# HETEROGENOUS MEMORY AUGMENTED NEURAL NETWORKS

Zihan Qiu\* Tsinghua University

**Zhen Liu**University of Montréal

Shuicheng Yan BAAI **Shanghang Zhang** Peking University

Jie Fu<sup>†</sup> HKUST

# **ABSTRACT**

It has been shown that semi-parametric methods, which combine standard neural networks with non-parametric components such as external memory modules and data retrieval, are particularly helpful in data scarcity and out-of-distribution (OOD) scenarios. However, existing semi-parametric methods mostly depend on independent raw data points - this strategy is difficult to scale up due to both high computational costs and the incapacity of current attention mechanisms with a large number of tokens. In this paper, we introduce a novel heterogeneous memory augmentation approach for neural networks which, by introducing learnable memory tokens with attention mechanism, can effectively boost performance without huge computational overhead. Our general-purpose method can be seamlessly combined with various backbones (MLP, CNN, GNN, and Transformer) in a plug-and-play manner. We extensively evaluate our approach on various image and graph-based tasks under both in-distribution (ID) and OOD conditions and show its competitive performance against task-specific state-of-the-art methods. Code is available at https://github.com/qiuzh20/HMA.

# 1 Introduction

Semi-parametric methods, which parametrize the mapping from an input domain  $\mathcal{X}$  to an output domain  $\mathcal{Y}$  with both neural net parameters and non-parametric data, are widely used in a variety of tasks including but not limited to meta-learning [42, 53], energy-based models [2] and planning [27]. With a non-parametric component in the neural net architecture, the model may better incorporate priors like distances between data points and characterize attributes such as data uncertainty. A typical design is to retrieve data associated with the current input with k-nearest neighbors (kNN). For instance, kNN-LM [29] proposes to perform inference with data retrieval from a small batch of inputs, similar to the concept of support set used in few-shot learning [54]. In practice, the retrieval strategies and parameters, such as the number of nearest neighbors, must be carefully designed for performance and low computational overhead [16, 47].

Instead of using full datasets, one may augment the neural net architecture with *external* memory. With an external memory module, one can store a tiny amount of data points or latent features obtained through dataset distillation [60], random selection [31], or learned policies [47]. Especially with the attention mechanism employed by NPT [31], a model with external memory can utilize non-parametric components more flexibly than traditional methods such as Gaussian processes [12] due to the learnable dependencies instead of a fixed kernel function. Still, it suffers from the same scalability problem as the attention has to operate directly on a potentially large data set or features.

Inspired by some recent work that effectively learns a tiny amount of synthetic data for teaching student networks [60, 77], we propose Heterogeneous Memory Augmentation (HMA), a general approach for learning dependencies between data and augmenting networks without high computational overhead for various downstream tasks. HMA sequentially employs on the feature space of some backbone network (a) a real memory augmentation module (RMA), with classical memory augmentation methods following [42, 22], and (b) a synthetic memory augmentation module (SMA). Specifically, SMA learns compressed memory entries that encode *dataset-relevant* information and leverages attention across datapoints (ABD) which consider the cross-relationship between the input batch in a manner similar

<sup>\*</sup>Work done while interning at HKUST and BAAI: qzh11628@gmail.com

<sup>&</sup>lt;sup>†</sup>Corresponding author: jiefu@ust.hk

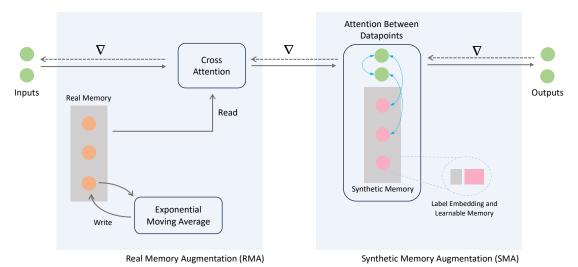


Figure 1: The overall framework of HMA. For inputs from some backbone, we concatenate them with label embeddings from **known** token similar to CLS. In RMA, a momentum memory queue and attention-based reader provide inputs with cross-batch information. During training, the real memory is updated with features from the momentum encoder and **true** label embeddings. In SMA, attention between datapoints is used to pass information among inputs and synthetic memory slots.

to NPT. These learned memory entries assist SMA in working effectively without relying on large batch sizes or additional selection methods. Notably, our HMA is architecture-agnostic and can be plugged into almost any backbone architecture in a few lines of code, without any additional training phases or losses.

In summary, our contributions are:

- We introduce heterogeneous memory augmentation (HMA), a better semi-parametric module that leverages both synthetic (learnable) and real (past data) memory. Compared to previous methods, our HMA can better capture the data dependencies for semi-parametric inference.
- The key component, synthetic memory augmentation (SMA), utilizes attention mechanisms to perform nonparametric inference on top of learned memory slots and inputs. The use of attention across datapoints allows the module to better capture the data dependencies 1) between the input batch and the memory slots as well as 2) between individual datapoints in the input batch, compared to traditional non-parametric techniques like Gaussian processes.
- Our model-agnostic HMA directly operates on the feature space and, therefore, can be incorporated into various backbones in a plug-in-play fashion.
- With learnable memory slots in SMA, HMA is more tailored to downstream tasks. Extensive experiments show its effectiveness in various in-distribution and out-of-distribution tasks.

# 2 Related Work

### 2.1 Semi-parametric Methods

Retrieval methods leverage the similarity between data points to provide information for prediction. For input x, the prediction probability is  $p = \lambda p_{\rm kNN} + (1-\lambda) p_{\rm BaseModel}$  to improve its ability to handle long-tail data and robustness [29]. During testing, the model can directly utilize similar training data for predictions. These characteristics make the retrieval method bring substantial improvements in various tasks of NLP [28, 71, 37] and graph tasks [59]. GNN-LM [40] also builds a graph by combining the samples with their corresponding retrieval data and further enhances the sample features through graph aggregation. Non-Parametric Classifier [44, 69] stores the training data's features and labels, enhancing similarity-based classification. Prototypical Networks [53, 45, 55] employ specifically designed objectives to learn a metric space. Each class's non-parametric "prototype representations" are easily obtained and used for few-shot and zero-shot tasks in this space. Compared to these methods, HMA learns synthetic memory slots directly with gradient descent updates, which can be more tailored for downstream tasks.

#### 2.2 Attention

Attention [58] can integrate various information and offer a versatile mechanism to uncover and exploit relationships between data. BatchFormer [25] performs attention in batches, allowing the model weights to learn the relationships. NPT [31] deploys Attention between Attributes and Attention between Datapoints to incorporate the learning of representation and relationships between data. However, ABD in NPT leads to a computational complexity of  $O(n^2)$ , which makes it difficult to incorporate a large amount of data and further hinders the feature learning process. To address the efficiency issue, [50] presents inducing points to avoid expensive full self-attention.

# 2.3 Neural Memory

Memory mechanisms can be used to discover and exploit the relationships between data through contrastive learning [22, 48]. Such relationships are also pervasive in meta-learning. [70] proposes a learnable memory to capture the meta-class information of semantic segmentation during the base class training procedure. Unlike the works mentioned above that use memory in a no-parametric fashion, [54] utilizes relevant features in the input to recall memory and uses them to modify the network parameters. Meta Networks [42] and Neural Stored-program Memory [35] directly store network parameters in different training stages in memory.

Soft prompts [39, 38, 52] can also be regarded as a form of learnable memory that captures task-relevant information during fine-tuning and aids in leveraging the knowledge within pre-trained models. L2P [64, 63] stores a series of soft prompts and reads them using key-value matching, resulting in remarkable performance during continued learning.

# 3 Preliminaries

**Attention mechanism** [58] is a fundamental component in deep learning models that enables the model to focus on relevant information selectively. It is commonly formulated as follows:

$$Att(Q, K, V) = softmax(QK^{T} / \sqrt{d_k})V, \tag{1}$$

where Q,K,V are stacked matrixes of queries, keys, and values. Typically, multi-head attention (MultiHA) is used for large model capacity. Using multiple attention modules in parallel, the model can capture different aspects of the input data and combine their results. The outputs of the attention heads are concatenated to yield the final output:

$$\operatorname{MultiHA}(Q,K,V) = \operatorname{concat}(\operatorname{head}_1, \cdots, \operatorname{head}_h), \text{ where } \operatorname{head}_i = \operatorname{Att}(QW_i^Q, KW_i^K, VW_i^V)$$

where  $W_i^Q, W_i^K$  and  $W_i^V$  are learnable matrices for the  $d_k$ -dimensional queries, keys and values for i-th head. By AB with  $A \in \mathbb{R}^{n \times a \times b}$  and  $B \in \mathbb{R}^{n \times b \times c}$ , we mean batched matrix multiplication in Einstein summation notation einsum(A, B, `nab,nbc->nac'). The output of MHA is computed with additional components, including feed-forward layer (FF), layer normalization (LN) and residual connection:

$$MHA(Q, K, V) = H + FF(LN(H)), \text{ where } H = V + MultiHA(LN(Q), LN(K), LN(V))$$
(2)

Non-parametric transformer (NPT) [31] extends the standard transformer in a semi-parametric way by leveraging the dependencies between datapoints with the attention among large batched inputs. Specifically, NPT first encodes the input batch  $H \in \mathbb{R}^{bs \times d}$  and reshape the output into  $H' = \mathbb{R}^{1 \times bs \times (d*h)}$ , and then applies attention across datapoints (ABD) by treating the input batch dimension as the token dimension:

$$H'_{ABD} = ABD(H) = MHA(H', H', H') \in \mathbb{R}^{1 \times bs \times (d*h)}$$
(3)

Once information has been exchanged between datapoints in the input batch, the ABD output  $H'_{ABD} \in \mathbb{R}^{bs \times 1 \times (d*h)}$  is reshaped back into  $H_{ABD} \in \mathbb{R}^{bs \times d \times h}$  to apply a self-attention operation on the feature dimension d, which we term attention across attributes (ABA) to distinguish it from ABD:

$$ABA(H_{ABD}) = MHA(H_{ABD}, H_{ABD}, H_{ABD}) \in \mathbb{R}^{bs \times d \times h}$$
(4)

ABD and ABA can be iteratively applied for several times before feeding the resulted (and reshaped) feature into a classifier or some other neural net.

# 4 Heterogenous Memory Augmentation

HMA performs two consecutive steps on the extracted feature batch 4.1: 1) real memory augmentation (RMA) 4.2, and 2) synthetic memory augmentation (SMA) 7.4.2. As semi-parametric methods heavily rely on the quality of non-parametric components, we employ RMA to *expand the non-parametric part implicitly*. Subsequently, we combine it with synthetic memory and apply attention between datapoints. The final output feature from SMA is fed into a linear projection head to compute logits for classification. We summarize HMA in the supplementary material.

#### 4.1 Feature Extraction

Given input X and label Y with batch size bs, We compute the feature  $h^1 = E(X) \in \mathbb{R}^{bs \times d_1}$ , where  $E(\cdot)$  can be any backbone, and  $d_1, d_2$  is the feature embedding dimension. In line with the setup of NPT, we incorporate labels as attributes. However, directly adding the embeddings of the true labels would leak information. To address this, for an n-classification problem, we replace the true labels Y with an additional label, denoted as n+1, and treat it as the CLS token as common in transformer literature [14]. We compute label embedding  $e = E_{\text{label}}(\mathbf{n+1}) \in \mathbb{R}^{bs \times d_2}$ , where  $d_2$  is the dimension of the label embedding. After concatenating, we obtain aggregated features  $C^1 = [h^1; e] \in \mathbb{R}^{bs \times d}$ ,  $d = d_1 + d_2$ . This concatenated representation  $C^1$  is used for later processes.

# 4.2 Real Memory Augmentation (RMA)

The properties of non-parametric components greatly influence the effectiveness of semi-parametric methods. NPT uses large batch sizes to reduce the influence of randomly selected non-parametric components. Retrievers ensure reference data quality through top-k selection. We aim to develop a method that effectively provides information beyond individual batches.

Drawing inspiration from MoCo [22], which employs a momentum-updated memory queue to provide stable and extensive contrastive samples, we introduce a real memory buffer  $M_{\text{buffer}} \in \mathbb{R}^{m_1 \times d}$  with size  $m_1$  (this buffer is empty at the beginning) and a momentum encoder  $E_m$ .

**Reading.** We utilize MHA2 to extract information from the memory buffer, thereby implicitly providing cross-batch information for subsequent SMA. To achieve this, we first transform  $M_{\text{buffer}}$  from  $\mathbb{R}^{m_1 \times d}$  to  $\mathbb{R}^{1 \times m_1 \times d}$  and  $C^1$  from  $\mathbb{R}^{bs \times d}$  to  $\mathbb{R}^{bs \times 1 \times d}$ . Then, we repeat  $M_{\text{buffer}}$  bs times in the first dimension to obtain  $M'_{\text{buffer}} \in \mathbb{R}^{bs \times m_1 \times d}$ . Subsequently, we concatenate  $M'_{\text{buffer}}$  and  $C^1$  along the second dimension, resulting in  $C^1_{\text{aug}} \in \mathbb{R}^{bs \times 1 + m_1 \times d}$ . RMA computes the output feature  $C^2$  by:

$$C^2 = \text{MHA}(C^1, C_{\text{aug}}^1, C_{\text{aug}}^1) \in \mathbb{R}^{bs \times 1 \times d}$$

$$\tag{5}$$

After RMA,  $C^2$  incorporates information from real data beyond a single batch. In 5.3, we observed that using RMA alone seldom brings improvements. Moreover, while using SMA alone can lead to some improvements, adding RMA achieves pronounced improvements.

Writing. We insert new aggregated input features  $[E_m(X), E_{\text{label}}(Y)] \in \mathbb{R}^{bs \times d}$  into the memory queue, and update the momentum encoder momentum update rule as in MoCo:  $E_m^{t+1} = \lambda E_m^t + (1-\lambda)E^{t+1}$ , in which  $E^{t+1}$  is the SGD-updated encoder from  $E^t$ .

# 4.3 Synthetic Memory Augmentation (SMA)

In NPT, the ABD mechanism allows models to utilize non-parametric components flexibly. We can also leverage this mechanism to introduce learnable synthetic memory, which captures dataset-relevant information. These synthetic memories also serve as compact non-parametric components for the ABD mechanism, eliminating the need for NPT's reliance on large batch sizes.

Although we can directly initialize a series of learnable parameters as memory slots with dimensions  $d_1+d_2$ , we also aim to incorporate label information into the inference process of SMA. Specifically, for each category  $i\in[n]$ , we initialize  $m_2$  learnable memory slots. Each memory slot is in  $\mathbb{R}^{d_1}$  and has corresponding label  $y_i$ . With the label embedder  $E_{\mathrm{emb}}$ , we map the label  $y_i$  to the label embedding  $e_{\mathrm{syn},i}\in\mathbb{R}^{d_2}$ . By concatenating all learnable memory slots and label embeddings, we obtain a learnable memory  $M_{\mathrm{SMA}}\in\mathbb{R}^{(n*m_2)\times(d_1+d_2)}$ . We reshape  $C^2$  and  $M_{\mathrm{SMA}}$ , and concatenate them to obtain  $C_{\mathrm{aug}}^2\in\mathbb{R}^{1\times(bs+m_2*n)\times d}$ . The SMA output features are obtained by

$$C^3 = \text{MHA}(C^2, C_{\text{aug}}^2, C_{\text{aug}}^2) \in \mathbb{R}^{1 \times bs \times d}.$$
 (6)

Finally, we use a linear projection head to compute the logits from  $C^3$ . During the training phase, the learnable memory slots are updated by gradient, while the label embeddings are regenerated using the updated embedder  $E_{\rm emb}$ .

# 5 Experiments

In this section, we evaluate the performance of HMA and explore its characteristics. Section 5.1 demonstrates that HMA can be combined with various backbones and improve image and graph tasks. Following the success of semi-parametric methods on OOD tasks, in section 5.2, we examine HMA's out-of-distribution generalization capabilities under different distributional shifts. Finally, in Section 5.3, we demonstrate the characteristics of both real and synthetic memory slots. Ablation study shows while ABD itself can bring improvements, with the help of real memory and synthetic memory slots, HMA consistently achieves competitive performance against task-specific state-of-the-art methods.

**Experiment Setup.** Our experiments strictly follow the basic settings like batch size, learning rate schedule, number of epochs, optimizer, evaluation metric, and test model selection in previous works for a fair comparison. In CIFAR-10 task, we use the codebase<sup>3</sup>. Each method is trained 200 epochs, and the final test accuracies over three random seeds are reported. In tuning ViT [15] task, we follow [52] to tune the ViT-B/32 model, which was pretrained on Imagenet-21K [13]. In Colored MNIST task 2, we use code base from [3]. In obg molecular property prediction tasks 3, we use the official code base. In real-world graph ID tasks 4 and graph OOD tasks 65, we follow the official implementation of G-mixup [62] and CIGA [21]. More details can be found in the supplementary material.

**Table Notations.** We have **bolded** the best results and added an <u>underline</u> to the second-best results in all the tables. To present the effects of different components in HMA and provide a comprehensive view, we included the results of ablations for each component in the subsequent tables: 1) RMA represents only real memory augmentation from 4.2. 2) ABD and 3) ABD+SYN represent the use of only attention between datapoints from 7.4.2 without and with learnable synthetic memory, respectively. 4) ABD+RMA represents using attention between datapoints and RMA.

# 5.1 HMA is a General Augmentation Mechanism

We initially tested HMA on CIFAR-10. We adopted a ResNet-18 [23] encoder, similar to the one used in NPT [31]. The original NPT framework reported a performance of 93.7% on CIFAR-10, which is even lower than using ResNet alone (93.9%). HMA achieves an accuracy of 95.54±0.11%, slightly outperforming the backbone ResNet (which achieves 95.43±0.01%). We acknowledge that minor differences may be attributed to the fact that this setting has already been well-studied.

To evaluate the performance of HMA on larger models and its adaptability to pre-trained models, we conducted experiments following the setup of ViT [15] on the pets37 [46], flowers102 [43], CIFAR100 [32], and DTD [10] datasets. We performed a simple hyperparameter search for HMA using different values of  $m_1 = 1, 2, 4 \times bs$  and  $m_2 = 4, 8, 16$ . In addition to directly tuning the ViT backbone, we compared HMA with the SOTA methods [52] that introduced soft prompts (referred to as learnable memory in the paper). "prompt-p" in the table indicates adding soft prompts only at the input layer, similar to p-tuning [39]. "prompt-a" refers to independently adding soft prompts at each layer of ViT, similar to p-tuning v2 [38]. PARAM refers to the ablation where non-parametric memory is not introduced, and only the parameters of the MHA 2 module are used. We report the mean and standard deviation for 3 seeds. The results are summarized in Table 1.

Table 1: Accuracy of full fine-tuning pretrained ViT on 4 image datasets. PROMPT-P and PROMPT-A indicate adding soft prompts only at the input layer and at each layer, respectively. PARAM is the parameter ablation. Other notations are introduced in 5.

| DATASET  | DTD   | PETS37   | FLOWERS102  | CIFAR100   | AVG   |
|--|---|--|---|--|---|
| VIT<br>PROMPT-P<br>PROMPT-A                      | $75.11 \pm 0.24 75.69 \pm 0.56 75.35 \pm 0.81$  | $\begin{array}{c} 90.71 \pm 0.07 \\ 90.49 \pm 0.22 \\ 90.47 \pm 0.28 \end{array}$                        | $98.48 \pm 0.40$<br>$98.59 \pm 0.22$<br>$98.42 \pm 0.11$                                  | $\begin{array}{c} 91.48{\pm}0.27 \\ 91.61{\pm}0.14 \\ 91.62{\pm}0.07 \end{array}$  | 88.95<br>89.09<br>88.97                                   |
| PARAM<br>RMA<br>ABD<br>ABD+RMA<br>ABD+SYN<br>HMA | $75.12\pm0.67$ $75.67\pm0.16$ $75.89\pm1.04$ $75.69\pm0.40$ $75.69\pm0.75$ <b>76.05</b> $\pm0.39$ | $90.27\pm0.05$<br>$89/96\pm0.31$<br>$90.66\pm0.20$<br>$90.54\pm0.77$<br>$90.09\pm1.10$<br>$91.08\pm0.22$ | $98.46\pm0.29$ $98.61\pm0.05$ $98.57\pm0.05$ $98.69\pm0.14$ $98.49\pm0.15$ $98.70\pm0.16$ | $\begin{array}{c} \underline{91.69} {\pm 0.08} \\ \underline{91.63} {\pm 0.31} \\ \underline{91.65} {\pm 0.14} \\ \underline{91.63} {\pm 0.19} \\ \underline{91.72} {\pm 0.08} \\ \underline{\textbf{91.93}} {\pm 0.19} \end{array}$ | 88.89<br>88.97<br>89.19<br>89.14<br>89.00<br><b>89.44</b> |

From Table 1, we can observe that the complete HMA consistently achieves better results than the best prompt tuning. The inferior performance of prompt tuning may be attributed to the interference caused by inserting new randomly initialized parameters. In contrast, HMA operates solely in the feature space of the backbone's output, avoiding such interference and allowing for more effective utilization of the original parameters.

We also conducted experiments following [52] in the scenario where only the CLS token and head of the pretrained model are tuned, and the results are shown in Table 2. Since HMA and prompt tuning are orthogonal techniques, we also examined the effects of adding HMA on top of prompt tuning. Aloughth HMA, operating solely in the feature space, does not show as significant improvement as prompt tuning when most of the model parameters are fixed, it can be combined with prompt tuning to achieve more pronounced improvements.

<sup>&</sup>lt;sup>3</sup>https://github.com/kuangliu/pytorch-cifar

Table 2: Accuracy of only tuning CLS token and head of ViT. +HMA indicates adding HMA to the original method. The results on the right side of  $\rightarrow$  represent the performance after adding HMA.

| DATASET       | DTD   | PETS37  |  |
|---------------|---|---|--|
| VIT→+HMA      | $74.01\pm0.29 \rightarrow 74.91\pm0.39$             | $89.17\pm0.07 \rightarrow 89.72\pm0.22$             |  |
| PROMPT-P→+HMA | $73.98\pm0.56 \rightarrow 73.95\pm0.73$             | $89.22\pm0.22 \rightarrow 89.58\pm0.60$             |  |
| PROMPT-A→+HMA | $\underline{74.75}\pm0.86 \rightarrow 75.44\pm0.06$ | $\underline{89.88}\pm0.28 \rightarrow 90.71\pm0.27$ |  |

To further validate the effectiveness of HMA with diverse backbones, we combine HMA with GCN [30], GIN [72] and their variants GCN-v, GIN-v[18, 36] backbones on three molecular property prediction tasks on the ogb benchmark [26]. The results are summarized in Table 3. HMA brings improvement among 10 of 12 reported AUROCs.

Table 3: AUROC and standard deviation of backbones and HMA on ogb benchmark. We have marked the relative improvements over the backbone with an upward arrow (↑) for the average results.

| DATASET            | BACE                               | BBBP                               | HIV                                | AVG                   |
|--------------------|------------------------------------|------------------------------------|------------------------------------|-----------------------|
| GCN<br>GCN+HMA     | $79.15\pm1.44$<br>$80.59\pm1.15$   | $68.87{\pm}1.51 \\ 69.96{\pm}0.55$ | $76.06{\pm}0.97 \\ 77.26{\pm}1.42$ | 74.69<br>75.85(†1.16) |
| GCN-V<br>GCN-V+HMA | 68.93±6.95<br>72.53±1.36           | $67.80{\pm}2.35 \\ 68.26{\pm}0.85$ | $75.99{\pm}1.19 \\ 76.51{\pm}1.70$ | 70.91<br>72.73(†1.82) |
| GIN<br>GIN+HMA     | $72.97{\pm}4.00 \\ 77.18{\pm}2.20$ | 68.17±1.48<br>69.47±1.50           | $75.58{\pm}1.40 \\ 76.47{\pm}0.17$ | 72.24<br>74.38(†2.14) |
| GIN-V<br>GIN-V+HMA | $73.46 \pm 5.24 \\ 76.56 \pm 1.38$ | 69.71±1.92<br>68.43±1.19           | 77.07±1.40<br>75.83±1.23           | 73.41<br>73.61(†0.20) |

Table 4: Accuarcies of different augmentation methods on real-world graphs. We strictly follow the official implementation and report the baseline results from G-mixup [21].

| DATASET                                       | IMDB-B   | IMDB-M   | REDD-B   | REDD-M5  | REDD-M12  | AVG  |
|---|--|--|--|--|---|--|
| VANILLA D-EDGE D-NODE S-GRAPH M-MIXUP G-MIXUP | $71.55{\pm}3.53 \\ 72.20{\pm}1.82 \\ 72.16{\pm}0.28 \\ 68.50{\pm}0.86 \\ 70.83{\pm}1.04 \\ \underline{71.94}{\pm}3.00$ | 48.83±2.75<br>48.83±3.02<br>48.33±0.98<br>47.25±3.78<br>49.88±1.34<br><b>50.46</b> ±1.49                   | 92.59±0.86<br>92.00±1.13<br>90.25±0.98<br>90.33±0.87<br>90.75±1.78<br><b>92.90</b> ±0.87 | $55.19\pm1.02$<br>$55.10\pm0.44$<br>$53.26\pm4.99$<br>$54.60\pm3.15$<br>$54.95\pm0.86$<br>$55.49\pm0.53$ | $50.23\pm0.83$ $49.77\pm0.76$ $49.95\pm1.70$ $49.67\pm0.90$ $49.81\pm0.80$ $50.50\pm0.41$                       | 63.68<br>63.58<br>62.79<br>62.07<br>63.24<br>64.25 |
| RMA<br>ABD<br>ABD+SYN<br>HMA                  | 71.17±4.75<br>70.67±3.25<br>71.67±3.25<br><b>72.62</b> ±0.48   | $\begin{array}{c} 49.89 \pm 2.50 \\ \hline 47.44 \pm 1.84 \\ 48.97 \pm 2.97 \\ 49.78 \pm 0.74 \end{array}$ | $91.92\pm1.70$<br>$92.42\pm0.14$<br>$91.08\pm1.23$<br>$92.80\pm1.37$                     | $55.23\pm0.58$<br>$54.87\pm0.45$<br>$55.90\pm0.95$<br>$56.37\pm1.65$                                     | $\begin{array}{c} 49.32{\pm}1.45 \\ 49.08{\pm}0.86 \\ 49.29{\pm}1.00 \\ \underline{50.27}{\pm}0.60 \end{array}$ | 63.51<br>62.90<br>63.38<br><b>64.37</b>            |

Furthermore, we follow G-mixup [21] and compare it with various augmentation methods, including DropEdge [51], DropNode [74], Subgraph [61], and Manifold-Mixup [62] on different datasets. All methods use the same GIN [72] backbone. The results are summarized in Table 4. We can see the three components of HMA have a similar effect to CIFAR-10, and full HMA consistently outperforms vanilla GIN backbone. By doing meaningful mixup with the help of elaborately designed graphon, G-mixup achieves better performance than simply mixing graphs in feature space [62]. Although HMA also only operates in feature space, it significantly outperforms Manifold-Mixup and shows competitive results to G-Mixup.

# 5.2 HMA Performs Competitively on OOD Tasks

We test the OOD performance of HMA on the Colored MNIST dataset introduced in IRM [3]. This task is distinct from traditional MNIST as it introduces misleading correlations between labels and colors. Specifically, digits 0-4 and 5-9

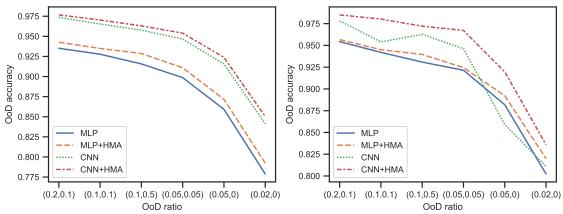


Figure 2: OOD results for colored MNIST (left: original size, right: downsized) experiments. We change the OOD sample ratio from (0.2, 0.1) to (0, 0.02)

are considered two separate classes. Two domains  $p_1, p_2$  can be accessed during training: in the first domain, green digits have a probability of  $p_1$  to be in 0-4; in the second, the probability is  $p_2$ . During testing, there are two sets: 1. reversed color: green digits have a 90% chance of being 0-4; 2. gray: blue and red images are turned into gray. The original image size (28,28) is downsized for computational convenience to (14,14). HMA is tested with MLP from IRM [3] and CNN from [19] under both downsized and original image sizes. We modify  $p_1, p_2$  from (0.2, 0.1) to (0, 0.02) to more accurately demonstrate the model's performance with distribution shifts.

This task provides a proof of concept for the semi-parametric approach in addressing out-of-distribution (OOD) tasks. While there is a distribution shift, there are still certain data points within the training set that share the same distribution as the test data. The semi-parametric approach effectively reduces the distribution shift by leveraging these data points as a support set. The average accuracy of the two test domains is presented in Figure 2. With different backbones, image sizes, and distribution shifts, we can observe that HMA consistently improves OOD performance.

In the following graph tasks, we follow CIGA [8] to examine HMA on diverse distribution shits. Our baselines include ERM [57], SOTA interpretable GNNs like ASAP Pooling[49], and DIR[66], and SOTA OOD objectives like IRM[3], v-Rex[33], and IB-IRM[1].

In Table 6, we examine HMA on graph size shift scenarios on datasets converted from TU benchmarks[41]. Distribution shifts are generated through size-specific dataset splits following [8] [73]: the training set comprises graphs with sizes smaller than the 50th percentile, 10% of which are held-out for the validation set, and the test set consists of those with sizes larger than the 90th percentile. Matthews correlation coefficient [4] is adopted due to the class imbalance. In Table 5, we further test HMA on larger real-world datasets with more complicated node degree biases. We use the Graph-SST

Table 5: Acc of OOD algorithms on averaged degrees shift.

| DATASET                                    | SST5   | TWITTER   | AVG                                       |
|--|--|---|---|
| ERM<br>ASAP<br>DIR                         | $43.72\pm0.59$ $44.16\pm1.36$ $41.12\pm1.96$                               | $\frac{63.54}{60.68 \pm 2.10}$ $59.85 \pm 2.98$   | 53.63<br>52.42<br>50.49                   |
| IRM<br>V-REX<br>IB-IRM<br>CIGAV1<br>CIGAV2 | $43.69\pm1.26$ $43.28\pm0.52$ $40.85\pm2.08$ $43.70\pm1.98$ $43.30\pm0.90$ | $\begin{array}{c} 63.50{\pm}1.23 \\ 63.21{\pm}1.57 \\ 61.26{\pm}1.20 \\ 62.02{\pm}2.28 \\ 61.80{\pm}2.03 \end{array}$ | 53.60<br>53.25<br>51.06<br>52.68<br>52.55 |
| HMA  | 44.03±1.01   | <b>65.12</b> ±2.17  | 54.58                                     |

datasets following CIGA[8]. Graphs are assigned based on their average degree to generate distribution shifts: the training set includes graphs with average degrees smaller than the 50th percentile, the validation set includes those with average degrees larger than the 50th percentile and smaller than the 80th percentile, and the test set includes the left. Graphs in Twitter are assigned in an inversed order to investigate OOD generalization ability from large degree graphs to small ones.

ERM is a strong baseline for these real-world tasks compared with specifically designed algorithms [19]. Our HMA consistently improves ERM baselines without introducing complex OOD objectives or learning strategies.

### **5.3** Heterogenous Memory Shows Reasonable Properties

We use T-SNE [56] to visualize the relationships between the synthetic memory features and real data features in  $C^2$ . We excluded the label embeddings to focus on the synthetic memory slots since they would result in identical embeddings for instances with the same label and affect the T-SNE results.

Table 6: Matthews correlation coefficients of different OOD algorithms on graph size shifts for real-world graphs. We strictly follow CIGA [8] and reproduce the results of ERM, GIGAv1&v2.

| DATASET | NC11                          | NC109             | PROTEINS        | DD                          | AVG   |
|---------|-------------------------------|-------------------|-----------------|-----------------------------|-------|
| ERM     | $0.15 \pm 0.03$               | $0.16 \pm 0.07$   | $0.19 \pm 0.07$ | $0.29 \pm 0.1$              | 0.198 |
| ASAP    | $0.16\pm0.10$                 | $0.15 {\pm} 0.07$ | $0.22 \pm 0.16$ | $0.21 \pm 0.08$             | 0.185 |
| DIR     | $\underline{0.21} {\pm 0.06}$ | $0.13{\pm}0.05$   | $0.25{\pm}0.14$ | $0.20\pm0.10$               | 0.198 |
| IRM     | $0.17 \pm 0.02$               | $0.14 \pm 0.01$   | $0.21 \pm 0.09$ | $0.22 \pm 0.08$             | 0.185 |
| V-REX   | $0.15{\pm}0.04$               | $0.15{\pm}0.04$   | $0.22{\pm}0.06$ | $0.21 \pm 0.07$             | 0.183 |
| IB-IRM  | $0.12{\pm}0.04$               | $0.15{\pm}0.06$   | $0.21 \pm 0.06$ | $0.15{\pm}0.13$             | 0.158 |
| CIGAv1  | $0.25{\pm}0.08$               | $0.26 \pm 0.09$   | $0.35 \pm 0.17$ | $0.21{\scriptstyle\pm0.12}$ | 0.268 |
| CIGAv2  | $0.28 \pm 0.10$               | $0.27 \pm 0.10$   | $0.30{\pm}0.09$ | $0.19{\pm}0.12$             | 0.260 |
| HMA     | $0.19 \pm 0.05$               | $0.18 \pm 0.04$   | $0.34 \pm 0.07$ | <b>0.32</b> ±0.03           | 0.258 |

We present the synthetic memory slots' relationships in Figure 3 (left). It can be observed that slots belonging to the same class cluster together, indicating that the synthetic memory slots can learn class-specific information with the help of label embeddings. In Figure 3 (right), we show the relationships between three classes of synthetic memory slots and their corresponding real data features. It can be observed that different classes of real data features form separate clusters, while three separate clusters in Figure 3 (left) merge into one. This suggests that the learned memory in the synthetic memory slots differs from the real data features. The improvement in 5.1 and 5.2 may come from the information beyond the instance level in the synthetic memory slots.

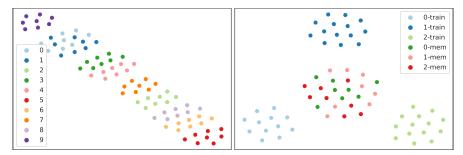


Figure 3: T-SNE on CIFAR-10 for synthetic memory features (left) and real data features with synthetic memory features (right). '0-train' and '0-mem' refer to real and memory data features, respectively. The preceding label corresponds to the class from CIFAR10.

We also visualize the synthetic memory slots features learned in graph tasks in Figure 4. Although the synthetic memory slots of class 1 scatter over other classes' clusters Figure 4 left, there is an interesting observation that the features of class 1 mix with the other three classes in the feature space correspondingly, as shown in Figure 4 right. The correspondence indicates that the synthetic memory slots might capture class-level information, similar to prototype representations [53].

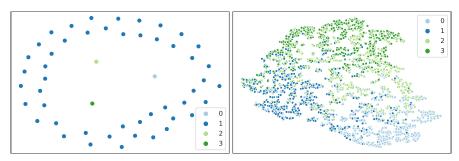


Figure 4: T-SNE on REDD-M5 for synthetic memory features (left) and input features (right)

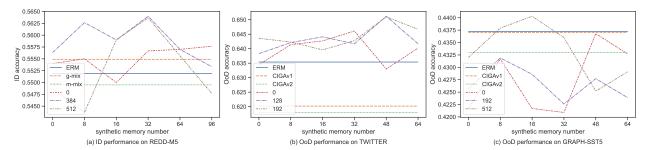


Figure 5: Performance of HMA on REDD-M5, Twitter and SST5 with different memory hyperparameters. The different colored dashed lines represent different sizes  $m_1$  of real Memory Buffer (0 means no RMA), and it can be seen that introducing  $m_1$  can make the model more stable; the horizontal axis represents different  $m_2$  (0 represents using ABD without synthetic memory)

**Ablation Study.** The in-distribution performances on REDD-M5 are depicted in Figure 5(a) for different memory configurations, while the out-of-distribution performances on Twitter and SST are illustrated in Figure 5(b) and (c) respectively. It can be seen that while a suitable increase in the synthetic memory size  $m_2$  can bring improvement, larger  $m_2$  will hurt HMA's performance. One potential explanation could be that a large  $m_2$  increases the optimization difficulty of ABD. Furthermore, it can be seen that with the addition of real memory augmentation, an increase in  $m_2$  provides more improvement.

From Table 1 and Table 4, we can find that singly using ABD or SMA proves beneficial in certain scenarios but not significant. The possible reason for this observation is the sensitivity of semi-parametric approaches. By solely relying on SMA without any data filtering, the quality of the non-parametric elements remains unassured. However, upon incorporating RMA, the effectiveness of HMA surpasses that of individual deployments of RMA and SMA. This aligns with our conjecture that RMA explicitly offers more information to the synthetic memory learning process.

# 6 Discussions, Conclusions, Limitations and Future work

**Two-stage Augmentation.** While it seems more natural to perform both RMA and SMA in one layer, *i.e.* to compute MHA with features aggregated from the input features, buffer features and the learnable features, the overall complexity is quadratic due to the SMA operations. It is possible to alternatively use linear approximations to standard attention (*e.g.*, Performer [9]), it is beyond our scope and we leave it to future work.

We propose a universal memory augmentation method that can be seamlessly integrated with various backbone architectures. SMA leverages parametric and non-parametric approaches in the complementary combination of Attention Between Datapoints and learnable memory, thereby enabling the capture of class-specific and task-aware information, which is beneficial for ID and OOD downstream tasks. Additionally, based on momentum memory queue and attention-based reading, RMA enhances SMA with cross-batch information. Our experimental results demonstrate that HMA can consistently improve the performance of backbones and performs competitively with task-specific best-performing designs.

# References

- [1] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 34:3438–3450, 2021.
- [2] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint* arXiv:2003.05033, 2020.
- [3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv* preprint arXiv:1907.02893, 2019.
- [4] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *International Conference on Machine Learning*, pages 837–851. PMLR, 2021.
- [5] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.
- [6] Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. Memory transformer. *arXiv preprint* arXiv:2006.11527, 2020.

- [7] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022.
- [8] Yongqiang Chen, Yonggang Zhang, Yatao Bian, Han Yang, MA KAILI, Binghui Xie, Tongliang Liu, Bo Han, and James Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. In *Advances in Neural Information Processing Systems*.
- [9] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Martin Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *accepted to ICLR 2021 (oral presentation)*, 2021.
- [10] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv* preprint arXiv:1901.02860, 2019.
- [12] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.
- [13] 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.
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* preprint arXiv:1810.04805, 2018.
- [15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [16] Andrew Drozdov, Shufan Wang, Razieh Rahimi, Andrew McCallum, Hamed Zamani, and Mohit Iyyer. You can't pick your neighbors, or can you? when and how to rely on retrieval in the *k* nn-lm. *arXiv preprint arXiv:2210.15859*, 2022.
- [17] Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. Augmenting transformers with knn-based composite memory for dialog. *Transactions of the Association for Computational Linguistics*, 9:82–99, 2021.
- [18] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [19] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020.
- [20] Ankit Gupta and Jonathan Berant. Gmat: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274*, 2020.
- [21] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. *arXiv preprint arXiv:2202.07179*, 2022.
- [22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
- [25] Zhi Hou, Baosheng Yu, and Dacheng Tao. Batchformer: Learning to explore sample relationships for robust representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7256–7266, 2022.
- [26] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems, 33:22118–22133, 2020.
- [27] Peter Humphreys, Arthur Guez, Olivier Tieleman, Laurent Sifre, Théophane Weber, and Timothy Lillicrap. Large-scale retrieval for reinforcement learning. Advances in Neural Information Processing Systems, 35:20092–20104, 2022.

- [28] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.
- [29] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.
- [30] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
- [31] Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 34:28742–28756, 2021.
- [32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [33] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.
- [34] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. General multi-label image classification with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16478–16488, 2021.
- [35] Hung Le, Truyen Tran, and Svetha Venkatesh. Neural stored-program memory. *arXiv preprint arXiv:1906.08862*, 2019.
- [36] Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017.
- [37] Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. Bertgen: Transductive text classification by combining gen and bert. *arXiv preprint arXiv:2105.05727*, 2021.
- [38] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.
- [39] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.
- [40] Yuxian Meng, Shi Zong, Xiaoya Li, Xiaofei Sun, Tianwei Zhang, Fei Wu, and Jiwei Li. Gnn-lm: Language modeling based on global contexts via gnn. *arXiv preprint arXiv:2110.08743*, 2021.
- [41] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.
- [42] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.
- [43] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.
- [44] Emin Orhan. A simple cache model for image recognition. *Advances in Neural Information Processing Systems*, 31, 2018.
- [45] Frederik Pahde, Mihai Puscas, Tassilo Klein, and Moin Nabi. Multimodal prototypical networks for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2644–2653, 2021.
- [46] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [47] Guangyue Peng, Tao Ge, Si-Qing Chen, Furu Wei, and Houfeng Wang. Semiparametric language models are scalable continual learners. *arXiv preprint arXiv:2303.01421*, 2023.
- [48] Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*, 2020.
- [49] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

- [50] Richa Rastogi, Yuntian Deng, Ian Lee, Mert R Sabuncu, and Volodymyr Kuleshov. Semi-parametric deep neural networks in linear time and memory. *arXiv preprint arXiv:2205.11718*, 2022.
- [51] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.
- [52] Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Andrew Jackson. Fine-tuning image transformers using learnable memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12155–12164, 2022.
- [53] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.
- [54] Pablo Sprechmann, Siddhant M Jayakumar, Jack W Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. Memory-based parameter adaptation. arXiv preprint arXiv:1802.10542, 2018.
- [55] Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. Hierarchical attention prototypical networks for few-shot text classification. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 476–485, 2019.
- [56] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [57] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.
- [58] 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.
- [59] Dingmin Wang, Shengchao Liu, Hanchen Wang, Bernardo Cuenca Grau, Linfeng Song, Jian Tang, Song Le, and Qi Li. Retrieval-enhanced graph neural networks for graph property prediction. 2022.
- [60] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. arXiv preprint arXiv:1811.10959, 2018.
- [61] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Graphcrop: Subgraph cropping for graph classification. *arXiv preprint arXiv:2009.10564*, 2020.
- [62] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, pages 3663–3674, 2021.
- [63] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 631–648. Springer, 2022.
- [64] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.
- [65] Qingyang Wu, Zhenzhong Lan, Jing Gu, and Zhou Yu. Memformer: The memory-augmented transformer. arXiv preprint arXiv:2010.06891, 2020.
- [66] Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*, 2022.
- [67] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. arXiv preprint arXiv:2203.08913, 2022.
- [68] Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. An efficient memory-augmented transformer for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2210.16773*, 2022.
- [69] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.
- [70] Zhonghua Wu, Xiangxi Shi, Guosheng Lin, and Jianfei Cai. Learning meta-class memory for few-shot semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 517–526, 2021.

- [71] Jitao Xu, Josep M Crego, and Jean Senellart. Boosting neural machine translation with similar translations. In *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track)*, pages 282–292, 2022.
- [72] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.
- [73] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.
- [74] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.
- [75] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.
- [76] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.
- [77] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023.
- [78] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *ICLR*, 1(2):3, 2021.

# 7 Appendix

### 7.1 Discussions, Limitations and Future works

Our work also shares similarities with the approach in [34] that both incorporate labels into the learning process and utilize dependencies within the data. However, [34] explores the presence of different class labels within the same image, for example, 'dolphin is unlikely to co-occur with grass, while a knife is more likely to appear next to a fork', while HMA upon the observation in NPT and consider the correlations between different data points to improve performance.

We have observed that HMA brings more significant improvements when the dataset size is small. This may be due to the current design of HMA, which limits the capacity of the non-parametric components or struggles to capture complex information in larger datasets. In some scenarios, HMA's performance is still weaker than the SOTA methods. This could be attributed to the "trade-off" between efficiency and effectiveness in the two stages of HMA's design: in SMA, we introduce learnable memory tokens to approximate dataset-related information, while in RMA, we use a momentum memory buffer to approximate cross-batch information. These design choices allow HMA to provide improvements when combined with various backbones without significant computational overhead.

Future work could focus on more finely constructing the non-parametric components. For example, using Dataset Distillation [60] to initialize learnable memory, introducing a retriever [47] to enhance the quality of the non-parametric components, incorporating contrastive loss to assist in non-parametric learning, and obtaining memory with improved interpretability. Furthermore, introducing learnable non-parametric components like HMA can also be applied to scenarios and used to improve the efficiency of semi-parametric methods like retrievers [47].

### 7.2 Related Work – Continued

### 7.2.1 Neural Memory

Memory mechanism is widely used due to its capacity to process long-dependency information. LSTM [24], Transformer XL [11], Recurrent Memory Transformer [5] enable the model to handle long context information by passing hidden states. [75, 20, 6] incorporate global memory tokens to reduce the computational cost of full self-attention and synthesize information within the same window. Memformer [65], Memorizing Transformer [67] introduces external memory to mitigate the issues caused by the limited transformer window size. Other works [68, 17] use the preprocessed knowledge base as additional memory, enabling the model to access more knowledge in dialogue and question-answering tasks, analogous to retrieval methods.

While there are many similarities between the synthetic memory in HMA and other learnable memories or soft prompts, our method exhibits two key differences: 1. We introduce learnable synthetic memory slots specifically in the semi-parametric ABD process to capture *dataset-relevant* information effectively. 2. The synthetic memory learned during

the ABD process aids the semi-parametric approach without relying on large batch sizes or additional selection methods. Furthermore, experimental results indicate that with the indirect guidance of label embeddings, the learned synthetic memory in HMA also contains class-specific information.

### 7.2.2 Dataset Distillation

Dataset distillation (DS) attempts to condense the information of an entire dataset into a few synthetic images. Initial work [60] treated the synthetic images as hyperparameters and used gradient-based hyperparameter optimization methods to update them. Subsequent studies [78, 76, 7] shift the optimization goal to minimizing the training gradient differences. However, these methods have two drawbacks: they are expensive due to the extra networks and second-order gradients, and the obtained synthetic images are only focused on aiding network training, which could be seen as a *condensed gradient* with no direct connection to downstream tasks. [60, 77] optimized the synthetic images by reducing the distribution distance in the feature space. Drawing inspiration from DS, we use class-specific synthetic memory slots in ABD to capture information beyond individual data points. Moreover, our synthetic memory slots are optimized with gradients directly from the current task.

### 7.3 Experiments – Continued

# 7.3.1 Training Details

The training process for all experiments remains consistent with the original experimental settings, allowing HMA to be directly integrated with various backbones.

In the ViT tuning experiments, the baselines follow the original ViT's [15] training step settings (CIFAR100=10000, pets37=500, flowers102=500, dtd=1000). The learning rate was searched within the range of [0.001, 0.003, 0.01, 0.03]. In the first few HMA experiments, we found that lr=0.003 in full-tuning experiments and lr=0.01 in part-tuning experiments yielded satisfactory results. Therefore, for all HMA experiments, only these two learning rates were used respectively to reduce hyperparameter searches. For the size of the real memory buffer  $m_1$ , we searched within the range of the original experiment's default batch size multiplied by [1, 2, 3]. For the number of synthetic memories  $m_2$ , we searched within [4, 8, 16, 32]. For the label embedding size  $d_2$ , we used  $d_2 = 64$  in ViT and all subsequent experiments.

In the remaining experiments, all hyperparameters were kept the same as the original settings, and the two hyperparameters introduced by HMA,  $m_1$  and  $m_2$ , were searched similarly as in ViT. Specifically, in the graph experiments, where the number of label categories is smaller, we adjusted the search range of  $m_2$  to [8, 16, 32, 64].

### 7.3.2 Computation

Experiments were conducted on NVIDIA Tesla P100 16 GB GPUs (one experiment was conducted on one GPU).

In the tuning experiment of ViT [15] on the DTD [10] dataset, the baseline, which involves tuning the head and the CLS token, has 87.5 M parameters. Among them, there are 36.9 K trainable parameters. These whole occupy approximately 4539 MiB of GPU memory. The training process for 1000 steps takes around 360 seconds. After incorporating HMA, the total number of parameters increases to 119 M, with 32.4 M trainable parameters. It occupies approximately 5159 MiB of GPU memory, and the training time for 1000 steps is comparable to the backbone alone without any significant difference. Although it introduces additional parameters, our ablation study demonstrates that directly increasing the number of parameters may lead to worse performance. Additionally, the additional memory overhead caused by HMA is relatively small.

It is worth noting that the number of parameters introduced by HMA depends only on the output feature dimension of the backbone and the label embedding. If the feature dimension is reduced, the number of parameters and computational overhead of HMA can be further reduced.

In the graph OOD experiment on the Graph-Twitter dataset, the standard ERM [57] method takes approximately 10 minutes for a single experiment. The state-of-the-art method, CIGA [8], which involves additional training stages and loss takes around 50 minutes for a single experiment. On the other hand, HMA completes a single experiment in approximately 20 minutes. Despite taking significantly less time than CIGA, HMA achieves superior performance on this dataset. Similarly, for the Graph-SST5 dataset, the single experiment runtime for ERM, CIGA, and HMA is approximately 15 minutes, 140 minutes, and 60 minutes, respectively.

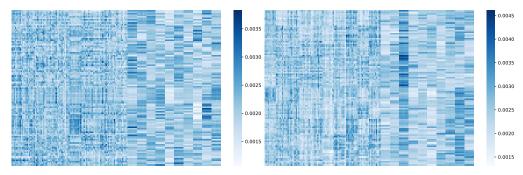


Figure 6: The visualization of attention scores: the left part illustrates the results for the top 10 classes with 10 samples each on the DTD dataset, while the right part depicts the visualization for the top 10 classes with 10 samples each on the Pets37 dataset. The y-axis corresponds to the input samples, with samples from each class grouped. The x-axis represents the corresponding samples and synthetic memory. The left side of each image exhibits the attention scores among the samples, while the right side illustrates the scores between the samples and synthetic memory.

# 7.3.3 Extended Visualization Results

We also visualized the attention scores in SMA, as shown in Figure 6. We selected 10 images from 10 different classes in two datasets, forming a 10x10 batch for classification, and visualized the attention scores in SMA. We can observe the following: (1) There is a clear distinction between the attention patterns among real data and the patterns between real data and synthetic memory (left vs. right). (2) Data from the same class exhibit more similar attention scores towards the synthetic memory (as evident in the attention scores on the right, with noticeable block patterns along the vertical axis). Observation (1) may be attributed to the fact that synthetic memory of the same class shares the same label embedding, and similar initialization methods lead to the learning of similar memory features. As a result, keys generated from the same class of synthetic memories tend to exhibit similar behaviors. Observation (2) suggests that synthetic memory captures information at the class level. Additionally, the ability of SMA to help inputs find similar reference sample is likely a contributing factor to the improvement in OOD performance.

### 7.3.4 Extended Results

Table 7: Accuracy of Resnet18 and HMA on CIFAR-10. There are three components of our approach: Real Memory Augmentation (RMA), Attention between Data-points (ABD), and Synthetic Memory (SYN).

| RMA    | ABD          | SYN | ACC  |
|--------|--------------|-----|--|
| √<br>√ | √<br>√<br>√. | √,  | 95.43±0.01<br>95.40±0.15<br>95.43±0.14<br>95.35±0.03<br>95.49±0.17 |
|        |              |     | <b>95.54</b> ±0.11   |

On CIFAR-10, we adopted a ResNet-18 encoder, similar to the one used in NPT[31]. From the accuracy results in Table 7, we can observe the impact of the three components of our approach: Real Memory Augmentation (RMA), Attention between Datapoints (ABD), and Synthetic Memory (SYN). If none of them is used, the model degrades to the backbone. We have the following observations: using ABD alone can hurt the performance similar to NPT; adding SMA during ABD can bring an improvement; while RMA alone doesn't bring a significant effect, combining RMA and SMA yields the best results. Although the variance is larger than the backbone, our synthetic memory does improve the best results.

In the scenario where only the head and the CLS token are tuned 8, we can observe that although adding HMA improves upon the backbone, its performance falls short of SOTA prompt tuning [52]. We attribute this to the fact that HMA only adjusts the final feature space, and when most model parameters are fixed, its impact is weaker compared to the method of independently incorporating soft prompts at each layer. However, as demonstrated in the main text, HMA can be effectively combined with prompt tuning to achieve superior performance compared to prompt tuning alone.

FLOWERS 102 DTD PETS37 CIFAR 100 AVG DATASET  $98.53 \pm 0.17$  $74.01 \pm 0.29$  $89.17 \pm 0.07$  $88.40 \pm 0.06$ 87.52 VIT  $98.62 \pm 0.21$  $90.01 \pm 0.21$ PROMPT-P  $73.98\pm0.56$  $89.22 \pm 0.22$ 87.96 PROMPT-A  $74.75 \pm 0.86$ 89.88±0.28  $98.57 \pm 0.09$ 91.79±0.19 88.75  $72.98 \pm 0.05$  $89.44 \pm 0.05$  $98.59 \pm 0.11$  $88.04 \pm 0.12$ 87.26 PARAM  $89.52{\pm0.31}$  $98.61 \pm 0.05$  $88.35 \pm 0.24$  $74.31 \pm 0.45$ 87.70 RMA $74.18 \pm 0.20$  $89.59 \pm 0.20$  $98.57 \pm 0.05$  $88.37 \pm 0.12$ 87.68 ABD  $74.75 \pm 0.27$  $89.52 \pm 0.77$  $98.62 \pm 0.21$  $88.51 \pm 0.18$ 87.85 ABD+RMA  $73.04\pm0.28$  $88.90 \pm 1.10$  $98.40 \pm 0.04$  $88.48 \pm 0.16$ 87.48 ABD+SYN **74.91**±0.39 98.67±0.18  $89.72 \pm 0.22$  $88.52 \pm 0.14$ 87.96 R+A+S

Table 8: Accuarcy of tuning head and cls token of pretrained ViT on 4 image datasets.

# 7.4 Algorithm

### 7.4.1 Real Memory Augmentation (RMA)

**Reading.** We utilize Multi-head attention to extract information from the memory buffer, implicitly providing cross-batch information for subsequent SMA. To achieve this, we first transform  $M_{\text{buffer}}$  from  $\mathbb{R}^{m_1 \times d}$  to  $\mathbb{R}^{1 \times m_1 \times d}$  and  $C^1$  from  $\mathbb{R}^{bs \times d}$  to  $\mathbb{R}^{bs \times 1 \times d}$ . Then, we repeat  $M_{\text{buffer}}$  bs times in the first dimension to obtain  $M'_{\text{buffer}} \in \mathbb{R}^{bs \times m_1 \times d}$ . Subsequently, we concatenate  $M'_{\text{buffer}}$  and  $C^1$  along the second dimension, resulting in  $C^1_{\text{aug}} \in \mathbb{R}^{bs \times 1 + m_1 \times d}$ . RMA computes the output feature  $C^2$  by:

$$C^2 = \text{MHA}(C^1, C^1_{\text{ans}}, C^1_{\text{ans}}) \in \mathbb{R}^{bs \times 1 \times d}$$

$$\tag{7}$$

Writing. We insert new aggregated input features  $[E_m(X), E_{\text{label}}(Y)] \in \mathbb{R}^{bs \times d}$  into the memory queue, and update the momentum encoder momentum update rule as in MoCo:  $E_m^{t+1} = \lambda E_m^t + (1-\lambda)E^{t+1}$ , in which  $E^{t+1}$  is the SGD-updated encoder from  $E^t$ .

# 7.4.2 Synthetic Memory Augmentation (SMA)

For each category  $i \in [n]$ , we initialize  $m_2$  learnable memory slots. Each memory slot is in  $\mathbb{R}^{d_1}$  and has corresponding label  $y_i$ . With the label embedder  $E_{\mathrm{emb}}$ , we map the label  $y_i$  to the label embedding  $e_{\mathrm{syn},i} \in \mathbb{R}^{d_2}$ . By concatenating all learnable memory slots and label embeddings, we obtain a learnable memory  $M_{\mathrm{SMA}} \in \mathbb{R}^{(n*m_2) \times (d_1 + d_2)}$ . We reshape  $C^2$  and  $M_{\mathrm{SMA}}$ , and concatenate them to obtain  $C_{\mathrm{aug}}^2 \in \mathbb{R}^{1 \times (bs + m_2*n) \times d}$ . The SMA output features are obtained by

$$C^3 = \text{MHA}(C^2, C_{\text{aug}}^2, C_{\text{aug}}^2) \in \mathbb{R}^{1 \times bs \times d}. \tag{8}$$

Finally, we use a linear projection head to compute the logits from  $C^3$ . During the training phase, the learnable memory slots are updated through gradient updates, while the label embeddings are regenerated using the updated embedder  $E_{\rm emb}$ .

# 7.5 Societal Impacts

Due to its lack of additional training targets and stages, as well as its compatibility with various backbone architectures without modifications, HMA can be easily combined with numerous backbones. We believe that HMA has the potential to play a significant role in practical applications of deep learning.

# Algorithm 1 Heterogeneous Memory Augmentation

```
Require: data \mathcal{D}, encoder E, momentum encoder E_m, label embedder E_{\text{label}}, class number n
Require: batch size bs; buffer size m_1, synthetic memory size m_2; feature and label embedding dimension d_1, d_2
 1: initialize empty real memory buffer M_{\text{buffer}} = \{\}, synthetic memory M_{\text{syn}}
 2: for (X,Y) \in D do
       Get input feature h^1 = E(X) and unknown embedding e = E_{label}(\mathbf{n+1})
 3:
 4:
       Read from M_{\text{buffer}} with cross attention according to (5)
 5:
       Read from M_{\rm syn} with attention between datapoints according to (6)
       Predict with augmented embeddings
 6:
 7:
       if is training then
          Update M_{\text{buffer}} with [E_m(X), E_{label}(Y)]
 8:
 9:
          Compute loss and update E, E_{label}, M_{syn} and attention weights with gradients
10:
          Update E_m = \gamma E_m + (1 - \gamma)E
       end if
11:
12: end for
```