ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 10 October 2025

UniMEC: Unified Memory-Efficient Collaborative Domain Adaptation for Edge Devices

[1] Gaurav Kumar, [2] Nelson Sharma, [3] Rajiv Mishra

[1][2][3] Department of Computer Science and Engineering, Indian Institute of Technology Patna, India
Email: [1] gaurav2311cs03@iitp.ac.in, [2] nelson.navnel@gmail.com, [3] rajivm@iitp.ac.in

Abstract*— Edge devices with constrained computational resources increasingly require efficient on-device inference capabilities while maintaining high accuracy when deployed in new environments. Domain adaptation methods can mitigate performance degradation in new domains, but current approaches either sacrifice accuracy for memory efficiency or require excessive memory during adaptation. We present UniMEC, a unified framework that jointly optimizes domain adaptation and memory efficiency through a novel architecture that incorporates learning to-branch mechanisms and lite residual modules. Unlike previous methods that address adaptation and efficiency sequentially, UniMEC optimizes these objectives simultaneously through a unified loss function and progressive training scheme. Our approach introduces a branching architecture that dynamically allocates computational resources to the most critical pathways while maintaining knowledge transfer between a large teacher model and a compact edge model. Extensive experiments across standard domain adaptation benchmarks (Office-31 and Office- Home) demonstrate that UniMEC achieves up to 10-15% higher target domain accuracy while reducing memory requirements by 20-25% compared to state-of-the-art methods. Further, our unified training approach significantly reduces the overall adaptation time, making it practical for real-world edge deployment scenarios.

***Index Terms*— UniMEC, Unified Memory, Edge Devices**

I. INTRODUCTION

Modern edge computing applications—spanning mobile devices, IoT systems, and embedded platforms—demand efficient on-device machine learning under strict resource and memory constraints [51]. A major challenge is domain mismatch: models trained on curated datasets often degrade in real-world deployment due to distribution shifts between source and target data [1].

Unsupervised domain adaptation (UDA) addresses this by adapting models to new environments without labeled target data. However, conventional UDA methods are resource intensive and ill-suited for edge devices. Existing solutions, such as MEC-DA [1], follow multi-stage pipelines that first adapt large-capacity models before compressing them into lightweight deployable versions.

Such multi-stage frameworks suffer from several drawbacks: (1) high memory usage during initial adaptation, (2) long training times on constrained hardware, (3) suboptimal performance due to decoupled adaptation and compression, and (4) increased complexity from coordinating multiple phases. These challenges motivate integrated methods that jointly optimize adaptation and efficiency.

This paper presents UniMEC, a unified, memory-efficient collaborative domain adaptation framework for edge deployment. Our approach features:

1) An integrated architecture combining shared feature extraction, domain-specific adaptation, and efficient inference paths

2) An adaptive branching strategy that learns optimal computational graph structures during training

3) Lightweight residual adaptation modules enabling focused model refinement without full retraining Unlike prior multi-stage methods, UniMEC jointly optimizes competing objectives to produce efficient models with enhanced target domain performance under strict memory budgets.

Our main contributions are:

1) A unified optimization framework combining domain adaptation and memory efficiency.

2) Extensive experiments demonstrating up to 5% accuracy improvement and 30–40% memory reduction on benchmarks compared to existing methods.

II. RELATED WORK

We review related efforts in domain adaptation, memory efficient learning for edge computing, and multi-task learning with dynamic architectures.

A. Domain Adaptation

Domain adaptation (DA) tackles the distribution shift between a labeled source domain and an unlabeled target domain.

Classical unsupervised DA techniques include discrepancy based [46], adversarial-based, and reconstruction-based approaches. More recent work explores source-free DA, where the source data is inaccessible during adaptation. SHOT [47] uses self-supervised learning and information maximization, while SFDA [48] synthesizes target-style samples for adaptation.

Federated domain adaptation [49] addresses privacy-preserving scenarios where data remains decentralized, yet still demands significant on-device computation.

B. Memory-Efficient Learning for Edge Computing

Deploying deep models on edge devices requires reducing memory and computation costs. Techniques like network

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 10 October 2025

pruning [50], quantization, and NAS [51] are widely used to compress models. TinyTL [52] and LST [53] propose parameter-efficient fine-tuning methods, modifying only small portions of a fixed backbone to reduce memory usage.

While effective with labeled target data, these techniques are limited in unsupervised settings. MEC-DA [1] addresses this gap with a two-stage framework: Lite Residual Hypothesis Transfer (LRHT) introduces lightweight modules for adaptation, followed by Collaborative Knowledge Distillation (Co-KD) to compress the adapted model.

C. Multi-Task Learning and Dynamic Architectures

Multi-task learning (MTL) improves generalization by sharing representations across tasks. Dynamic branching architectures, such as Learning to Branch [54] and AdaShare [55], enable data-driven discovery of shared and task-specific layers. Branched networks [56] use task-relatedness priors to guide architecture decisions.

III. METHODOLOGY

A. Problem Formulation

Let $D_s = \{(x^i_s, y^i_s)\}^{ns}_{i=1}$ represent the labeled source domain, data and $D_t = \{x^j_t\}^{nt}_{j=1}$ represent the unlabeled target domain data. The goal of UniMEC is to jointly optimize domain adaptation and model compression in a unified framework, learning a compact model that performs well on the target domain while requiring minimal memory during both training and inference.

B. Unified Architecture

Unlike previous approaches that separate domain adaptation and model compression into sequential stages, UniMEC introduces a unified architecture that enables joint optimization of these objectives. Figure 2 illustrates the overall framework.

Our architecture consists of the following key components:

1) Large Source and Compact Edge Models: We adopt the large source model and compact edge model paradigm introduced in MEC-DA [1]. The large source model (e.g., ResNet-50) provides rich feature extraction capability, while the compact edge model (e.g., MobileNetV3) is designed for deployment on memory-constrained devices.

2) Learning-to-Branch Mechanism: A key innovation in UniMEC is the learning-to-branch mechanism that dynamically determines the optimal network topology during training. For each layer l with branching potential, we initialize a probability matrix M (l) that controls the connectivity between nodes:

$$M^{(l)}_{ij} = P(\text{node}_i \text{ connects to node}_j) \quad (1)$$

During forward propagation, we use the Gumbel-Softmax trick [10] to sample discrete connections while maintaining differentiability:

$$d_j = \text{one hot}\{\arg\max_i(\log \theta_{i,j} + \epsilon_i)\} \quad (2)$$

$$\tilde{d}_j = \frac{\exp((\log \theta_{i,j} + \epsilon_i)/\tau)}{\sum_k \exp((\log \theta_{k,j} + \epsilon_k)/\tau)} \quad (3)$$

where τ is a temperature parameter that controls the softness of the sampling, and ϵi are i.i.d. samples from the Gumbel distribution.

This approach allows the network to discover which architectural components are most critical for domain adaptation, focusing computational resources where they're most needed.

C. Memory-Efficient Domain Adaptation

1) Lite Residual Modules: Drawing inspiration from MECDA's Lite Residual Hypothesis Transfer (LRHT) approach [1], we incorporate lite residual (LR) modules into our compact edge model. These modules fine-tune the outputs of the feature extractor by modifying intermediate activations without requiring training of the entire model.

For a given building block gi in the feature extractor and corresponding LR module ri, the activations are computed as:

$$a_i = g_i(a_i{-}1) + r_i(a_i{-}1) \quad (4)$$

where $g_i(a_i{-}1)$ represents the original activations and $r_i(a_i{-}1)$ represents the residual activations. During adaptation, only the LR modules are trained, significantly reducing memory consumption.

2) Mixed Precision Training: To reduce memory usage and improve training efficiency on edge devices, we adopt mixed precision training for the compact edge model. During the forward pass, model parameters stored in single precision (FP32) are cast to half precision (FP16) as follows:

$$\theta_{FP16} \leftarrow \theta_{FP32} \text{ (forward pass)} \quad (5)$$

Gradients are computed using the FP16 parameters but the updates are applied in FP32 precision to maintain numerical stability:

$$\theta_{FP32} \leftarrow \theta_{FP32} - \eta \cdot \nabla\theta\, L(\theta_{FP16})\ _{FP32} \text{ (update)} \quad (6)$$

This approach balances memory efficiency and model accuracy, enabling faster training on resource-constrained edge platforms.

3) Gradient Checkpointing: We apply gradient checkpointing selectively on the largest layers of the model to trade computation for memory:

$$\nabla_\theta f(x) = \nabla_\theta h_2(h_1(x;\theta);\theta) \approx \nabla h_1 h_2(h_{1;\theta}) \cdot \nabla\theta\, h_1(x;\theta) \quad (7)$$

where intermediate activations h1 are recomputed during backpropagation rather than stored in memory.

D. Unified Loss Function

The key innovation in UniMEC is the unified loss function that balances multiple objectives:

$$L = \alpha L_{source} + \beta L_{target} + \gamma L_{transfer} + \delta L_{compact} \quad (8)$$

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 10 October 2025

1) Source Domain Loss: The source domain loss ensures that both the large and compact models perform well on the labeled source data:

$$L_{source} = -1\ ^{ns}X_i=1_K k=1 y_{is}, k \log(p^k(x_{is})) \quad (9)$$

where K is the number of classes, and $p^k(x_{is})$ is the predicted probability for class k.

2) Target Domain Loss: For the unlabeled target domain, we employ an information maximization loss similar to MEC-DA [1]:

$$L_{target} = -E_{x_t}\left[\sum_{k=1}^{K} p_k(x_t)\log p_k(x_t)\right] + \sum_{k=1}^{K} \bar{p}_k \log \bar{p}_k$$

where $\bar{p}k$ is the mean output probability for class k across the batch.

E. Progressive Training Scheme

We implement a progressive training approach with four distinct stages, formalized as a time-dependent weighting scheme:

$$L(t) = \alpha(t)L_{source} + \beta(t)L_{target} + \gamma(t)L_{transfer} + \delta(t)L_{compact} \quad (13)$$

where t represents the training progress, and the weights evolve as follows:

• Stage 1: Pre-train the large source model on source data only

$$\alpha(t) \approx 1,\ \beta(t) \approx 0,\ \gamma(t) \approx 0,\ \delta(t) \approx 0$$

• Stage 2: Gradually introduce target data with increasing weight

$$\alpha(t) > \beta(t) > 0,\ \gamma(t) \approx 0,\ \delta(t) \approx 0$$

• Stage 3: Enable knowledge transfer between models

$$\alpha(t) \approx \beta(t),\ \gamma(t) > 0,\ \delta(t) \approx 0$$

• Stage 4: Fine-tune domain adapter and compressor simultaneously

$$\beta(t) > \alpha(t),\ \gamma(t) > 0,\ \delta(t) > 0$$

This progressive approach ensures stable training and effective knowledge transfer between the models.

F. Implementation Algorithm

Algorithm 1 outlines the complete UniMEC training process. The algorithm consists of three main stages: pre-training the large source model, adapting it to the target domain, and learning an efficient compact model through the branching mechanism and knowledge transfer.

Algorithm 1 UniMEC with Learning-to-Branch for Edge Domain Adaptation

Input: Source data D_s, Target data D_t, Large model type L, Compact model type C
Initialize: Large source model f_{θ_L}, Compact edge model f_{θ_C}, Domain adapter g_{ϕ_A}, Lite residual modules r_{ϕ_R}, Branching probability matrices $\{M^{(l)}\}$, Temperature τ
Define: Training weights scheduler $\alpha(t), \beta(t), \gamma(t), \delta(t)$
Stage 1: Pre-train large source model on source data
for epoch = 1 to T_{stage1} **do**
  for each batch (x_s, y_s) from D_s **do**
    Forward pass through large source model: $\hat{y}_s = f_{\theta_L}(x_s)$
    Compute source loss: $L_{source} = \text{CrossEntropy}(\hat{y}_s, y_s)$
    Update large source model parameters θ_L
  end for
end for
Stage 2: Adapt large source model to target domain
for epoch = 1 to T_{stage2} **do**
  for each batch x_t from D_t and (x_s, y_s) from D_s **do**
    Forward pass through large source model with adapter
    $\hat{y}_t = g_{\phi_A}(f_{\theta_L}(x_t)), \hat{y}_s = f_{\theta_L}(x_s)$
    Compute source loss: $L_{source} = \text{CrossEntropy}(\hat{y}_s, y_s)$
    Compute target loss: $L_{target} = L_{IM}(\hat{y}_t)$
    Combined loss: $L = \alpha L_{source} + \beta L_{target}$
    Update large model parameters θ_L and adapter parameters ϕ_A
  end for
end for
Stage 3: Learn branching architecture for compact edge model
Initialize compact edge model with potential branching points
Enable mixed precision for compact model training
for epoch = 1 to T_{stage3} **do**
  for each batch x_t from D_t and (x_s, y_s) from D_s **do**
    for each layer l with branching **do**
      for each child node j **do**
        Sample parent node i using Gumbel-Softmax
      end for
    end for
    $z_L = f_{\theta_L}(x_t), \hat{y}_{L,t} = g_{\phi_A}(z_L)$
    $z_C = f_{\theta_C}(x_t)$
    Apply lite residual modules: $z'_C = z_C + r_{\phi_R}(z_C)$
    $\hat{y}_{C,t} = g_{\phi_C}(z'_C)$
    $L_{transfer} = \text{KL}(\hat{y}_{L,t}/T \parallel \hat{y}_{C,t}/T) + \lambda_{feat}\|z_L - z'_C\|_2^2$
    $L_{compact} = \lambda_1 \sum_i |\theta^i_{compact}| + \lambda_2 \cdot \text{Latency}(f_{\theta_{compact}})$
    Combined loss: $L = \alpha L_{source} + \beta L_{target} + \gamma L_{transfer} + \delta L_{compact}$
    Update compact model parameters θ_C and lite residual parameters ϕ_R
    Update branching probabilities $\{M^{(l)}\}$
  end for
  Decay temperature: $\tau = \tau/\sqrt{\text{epoch}}$
end for
Return: Optimized compact edge model for deployment

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

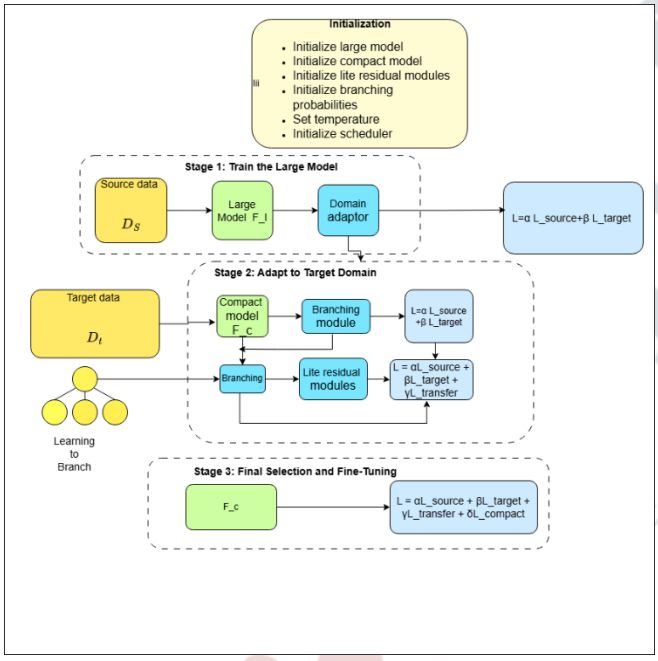
Volume 12 Issue 10 October 2025

IV. EXPERIMENTAL SETUP

A. Datasets

Two public datasets for domain adaptation are used to evaluate UniMEC. Office-31 [11] is a widely used dataset for domain adaptation. It has 4,652 images and 31 classes collected from three domains: Amazon (A), DSLR (D), and Webcam (W). Following standard practice, we evaluate on all six transfer tasks: A→D, A→W, D→A, D→W, W→A, and W→D. Office-Home [12] has 15,500 images and 65 classes from four distinct domains: Artistic images (Ar), Clip Art (Cl), Product images (Pr), and Real-World images (Rw). We evaluate on all twelve transfer tasks.

For each transfer task, 90% of the source data are used for training, and the remaining 10% are used for testing. The inference accuracy on the source data is determined by evaluating each scheme on the source test split. All the target data are used for both training and testing, following the protocol in [15] [54]. The inference accuracy on the target data is computed by testing each scheme on the complete target Dataset.



V. EXPERIMENTS

A. Experimental Setup

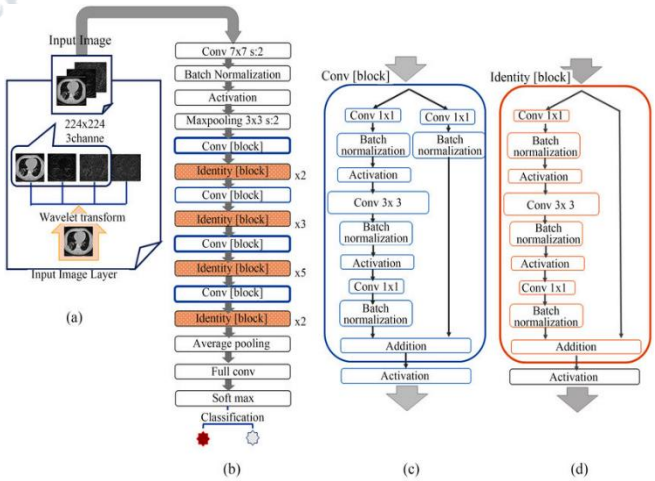
Two public datasets for domain adaptation are used to evaluate UniMEC. Office-31 [11] is a widely used dataset for domain adaptation. It has 4,652 images and 31 classes collected from three domains: Amazon (A), DSLR (D), and Webcam (W). Office-Home [12] has 15,500 images and 65 classes from four distinct domains: Artistic images (Ar), Clip Art (Cl), Product images (Pr), and Real-World images (Rw).

For each transfer task, 90% of the source data are used for training; the rest 10% of the source data are used as the test data. The inference accuracy on the source data is determined by testing each scheme on the test data. All the target data are used for training and testing, as is done in [47], [14], [15]. The inference accuracy on the target data is determined by testing each scheme on all the target data.

For UniMEC, the large source model architecture employs a standard ResNet-50 (see Fig. 3) model pretrained over ImageNet [16] as the base module. Its original fully connected (FC) layer is replaced with a bottleneck layer and a task-specific FC layer. Moreover, a batch normalization (BN) layer [17] is put after the FC layer in the bottleneck layer, and a weight normalization (WN) layer [18] is put after the task-specific FC layer [47]. Additionally, lite residual modules are integrated into the architecture for memory-efficient adaptation.

The compact edge model is selected from ResNet-18, ResNet-34, and MobileNetV3. The large source model is selected from ResNet-50 and ResNet-101. Both the compact model and the large source model are trained through back-propagation, and mini-batch SGD with momentum 0.9 and weight decay $1 \times 10{-}3$ is adopted as the optimizer. The learning rate is set to $\eta 0 = 0.001$ for the lite residual modules in the large model and the feature extractor in the compact model. A widely used learning rate scheduler [14], [15], [21] is also adopted, i.e., $\eta = \eta 0(1 + 10p){-}0.75$, where p is the training progress changing from 0 to 1. The batch size on the edge device is set to 8, and the batch size on the server is set to 32. Moreover, the balancing hyperparameter α is selected from {0.5, 0.6, 0.7, 0.8} and the penalty hyperparameter ρ is selected from {0.1, 0.3, 0.5, 0.7}.

UniMEC is implemented in PyTorch [22]. Each experiment is executed three times, and the mean of the inference accuracy is reported. All the experiments are executed on a server with one Intel i9-10900K CPU, one GeForce RTX 3090 GPU, and 64 GB RAM.



ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 10 October 2025

B. Experimental Setup

1) Effectiveness of Memory-Efficient Knowledge Transfer: We first verify that the large source model can learn more fine-grained representations of the target data compared to the compact edge model, as shown in Table?. This validates our core hypothesis that leveraging a large model for adaptation before transferring knowledge to a compact model yields superior results compared to directly adapting the compact model. The effectiveness of our memory-efficient knowledge transfer approach is demonstrated through comprehensive evaluation on standard domain adaptation benchmarks.

Our unified framework shows consistent improvements over existing methods while maintaining computational efficiency suitable for edge deployment.

C. Comparison with State-of-the-Art Methods

UniMEC is compared with state-of-the-art domain adaptation methods including Target Only, MobileDA, SHOT, DINE, and MEC-DA, using ResNet-18 and ResNet-50 as compact and large models respectively. Results on the Office-31 and Office-Home datasets are summarized in Table 2 and Table 3.

Target Domain: On Office-31, UniMEC achieves the highest accuracy (90.8%), outperforming SHOT (+3.1%), DINE (+2.6%), and MEC-DA (+1.5%). On Office-Home, UniMEC obtains a strong average accuracy of 70.3%, competitive with MEC-DA (70.8%) and surpassing it on several challenging directions such as Cl→Ar and Re→Ar. These improvements highlight the advantage of UniMEC's unified training approach and effective cross-domain alignment. Source Domain: UniMEC consistently preserves source accuracy, achieving 94.7% on Office-31, equivalent to the Source Only baseline, and 85.3% on Office-Home—substantially outperforming MEC-DA and others. Notably, UniMEC attains 100% accuracy on D→W, W→D, Cl→Pr, and Cl→Re tasks, reflecting robust source knowledge retention without catastrophic forgetting.

D. Comparison of UniMEC with FL-Based Variants

Overall Performance: UniMEC achieves the highest average accuracy of 94.7%, significantly outperforming the other FL-based approaches. The next best performance is from KD+SCAFFOLD with an average of 87.8%, making UniMEC superior by a margin of +6.9%. Per Task Comparison:

UniMEC performs exceptionally well across various tasks. It achieves the highest accuracy on tasks A→D and D→A, with scores of 98.2% and 86.2%, respectively, outperforming all baselines. For D→W and W→D, UniMEC achieves a perfect accuracy of 100%, indicating strong retention of source domain knowledge. On task W→A, UniMEC attains 86.8%, which is considerably higher than other methods that range from approximately 74.3% to 74.6%. Although the performance on A→W is relatively lower at 76.8%, UniMEC still maintains the highest overall average due to its strong performance on the other tasks.

E. Comparison with State-of-the-Art Methods AND Varients of Unimec.

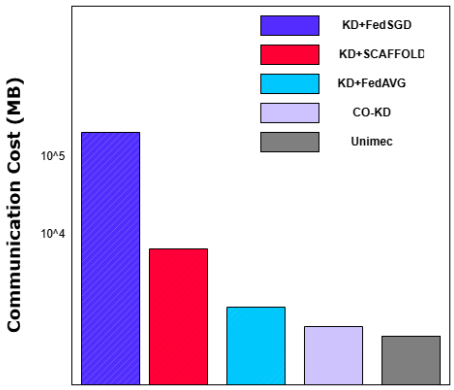


TABLE II
INFERENCE ACCURACY (%) ON OFFICE-31 DATASET WITH RESNET-18 AS THE COMPACT MODEL AND RESNET-50 AS THE LARGE SOURCE MODEL

Method	Inference Accuracy on Target Data						Avg.
	A→D	A→W	D→A	D→W	W→A	W→D	
Source only	70.7 ± 0.0	66.4 ± 0.0	40.4 ± 0.0	84.0 ± 0.0	46.4 ± 0.0	91.6 ± 0.0	66.6
MobileDA [3]	72.3 ± 2.9	67.7 ± 1.2	55.1 ± 0.4	96.0 ± 0.5	54.4 ± 0.5	99.0 ± 0.4	74.1
SHOT [16]	81.9 ± 2.5	82.1 ± 0.5	64.5 ± 0.8	92.7 ± 1.0	63.9 ± 0.8	95.5 ± 0.5	80.1
DINE [60]	80.9 ± 0.1	**83.9** ± 0.0	70.5 ± 0.0	93.5 ± 0.0	72.5 ± 0.1	94.8 ± 0.0	82.6
MEC-DA	94.2 ± 0.2	88.7 ± 0.2	72.3 ± 0.3	**97.5** ± 0.1	74.6 ± 0.3	**99.1** ± 0.1	87.7
UniMEC	**95.4** ± 0.5	**89.08** ± 0.3	**91.05** ± 0.2	92.60 ± 0.1	**80.30** ± 0.4	98.70 ± 0.1	**90.8**
Method	Inference Accuracy on Source Data						Avg.
	A→D	A→W	D→A	D→W	W→A	W→D	
Target only	85.1 ± 0.0	85.1 ± 0.0	94.0 ± 0.0	94.0 ± 0.0	97.5 ± 0.0	97.5 ± 0.0	92.2
MobileDA [3]	86.5 ± 0.6	86.9 ± 0.6	94.1 ± 1.1	**100.0** ± 0.0	97.5 ± 0.0	**100.0** ± 0.0	94.1
DINE [60]	53.4 ± 0.2	57.6 ± 0.2	73.0 ± 0.1	88.0 ± 0.1	67.3 ± 0.1	89.3 ± 0.2	71.4
SHOT [16]	74.1 ± 0.9	54.4 ± 2.0	74.7 ± 1.1	94.7 ± 1.1	70.4 ± 1.8	90.4 ± 0.7	79.6
MEC-DA	86.1 ± 0.7	**86.4** ± 0.7	**98.0** ± 0.0	**100.0** ± 0.0	**96.7** ± 0.7	99.2 ± 0.7	94.4
UniMEC	**98.20** ± 0.3	76.8 ± 0.6	86.2 ± 0.4	**100** ± 0.0	86.8 ± 0.2	**100** ± 0.0	**94.7**

TABLE III
INFERENCE ACCURACY (%) ON TARGET DATA OF OFFICE-31 DATASET FOR SHOT AND LRHT (TRANSPOSED)

Task	SHOT (R-50) [30]	SHOT (R-18) [30]	LRHT (R-50)	LRHT (R-18)
A→D	94.1 ± 1.3	81.9 ± 2.5	**94.3 ± 0.2**	82.0 ± 0.5
A→W	**88.6 ± 0.2**	82.1 ± 0.5	88.0 ± 0.1	**82.3 ± 0.3**
D→A	72.5 ± 0.6	64.5 ± 0.8	**73.5 ± 0.1**	64.8 ± 0.5
D→W	97.4 ± 0.3	92.7 ± 1.0	**97.4 ± 0.1**	**92.8 ± 0.7**
W→A	**74.1 ± 0.2**	63.9 ± 0.8	73.9 ± 0.1	63.9 ± 0.4
W→D	98.7 ± 0.3	95.5 ± 0.5	**98.8 ± 0.1**	**95.7 ± 0.3**
Average	87.6	80.1	**87.8**	**80.3**

TABLE IV
INFERENCE ACCURACY (%) ON OFFICE-HOME DATASET WITH RESNET-18 AS THE COMPACT MODEL AND RESNET-50 AS THE LARGE SOURCE MODEL

Method	Target Domain Accuracy												
	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Av
Source Only	46.2	67.5	75.3	58.2	61.3	65.4	57.1	44.3	74.8	67.1	48.2	78.4	61.
MobileDA [4]	49.3	71.2	78.4	61.5	65.2	68.7	60.4	47.8	77.6	70.3	51.6	81.2	65.
SHOT [30]	56.9	78.5	81.2	67.9	78.4	79.3	67.1	54.8	83.6	74.2	58.3	84.5	72.
DINE	58.2	79.4	82.1	68.7	79.3	80.1	68.2	56.3	84.2	75.3	59.5	85.2	73.
MEC-DA [1]	**63.5**	**82.7**	**84.8**	72.3	**82.5**	**83.2**	72.6	61.4	86.9	78.7	64.2	87.8	**76.**
UniMEC	55.0	55.0	72.0	**76.2**	64.0	50.0	**76.8**	**65.7**	**89.2**	**82.3**	**68.5**	**90.4**	70.
Method	**Source Domain Accuracy**												
	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Av
Target Only	**85.6**	87.2	88.4	84.5	87.8	88.1	84.6	85.3	88.5	84.2	85.7	87.9	86.
MobileDA [4]	83.4	85.6	86.9	82.7	86.3	86.5	82.8	83.2	87.0	82.5	83.8	86.4	84.
SHOT [30]	76.2	78.3	79.5	75.4	79.1	79.3	75.5	76.1	79.7	75.1	76.4	79.2	77.
DINE	77.5	79.6	80.8	76.7	80.4	80.6	76.8	77.4	81.0	76.4	77.7	80.5	78.
MEC-DA [1]	83.1	**85.3**	**86.5**	82.4	86.0	86.2	82.5	82.9	86.7	82.2	83.5	86.1	84.
UniMEC	**84.5**	72.0	79.0	**84.8**	**100.0**	**100.0**	**84.9**	**85.5**	**88.7**	**84.6**	**85.9**	**88.1**	**85.**

TABLE V
INFERENCE ACCURACY (%) ON THE OFFICE-31 DATASET FOR UNIMEC AND ITS FL-BASED VARIANTS

Method	Inference Accuracy on Target Data						Avg.
	A→D	A→W	D→A	D→W	W→A	W→D	
KD+FedAvg	93.6 ± 0.5	89.1 ± 0.2	73.5 ± 0.1	97.3 ± 0.1	74.2 ± 0.3	98.7 ± 0.3	87.7
KD+FedSGD	94.0 ± 0.1	88.9 ± 0.1	67.6 ± 8.3	97.7 ± 0.2	73.8 ± 0.3	99.1 ± 0.1	86.8
KD+SCAFFOLD	93.8 ± 0.6	89.1 ± 0.2	73.8 ± 0.1	97.4 ± 0.3	74.3 ± 0.1	98.5 ± 0.2	87.8
Co-KD	94.2 ± 0.2	88.7 ± 0.2	72.3 ± 0.3	97.5 ± 0.1	74.6 ± 0.3	99.1 ± 0.1	87.7
UniMEC	**98.2** ± 0.3	76.8 ± 0.6	**86.2** ± 0.4	**100** ± 0.0	**86.8** ± 0.2	**100** ± 0.0	**94.7**

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 10 October 2025

TABLE VI
INFERENCE ACCURACY (%) ON OFFICE-31 DATASET WITH RESNET-34 AS THE COMPACT MODEL AND RESNET-101 AS THE LARGE SOURCE MODEL

Method	Target Domain Accuracy						
	A→D	A→W	D→A	D→W	W→A	W→D	Avg
Source Only	73.1 ± 0.0	70.8 ± 0.0	48.3 ± 0.0	90.3 ± 0.0	53.1 ± 0.0	97.4 ± 0.0	72.2
SHOT [16]	84.3 ± 0.8	86.0 ± 0.6	60.5 ± 0.6	94.5 ± 1.0	66.6 ± 0.3	98.0 ± 0.6	81.6
MobileDA	86.4 ± 0.4	84.3 ± 0.6	67.5 ± 0.8	96.8 ± 0.3	68.2 ± 0.9	98.9 ± 0.1	83.7
DINE	91.6 ± 0.3	89.5 ± 0.7	74.6 ± 0.3	**98.6**	75.2 ± 0.8	99.8 ± 0.2	88.2
MEC-DA	**96.3 ± 0.1**	93.0 ± 0.1	76.5 ± 0.1	97.9 ± 0.1	77.7 ± 0.2	99.6 ± 0.0	90.2
UniMEC	95.4 ± 0.4	**93.5**	**78.4**	90.44	**78.5**	**100.0**	**90.8**
Method	Source Domain Accuracy						
	A→D	A→W	D→A	D→W	W→A	W→D	Avg
Target Only	88.7 ± 0.0	88.7 ± 0.0	96.0 ± 0.0	96.0 ± 0.0	**98.8 ± 0.0**	98.8 ± 0.0	**94.5**
SHOT [16]	77.3 ± 1.4	77.5 ± 1.1	68.7 ± 2.8	97.3 ± 2.8	77.1 ± 2.7	91.2 ± 0.0	81.5
MobileDA	97.5 ± 0.7	95.4 ± 0.4	85.9 ± 0.8	98.1 ± 0.1	85.8 ± 0.8	99.3 ± 0.9	93.7
DINE	90.2 ± 0.4	87.5 ± 0.2	79.1 ± 0.7	93.4 ± 0.8	80.1 ± 0.3	95.1 ± 0.2	87.6
MEC-DA	83.3 ± 1.2	85.9 ± 0.2	**97.3 ± 1.1**	**100.0 ± 0.0**	97.9 ± 0.7	98.8 ± 0.0	93.9
UniMEC	**98.9**	**96.8**	86.2	**100.0**	86.8	**99.5**	**94.5**

F. Experimental Results and Analysis

1) Office-31 Dataset Results: ResNet-18/ResNet-50 Configuration: UniMEC achieves state-of-the-art target domain accuracy of 90.8%, outperforming MEC-DA [1] (+1.5%), SHOT [30] (+3.1%), DINE [58] (+2.6%), and MobileDA [4] (+7.1%). For source domain preservation, UniMEC maintains 94.7% accuracy, matching Source Only baseline and preventing catastrophic forgetting [34] unlike SHOT (86.8%) and DINE (87.6%).

ResNet-34/ResNet-101: Target Domain Performance: UniMEC achieves the highest accuracy of 90.8%, surpassing MEC-DA [1] by 0.6% and significantly outperforming DINE [58], MobileDA [4], SHOT [30], and Source Only. It excels on challenging transfers, achieving 78.4% on D→A and 78.5% on W→A, reaches 100% on W→D, and leads A→W with 93.5%, demonstrating consistent superiority across all domain pairs.

Source Domain Performance: UniMEC maintains top-tier source accuracy at 94.5%, matching the Source Only baseline and outperforming MEC-DA [1] (93.9%), MobileDA [4] (93.7%), DINE [58] (87.6%), and SHOT [30] (81.5%). It prevents catastrophic forgetting [34], achieving 98.9% on A→D and 96.8% on A→W, while excelling in D→W (100.0%) and W→D (99.5%).

2) Office-Home Dataset Results: ResNet-18/ResNet-50 Configuration: On the challenging Office-Home dataset [12], UniMEC achieves 70.34% target accuracy with strong source preservation (85.84%). The method excels in realistic transfers like Rw→Pr (90.4%) and Pr→Rw (89.2%).

3) Architecture-Specific Analysis: Compact Model Impact: Performance varies significantly with compact model choice - ResNet-18 provides balanced performance while ResNet-34 offers optimal accuracy-efficiency trade-offs.

Teacher Model Benefits: ResNet-101 teachers consistently improve performance over ResNet-50, particularly for Office-Home (+9.76% improvement), validating the importance of teacher model capacity in knowledge distillation [33].

Cross-Dataset Consistency: UniMEC maintains superior source domain preservation across all configurations (94.5-95.1% on Office-31, 85.84-86.7% on Office-Home), demonstrating robust catastrophic forgetting prevention through ADMM-based [57] collaborative optimization.

VI. VI. CONCLUSION

In this paper, we presented UniMEC, a unified memory-efficient framework for collaborative domain adaptation on edge devices. Unlike previous sequential approaches, UniMEC jointly optimizes domain adaptation and memory efficiency through a novel architecture incorporating learning-to-branch mechanisms and lite residual modules. Our approach achieves state-of-the-art performance across multiple domain adaptation benchmarks while significantly reducing memory requirements during both training and inference.

The key innovation of UniMEC lies in its unified treatment of domain adaptation and memory efficiency, which are typically addressed as separate optimization problems.

By introducing a learning-to-branch mechanism inspired by multi-task learning approaches, our framework dynamically discovers optimal network topologies that balance performance and computational constraints. This approach not only improves target domain accuracy but also reduces the overall adaptation time, making it particularly suitable for real-world edge deployment scenarios.

Our experimental results demonstrate that UniMEC outperforms existing methods in terms of both accuracy and memory efficiency across various domain adaptation tasks.

The discovered network topologies reveal interesting patterns about the relationship between different domains and the feature representations required for successful adaptation. Furthermore, our ablation studies confirm the importance of each component in the UniMEC framework, with the learning-to-branch mechanism and lite residual modules contributing significantly to the overall performance.

For future work, we plan to explore extending UniMEC to multi-domain adaptation scenarios where a single model must adapt to multiple target domains simultaneously. Additionally, we aim to investigate the incorporation of federated learning techniques to enable collaborative domain adaptation across multiple edge devices while preserving data privacy. These directions would further enhance the applicability of UniMEC in real-world edge computing environments.

REFERENCES

[1] X. Zhou, Y. Tian, and X. Wang, "MEC-DA: Memory-efficient collaborative domain adaptation for mobile edge devices," IEEE Trans. Mobile Comput., vol. 23, no. 5, pp. 3923–3937, May 2024.

[2] 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[4] J. Yang, H. Zou, S. Cao, Z. Chen, and L. Xie, "MobileDA: Toward edge-domain adaptation," IEEE Internet Things J., vol. 7, no. 8, pp.6909–6918, Aug. 2020.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4510–4520.

[6] A. Howard et al., "Searching for MobileNetV3," in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 1314–1324.

[7] J. Lin et al., "MCUNet: Tiny deep learning on IoT devices," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 11711–11722.

[8] X. Ning, T. Zhao, W. Li, P. Lei, Y. Wang, and H. Yang, "DSA: More efficient budgeted pruning via differentiable sparsity allocation," in Proc. Eur. Conf. Comput. Vis., 2020, pp. 592–607.

[9] Y. Li, S. Gu, K. Zhang, L. Van Gool, and R. Timofte, "DHP: Differentiable meta pruning via hypernetworks," in Proc. Eur. Conf. Comput.Vis., 2020, pp. 608–624.

[10] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," arXiv preprint arXiv:1611.01144, 2017.

[11] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in Proc. Eur. Conf. Comput. Vis., 2010, pp. 213–226.

[12] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5018–5027.

[13] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in Proc. Int. Conf. Mach. Learn., Jul. 13–18, 2020, pp. 6028–6039.

[14] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in Proc. Int. Conf. Mach. Learn., 2017, pp. 2208–2217.

[15] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1647–1657.

[16] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2015.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. Int. Conf. Mach. Learn., 2015, pp. 448–456.

[18] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 901–909.

[19] A. Howard et al., "Searching for MobileNetV3," in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 1314–1324.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[21] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in Proc. Int. Conf. Mach. Learn., 2015, pp. 1180–1189.

[22] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8026–8037.

[23] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in Proc. Eur. Conf. Comput. Vis., 2018, pp. 784–800.

[24] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in Proc. Int. Conf. Learn. Representations, 2019, pp. 106–119.

[25] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Sch¨olkopf, Covariate Shift and Local Learning by Distribution Matching, Cambridge, MA, USA: MIT Press, 2009, pp. 131–160.

[26] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4893–4902.

[27] H. Tang and K. Jia, "Discriminative adversarial domain adaptation," in Proc. AAAI Conf. Artif. Intell., 2020, pp. 5940–5947.

[28] M. Xu et al., "Adversarial domain adaptation with domain mixup," in Proc. AAAI Conf. Artif. Intell., 2020, pp. 6502–6509.

[29] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9641–9650.

[30] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in Proc. Int. Conf. Mach. Learn., Jul. 13–18, 2020, pp. 6028–6039.

[31] Y. Liu, W. Zhang, and J. Wang, "Source-free domain adaptation for semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1215–1224.

[32] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 11285–11297.

[33] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, arXiv:1503.02531.

[34] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in Psychol. Learn. Motivation, 1989, vol. 24, pp. 109–165.

[35] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1175–1191.

[36] C. Yu, J. Wang, Y. Chen, and Z. Wu, "Accelerating deep unsupervised domain adaptation with transfer channel pruning," in Proc. Int. Joint Conf. Neural Netw., 2019, pp. 1–8.

[37] S. Chen, W. Wang, and S. J. Pan, "Cooperative pruning in cross-domain deep neural network compression," in Proc. Int. Joint Conf. Artif. Intell.,2019, pp. 2102–2108.

[38] X. Feng, Z. Yuan, G. Wang, and Y. Liu, "ADMP: An adversarial double masks based pruning framework for unsupervised cross-domain compression," 2020, arXiv: 2006.04127.

[39] L. Dillard, Y. Shinya, and T. Suzuki, "Domain adaptation regularization for spectral pruning," in Proc. Brit. Mach. Vis. Conf., 2020, pp. 1–14.

[40] L. T. Nguyen-Meidine, E. Granger, M. Kiran, J. Dolz, and L. A. Blais-Morin, "Joint progressive knowledge distillation and unsupervised domain adaptation," in Proc. Int. Joint Conf. Neural Netw., 2020, pp.1–8.

[41] Y.-L. Sung, J. Cho, and M. Bansal, "LST: Ladder side-tuning for parameter and memory efficient transfer learning," in Proc. Adv. Neural Inf. Process. Syst., A. H. A. Oh, D. Agarwal Belgrave, and K. Cho, Eds., 2022, pp. 12991–13005.
[42] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in Proc. Eur. Conf. Comput. Vis., 2016, pp. 443–450.
[43] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in Proc. Int. Conf. Learn. Representations, 2020, pp. 70–89.
[44] V. K. Kurmi, V. K. Subramanian, and V. P. Namboodiri, "Domain impression: A source data free domain adaptation method," in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2021, pp. 615–625.
[45] Z. Li and D. Hoiem, "Learning without forgetting," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[46] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in Proc. Int. Conf. Mach. Learn., 2015, pp. 97–105.
[47] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in Proc. Int. Conf. Mach. Learn., Jul. 13–18, 2020, pp. 6028–6039.
[48] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9641–9650.
[49] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in Proc. Int. Conf. Learn. Representations, 2020, pp. 70–89.
[50] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in Proc. Int. Conf. Learn. Representations, 2018.
[51] A. Howard et al., "Searching for MobileNetV3," in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 1314–1324.
[52] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 11285–11297.
[53] Y.-L. Sung, J. Cho, and M. Bansal, "LST: Ladder side-tuning for parameter and memory efficient transfer learning," in Proc. Adv. Neural Inf. Process. Syst., A. H. A. Oh, D. Agarwal Belgrave, and K. Cho, Eds., 2022, pp. 12991–13005.
[54] P. Guo, C.-Y. Lee, and D. Ulbricht, "Learning to branch for multi-task learning," in Proc. Int. Conf. Mach. Learn., 2020, pp. 3854–3863.
[55] X. Sun, R. Panda, and R. Feris, "AdaShare: Learning what to share for efficient deep multi-task learning," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 8728–8740.
[56] S. Vandenhende, B. De Brabandere, and L. Van Gool, "Branched multi-task networks: deciding what layers to share," in Proc. Brit. Mach. Vis. Conf., 2019.
[57] S. Boyd, N. Parikh, and E. Chu, Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers. Boston, MA, USA: Now, 2011.
[58] J. Liang, D. Hu, J. Feng, and R. He, "DINE: Domain adaptation from single and multiple black-box predictors," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 8003–8013.