], [51], [37], [24], [11]) learn and encode the complex discrete graph data by summarizing the high-level feature information in the graph-level embedding. The graph representation learning presents an alternate paradigm of approach for automatic identification of nanomaterials from their morphology or shape in the electron micrographs. The graph representation of the complex nanomaterial images has the spatial hierarchies of the high-level visual features embedded in a wide spectrum of graph structural-property information spanning across nodes, edges, motifs, and subgraphs. There is a need and necessity to tailor the architecture of traditional GNNs to effectively learn the structural characteristics of the graph for improved performance in the graph-level classification tasks. This work presents an overarching view of electron micrographs by effectively learning the multilevel and multiview representations for the global reasoning of the complex visual content embedded in the graph. In contrary to traditional CNNs, the novel framework identities the discrete entities(low-level visual elements) and their pairwise relationships from the fixed-graph topology for better visual perception of nanostructures in the images. The proposed framework offers better generalization and scalability for large datasets.

*Both authors contributed equally to this research.

2 OUR APPROACH

This section presents the Electron Micrographs Classification Network for brevity, EMCNet for nanomaterials identification. Our framework consists of six main steps, (1) tokenization of the nanoscale images; splitting each image into patches and representing it as a patch-attributed grid graph with diagonal edges, (2) Graph encoder, for brevity, **GEnc**; performs the iterative neighborhood-local aggregation schemes on the augmented graphs for learning the abstract graph representations, (3) Hierarchical graph encoder, for brevity, **HGEnc**; performs the layer-wise local-graph pooling and higher-order message-passing schemes to learn the implicit multi-grained hierarchical representations, (4) tree decomposition; we perform the tree decomposition of the grid graph to obtain a clique tree graph,(5) Clique tree encoder, for brevity, **CTEnc**; performs the iterative neighborhood message-passing schemes on the clique tree graph to learn the local substructures in the graph, and (6) the output layer; applies a weighted linear transformation on the learned representations and predicts the image category. The overview of the EMCNet framework is illustrated in Figure 1.

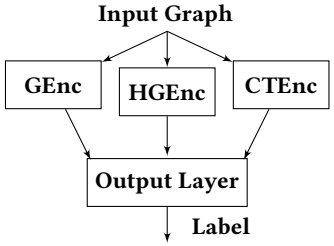


Figure 1: Overview of EMCNet framework, (a) we represent each image as a grid graph, (b) the GEnc and HGEnc modules of the framework computes the respective grid graph representations, (c) the CTEnc module determines the tree representation, and (d) the output layer predicts the image category.

2.1 Tokenization of Images

Let us assume that H, W, and C denote the dimensions of an RGB image, where H is the height, W is the width, and C is the number of channels. For a given 3D image, $\mathrm{I} \in \mathbb{R}^{H \times W \times C}$ and the resolution of each patch (P, P, C). We split the 3D image into "**N**" non-overlapping patches(also known as tokens). **N** is determined by $\frac{\mathrm{HW}}{\mathrm{P}^2}$. In a nutshell, each 3D image is reshaped to obtain a sequence of flattened 2D patches, $\mathrm{I}_P \in \mathbb{R}^{N \times P^2 C}$. The independent patches are projected into a d-dimensional embedding space to obtain the low-level patch representations, $\mathrm{I}_P \in \mathbb{R}^{N \times d}$. It is described below,

$$I_P = I_P \mathbf{E} \tag{1}$$

The trainable patch embeddings, $\mathbf{E} \in \mathbb{R}^{P^2C \times d}$ are the semantic representations of the patches in vision tasks. The patch representations suffer from the inherent drawback of lack of patch locality-preserving information. The position embeddings, $P_E \in \mathbb{R}^{N \times d}$ are linearly added to the patch representations to enable position awareness, refer to Figure 2. The position embeddings along with the patch embeddings are randomly initialized and are jointly updated along with the network parameters during the training of the model. We represent the patches of each image as the nodes of a regular static graph. In general, there is no natural ordering of the nodes in the graph. Here, we expose our model to the positional information of the patches to effectively exploit the locality information to perform better on the evaluation metrics. Figure 3 depicts the illustration of the tokenization of the images.

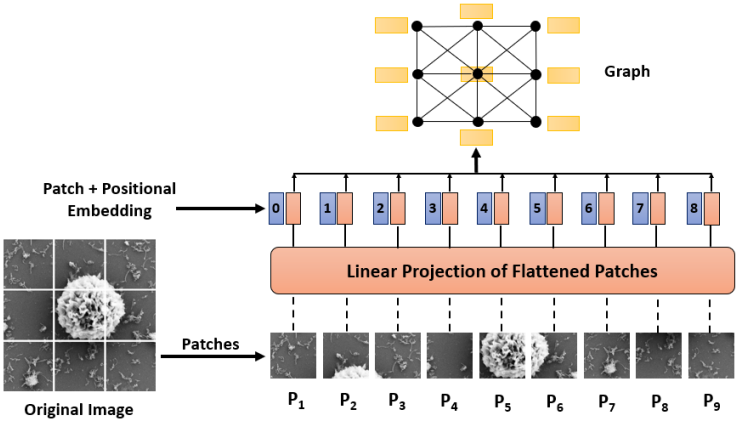


Figure 2: For illustration purpose, we split the image into 3×3 patches and represent it as an undirected graph.

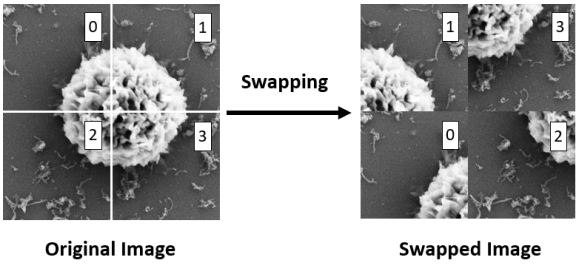


Figure 3: For illustration purpose, we split an image into 2×2 patches. In the absence of positional information, the GNNs cannot distinguish the images on the left and right.

2.2 Graph Representation

We represent each nanoscale image as the node-attributed undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node set, $\mathcal{V}$ denote the sequence of patches. The node feature matrix is described by $X \in \mathbb{R}^{N \times d}$. $N(|\mathcal{V}|)$ denotes the number of nodes. d is the dimensionality of the node feature vector. The patches are connected through an edge to adjacent patches. $\mathcal{E}$ denotes the edge-set. It is obtained from the graph structure information. The graph structure can also be represented by the adjacency matrix, $\mathcal{A} \in \mathbb{R}^{N \times N}$. $\mathcal{A}[v, u] = 1$ if $(v, u) \in \mathcal{E}$ or else $\mathcal{A}[v, u] = 0$. Here, u & $v \in \mathcal{V}$ refers to the local-graph neighbors and are connected by an arbitrary edge characterized by an empty edge attribute, $(v, u) \in \mathcal{E} \leftrightarrow (u, v) \in \mathcal{E}$, $\forall v \in \mathcal{N}(u)$. The local-graph neighbors of each node are described by $\mathcal{N}(u) = \{v \in \mathcal{V} \mid (v, u) \in \mathcal{E}\}$. The graph representation of the nanoscale images has a grid-like structure and has fixed dependency relationships among the nodes of the graph across the diverse set of images. Given a dataset $(\mathcal{G}_i, y_i) = \{(\mathcal{G}_1, y_1), \cdots, (\mathcal{G}_n, y_n)\}$, where y_i is the ground truth label of $\mathcal{G}_i$, the objective of the graph-level classification task is to learn a novel mapping function $f : \mathcal{G}_i \rightarrow y_i$ that maps the discrete graphs to the set of labels.

2.3 Graph Encoder(GEnc)

The objective of the graph encoder is to map the high-dimensional discrete graphs information to the low-dimensional graph-level representations in the Euclidean space. We propose a message-passing neural network to model the discrete graphs, $\mathcal{G}_i$ for determining both the node-level $\mathbf{z}_u, u \in \mathcal{V}(\mathcal{G}_i)$, and the graph-level abstract representations $\mathbf{z}_{\mathcal{G}_i}$, whilst maximally preserving the high-level visual feature information embedded in the graphs. We augment each graph with a virtual master node. The master node is bidirectionally connected to all nodes of the graph through virtual edges. The virtual edges represent the pairwise relationships between each "real" node and the master node in the graph. The augmented graph is not a fully-connected graph. The virtual master node representation contains the global-information of the nodes. Each node reads and writes to transform the virtual master-node representation through the iterative message-passing schemes. Neural messages are communicated across the nodes of the augmented graphs through the edges connecting them. The neural messages encapsulate the immediate graph neighbors information. The neighborhood-local aggregation schemes learn the node-level abstract representations. The node representations capture the rich local and informative long-range pairwise interactions spanning across the graph. The virtual master node helps in providing an additional path for message propagation to promote the information diffusion between the distant nodes of the graph. It helps in learning the global structural properties for a graph-wide representation. Each node in the graph receives and sends neural messages to its local-graph neighbors. At first, the source node $v \in \mathcal{V}$ receives neural messages Φ_{wv} from its local-graph neighbors, $w \in \mathcal{N}(v) \backslash u$. Φ_{wv} is modeled by a linear operator to encode the neighboring node's abstract representations. The subscript, wv indicates the message propagation from the node w to node v. Subsequently, the source node, $v \in \mathcal{N}(u)$ modifies the neural message vectors Φ_{wv} with its feature vector x_v, and applies non-linearity to compute the neural-message vector Φ_{vu}. The source node sends the neural message Φ_{vu} to the sink node u to refine the node embedding $\mathbf{z}_u$. In short, each source node, $v \in \mathcal{V}$ dispatches neural messages to the sink node, $u \in \mathcal{V}$ only after it receives all the incoming neural messages from its neighbors, $\Phi_{wv}, w \in \mathcal{N}(v) \backslash u$. Algorithm 1 presents our Graph Encoder(GEnc). For each message-passing iterative step, the receptive field increases by one hop. We compute the refined neural message vector by instigating additional rounds of message-passing to embed the larger neighborhood of each node by iterating for T steps. Each node in the graph perceives the refined neural messages sent from its immediate graph neighbors to transform its abstract

representation. The transformed node representations encode the local substructure information about its T-hop neighborhood. We determine the fixed-size graph-level embedding $\mathbf{z}_{\mathcal{G}_i}$ by performing the sum-pooling operation on the node-level embeddings. The weight matrix, W^g applies a shared linear transformation on each node feature vector. The trainable parameters, U_1^g and U_2^g of the iterative graph message-passing schemes are shared across nodes of the graph. τ is a non-linear sigmoidal function. The optimal selection of message-passing iterative-steps (neighborhood aggregation) trade-offs the under-representation and the over-smoothing of the node-level representations of the augmented graph to encode the larger local-graph neighborhood. The graph-level representation obtained from the node-level representations of the augmented graphs with fixed graph topology and varying node features are discriminative enough to achieve higher classification accuracy. In summary, the graph encoder maps the discrete graph data to the optimal graph-level representations by learning to model the visual information embedded in the complex structural properties of the graph. It is utilized in the downstream tasks to perform inference on the graphs through label predictions.

Algorithm 1 Iterative Graph Message Passing Mechanism

Initialize messages: $\Phi_{wv}^{(0)} = 0$
for $t = 1$ **to** T **do**
$\Phi_{wv}^{(t)} = \tau\big(\mathrm{W}^g x_w + \sum_{w \in \mathcal{N}(v)\setminus u} \Phi_{wv}^{(t-1)}\big)$
$\Phi_{vu}^{(t)} = \tau\big(\mathrm{W}^g x_v + \Phi_{wv}^{(t)}\big)$

end for
$\mathbf{z}_u = \tau\big(\mathrm{U}_1^g x_u + \sum_{v \in \mathcal{N}(u)} \mathrm{U}_2^g \Phi_{vu}^{(T)}\big)$
Return $\mathbf{z}_{\mathcal{G}_i} = \sum_u \mathbf{z}_u / |\mathcal{V}|$.

2.4 Hierarchical Graph Encoder(HG-Enc)

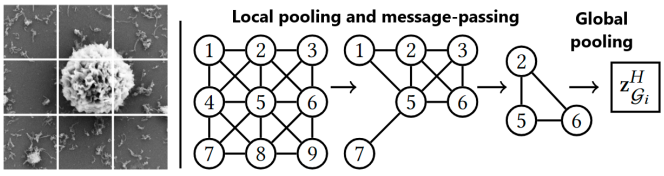


Figure 4: The nodes are labeled for illustration. We gradually reduce the graph size in progressive layers, by rejecting the nodes of lower importance and learn the high-level representations through the self-attention based message-passing schemes.

The HG-Enc module learns the prominent subgraph structures to determine the hierarchical representations of the graphs. The HG-Enc module overcomes the following shortcomings, (1) traditional GNNs treat all the nodes equally by disregarding the subset of nodes of high importance; explicitly, it models all the patches of an image without consideration of the visual content, (2) the tokenization of the images results in non-overlapping patches and this skips the boundary-level information present in the adjacent patches of an image, and (3) there is a need to learn the scale-variant visual elements in the underlying graph, and this would be intractable for the conventional GNNs to encode from the independent patches of a fixed scale. Figure 4 depicts the illustration of the hierarchical graph encoder. The key steps in the HG-Enc module are (a) the basis for pooling the graph to obtain the pooled graph, (b) structure learning of the pooled graph, and (c) determining the higher-order node representations of the pooled graph. The HG-Enc module performs the non-linear down-sampling by adaptively rejecting a subset of nodes in the main graph to form an induced subgraph through an importance score measure. The importance score of each node is determined by performing a scalar projection of the node-level embedding on the trainable projection vector. The node embeddings are determined through the spatial-graph filtering operations.

$$z_u^{(l+1)} = \mathrm{ReLU}\left(\alpha_{u,u} W^{(l)} z_u^{(l)} + \sum_{v \in \mathcal{N}^{(l)}(u)} \alpha_{v,u} W^{(l)} z_v^{(l)}\right) \tag{2}$$

where $z_u^0 = x_u \in \mathbb{R}^d$ is the feature vector of node u. $z_u^{(l)} \in \mathbb{R}^d$ denotes the abstract representation of node u. The superscript, l denotes the layer. The set of neighbors of a node u obtained from the adjacency matrix $\mathcal{A}^{(l)}$, is $\mathcal{N}^{(l)}(u) = \left\{v \mid \mathcal{A}^{(l)}[v,u] > 0\right\}$. The trainable weight matrix, $\mathrm{W}^{(l)} \in \mathbb{R}^{2d \times d}$ applies a shared linear transformation on the node feature vector, and the attention coefficients $\alpha_{v,u}$ are determined as,

$$\zeta(v,u) = \mathrm{ReLU}\left(a^\top (r_u \oplus r_v)\right); r_u = W z_u^{(l)} \tag{3}$$

$$\alpha_{v,u} = \frac{\exp(\zeta(v,u))}{\sum_{k \in \mathcal{N}^{(l)}(u) \cup \{u\}} \exp(\zeta(k,u))} \tag{4}$$

Where a is the vector of the weighted coefficients for the attention mechanism. $\oplus$ denotes the concatenation operation. r_u is the linearly transformed patch representations. ReLU is the non-linear activation function. The layer-wise forward propagation of the HG-Enc operator is described by,

$$s_u = z_u^{(l+1)} \frac{p^{(l+1)}}{\|p^{(l+1)}\|} \tag{5}$$

where $z_u^{(l+1)}, p^{(l+1)} \in \mathbb{R}^{2d}$ & $s \in \mathbb{R}^{|\mathcal{V}^{(l)}|}$. $p^{(l+1)}$ denotes the trainable projection vector to map the node representations into the importance score. The scalar projection of the node-attributes on the normalized projection vector is described by s. The importance score is utilized to rank the nodes in the graph to reduce the graph size. For a given pooling ratio p_r, we select a subset of top m-ranked nodes in the main graph. The pooling ratio is described by, $p_r = \frac{m}{|\mathcal{V}^{(l)}|}, \forall\ p_r \in (0,1]$. The node ranking operation, $idx = \mathrm{rank}(s,m)$ utilizes the scalar projection scores to sample the top m-prominent nodes. idx describes the indices of the selected top m-nodes in the graph. We obtain the pooled graph $\mathcal{G}^{(l+1)}$ from the main graph $\mathcal{G}^{(l)}$ by utilizing the prominent nodes indices obtained through the node-ranking operation. The pooled graph-structure and node attribute information is described as,

$$\mathcal{A}^{(l+1)} = \mathcal{A}^{(l)}(idx, idx);\ \mathcal{A}^{(l+1)} \in \mathbb{R}^{m \times m},\ \mathcal{A}^{(l)} \in \mathbb{R}^{|\mathcal{V}^{(l)}| \times |\mathcal{V}^{(l)}|}$$

where $\mathcal{A}^{(l+1)}$ is the adjacency matrix of the pooled graph. The node attribute matrix of the pooled graph $\mathcal{G}^{(l+1)}$ is described by,

$$\tilde{Z}^{(l+1)} = Z^{(l)}(idx, :);\ \tilde{Z}^{(l+1)} \in \mathbb{R}^{m \times 2d},\ Z^{(l)} \in \mathbb{R}^{|\mathcal{V}^{(l)}| \times 2d} \tag{6}$$

A gating operation is performed to regulate the information flow to avoid inadvertent over-smoothing of the node representations,

$$\tilde{s} = \mathrm{ReLU}\big(s(idx)\big) \tag{7}$$

$$Z^{(l+1)} = \tilde{Z}^{(l+1)} \odot \left(\tilde{s} \mathbf{1}_{2d}^T\right) \tag{8}$$

$\mathbf{1}_{2d} \in \mathbb{R}^{2d}$ is a vector of size $2d$ and with each component value of 1. $\odot$ denotes the element-wise product. We will apply the self-attention mechanism to learn the compact hierarchical node representations of the pooled graph from the input node representations as given by the node attribute matrix, $Z^{(l+1)}$, refer Equation 8. The hierarchical node representation is computed as the weighted sum of the node attributes of the pooled graph,

$$z_u^{(l+1)} = \sum_k \alpha_{ku} \left(z_k^{(l+1)} W_V^{(l+1)} \right), k \in \mathcal{N}^{(l+1)}(u) \cup \{u\} \quad (9)$$

where $z_u^{(l+1)} \in \mathbb{R}^d$. Each weight coefficient α_{ku} is computed using a softmax as,

$$\alpha_{ku} = \frac{\exp(e_{ku})}{\sum_k \exp(e_{ku})}, \quad (10)$$

where e_{vu} is calculated using scaled dot-product attention. It is described by,

$$e_{ku} = \frac{\left(z_u^{(l+1)} W_Q^{(l+1)}\right)\left(z_k^{(l+1)} W_K^{(l+1)}\right)^T}{\sqrt{d}} \quad (11)$$

The trainable projections $W_Q^{(l+1)}, W_K^{(l+1)}, W_V^{(l+1)} \in \mathbb{R}^{2d \times d}$ are unique per layer. Here, we stack 3-layers of HG-Enc operators each with a pooling ratio of $p_r = 0.75$. We perform the global average pooling operation on the node-level embeddings of the pooled graph to obtain $\mathbf{z}_{\mathcal{G}_i}^H$. There exist various techniques to perform down-sampling on graphs. The local-graph pooling technique is fundamentally different from the graph coarsening(contraction) technique on the graph reduction method. The local-graph pooling technique ranks the nodes based on the complex visual content it contains. The low-ranking nodes are characterized by noise or contain the no-visual element. The HG-Enc operator drops the nodes of lower importance through the local-graph pooling technique and models the task-relevant induced subgraph (high-ranking nodes) from the main graph. In comparison, the graph coarsening technique[51] [37] clusters the nodes into supernodes based on the learned node representations. It implicitly models the noise, learns sub-optimal node representations, and are less-effective on the graph-level classification tasks. In summary, the HG-Enc operator learns the long-range, multi-level dependencies of the graph.

2.5 Clique Tree Encoder(CTEnc)

We perform the tree-decomposition[1] of the graph $\mathcal{G}$ to obtain the clique tree representation $\mathcal{T}$. It presents the tree-structured scaffold of the local graph substructures(*i.e.*, motifs and subgraphs) and their pairwise relationships. Figure 5 depicts an illustration of the tree decomposition of the graph representation of the nanoscale images. Each supernode of the clique tree, $C_i = (\mathcal{V}_i, \mathcal{E}_i)$ is an induced subgraph of the main graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each supernode of the clique tree, C_i is labeled with a $\mathcal{V}_i \subseteq \mathcal{V}$ and $\mathcal{E}_i \subseteq \mathcal{E}$. Here, i refers to the index of the supernode in $\mathcal{T}$. Each supernode, C_i in the clique tree $\mathcal{T}$ is described by a feature vector x_i. As stated earlier, each supernode in the clique tree is an induced subgraph of the main graph. We apply a shared linear transformation on the concatenated matrix of the node features in the subgraph to obtain the feature vector of the supernode. We utilize tree-based message-passing schemes to operate on the topology of the clique tree $\mathcal{T}$ to determine its low-level embedding $\mathbf{h}^{\mathcal{T}}$. A random leaf supernode is selected as the root supernode of the clique tree. Neural messages are computed and propagated across the supernodes of the clique tree in two phases. The neural messages encode the neighboring supernode's information in the clique tree and the local relations among the supernodes. At first, in the bottom-up phase, neural messages are propagated from the leaf supernodes iteratively towards the root supernode of the clique tree. In the top-down phase, neural messages are dispatched from the root supernode to its child supernodes down to the leaf supernodes of the clique tree. The neural messages m_{ij} and m_{ji} are dispatched from the supernode C_i to the supernode C_j through the superedge (C_i, C_j) and the vice-versa in the clique tree.

Algorithm 2 Iterative Tree-Message Passing Mechanism

Initialize messages: $\mathbf{m}_{ji}^{(0)} = \mathbf{0}$
for $t = 1$ **to** T **do**
  $\mathbf{m}_{ji}^{(t)} = \text{GRU}\left(x_j, \{\mathbf{m}_{kj}^{(t-1)}\}_{k \in \mathcal{N}(j) \setminus i}\right)$.
end for
$\mathbf{h}_i = \tau\left(\mathbf{W}_1^{\mathcal{T}} x_i + \sum_{j \in \mathcal{N}(i)} \mathbf{W}_2^{\mathcal{T}} \mathbf{m}_{ji}^{(T)}\right)$.
Return $\mathbf{h}^{\mathcal{T}} = root\ node, \mathbf{h}_i$.

Algorithm 3 Gating Mechanism on Tree-Structure

$$\mathbf{s}_{ji} = \sum_{k \in \mathcal{N}(j) \setminus i} \mathbf{m}_{kj}$$
$$\mathbf{z}_{ji} = \sigma\left(\mathbf{W}^z x_j + \mathbf{U}^z \mathbf{s}_{ji} + \mathbf{b}^z\right)$$
$$\mathbf{r}_{kj} = \sigma\left(\mathbf{W}^r x_j + \mathbf{U}^r \mathbf{m}_{kj} + \mathbf{b}^r\right)$$
$$\widetilde{\mathbf{m}}_{ji} = \tanh\left(\mathbf{W} x_j + \mathbf{U} \sum_{k \in \mathcal{N}(j) \setminus i} \mathbf{r}_{kj} \odot \mathbf{m}_{kj}^{(t-1)}\right)$$
$$\mathbf{m}_{ji} = \left(1 - \mathbf{z}_{kj}\right) \odot \mathbf{s}_{ji} + \mathbf{z}_{kj} \odot \widetilde{\mathbf{m}}_{kj}$$

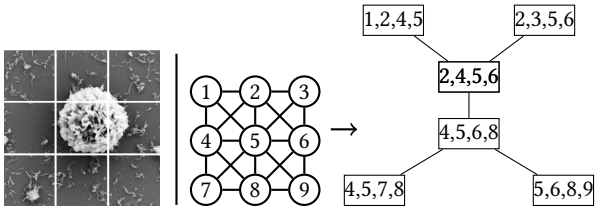


Figure 5: The nodes are labeled for illustration. We transform the graph into a hierarchical clique tree. The supernodes are known as cliques. The top-left-corner supernode of the clique tree is an induced subgraph(consists of nodes with labels 1, 2, 4, 5, and all of the edges) of the main graph and similarly the other supernodes.

Algorithm 2 presents our Clique Tree Encoder(CTEnc). The neural message $\mathbf{m}_{ji}$ is computed through the gating mechanism to encapsulate the long-range interactions across the supernodes of the clique tree by regulating the information diffusion. Algorithm 3 presents the gating mechanism on the tree structure. After performing, the T-steps of message-passing iterations. We perform the sum-pooling operation on the weighted neural messages dispatched from the local-tree neighbors, $j \in \mathcal{N}(i)$ to compute a single-message vector. The supernode C_i perceives the aggregated message vector to refine its embedding $\mathbf{h}_i$. The transformed supernode embeddings of the clique tree incorporate information about their local T-hop neighbors. The tree-level embedding $\mathbf{h}^{\mathcal{T}}$ is the root-node embedding, $\mathbf{h}_{\text{root}}$ and it summarizes the subgraph patterns in the main graph. The trainable parameters of the learning algorithm $\mathbf{W}_1^{\mathcal{T}}$, $\mathbf{W}_2^{\mathcal{T}}$ are shared across the supernodes of the

clique tree. τ is the sigmoidal function that introduces the non-linearity in the message-passing schemes. In summary, the tree encoder maps the discrete clique trees to determine the low-level tree representations.

3 OUTPUT LAYER

The outputs of the parallel operating modules, GEnc, HGEnc, and CTEnc are further transformed by the linear operator. We apply softmax-activation on the output of the linear operator for the multi-class classification task. It is mathematically described as follows,

$$\mathbf{q}_i = \mathbf{softmax}\big(\mathbf{W}_1 \mathbf{z}_{\mathcal{G}_i} + \mathbf{W}_2 \mathbf{z}^H_{\mathcal{G}_i} + \mathbf{W}_3 \mathbf{h}^{\mathcal{T}_i}\big) \tag{12}$$

where $\mathbf{q}_i$ is the probability distribution over the image categories in the dataset. $\mathbf{W}_1, \mathbf{W}_2$ & $\mathbf{W}_3$ are the training parameters of the model which are jointly-optimized along with other model parameters. We apply the argmax operation on $\mathbf{q}_i$ for determining the model predictions on the image label, y^o_i. In summary, the hierarchical graph encoder(Hg-Enc) learns the multi-spatial scale patterns in the nanoscale images. The Graph Encoder(GEnc) and clique tree encoder(CTEnc) are tailored-designed to overcome the inherent limitations of the high intra-class dissimilarity and high inter-class similarity in the nanomaterial images classification task.

4 DATASETS

We conduct experiments on the Scanning Electron Microscopy dataset[2] for nanomaterials identification. The labeled dataset contains ≈21,283 high-resolution images at the nanoscale. The dataset contains a wide collection of nano-objects. The images are classified into 10 categories, and they span a broad range of *particles, nanowires, patterned surfaces, microelectromechanical devices, etc.* The images are augmented by geometric transformations such as shearing, and resizing to improve the quality of training data and reduce over-fitting. We randomly split the data. The test set comprises 4256 samples. The validation set contains 2128 samples, and the remaining are for the training set. The dataset is highly imbalanced. We utilize 10-Fold cross-validation for training the model to learn from the training set. In addition, we utilized several open-source material benchmark datasets to evaluate our proposed method. The details are reported in the supplementary material.

5 EXPERIMENTAL SETUP

The model was trained in a supervised-learning approach for the classification task. The data pre-processing involves feature scaling to obtain normalized images. The resolution of each RGB image in our dataset is $1024 \times 768 \times 3$ pixels. We perform resizing of the image to obtain a relatively lower spatial resolution, $256 \times 256 \times 3$ pixels. We split the downscaled image into non-overlapping patches with resolution $32 \times 32 \times 3$ pixels. The positional and patch embedding size(d) is 64. The batch size is 24. The iterative message-passing steps(T) for GEnc & CTEnc operators is 6. The training number of epochs is 100. The initial learning rate is $5e^{-3}$. We reduce the learning rate by half if the evaluation metric shows no improvement on the validation set for a waiting number of 10 epochs. We run the gradient descent algorithm to minimize the cross-entropy loss between the ground-truth labels and the model predictions. We report the ensemble average of the results obtained from five computational experiments. Each computational experiment is run for a unique random seed. The experimental results reported are the average value of the different random seeds-based experimental run outputs. We utilized NVIDIA Tesla T4 GPUs for the training of GNNs built upon the PyTorch framework. Early stopping is implemented on the validation set to prevent the model from over-fitting and for model selection. We evaluate the model performance and report the evaluation metric on the test set.

6 EXPERIMENTS

In this work, we find an answer to the following research questions.

- RQ1 : How does our proposed method perform in classification tasks compared to the ConvNets(CNNs), Vision Transformers(ViTs), and traditional GNNs-based algorithms?
- RQ2 : How helpful are the various modules of our proposed method for the improved overall performance of the model?
- RQ3 : What about the impact of positional encoding schemes?
- RQ4 : How do the baselines modules of similar functionality perform compared to the modules in our framework?
- RQ5 : What is the impact of image categories on each modules?
- RQ6 : What about the quality of learned embeddings in self-supervised settings?

7 RESULTS

7.1 RQ1: Benchmarking algorithms

The initial experimental results on the dataset are reported by [41]. They evaluate the well-known inception model and its variant's performance on the subset of the original dataset [2], which contains a set of 10 categories for a total of 18,577 images. Due to the unavailability of the subset dataset publicly, we conducted experiments on the original dataset [2] which contains 12% higher samples. However, the original dataset [2] curator doesn't provide the predefined train/vali-dation/test dataset. So, we utilize the k-fold cross-validation method to evaluate our model performance on the dataset. Table 1 presents the performance of the baseline models based on ConvNets(CNNs), and Vision Transformers(ViTs) architectures in comparison with our method. We utilize the neural network architecture reported in the literature for the baseline models. For fair and rigorous comparison, we adopt identical experimental settings to generate the results of the baseline models. The evaluation metric is the conventional Top-N accuracy, where $N \in \{1, 2, 3, 5\}$. The standard deviation values are less than almost 2% of the mean value. For further comparisons with the recent advances in the GNNs. We present a reasonable comparison under identical experimental settings with the baseline GNNs techniques in Table 2. The baseline GNNs performance is evaluated on the graph representation of the nanoscale images. In comparison with the baseline models, our proposed method attains significant gains and demonstrates the best performance consistently across the Top-N accuracy classification scores. We also compare our model performance with the contrastive learning algorithms on vision and graph data for classification tasks. We report the results in Tables, 1 and 2. We also evaluate our model performance in terms of precision (fraction of positive classified images are actually correct) and recall (fraction of actual positives identified correctly). We report an average precision of 0.705 and an average recall of 0.763 across the image categories.

Table 1: Comparative study of our proposed method and the baseline algorithms. We also report the performance of the Self-Supervised(VSL) learning algorithms on Vision tasks.

	Algorithms	Parameters	Top-1	Top-2	Top-3	Top-5
ConvNets	AlexNet	5.70E+07	0.493	0.582	0.673	0.793
	DenseNet	2.39E+05	0.539	0.750	0.875	0.906
	ResNet	2.72E+05	0.512	0.766	0.891	0.906
	VGG	3.44E+07	0.517	0.644	0.717	0.779
	GoogleNet	2.61E+05	0.560	0.844	0.906	0.938
	SqueezeNet	7.41E+05	0.436	0.469	0.609	0.656
VSL	Barlowtwins[57]	8.99E+06	0.138	0.250	0.328	0.453
	SimCLR[13]	8.73E+06	0.157	0.234	0.359	0.469
	byol[28]	8.86E+06	0.130	0.234	0.281	0.422
	moco[31]	8.73E+06	0.158	0.188	0.250	0.438
	nnclr[21]	9.12E+06	0.144	0.266	0.313	0.531
	simsiam[14]	9.01E+6	0.170	0.266	0.391	0.500
Vision Transformers(ViTs)	CCT[29]	4.10E+05	0.600	0.781	0.875	0.969
	CVT[54]	2.56E+05	0.537	0.750	0.828	0.953
	ConViT[16]	6.00E+05	0.582	0.734	0.828	0.938
	ConvVT[54]	9.23E+04	0.291	0.563	0.734	0.875
	CrossViT[10]	8.35E+05	0.466	0.719	0.828	0.938
	PVTC[53]	1.30E+06	0.567	0.766	0.813	0.922
	SwinT[40]	2.78E+07	0.675	0.766	0.891	0.938
	VanillaViT[19]	1.79E+06	0.623	0.828	0.859	0.938
	Visformer[15]	1.21E+05	0.371	0.578	0.641	0.797
	ATS[23]	3.26E+06	0.511	0.703	0.828	0.938
	CaiT[49]	3.84E+07	0.616	0.750	0.906	0.938
	DeepViT[59]	3.26E+06	0.512	0.734	0.875	0.938
	Dino[8]	2.02E+07	0.047	0.219	0.391	0.432
	Distallation[48]	2.06E+06	0.516	0.719	0.844	0.938
	LeViT[27]	1.68E+07	0.597	0.813	0.875	0.953
	MA[30]	3.87E+06	0.192	0.288	0.350	0.459
	NesT[58]	1.61E+07	0.636	0.828	0.891	0.953
	PatchMerger[43]	3.26E+06	0.549	0.719	0.859	0.922
	PiT[33]	4.48E+06	0.520	0.703	0.828	0.953
	RegionViT[9]	1.22E+07	0.575	0.797	0.859	0.922
	SMIM[55]	2.38E+06	0.163	0.297	0.453	0.609
	T2TViT[56]	1.03E+07	0.702	0.859	0.906	0.938
	ViT-SD[38]	4.47E+06	0.613	0.766	0.906	0.953
	EMCNet	9.70E+05	**0.783**	**0.876**	**0.952**	**0.984**

7.2 RQ2: Study of Modules

Our proposed method comprises of **GEnc**, **HGEnc**, and **CTEnc** modules. We perform experiments on the SEM dataset by gradually removing each module in a controlled setting to examine the efficacy of the key modules that be responsible for the improved overall performance of the model. We refer to the **EMCNet** model in the absence of different operators as follows:

- **w/o GEnc**: **EMCNet** model without the **GEnc** operator.
- **w/o HGEnc**: **EMCNet** model without the **HGEnc** operator.
- **w/o CTEnc**: **EMCNet** model without the **CTEnc** operator.

We report in Table 3 the results obtained on the test dataset in terms of Top-1 classification accuracy. The results demonstrate the effectiveness of the operators with negligible add-on computational complexity. To be specific, the **HGEnc** operator is of advantage

Table 2: Comparative study of our proposed method and the baseline algorithms. We also report the performance of the Graph Self-Supervised(GSL) learning algorithms.

	Algorithms	Parameters	Top-1	Top-2	Top-3	Top-5
GSL	GBT[6]	7.09E+05	0.513	0.595	0.686	0.778
	GRACE[60]	7.44E+05	0.581	0.646	0.711	0.773
	BGRL[46]	6.92E+05	0.573	0.629	0.671	0.728
	InfoGraph[45]	6.82E+05	0.560	0.631	0.694	0.756
Graph Convolution Networks	APPNP[37]	7.35E+05	0.604	0.713	0.792	0.823
	AGNN[47]	5.22E+05	0.517	0.733	0.841	0.943
	ARMA[5]	4.57E+05	0.553	0.747	0.848	0.925
	DNA[24]	8.48E+05	0.593	0.677	0.786	0.891
	GAT[51]	6.31E+05	0.507	0.724	0.807	0.914
	GGConv[39]	8.05E+05	0.583	0.778	0.841	0.944
	GraphConv[42]	5.85E+05	0.623	0.787	0.875	0.953
	GCN2Conv[12]	6.18E+05	0.697	0.813	0.867	0.945
	ChebConv[17]	5.00E+05	0.547	0.762	0.834	0.896
	GraphConv[42]	6.79E+05	0.533	0.727	0.847	0.961
	GraphUNet[25]	9.57E+05	0.622	0.738	0.866	0.912
	MPNN[26]	5.22E+05	0.643	0.792	0.873	0.959
	RGGConv[7]	6.58E+05	0.633	0.727	0.886	0.928
	SuperGAT[36]	5.54E+05	0.561	0.676	0.863	0.935
	TAGConv[20]	5.74E+05	0.614	0.739	0.803	0.946
	EMCNet	9.70E+05	**0.783**	**0.876**	**0.952**	**0.984**

in learning the long-range dependencies and complex hierarchical representations of the graph in comparison to the flat message-passing schemes in GNNs. Similarly, **CTEnc** operator learns the complex structural characteristics in the graphs through the tree-structured scaffolds representations. The **GEnc** operator helps in better learning of the rich local-graph neighborhood information.

Table 3: The table reports the ablation studies to characterize the effect of each module on the overall model performance.

Methods	**EMCNet**	**w/o GEnc**	**w/o HGEnc**	**w/o CTEnc**
Accuracy	**0.783** ± 0.012	**0.717** ± 0.014	**0.594** ± 0.015	**0.664** ± 0.019

7.3 RQ3: Study on Tokenization of Images

We conduct experiments to investigate the impact of positional embeddings that contribute to the better performance of the model. We refer to the **EMCNet** model without incorporating the positional embeddings as follows: **w/o PosEmb**: **EMCNet** model. We report in Table 4 the results obtained on the test dataset in terms of Top-1 accuracy. The impact is significant on the model performance.

Table 4: The ablation studies to determine the impact of positional embeddings on the model performance.

Methods	**EMCNet**	**w/o PosEmb**
Accuracy	**0.783** ± 0.012	**0.515** ± 0.012

We further investigate the model performance by modeling the positional encoding scheme(PEs) by the Laplacian PEs[22], Self-attention RPEs,[44], Signed RPEs[34], and Sinusoidal PEs[50] compared to our approach of learning the optimal position embeddings through training parameters of the model. We refer to the **EMCNet** model by incorporating the following PEs techniques as follows:

- **w/ LPEs**: **EMCNet** model with **Laplacian PEs**.
- **w/ SPEs**: **EMCNet** model with **Sinusoidal PEs**.
- **w/ SARPEs**: **EMCNet** model with **Self-attention RPEs**.
- **w/ APEs**: **EMCNet** model with **Signed RPEs**.

We report in Table 5 the results on the test dataset in terms of Top-1 classification accuracy. The impact of various PEs techniques is marginal on the model predictive performance in comparison with our method.

Table 5: The table reports the ablation studies on positional-encoding schemes.

Methods	EMCNet	w/ LPEs	w/ SPEs	w/ SARPEs	w/ SRPEs
Accuracy	**0.783** ± 0.012	**0.772** ± 0.008	**0.767** ± 0.010	**0.790** ± 0.003	**0.761** ± 0.007

7.4 RQ4: Study of GEnc module

We investigate the GEnc module efficacy in comparison with the popular algorithms of similar functionality. The GEnc module consists of the graph convolution operator to perform convolutional operations on graphs and the global average pooling operator to determine the graph-level representation. We utilize well-known methods as baseline operators to perform convolution on the graphs. The list includes, GAT([51]), APPNP([37]), DNA([24]), and GCN2([11]). For the global graph-pooling, the list includes GraphMultisetTransformer(GMT, [3]), GlobalAttention(GA, [39]), Set2Set([52]), and Global Summation Pooling(GSM). We refer to the **EMCNet** model with the baseline graph convolution operators are as follows:

- **w/ GAT**: The **EMCNet** model with **GAT** operator.
- **w/ APPNP**: The **EMCNet** model with **APPNP** operator.
- **w/ DNA**: The **EMCNet** model with **DNA** operator.
- **w/ GCN2**: The **EMCNet** model with **GCN2** operator.

The Top-1 accuracy is reported in Table 6. The results support our graph convolution operator based approach to overcome the shallow learning mechanisms of the baseline operators.

Table 6: The table reports the comparative study of the graph convolution baseline operators on the model performance.

Methods	EMCNet	w/ GAT	w/ APPNP	w/ DNA	w/ GCN2
Accuracy	**0.783** ± 0.012	**0.747** ± 0.012	**0.759** ± 0.010	**0.767** ± 0.009	**0.749** ± 0.013

We refer to the **EMCNet** model with the baseline operators for modeling the global read-out function in the **GEnc** module as:

- **w/ GMT**: The **EMCNet** model with **GMT** operator in **GEnc**.
- **w/ GA**: The **EMCNet** model with **GA** operator in **GEnc**.
- **w/ Set2Set**: The **EMCNet** model with **Set2Set** in **GEnc**.
- **w/ GSM**: The **EMCNet** model with **GSM** operator in **GEnc**.

We report the results obtained on the test dataset in terms of Top-1 classification accuracy. The results reported in Table 7 demonstrate no significant improvements in the model performance in comparison to our method with the global average pooling operator. Overall, our graph encoder proves to be effective by learning to compute optimal graph-level representations.

Table 7: The table reports the comparative study of the baseline global pooling operators on the model performance

Methods	EMCNet	w/ GMT	w/ GA	w/ Set2Set	w/ GSM
Accuracy	**0.783** ± 0.012	**0.797** ± 0.011	**0.791** ± 0.007	**0.786** ± 0.008	**0.772** ± 0.013

7.5 RQ4: Study of HGEnc module

We probe the HGEnc module effectiveness in comparison with the well-known algorithms of identical functionality. The **HGEnc** module operates in two phases. The first phase performs the graph coarsening and the high-order message-passing schemes to compute the hierarchical representations of the nodes in the coarser graph. The successive phase performs the global average pooling of the node representations to compute the graph-level representation. We utilize popular methods as the baseline operators for modeling the hierarchical graph coarsening and message-passing schemes. The list includes, MemPool([51]), ASAPool([37]), SAGPool([24]), and TopKPool([25]). We refer to the **EMCNet** algorithm with the baseline hierarchical graph convolution operators are as follows:

- **w/ MemPool**: **EMCNet** model with **MemPool** operator.
- **w/ ASAPool**: **EMCNet** model with **ASAPool** operator.
- **w/ SAGPool**: **EMCNet** model with **SAGPool** operator.
- **w/ TopKPool**: **EMCNet** model with **TopKPool** operator.

Table 8: The table reports the study of baseline hierarchical graph convolution operators on the model performance.

Methods	EMCNet	w/ MemPool	w/ ASAPool	w/ SAGPool	w/ TopKPool
Accuracy	**0.783** ± 0.012	**0.713** ± 0.014	**0.725** ± 0.011	**0.709** ± 0.008	**0.725** ± 0.013

The Top-1 classification accuracy is reported in Table 8 on the test dataset. The results demonstrate the efficacy of **HGEnc** module to model the complex local substructures to learn the hierarchical representations of the graph. We refer to the **EMCNet** model with the following baseline operators to perform the global average pooling in the **HGEnc** module as:

- **w/ GMT**: The **EMCNet** model with **GMT** operator in **HGEnc**.
- **w/ GA**: The **EMCNet** model with **GA** operator in **HGEnc**.
- **w/ Set2Set**: The **EMCNet** model with **Set2Set** in **HGEnc**.
- **w/ GSM**: The **EMCNet** model with **GSM** operator in **HGEnc**.

The results are reported on the test dataset in Table 9 in terms of Top-1 classification accuracy. It shows no significant improvement or deterioration in the model performance in comparison to our method with the global average pooling operator.

Table 9: The table reports the comparative study of the graph read-out baseline operators on the model performance.

Methods	EMCNet	w/ GMT	w/ GA	w/ Set2Set	w/ GSM
Accuracy	**0.783** ± 0.012	**0.779** ± 0.006	**0.789** ± 0.003	**0.775** ± 0.011	**0.790** ± 0.007

7.6 RQ5: Study of image categories impact on the modules

Each image category in the SEM dataset consists of a mixture of easy(visually similar images) and hard samples(visually diverse), characterized by the complexity of multi-scale spatial features, degree of detail, and information density. The hard samples greatly influence learning the parameters of the modules. Similarly, each module learns distinct and dominant patterns from easy samples and generalization ability from the hard samples of each image category. Our proposed framework generalizes well despite the complexity of patterns across the wide spectrum of image categories. We study the impact of the GEnc, HGEnc, and CTEnc modules in isolation with the absence of other modules on the classification task of each image category. We perform additional experiments on

the SEM dataset by realizing our proposed method with the module under investigation to support the rationale of each module on the classification task. We refer to the EMCNet model realized with the different operators as follows:

- **w/ GEnc**: **EMCNet** model with the **GEnc** operator.
- **w/ HGEnc**: **EMCNet** model with the **HGEnc** operator.
- **w/ CTEnc**: **EMCNet** model with the **CTEnc** operator.

We utilize the identical experimental settings, refer to section 5 for performing this study. We report the results in Table 10, obtained on each image category of the test dataset in terms of Top-1 classification accuracy. We utilize 10-fold cross-validation to evaluate the model performance. The GEnc and CTEnc modules effectively capture the rich local information spread across each image for the image categories such as *biological, tips, films coated surface, and powder*. The long-range and hierarchical information inherent in image categories such as *fibres, porous sponge, patterned surface, nanowires, particles, MEMS devices, and powder* are effectively modeled by the HGEnc module.

Table 10: The table reports the efficacy of each module on learning different image categories. The underline score represents the best performance among the modules.

Category	Modules			EMCNet
	GEnc	HGEnc	CTEnc	
Biological	0.658	0.525	0.628	**0.727**
Tips	0.712	0.678	0.708	**0.811**
Fibres	0.485	0.657	0.523	**0.783**
Porous Sponge	0.567	0.665	0.595	**0.722**
Films Coated Surface	0.623	0.537	0.679	**0.725**
Patterned surface	0.576	0.712	0.625	**0.771**
Nanowires	0.624	0.749	0.671	**0.789**
Particles	0.563	0.673	0.654	**0.729**
MEMS devices	0.625	0.748	0.658	**0.812**
Powder	0.672	0.613	0.648	**0.771**

7.7 RQ6: Self-Supervised Learning

The graph contrastive learning(GCL) algorithms construct numerous arbitrary-sized graph views through stochastic data augmentation techniques. GCL is a self-supervised learning algorithm that learns a discriminative model by contrasting multiple positive-paired augmented views of the same graph, as opposed to the independently sampled negative-paired views of different graphs in the embedded space. We utilize GCL as a pre-training model to learn the node- and graph-level unsupervised representations through mutual information maximization, while optimally preserving the structural and attributional characteristics of the graph. We model the graph encoder of the contrastive techniques with our proposed, **GEnc** or **HEnc** operators. We leverage a broad range of ML techniques such as Support Vector Machines, Random Forest, and the Gradient Boosted Trees for the downstream task of modeling the target label of the graph data as a function of the generated unsupervised representations. We evaluate and report the results on the test dataset in terms of Top-1 classification accuracy. The experimental results are reported in Table 11 and Table 12, where we additionally report EMCNet model performance in comparison with Graph contrastive techniques. The classification scores show the effectiveness of the latent representations computed by the contrastive techniques through our design variants-based approach.

Table 11: The table reports the performance of the contrastive techniques. The graph encoder of the contrastive techniques is modeled by GEnc operator.

Methods	SVM	RF	XGBoost	EMCNet
BGRL, [46]	0.674 ± 0.09	0.721 ± 0.09	0.744 ± 0.08	**0.783** ± 0.012
GBT, [6]	0.693 ± 0.03	0.737 ± 0.05	0.756 ± 0.12	
GRACE, [60]	0.713 ± 0.05	0.742 ± 0.06	0.773 ± 0.04	
InfoGraph, [45]	0.723 ± 0.03	0.751 ± 0.11	0.764 ± 0.06	

Table 12: The table reports the performance of the contrastive techniques. The graph encoder of the contrastive techniques is modeled by HEnc operator.

Methods	SVM	RF	XGBoost	EMCNet
BGRL, [46]	0.693 ± 0.07	0.742 ± 0.09	0.769 ± 0.013	**0.783** ± 0.012
GBT, [6]	0.701 ± 0.05	0.758 ± 0.03	0.761 ± 0.06	
GRACE, [60]	0.717 ± 0.09	0.762 ± 0.09	0.771 ± 0.09	
InfoGraph, [45]	0.724 ± 0.08	0.752 ± 0.04	0.775 ± 0.06	

7.8 Additional Experiments

Here, we perform an additional set of experiments by resizing the input images of resolution $1024 \times 768 \times 3$ pixels to obtain a relatively higher spatial resolution of $512 \times 512 \times 3$ pixels. Here, we consider two experimental settings as follows:

- In the first setting(FS), we split the image into the patches of the resolution, $64 \times 64 \times 3$ pixels. We project the flattened patch representations into the embedding space to obtain a size of 144×64.
- In the second setting(SS), we split the image into the patches of the resolution, $32 \times 32 \times 3$ pixels. We project the flattened patch representations into the embedding space to obtain a size of 256×64.

We refer to the **EMCNet** model with the aforementioned experimental settings as follows,

- **w/ FS**: **EMCNet** model trained with the first settings.
- **w/ SS**: **EMCNet** model trained with the second settings.

Table 13: The table reports the effect of the patch size and the number of patches on the model performance.

Methods	EMCNet	w/ FS	w/ SS
Accuracy	**0.783** ± 0.012	**0.846** ± 0.009	**0.867** ± 0.005

Table 14: Performance comparison of our clustering approach based EMCNet and the baseline EMCNet models.

Methods	EMCNet	w/ EMCNet : Clusters
Accuracy	**0.783** ± 0.012	**0.921** ± 0.0013

The results reported in Table 13 show significant improvements in the model performance. To overcome the limitations of learning from images having a high degree of similarity corresponding to different image categories. We perform the K-Means clustering on the SEM dataset to assign cluster labels to the images based on similarity. We partition the images into k-fixed apriori clusters in which each image belongs to a specific cluster. In this work, the optimal setting for k is **2**. We will train a single EMCNet model to learn from the images corresponding to each cluster in the supervised learning approach. We utilize an identical train/validation/test split

of 70%/20%/10% for images assigned to each cluster. We report the results in Table 14 denoted by **w/ EMCNet : Clusters**. The results demonstrate exceptional performance in the Top-1 classification accuracy in comparison with the baseline EMCNet model.

8 CONCLUSION AND FUTURE WORKS

We conduct the first in-depth study of the graph-neural networks for solving the challenging electron micrographs classification tasks. We propose a joint optimization framework to determine the expressive graph-level representations by effectively summarizing the complex hierarchical visual feature maps from electron micrographs. The experimental results support our framework efficacy to achieve better performance in comparison to the state-of-art methods. In the future, we would like to extend our work on other electron micrograph datasets like REM, SEM, STEM etc., and also develop graph neural network-based models for object detection, segmentation for nanomaterial microstructures.

REFERENCES

[1] Salvador Aguinaga, David Chiang, and Tim Weninger. 2018. Learning hyperedge replacement grammars for graph generation. *IEEE transactions on pattern analysis and machine intelligence* 41, 3 (2018), 625–638.

[2] Rossella Aversa, Mohammad Hadi Modarres, Stefano Cozzini, Regina Ciancio, and Alberto Chiusole. 2018. The first annotated set of scanning electron microscopy images for nanoscience. *Scientific data* 5, 1 (2018), 1–10.

[3] Jinheon Baek, Minki Kang, and Sung Ju Hwang. 2021. Accurate learning of graph representations with graph multiset pooling. *arXiv preprint arXiv:2102.11533* (2021).

[4] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. 2021. The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision* 129, 4 (2021), 1038–1059.

[5] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[6] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. 2021. Graph Barlow Twins: A self-supervised representation learning framework for graphs. *arXiv preprint arXiv:2106.02466* (2021).

[7] Xavier Bresson and Thomas Laurent. 2017. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553* (2017).

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9650–9660.

[9] Chun-Fu Chen, Rameswar Panda, and Quanfu Fan. 2021. Regionvit: Regional-to-local attention for vision transformers. *arXiv preprint arXiv:2106.02689* (2021).

[10] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. 2021. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 357–366.

[11] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 1725–1735.

[12] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 1725–1735.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[14] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15750–15758.

[15] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. 2021. Visformer: The vision-friendly transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 589–598.

[16] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. 2021. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. *arXiv preprint arXiv:2103.10697* (2021).

[17] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[18] Aditya M Deshpande, Ali A Minai, and Manish Kumar. 2020. One-shot recognition of manufacturing defects in steel surfaces. *Procedia Manufacturing* 48 (2020), 1064–1071.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[20] Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. 2017. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370* (2017).

[21] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2021. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9588–9597.

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

[23] Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. 2021. Ats: Adaptive token sampling for efficient vision transformers. *arXiv preprint arXiv:2111.15667* (2021).

[24] Matthias Fey. 2019. Just jump: Dynamic neighborhood aggregation in graph neural networks. *arXiv preprint arXiv:1904.04849* (2019).

[25] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *international conference on machine learning*. PMLR, 2083–2092.

[26] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[27] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. 2021. LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12259–12269.

[28] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems* 33 (2020), 21271–21284.

[29] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. 2021. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704* (2021).

[30] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377* (2021).

[31] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[33] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. 2021. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11936–11945.

[34] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. *arXiv preprint arXiv:2009.13658* (2020).

[35] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[36] Dongkwan Kim and Alice Oh. 2022. How to find your friendly neighborhood: Graph attention design with self-supervision. *arXiv preprint arXiv:2204.04879* (2022).

[37] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[38] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. 2021. Vision Transformer for Small-Size Datasets. *arXiv preprint arXiv:2112.13492* (2021).

[39] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[40] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10012–10022.

[41] Mohammad Hadi Modarres, Rossella Aversa, Stefano Cozzini, Regina Ciancio, Angelo Leto, and Giuseppe Piero Brandino. 2017. Neural network for nanoscience scanning electron microscope image recognition. *Scientific reports* 7, 1 (2017), 1–12.

[42] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[43] Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. 2022. Learning to Merge Tokens in Vision Transformers. *arXiv preprint arXiv:2202.12015* (2022).

[44] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).

[45] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[46] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. 2021. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

[47] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735* (2018).

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*. PMLR, 10347–10357.

[49] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. 2021. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 32–42.

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

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

[52] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* (2015).

[53] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2022. PVT v2: Improved baselines with Pyramid Vision Transformer. *Computational Visual Media* (2022), 1–10.

[54] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22–31.

[55] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2021. Simmim: A simple framework for masked image modeling. *arXiv preprint arXiv:2111.09886* (2021).

[56] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 558–567.

[57] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*. PMLR, 12310–12320.

[58] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Arik, and Tomas Pfister. 2022. Nested Hierarchical Transformer: Towards Accurate, Data-Efficient and Interpretable Visual Understanding. (2022).

[59] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. 2021. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886* (2021).

[60] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

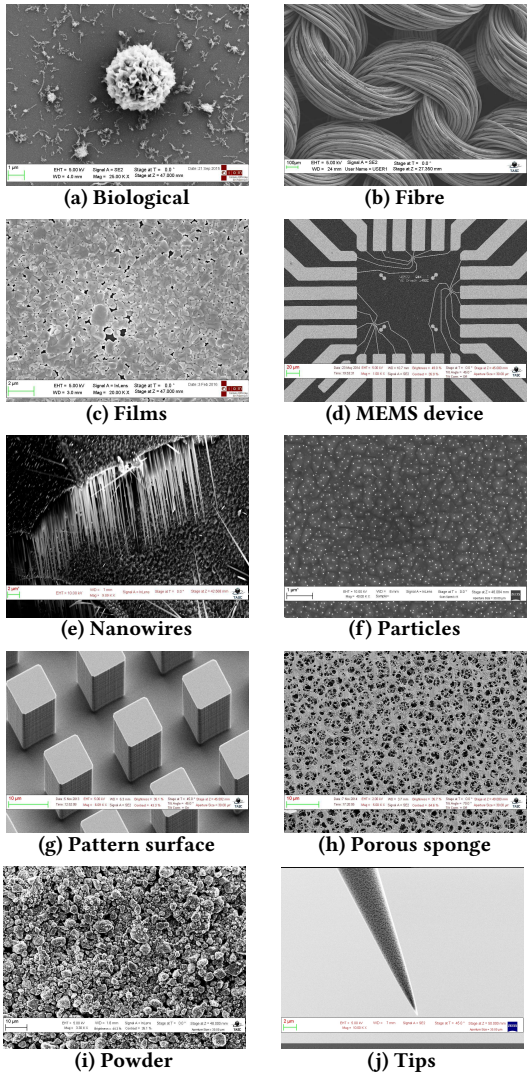
9 SUPPLEMENTARY MATERIAL

9.1 SEM dataset

In the Figure 6 we show the representative images of the 10 different image categories in the dataset. The SEM dataset[2], please refer to Table 15 has unequal distributions in the total count of each image category in the dataset.

9.2 Hyperparameters Study

We perform an in-depth analysis of the hyperparameters search to determine the effective model complexity of our proposed method, **EMCNet**. The hyperparameters of the algorithm are the dimensionality of embedding(d), the graph-pooling ratio(p_r), and the batch size(bs). The hyperparameter tuning is based on grid-search technique. We learn the tuple of hyperparameters that yield the optimal performance of our proposed method on the validation set in terms



(a) Biological (b) Fibre (c) Films (d) MEMS device (e) Nanowires (f) Particles (g) Pattern surface (h) Porous sponge (i) Powder (j) Tips

Figure 6: The SEM dataset

Table 15: The SEM dataset: The segregation of the number of images per category is presented here.

Category	Number of images
Biological	973
Tips	1625
Fibres	163
Porous Sponge	182
Films Coated Surface	327
Patterned surface	4756
Nanowires	3821
Particles	3926
MEMS devices and electrodes	4591
Powder	918
Total	21282

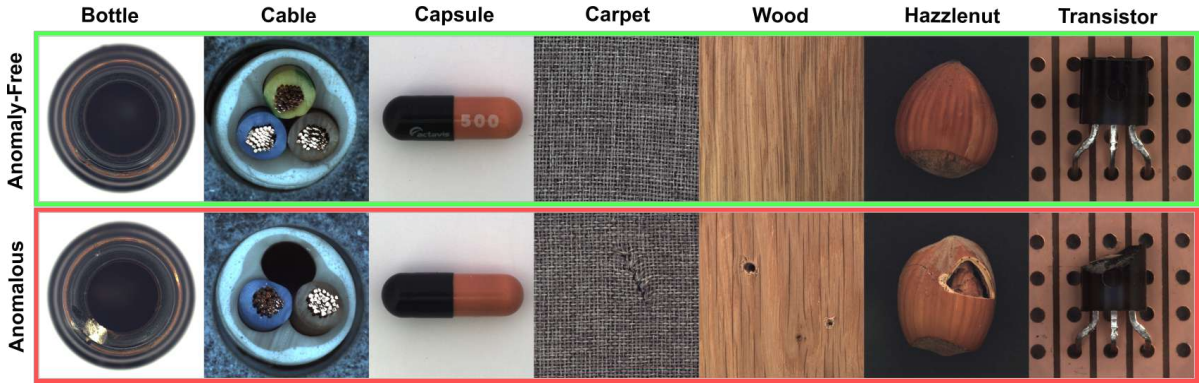


Figure 7: Illustration of seven catogories from MVTec anomaly detection dataset.

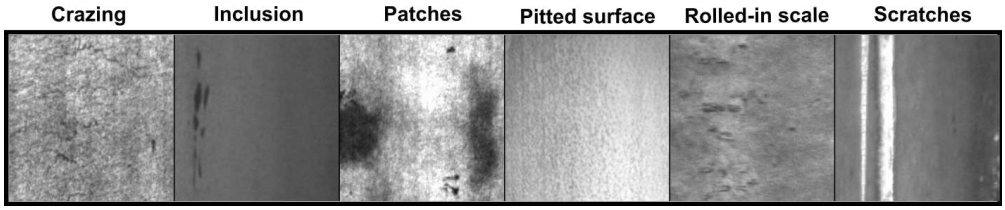


Figure 8: Illustration of six defect categories from NEU surface defect dataset of hot-rolled steel strip.

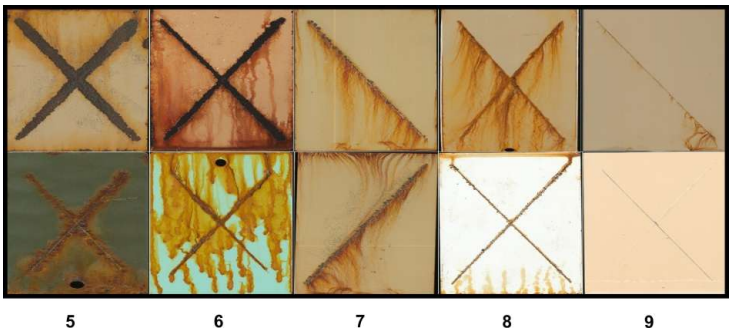


Figure 9: Illustration of five classes of corrosion rating assigned by ASTM

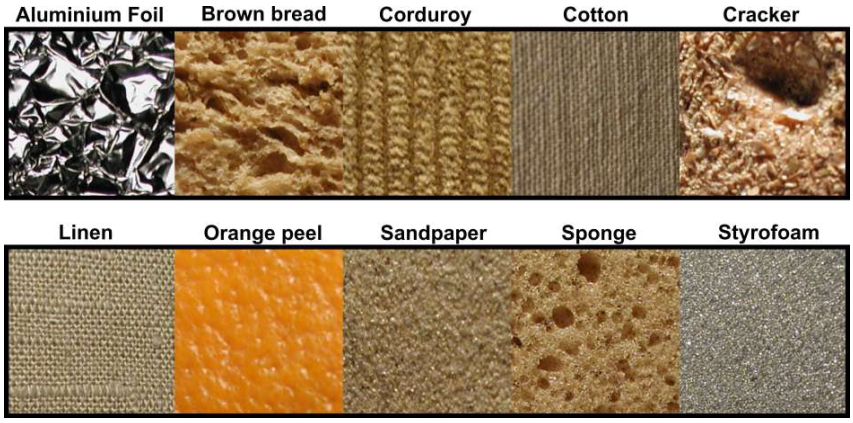


Figure 10: Illustration of the different materials from KTH-TIPS dataset

of the Top-1 classification score. In each experiment, we change the hyperparameter under investigation and the remaining hyperparameters are kept constant as described in section 5 to determine the effect of the hyperparameter on the model performance. The optimal hyperparameters determined from the hyperparameter search are as follows, d is 64, p_r is 0.75 and bs is 24.

Table 16: Table reports the hyperparameter study

d	32	64	96	128
	0.712 ± 0.05	**0.783 ± 0.012**	0.789± 0.07	0.791 ± 0.08
p_r	0.25	0.5	0.75	0.95
	0.667 ± 0.08	0.725± 0.03	**0.783± 0.012**	0.748± 0.06
bs	12	24	48	64
	0.706 ± 0.04	**0.783 ± 0.012**	0.790± 0.07	0.786± 0.05

9.3 Benchmarking on Material datasets

- **MVTec dataset**[1][4] is an open-sourced anomaly detection benchmark dataset of industrially driven products. The database contains ≈5000 high-resolution RGB images. The images are classified into 15 different categories of objects(e.g *bottle, cable, metal nut, hazelnut, toothbrush, capsule, pills, screw, transistor, zipper*) and textures(e.g *carpet, grid, leather, tile, and wood*). The training/validation set contains 3629 images. The test set contains the remaining images. The training set contains normal data(anomaly-free), whereas the test set contains both the normal and the anomalies(consists of different kinds of defects). A few representative images of a sample of defect categories, along with the corresponding defect-free images are shown in Figure 7. The dataset is highly class imbalanced. We evaluate the performance of our proposed method on the supervised anomaly detection task(binary classification) in comparison with several baseline algorithms.
- **NEU surface defect dataset(NEU-SDD)** [2][18] contains 1800 gray-scale images of hot-rolled steel strip. The database is classified into six classes of surface defects that includes *crazing, inclusion, patches, pitted surface, rolled-in scale, and scratches*. Each class has 300 sample images. A few representative images of the surface defects are shown in Figure 8. We evaluate the performance of our proposed method on the multiclass classification task on the balanced dataset in comparison with several baseline algorithms.
- **Corrosion Image Dataset(CMI)**[3] is utilized to benchmark various algorithmic approaches for automation of corrosion assessment in materials. The dataset contains 600 images of corroding panels of resolution 512×512 pixels. Each panel in the dataset is annotated by the experts to provide the measure of the corrosion through the ASTM standards. In general, the corrosion rating is assigned on a scale of discrete integers from 1 to 10. The corrosion rating of 10 implies that the panel is in the initial stage of corrosion. The data presenters projects it as a class-balanced dataset by rejecting the panels with high corrosion ratings(1-4). Thus the dataset is classified into five corrosion ratings(5-9). Each corrosion rating based image category has 120 images. The sample images per corrosion rating are shown in Figure9. We evaluate the performance of our proposed method on the multi-class classification task in comparison with several baseline algorithms.
- **KTH-TIPS**[4] is a texture database which contains images of varying illumination, pose and scale of ten different materials. It contains a sample of 810 images belonging to different classes of materials such as *crumpled aluminum foil, brown bread, corduroy, cotton, cracker, linen, orange peel, sandpaper, sponge, and styrofoam.* A few example images per category are shown in Figure10. The image resolution in the dataset is 200×200 pixels. We evaluate the performance of our proposed method on the multi-class classification task in comparison with several baseline algorithms.

We report the results of the baseline algorithms and our model performance in Table 17 on the open-sourced benchmark datasets. The evaluation metric is the Top-1 classification accuracy. We achieve SOTA performance on all the datasets.

Table 17: Performance comparison on the datasets.

Algorithms		**MVTec**	**NEU-SDD**	**CMI**	**KTH-TIPS**
Baselines	ResNet	0.912	0.926	0.911	0.932
	GoogleNet	0.926	0.933	0.913	0.924
	SqueezeNet	0.931	0.947	0.937	0.958
	VanillaViT	0.954	0.953	0.955	0.966
	EMCNet	**0.983**	**0.978**	**0.971**	**0.992**

[1]Datasource: https://www.mvtec.com/company/research/datasets.
[2]Datasource: http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html
[3]https://arl.wpi.edu/corrosion_dataset
[4]https://www.csc.kth.se/cvap/databases/kth-tips/index.html