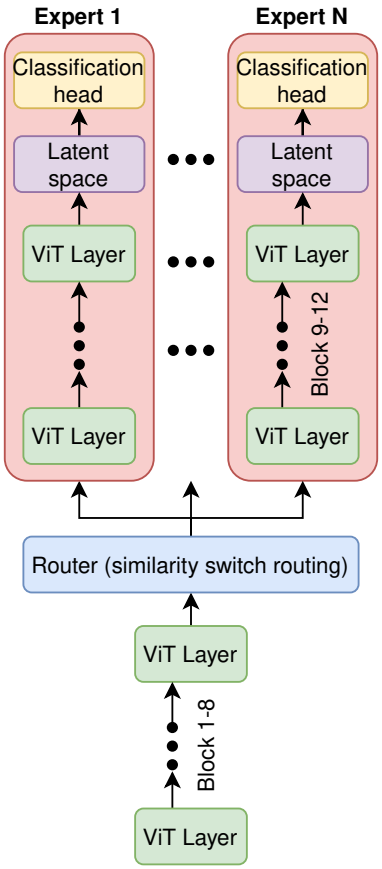


Mixture-of-Experts Architectures Through the Lens of Continual Learning

Master's Thesis in Complex Adaptive Systems

IAN COSS MAC LEOD

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Degree project report 2026

Mixture-of-Experts Architectures Through the Lens of Continual Learning

IAN COSS MAC LEOD

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Mixture-of-Experts Architectures Through the Lens of Continual Learning
IAN COSS MAC LEOD

Supervisor: Viktor Valadi, Machine Learning Engineer, Scale Out Systems
Examiner: Mats Granath, Complex Adaptive Systems Director

Degree project report 2026
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Diagram of solution variant 2 for adapting mixture-of-experts for domain incremental learning.

Typeset in LaTeX

Gothenburg, Sweden 2026

Mixture-of-experts architectures through the lens of continual learning
IAN COSS MAC LEOD
Department of Physics
Chalmers University of Technology

Abstract

Mixture-of-experts architectures on a vision transformer backbone are compared against standard architectures for image classification in continual learning challenges with the constraints found in autonomous vehicle onboard systems and a novel routing algorithm is presented for improving MoE performance in this setting. Domain incremental learning without domain labels and class imbalanced datasets are used with continual learning and imbalanced learning metrics to describe when MoE architectures become useful and what advantages and drawbacks one should consider. Results show that MoE should be used in highly complex datasets with domain focused routing to improve the architectures natural resistance to catastrophic forgetting but with current MoE strategies, large gains are not yet realized. Suggestions for strategies to pair with MoE for continual learning are given alongside guidance for MoE training in this environment.

Keywords: Image classification, mixture of experts, deep learning, continual learning, domain incremental learning, new instance classification, vision transformers, geometric router.

Acknowledgements

Thank you to Viktor Valadi, Salman Toor, Mauricio Muñoz, and Mats Granath for their expertise and patience in the exploration of this topic. Thank you to the AI Sweden team for their gratuitous and welcoming use of their facilities and servers to complete this project. Thank you to Julia and my family for their endless support in my efforts. And thank you to Chalmers and the Complex Adaptive Systems staff for the opportunity to learn these fascinating topics.

Ian Coss MacLeod, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

$A_I A$	Average Incremental Accuracy
AI	Artificial Intelligence
BWT	Backward Transfer
CL	Continual Learning
ConvNeXt	Convolutional neural network NeXt generation
DIL	Domain Incremental Learning
EMA	Exponential Moving Average
FFN(N)	Feed-Forward Neural Network
FWT	Forward Transfer
G-Mean	Geometric Mean
IID	Independent and Identically Distributed
IM	Intransigence Measure
LLM	Large Language Model
MAUC	Multiclass Area Under the receiver operator characteristic Curve
ML	Machine Learning
MLP	Multi-Layer Perceptron
MoE	Mixture-of-Experts
NC	New Class
NI	New Instance
NIC	New Instance and Class
TIL	Task Incremental Learning
ViT	Vision Transformer

Contents

1

Introduction

1.1 Context

Traditional neural network training follows a standard formula: pre-train, finetune, and then freeze the model weights for inference use. If this were to be likened to a human, this is the equivalent of anterograde amnesia after learning a topic [1]. When the topic that someone learned evolves, such as what a computer looks like in the 1980 versus in 2026, they would no longer be able to identify it.

While traditional neural network training is sufficient for less complex scenarios or for situations where the data representations are stable and a clean dataset can be generated, in real world situations the data representation can shift over time but old information must be remembered. Continual learning questions the need for freezing the model and instead designs strategies to allow for the training to continue and for the model to continue to improve and adapt even after its initial training. Of the variety of complications that arise due to this new problem formulation, catastrophic forgetting sits at the forefront. With current training methodologies, old memories are lost due to the new memories gained, similar to retrograde amnesia. When a model is being trained in a traditional setting, the dataset is designed to be independent, identically distributed, and to represent all definitions possible, and then freezes the weights to retain the memory of the distribution. This is not an assumption that can be made in continual learning. Data is coming in as a stream as the model is used. If over time the data representation changes, then the model can start to forget what it has already learned. On the other hand, if the model does not update its definitions, it may not be able to adapt to its environment. In autonomous driving, this could be as car models change, being in the countryside versus a city, or driving in the summer versus in the winter. Balancing this ability to adapt against the ability to remember is called the stability-plasticity tradeoff.

One method that shows potential in reducing forgetting is a modified network structure called mixture-of-experts (MoE) [2]. MoE models are a method to sequester information into separate blocks. Instead of running a model with all of its parameters, a router selects which partition of the model to use for a given input. These partitions are made explicitly, and named experts. A router then has the job of learning which expert to use based on the input. An example of this could be if one had a two-stage process for sorting: First, whether the symbols are hieroglyphics or Braille, then giving an expert in that written language the task of translating it. This partitioning allows for a given data definition to be updated only when a expert is used and increases memory capacity by a factor of the number of experts.

In this way, MoE's are expected to be able to reduce the forgetting that a model has without significantly increasing compute at runtime. While this holds in traditional settings and in monolithic, massive neural networks, little research has been done to explore this in the continual learning setting or in smaller models. To explore this theory, standard and MoE model variants of a vision transformer model are compared to study forgetting in continual learning vision classification tasks, and a new routing methodology intended for continual learning is presented to compare with existing methods.

1.2 Goals

This project was done in cooperation with Scale Out Systems and AI Sweden to explore MoE's in the problem of catastrophic forgetting in continual learning scenarios such as autonomous driving. Use of smaller models, large datasets, and constraining to solutions that do not grow the model size over time and protect the privacy of users are used in order to align to this.

The goal of this research is to establish the efficacy of MoE models against continual learning issues and to search for methods to improve viability.

The research questions are as follows:

- How does behavior of an MoE model differ from standard model architectures in continual learning setups?
- Do MoE architectures have an inherent advantage in continual learning?
- What can be done to improve MoE behavior in domain incremental, domain agnostic setups where privacy and restricted memory must be considered?
- What existing continual learning techniques can be used with MoE architectures?

1.3 Limitations

Datasets used for this investigation are smaller in order to allow for more tests to be performed. There is little noise in these datasets, which does not match real-world conditions, nor are the images of identical size. In real-world continual learning the dataset size would be essentially infinite so instead behavior must be matched to different points in a model's life. Early learning would be random initialization and pre-training represents the long term learning with different tasks prior to learning the newest task.

The models used are small or tiny variants in order to allow for testing, fit the target environment of vehicle onboard devices, and to reduce the chance of the model storage being enough to fully learn the dataset. Hyperparameter tuning was done by heuristic testing, as parameter sweeps are intensive and the models were found to be decently stable with the parameters used. Also since it is a comparative study of various models, parameters had to be matched between models for fairness.

Existing continual learning methods are not implemented in the models unless absolutely necessary. Adding additional algorithms inside of the training would lead

to the necessity of ablation studies to detect whether the algorithms unique to this paper improved forgetting and accuracy.

Data replay and growing network strategies have been shown to be the most effective solutions in this setting. Data replay is not used due to data privacy considerations of the use case. Growing networks are not used due to the limited memory of vehicles. Similarly, network distillation is avoided due to the memory doubling needed for storing old model checkpoints.

1.4 Ethical considerations

As the topics in this work include deep learning, image data, and a model that can change over time post-deployment, certain ethical considerations must be considered.

Deep learning is an energy-intensive algorithm. The use-case must be considered and confirm whether other energy efficient solutions exist. In the case of image data, deep learning algorithms have been shown to be far better at class discrimination [3]. Also, mixture-of-experts specifically has been shown to be a method that encourages the use of smaller models or more sparsity within a model at runtime [4]. As with all deep learning algorithms, the inherent black-box nature of its decision making must also be recognized as a potential downside for use. Testing of the experts and of weight significance in decisions made could allow for better understanding of the underlying behaviors and biases of the systems described in this report.

The pre-trained backbones on ImageNet and the biases of the datasets used in this analysis do contain inherent biases of the datasets themselves [5], [6], [7], [8], [9]. As such, any person intending to use the model checkpoints generated by this thesis should consider such before use, however the specific checkpoints are not intended to be released to the public.

The architectures and use-case specified must also be considered. Autonomous vehicle technology is high-risk as in the European Artificial Intelligence Act section 6 [10], and model updates after deployment could have potential danger to the user. This work is not intended to be used in these scenarios without considering the "substantial modification" rule as in the European Union AI Act section 12 [11]. Checks should be done for the fundamental safety baselines, and traceability metrics for the routing decisions of the MoE system should be logged if to be used in a production-ready environment.

The design of this thesis should also consider dual-use, as the technologies described in this thesis could also potentially be used in warfare technology, but this is not the focus or intention of the work.

2
Theory

2.1 Vision transformer structure

First, a short delve into vision transformer architecture, as this will be used as the backbone for the MoE models tested. Multi-head attention and reasoning for the structural decisions behind a transformer will be left to the seminal paper *Attention is All You Need* [12]. Vision transformers [13] are a slight modification of the original architecture to allow images to become tokens and then apply attention to the tokens. Vision transformers are often used for image classification, so the encoder side is used and then a classification head is attached as shown in Figure 2.1. Additionally, a special patch called the class token is added after the image is fed through that is a learnable token to hold additional useful information about the potential class of the image. At the end of each transformer block, there is a latent space bottleneck that encodes the patches into it and then adds and normalizes this latent representation with the representation of the prior layer. The latent space starts without any specific meaning, but can be trained to have semantic meaning held within. Understanding this latent space bottleneck is essential in both the conversion of a ViT into a ViT-MoE and in how routing in an MoE function and will be discussed later in this chapter.

2.2 Mixture-of-experts

The basic idea of a Mixture-of-Experts architecture is to take a feed-forward network and replace it with multiple of the same size with a router ahead of it that decides which FFN expert gets each piece of data. By routing similar data to the same expert, it gains specialization in that topic. The router is simply an MLP that gets the same information as the expert but gates which expert the data runs through. The job of the router is to perform latent space clustering. The Mixture-of-experts models in the deep learning space were originally designed to improve the sparsity of the model at inference time. A smaller FFN can be used, but by using the same size of FFN as the original dense model instead for each expert, the model information capacity is several times greater than that of the actual number of parameters used each time. Large language models were a natural fit for this, as there were a huge number of parameters and little of the information available in the monolithic architecture was important to a single given input. As transformers and MoE architectures became more widely used, vision transformers with MoE layers [14] as shown in Figure 2.2, were created in order to improve abilities of models to

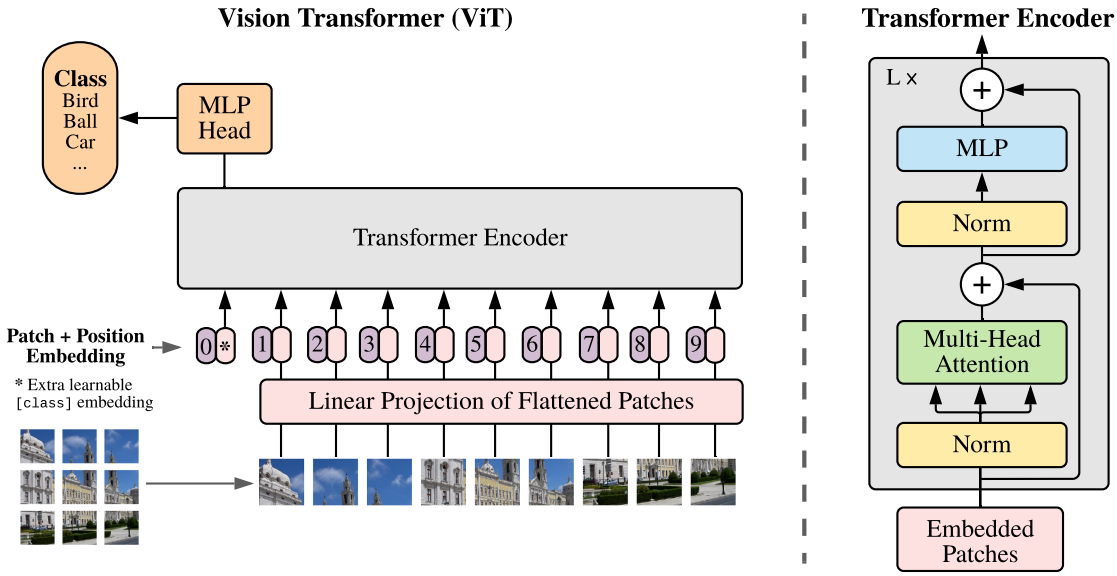


Figure 2.1: This figure adapted from Dosovitskiy et al (2021)'s [13] explanation of ViT structure shows how an image is fed into a ViT and the structure of a ViT encoder block, typically 12 blocks deep, for a ViT. The latent space representation bottleneck is at the end of each encoder block. Each numbered sub-image is then flattened into a vector and fed in to the ViT with positional embedding.

handle a variety of media and subjects. The bulk of work in the use of MoE in vision classification branches off of this design as the structure of transformers has a natural place to input MoE layers. Older, pre-deep learning styles of MoE mixed with deep learning MoE layers, opening up ensemble MoE [15], soft MoE [16], switch MoE [17], and top-k MoE (already introduced in Shazeer et al [2017]), where all or a subset of the experts are used at inference to handle the data, and the data could be routed by the full input or by token. Having the experts request tokens [18] or the tokens matched to experts through various mechanisms and structures have been explored to avoid expert collapse and enhance expert diversity. Shared experts add in a additional expert that is always used to handle standard features and then unshared experts are selected by the router to handle the more unique features in an input [19]. Routers have not changed much structurally but there have been new losses added to the training of the router to encourage use of experts past stochastic selection or selection of the expert that will perform best at the current moment during training [20]. Auxiliary loss functions applied to the router to force experts to be used is called load balancing loss [14] and to teach stable expert selection for a given input called consistency loss [21] where a single input with different augmentations are sent through and loss is based on whether it is correctly sent to the same expert each time. Setups involving hierarchies of routers - aptly named hierarchical MoE [22], [23] - to select an expert were also designed for decomposing problems even farther, and diversifying experts by structure of their neural network explored how to make experts definitively unique in what they learned. Uniqueness losses such as loss on expert-to-expert output difference [24] and loss on relying overly on specific experts via z-loss [25] are other ways that researchers have found

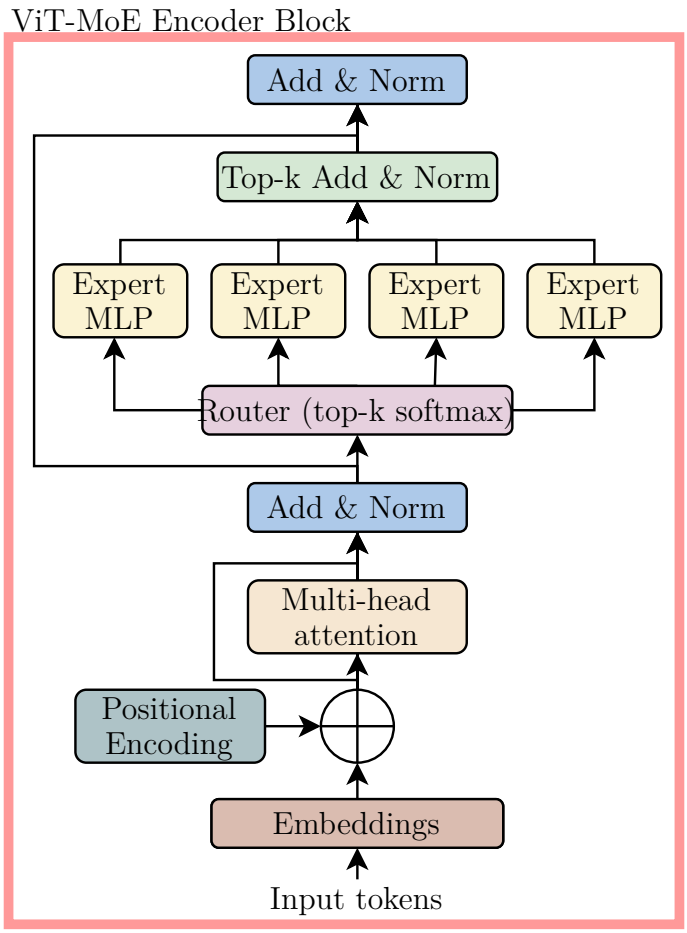


Figure 2.2: The MoE version of the ViT encoder block. Router selects a configurable number of experts (top-1 in this research) and the others are not activated. The experts output when multiple experts is added and normalized before adding to the carried over representation.

to try to force specialization to some success. However the bulk of this work assumes a single dataset and I.I.D data to create the expert specialization.

2.3 Continual learning

Continual learning has multiple setups, but there are common threads throughout all of it: Buffers of data that is used and then deleted, non-stationary distributions, and use of training throughout the inference use of the model. The buffers can be single pass or multiple pass, depending on whether it is true online learning. In settings such as vehicles, one would expect there to be a buffer gathered throughout the day and then training during the night to occur. The distribution changes can be from imbalanced class representation, changes in how a given class looks, changes in how all classes look, changes in what the labels represent due to a change in task, or additional classes being added over time. Domain incremental learning focuses on multi-pass buffers with changes in how all classes are represented over time, but there is a common underlying structure to each given class. Other types such as task-

incremental, are also of interest in research, and can be seen in reviews of continual learning for additional information. Task/domain agnostic learning is when the task/domain is not directly given as input and instead it must be inferred by the model, meaning that there is unsupervised domain identification occurring within the model. For the purposes of this investigation, imbalanced class representation and domain agnostic domain incremental learning are the focuses.

Standard continual learning issues of catastrophic forgetting and stability-plasticity tradeoff have multitudes of strategies for mitigation. An example of a network with no mitigation going through training can be shown in Figure 2.3. There are five mitigation categories: regularization, replay, optimization, representation, and architecture. An in-depth study into all methods can be found in *Wang et al*[26]. The below are included to explain their applicability to the constraints and topics in this project.

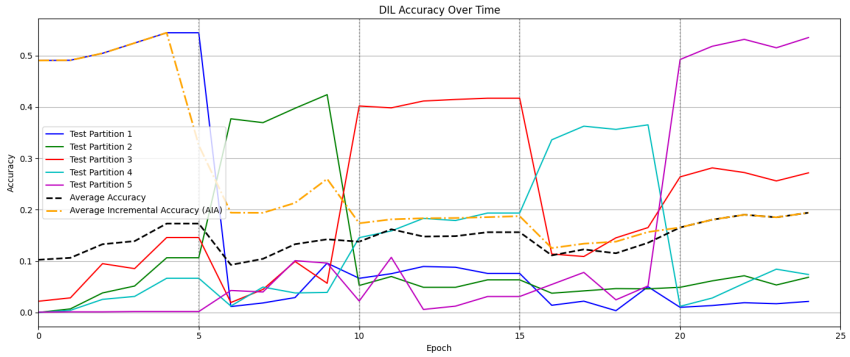


Figure 2.3: Behavior of a typical network in continual learning without mitigation strategies. Vertical lines are when a new domain is started on. Domain accuracy improves when training on that given domain, but the overall accuracy fails to improve.

Replay approaches have been the most successful and simple, where the issue of the data being non-I.I.D. is solved by saving old samples to be trained on with the new incoming samples, selecting samples to create the best statistical cover of the dataset and thus regaining a semblance of classical learning from the continual learning problem. The samples can be saved as raw inputs, latent features, or generated from the distributions from the model and fed back in. DER++ [27], iCaRL [28], and GDumb [29] are three popular methods. These methods do risk leakage of training data so for privacy aware methods this type of solution is often avoided, but in domain incremental domain agnostic learning, the only method that has shown positive results are replay based methods. There are some that instead use feature replay [30], which is better for privacy but still has risk. Better yet is generative replay [31] and generative feature replay [32], which make it difficult to extract actual image data from the model. The other issue that is still not fixed for the setting and constraints of this project is using compute time for old, previously ingested data instead of only new, novel data.

Regularization focuses on keeping important information less plastic than the rest of the model. Popular methods check each parameters importance to prior inputs or the sensitivity of the model to change of parameters for a given task and

then penalize changes to the important parameters. Learning without Forgetting [33], Elastic Weight Consolidation [34], Memory Aware Synapses [35], Synaptic Intelligence [36], and AR1 [37], are widely used, but they require knowledge of when a task is changing or data from prior tasks, which again are not desirable in online solutions where domain may not yet be known. LwF and others require checkpoints of the model to generate this meta-information when knowledge distillation losses are used, which doubles the memory footprint of the model, often undesirable in mobile applications. Ones require significant compute and memory additions such as EWC and MAS are powerful as well, but again increase the footprint significantly. The best methods tend to be the most compute heavy, however AR1* [8] has been shown to be both lightweight and not require checkpoints or significant memory additions.

Optimization approaches look at the information gradients and the loss landscape over time in order to identify how to align latent representations of each task/domain so that the model solves for the underlying structural representation matches. These options may also try to modify the loss gradients to be flatter, thus stabilizing training. Meta-learning falls under this umbrella, where a successful model checkpoint may be used to initialize at task boundaries. These solutions carry similar issues to replay and regularization approaches and tend to require underlying similarities in the overall data. The approaches here are often intermixed with the other approach types to obtain information that can be used to better understand the loss landscape.

Representation based ideas use pre-training or self-supervision. Pre-training has become a cornerstone of continual learning, as having well-organized latent spaces prior to learning allows for strong generalization and latent space clustering to guide training early without risk of gradient explosion. Self-supervised pre-training can also be used, depending on the structure of the model and data, to have a latent space that is well-organized for the dataset prior to training. Self-supervised contrastive loss [38], small adapters on a frozen backbone [39], and some forms of knowledge distillation are examples of this. The use of a pre-trained backbone for transfer learning is simple to implement and is well suited to our environment, so multiple methods are explored.

The final type is architectural, where the structure of the model itself can help solve the continual learning issues. Partitioning the model based on task [40], [41], expanding the network as capacity is reached [42], task-based parameter masking [43], [44], and MoE are all popular solutions here. Of note, batch normalization, slim networks, and lack of dropout all are found to cause forgetting. However, these methods do not have any inherent privacy concerns or memory expansion, albeit specific methods within this strategy do. Interestingly, groups of small, sparsely activated models tend to perform equally or better in accuracy with a decrease in forgetting [45]. The focus of the work done in this thesis all focus on the advantages that architecture can give, only using the other methods where absolutely necessary.

2.4 MoE's in continual learning

Architectural approaches is where the standard MoE model would be located with small changes to allow for better continual learning use of experts. Nearly all of the

methods here are heavily related to MoE and the vocabulary used in continual learning can be connected to the MoE vocabulary. Expanding the number of experts to add capacity with new tasks and merging/freezing old experts is the most standard method [40], but with fixed size compute and a lack of definitive domain boundaries or more domains/tasks than experts will lead to similar issues as in dense inference models. One strategy to overcome this is domain or task based clustering such as in [46]. However, the routing methodology must be heavily altered for the continual learning domain/task incremental regime. Unlike standard models in CL, MoE can forget in two ways: First, by overwriting the weights like a standard model, and second, routing to the wrong expert for a sample thus appearing to have forgot the information but instead forgot the routing. Thus changing the router to properly utilize the experts is needed, since classical load balancing loss causes approximately equal usage of experts enforced at all times. This load balancing would lead to experts with little semantic expertise and thus collapse or forgetting will occur as the distributions of the input change. Router strategy tends to collapse into entropy calculations with soft or ensemble training of experts or hard task-label based routing. The performance of the model in continual learning is primarily based on the quality of routing to the experts, as the information similar to the input needs to be selected for accurate results. Catastrophic forgetting takes a new form in this structure, where simply failing to route to the right expert can make it appear to have forgotten information while the real issue was accessing it. Routing similar domains to the same expert but also utilizing experts effectively is a difficult problem. When should one start using a new or different expert? How close are the domains even though one does not yet know all of the domains that exist? How does one keep old experts compatible with the overall architecture? This is where a new research space of routing solutions specific to continual learning begins.

Routing in continual learning MoE without domain labels has taken a few directions so far: online input clustering [47], domain label generation [48], or prompt-based experts [49], [50]. Some of these ideas have been done in non-continual learning setups, but would effectively apply to continual learning. Auxiliary losses as discussed in 2.2 can help to improve these methods as well. While these ideas can theoretically improve retention and cross-domain alignment, the reality is that there has yet to be a method for MoE continual learning that does not require meta-information for routing effectively, but there is research in a field known as geometric routing that shows promise.

The primary idea in geometric routing is to view the latent space as a finite area to split into sub-domains that each expert can specialize in. Some create latent basis vectors to describe each expert at each layer [51], [52], while others try to create clusters to match with experts [53], or have data about prior inputs and try to match the prior input data or EMA-update conditioned prototype with the expert [54], [55]. Geometric routing in general shows less improvement than other methods, but does not have the memory, privacy, or require freezing of the model that the other methods do, so this is the topic that the method designed in this thesis is based upon. The most similar research that has been done in this focuses on latent space partitioning by cosine similarity to expert centroids, however the methods of how the centroids are generated and used differ [56], [57], as does the

setting and metrics used to compare performance.

2.5 Continual learning metrics

Metrics used to describe continual learning have to encompass what is learned and when, what is forgotten and when, and whether the prior/future domains are helping or hurting the representations of each domain. It is essential to note that the general comparative trends of the metrics are important since it is continual learning. The exact values at the end are far less important than in standard model training.

The majority of the metrics are derived from the R matrix as described in [26] (R matrix measures), [58] (accuracy), and [59] (IM), which is the accuracy of the model on each domain test set at the end of training on each domain. A baseline model is used for the rest of the information needed for the metrics, which is a model with identical initial weights and training recipe but trained on only the single domain. Comparing the baseline model accuracy per domain against the actual model that has been trained sequentially on the domains allows for extrapolation of how much the other domains improve or diminish the training of each other. To have better resolution on what is occurring in the models, modifications to the equations were made to use the current epoch in a domain as a stand-in for final domain accuracy until the end of the domain so that the intra-domain dynamics could be captured. Table 2.1 briefly explains each metric and Appendix A yields the exact equations, with Appendix A.11 explaining the intermediate metric extension. Other metrics commonly used in continual learning are omitted, such as model size efficiency and sample size storage efficiency, since the constraints of the project would lead these metrics to be meaningless. Stability is usually defined as the forgetting measure and backwards transfer while plasticity is defined as the intransigence measure and forward transfer.

Imbalanced learning metrics are also applicable to continual learning. Class representation is often imbalanced in a given domain, so classical accuracy definitions do not effectively encompass balanced performance over all of the classes regardless of class size and discriminative ability of a classifier between classes. These can better summarize data one might find in a confusion matrix over time and account for the class imbalance simultaneously. Specifically the geometric mean and the receiver operating characteristic area under the curves extended to multi-class settings (MAUC) [60] are used to better understand classifier ability.

MoE research also has metrics, but these are less standardized and often do not effectively translate into continual learning setups. The main MoE metric used was in the expert usage per domain and overall. This allows one to see how much experts specialize in a domain versus in a classification subspace. Equal usage within a domain of experts would imply classification subspaces are being created, while heavily imbalanced usage could imply either expert collapse or domain specialization. Expert collapse is when only a single expert is ever used, thus wasting the actual expert system and collapsing to a non-MoE system. Domain specialization is when multiple experts are used over the dataset, but heavily imbalanced usage within each domain.

Classical metrics of overall accuracy and confusion matrices were used in tuning

and some are included in the additional data in Appendix C, but these are not essential to the findings of the research.

Due to the definition of the continual learning and imbalanced learning metrics being intended for classification performance analysis, this research uses classification to study the effects of MoE in continual learning.

2.6 Continual learning datasets

There are many versions of continual learning that are being studied and each requires its own setup to properly induce the setting that is being studied. Some of the types of continual learning, such as imbalanced class representation or class incremental learning, can use existing standard datasets with augmentations to the data itself or a controlled partitioning of the dataset. Imbalanced classes representation can be made by removing portions of the dataset based on class, Dirichlet partitioning [61], or static probabilistic partitioning. Dirichlet is a particularly useful case, as the total class representation is approximately equal, but each partition can have severe imbalance, often removing some classes entirely in a given partition . This means that the training must be carefully controlled to handle the class imbalance in each partition but that the overall representation of classes in the total dataset is still the same, allowing for comparisons against classical use of the dataset as an upper limit for performance. Static probabilistic partitioning is an easier problem where each domain has a representation randomly sampled using static probability during the generation of the partition. The exact probabilities for each class over all of the partitions should add to one, and the representation imbalance can be precisely controlled by the designer.

Domain incremental learning requires that the underlying structures of the class should have some shared concepts and that the types of overall structure are varied per domain. This means that when the concepts are disjoint, it is considered task incremental learning instead. Most datasets are not easily redefined for domain incremental setups, as all of the images would have to be reviewed and domains would have to be defined for each. As such, domain incremental learning datasets have been designed where each domain has all classes and there is an included domain label for each image. Typically the data is given to the network sequentially by domain so that the classes all have domain shifts at the same time and they are instant switches. Gradual domain shift is better simulated in blurred boundary continual learning. In domain incremental domain agnostic learning, which is one of the settings used for this investigation, the domain labels are never given to the network during training or inference. Domain agnostic learning is difficult since the class definitions drift as each domain arrives and the clusters in the latent space that most ViTMoE models use to route are no longer stationary targets and can cause forgetting due to poor routing.

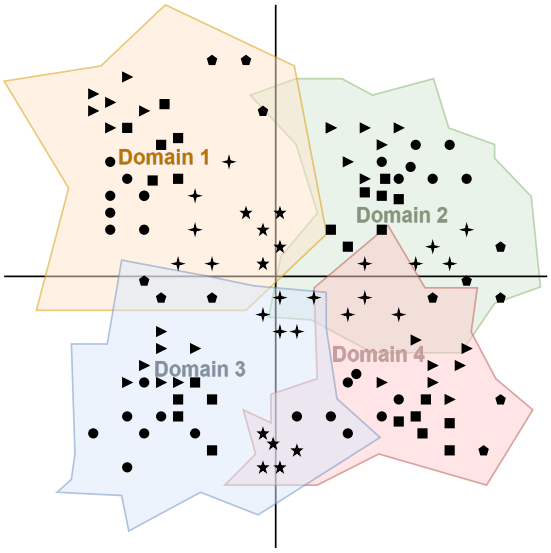


Figure 2.4: Simple examples of how class clustering occurs when strong domain separation may appear in the latent space. The black shapes are various classes.

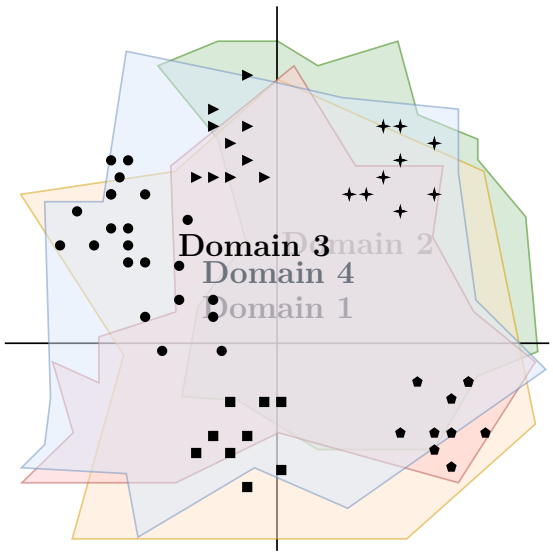


Figure 2.5: Simple examples of how class clustering occurs when domains are weak/non-existent. The black shapes are various classes.

2.7 Domain separability

Domains in domain incremental learning must be discussed for a moment. The degree of change from domain to domain can be essential in whether a network and training design will be able to handle it. When a domain is something that could nearly be an augmentation of the same image - such as winter versus summer or indoor versus outdoor - the domain change could almost be considered noise. This means that the system has to learn to handle noise and that the latent space will be unlikely to encode the domain as part of an input. Whereas domain change where it is fundamentally different, like a fast pencil sketch of an object on paper versus an image of the same object in a cluttered house, is likely to be part of the

latent encoding of the image. Now the system would need to be able to identify sub-clusters within each domain in the latent space in order to classify the object. There is no standardized way to define the domain change difficulty of a given DIL dataset, but this difference is essential in knowing whether a system can handle the degree of domain change that would be expected in the real-world continual learning of the system. The degree of domain prevalence in the latent space would also change based on what pre-training the model has, the size of the model, and the size of the latent space, among other factors. Examples of how clustering may appear in the latent space are shown in Figure 2.4 and Figure 2.5. Domain groups can also potentially be separable at different depths in the model. Strategies for mitigation must then also take the degree of domain separability into account.

Name	**Mathematical definition**	**Colloquial description**
Performance Matrix (R-matrix)	$R = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,D} \\ 0 & a_{2,2} & \cdots & a_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{D,D} \end{bmatrix}$	Matrix where $a_{i,j}$ represents the performance on domain j after learning domain i.
Average Incremental Accuracy ($A_I A$)	$\frac{1}{D}\sum_{d=1}^{D} a_d$	Mean accuracy over all seen domains up through domain D.
Forgetting Measure (FM)	$\frac{1}{D-1}\sum_{d=1}^{D-1}\left(\max_{l\in\{1,\ldots,D\}} a_{d,l} - a_{d,D}\right)$	Average loss of performance on previous tasks after learning domain D.
Backwards Transfer (BWT)	$\frac{1}{D-1}\sum_{d=1}^{D-1}(a_{d,D} - a_{d,d})$	Influence of learning new domain D on old domains d.
Forwards Transfer (FWT)	$\frac{1}{D-1}\sum_{d=2}^{D}(a_{d,d-1} - a_d^*)$	Effect of previous learning on future unseen domains.
Intransigence Measure (IM)	$a_D^* - a_{D,D}$	Inability to properly learn a new domain D compared to baseline model $*$.
Geometric Mean (G-Mean)	$\left(\prod_{c=1}^{K} \text{Recall}_c\right)^{\frac{1}{K}}$	Balanced performance across classes K using geometric mean of recalls.
Receiver Operating Characteristic Curve Integral Multi-Class Extension (MAUC)	$\frac{2}{K(K-1)}\sum_{i<j} AUC(i,j)$	Ability of a classifier to discriminate between multiple classes K independent of a decision threshold.
Per-domain Expert Utilization	$\frac{N_{d,i}}{N_{d,total}}$	Fraction of samples from domain d routed to or handled by expert i.

Table 2.1: Descriptions of the various metrics tracked in the project. They come from continual learning, imbalanced learning classifier metrics, and MoE metrics to give a better overall understanding of system performance. Continual learning definitions are those described for domain incremental setups. Check Appendix A for full mathematical descriptions and variable definitions if unfamiliar.

3
Methods

3.1 Experimental setups

The experiments done in this research are split into class imbalance partitioning and domain incremental learning without domain labels. CIFAR-10 [6], CoRE-50 [7], and Office-Home [9] datasets are used to better understand MoE in continual learning. CIFAR-10 allows for a small, easy dataset to show the difficulties presented by the continual learning setup in class imbalanced settings. CoRE-50 is innately set up to be used in continual learning domain incremental setups, however the domain shift itself is small. Each domain in the data has different lighting conditions and backgrounds for the objects to classify in the image. This is to explore when the domain shift is less significant than the inter-class shift, which means that the latent embedding space would be unlikely to present the domain shift within it as a primary dimensional factor. Office-Home is a dataset where there is larger domain shift. There are four domains: Art, clip-art, product, and real world. All three datasets are small datasets, which is the greatest weakness of the dataset selection compared to real-world continual learning, but allow for prototyping and testing of the various setups done in this project and enough time to run five seeds per experiment per model, allowing for statistics of the results and lowering the likelihood of the randomness of a single run to skew results.

To have an understanding of the behavior of models without pretraining or any continual learning mitigation strategies, a simple test on CIFAR-10 where the data is partitioned into five sections and training is done sequentially on each section is done. Class imbalance is also tested using either Dirichlet with a value of 0.3-0.5 or static sampling per partition, which simulates the prevalence of a subset of classes in a given buffer of data in true continual learning setups. The algorithm used to generate Dirichlet partitions is done as in [61]. These initial tests were also instrumental in understanding MoE trade-offs during tuning and checking behavior with different auxiliary losses.

CoRE-50 and Office-Home setups are all done in domain incremental fashion. The order of the domains is not changed, but the order of image presentation within a domain is random. The partitions are done by domain and can be ran for multiple epochs per partition. The motivation for this is to best represent the learning conditions found in AV, where a given set of data can be trained on while the vehicle is parked for extended periods.

Tests with models in the 10 million active parameters range and 30 million active parameters range are performed to see if model size has any significant effect.

The MLP ratio is tested to have two types of fair comparisons between the MoE variants and non-MoE models. One way to compare is approximately the same number of active parameters, meaning that the parameters used by the models in a given inference is the same. The other is having the same number of total parameters, meaning that the MoE MLP ratio is reduced in order to have the same number of total parameters as the dense models. The ViT models that were used as comparisons against the ViTMoE had identical structure except for the router and additional FFN experts. The ConvNeXt [62] model used had similar number of parameters to the models compared against, except for when the models that were compared against were smaller. This model is used as a benchmark for what is popularly used today.

Data augmentations used in the tests resized all images to the same size, used random resized crops, random flips over the vertical axis, and low levels of color jitter. Some CIFAR-10 experiments also include Gaussian blur and noise as compensation for the small dataset size.

All models use the same loss, learning rate, optimizers, warm-up ramps, pretraining, epochs, and random seeds for a given experiment. Weighted cross entropy loss was used to offset the problem of class imbalance, however a cap on weight was introduced to ensure an extremely low representation of class would not destabilize training. AdamW and Adam are popular choices in continual learning so these were also used in these experiments due to their robustness in performance in these scenarios. Pretraining was done in two ways: one was to take a hold-out subset from all domains and pretrain a randomly initialized model for some epochs on this subset, and the other was taking a pretrained ViT model and copying the model weights into each expert for the MoE variants. Models with no pretraining whatsoever were also explored to see if there were differences in how forgetting was portrayed in pretrained or randomly initialized models existed. The number of epochs per domain/partition was usually between 5-25 epochs, however some isolated experiments used 1 epoch. The seeds were controlled for each section of training and initialization of both the model and the dataset in order to have the dynamics of the model be comparable without concern for unfair stochasticity in the run itself. All experiments used 5 runs per model to have results that represent the standard training case. A full description of all experiments can be found in Appendix B.

The models used for understanding overall effectiveness of the MoE solutions and have a stand-in for current popular models were a standard ViT with identical setup to the MoE, sans the MoE layers, and a ConvNeXt model. MoE models explored in the initial phase of this exploration had various setups, including shared experts, token level routing, soft routing, and different layer combinations for which layers of the ViT have their FFN structures replaced with an MoE layer. Router balancing and router freezing was also tested to see the effect of stationary routing or stabler routing had on the MoE. Frozen routing, image level routing, router balancing, and shared experts are all methods that in the past have shown better training for MoE in continual learning. Router balancing was done via equations shown in 3.2.4 as an auxiliary loss.

For each model tested, a baseline model is trained alongside where it is identical to the initial condition of the model before training. If pretraining occurred, this is

inherited into the weights of the baseline model. The baseline model is then trained on only a single partition for the same number of epochs as the actual model on that domain. The accuracy of the baseline model allows for understanding whether the training of the actual model on all of the domains improves or hinders learning of each individual domain.

In order to control all of these variables and settings, a testing suite was designed from scratch to use model and experiment configuration files and sorted for fair comparisons of models and experimental setups within meta-configuration files to run comparative studies on models. Results for each experimental run were saved with per-epoch metrics as those described in Chapter 2. Table 3.1 shows explanations of the experiments of interest in the results section, however longer explanations of the exact training structure, values, and more can be found in Appendix B and in the configuration and meta-configuration files available on the public repository [63]. Additional experiments past those selected for interest in results are also available in the appendix.

3.2 EMA centroid cosine similarity router MoE designs

Mixture-of-experts models have two ways that catastrophic forgetting can manifest. One is the same way as standard models where the information is overwritten by shifting gradient landscapes. The other is in failing to access the information by routing to the wrong expert. Research in MoE continual learning recommends to freeze the routing after an initial learning session for the router, or post-pretrain [64]. But this effectively just freezes the model with old information and does not allow for experts to become experts in a specific topic over time, nor does it consider the implicit clustering that the router can perform as it learns . Others route via domain label, but this is not available during training in the domain agnostic setting. In order to mitigate both of these problems, the following solution is presented: route via domain, and when there are similar domains, use the same expert for similar domains. As this is done in a domain agnostic setting, the method of routing via domain without prior knowledge of domain is difficult. Then there is the issue of quantifying which expert is in charge of which domain and how to measure what domains are close to each other. Two strategies were created with this in mind, as well as the constraints and goals of where this solution may be used: a design that works for both smooth or sudden domain shifts, privacy aware, does not grow the number of parameters, utilizes the MoE structure, does not require meta data other than class label during training, and attempts to weaken the issues caused by continual learning.

3.2.1 Latent space leveraging

In order to have stable domain routing that is emergent as domains appear, the latent space of the transformer is leveraged to enforce similarity. A latent space should implicitly cluster similar structures together, as shown in Figure 3.2, which

shows ideally how the routing would behave. This is why a simple MLP layer after the final encoding latent space works for classification. In earlier layers, when domain is a prominent way that the data is presented, should be domain separable. A potential difficulty with this method is when the latent space is not yet stable, which can be mitigated by either using a pretrained encoder or to detach the encoder representation initially and later once stable to switch to using the newest latent space for the routing.

3.2.2 Similarity measurement

In order to measure similarity in the latent space, cosine similarity is used. Euclidean distance uses both magnitude and angle. Attention in transformers causes the latent space to be oriented via angle. L2 normalized Euclidean distance is proportional to cosine similarity, so this allows for a simple method to calculate angle similarity in a high-dimensional space.

$$\text{Similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

3.2.3 Expert-domain assignment

Routing is essentially creating a partition of the latent space. To make this explicit, experts are modeled as prototypes in the latent space. Other research has performed this in the past, but as an auxiliary signal for routing instead of explicit routing selection. The argument for explicit, non-MLP routing is that it protects the definition of a given expert better. The learning signal that is based off of which expert would perform best leads to expert collapse or class-based or feature-based partitioning of experts instead of domain-based partitioning. Since all classes are in each domain, all experts would get overwritten with each new domain. Thus, cosine similarity of the expert prototype in the latent space defines angular partitions of the latent space. The next step is to ensure that the prototype of each expert aligns with what the expert is becoming an expert in.

Online exponential moving average updates, as in the below equation, of an expert prototype can represent what the expert is being trained on and thus what it has learned.

$$\theta_{\text{EMA}}^{(t)} = \alpha \cdot \theta_{\text{EMA}}^{(t-1)} + (1 - \alpha) \cdot \theta_t$$

Where: $\theta_{\text{EMA}}^{(t)}$ is the updated moving average at the current step.$\theta_{\text{EMA}}^{(t-1)}$ is the average from the prior update.θ_t is the newly observed value.α is the momentum or decay rate, which defines how quickly historical data decays. This is used instead of simple online moving average to allow for the latent space to be restructured without collapsing the expert definition and to better represent the current usage of the expert. The expert is updated after each batch to the mean of the features of the mean of the tokens that were routed to it within the given batch.

A variant, which will be called variant 1, where each expert had a slow and a fast EMA centroid was also created, where the slow was updated at a much higher

α via the fast centroid location, and the fast centroid was updated as described previously, was also tested to see if more stable expert prototypes would behave better. This strategy was motivated by *Nested Learning* [1] and *Learning Fast and Learning Slow* [65]. The routing was then based off of a β value that weighted the fast centroid location against the slow centroid location for cosine similarity of the image in the latent space at each MoE block. The initialization of the prototypes simply used one image each from the first batch per expert. Domain clustering would be emergent as additional training samples were presented. In transformers, the latent space representation bottlenecks occur in each block, so this strategy can be used at each block.

One other variant, which will be called variant 2, uses two centroids as well: one as the mean of the patches of all of the images routed to a given expert each batch, and the other as the standard deviation of the same. Modern domain detection uses this as a strategy for detecting domains [66], so using it as a routing decision is a natural strategy. This strategy focused on using a pretrained ViT, copying it into experts, and then defining each expert prototype initial using means of tokens per domain with one image per class. In this case, there is only a single router to select which path to go through for the final 4 layers of the ViT structure and unique routing heads. Each expert is initialized as the average of the patches for a single domain using one image per class.

The architecture of each variant is shown in Figure 3.1.

The routing is done at the image level instead of the token level to farther encourage domain experts instead of classifier feature experts.

3.2.4 Additional auxiliary loss

When desiring to enforce the latent space to better represent the domains, an auxiliary contrastive-style loss can be applied. By pushing away representations assigned to different experts while pulling together representations assigned to the same expert, the model is encouraged to form clearer domain boundaries in the latent space. This is only used with variant 1.

The total auxiliary routing loss is defined as

$$\mathcal{L} = \lambda_{\text{fast}} L_{\text{fast}} + \lambda_{\text{slow}} L_{\text{slow}} + \lambda_{\text{align}} L_{\text{align}} + \lambda_{\text{cons}} + L_{\text{cons}} + \lambda_{\text{lb}}(t) L_{\text{balance}}.$$

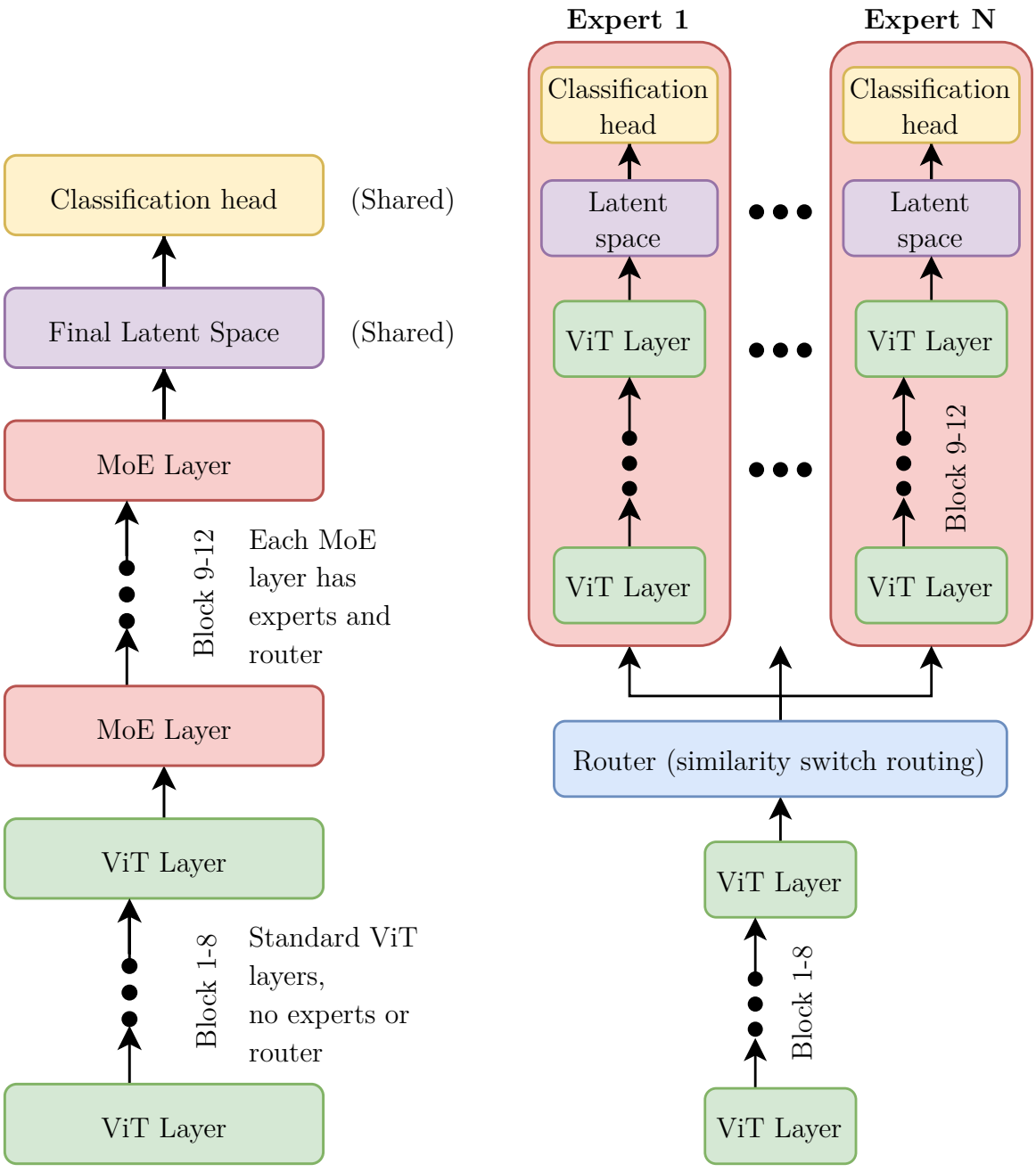
The annealed load-balancing weight is defined as

$$\lambda_{\text{lb}}(t) = \lambda_{\text{lb,init}} \cdot \max\left(0, ; 1 - \frac{t}{T_{\text{anneal}}}\right),$$

where t is the current global optimization step and T_{anneal} is the total number of annealing steps.

The alignment loss encourages image representations to remain close to the centroid of their assigned expert:

$$L_{\text{align}} = \frac{1}{M} \sum_{i=1}^{M} \left(1 - \frac{\mathbf{z} * i}{|\mathbf{z} * i|} \cdot \frac{\mathbf{c}^{\text{slow}} * a(i)}{|\mathbf{c}^{\text{slow}} * a(i)|}\right).$$



(a) Variant 1 architecture **(b)** Variant 2 architecture

Figure 3.1: Architectural differences between the two proposed variants. Refer back to Figures 2.1 and 2.2 for what is inside each ViT or MoE block. The router differences are not shown except for where the router itself resides.

This improves cluster cohesion by pulling samples toward the semantic region represented by their assigned expert centroid.

The consistency loss constrains the fast and slow centroids of each expert to remain aligned:

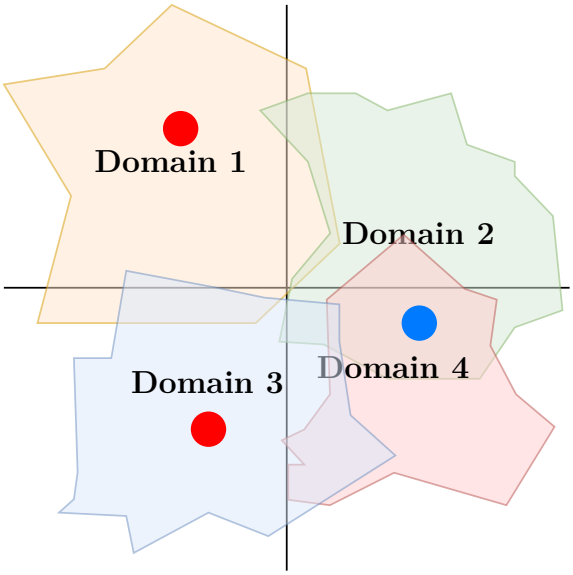


Figure 3.2: Example of the desired behavior of the expert centroids in the latent space. The red dots show 2 experts with domain specialization and the blue dot shows an expert that has clustered two similar domains as its specialization since there are fewer experts than distinct domains. This does not show the variants' second centroid, either a faster decaying centroid or a standard deviation centroid, respectively from variant 1 and 2.

$$L_{\text{cons}} = \frac{1}{N}\sum_{j=1}^{N}\left(1 - \frac{\mathbf{c}^{\text{fast}} * j}{|\mathbf{c}^{\text{fast}} * j|} \cdot \frac{\mathbf{c}^{\text{slow}} * j}{|\mathbf{c}^{\text{slow}} * j|}\right).$$

This stabilizes centroid dynamics by preventing rapidly updated centroids from drifting too far away from their long-term estimates.

The repulsion losses encourage expert centroids to specialize into distinct regions of the latent space.

For the fast centroids,

$$S^{\text{fast}} = \hat{C}^{\text{fast}}(\hat{C}^{\text{fast}})^{\top},$$

and

$$L_{\text{fast}} = \frac{1}{Nk}\sum_{e=1}^{N}\sum_{j\in\text{TopK}(e)} \max\left(S^{\text{fast}}_{e,j} - m, ; 0\right).$$

Similarly, for the slow centroids,

$$S^{\text{slow}} = \hat{C}^{\text{slow}}(\hat{C}^{\text{slow}})^{\top},$$

and

$$L_{\text{slow}} = \frac{1}{Nk}\sum_{e=1}^{N}\sum_{j\in\text{TopK}(e)} \max\left(S^{\text{slow}}_{e,j} - m, ; 0\right).$$

These losses penalize highly similar centroids, encouraging experts to occupy different semantic regions and reducing expert collapse.

The load-balancing loss encourages more uniform expert utilization:

$$L_{\text{balance}} = \sum_{e=1}^{N} p_e \log\left(\frac{p_e}{1/N}\right),$$

where p_e denotes the fraction of samples assigned to expert e within the current batch.

This term discourages routing collapse where only a small subset of experts receives assignments. This term was usually set to 0, but in the standard MoE baseline was the only auxiliary loss applied.

The variables are defined as follows:

- M: number of assigned samples in the current batch.
- N: number of experts.
- $a(i)$: mapping from sample i to its assigned expert index.
- m: centroid repulsion margin.
- TopK(e): the k most similar centroids to expert e, excluding itself.
- $\hat{C}$: row-wise normalized centroid matrix used for cosine similarity computations.

3.2.5 Experimentation

The strategies above were tested against the baseline models of a vision transformer and standard MoE with balancing loss to compare the new methods' efficacy in preventing catastrophic forgetting. The primary dataset used in the investigation for these models was Office-Home since it would have a strong domain representation. Both pretrained and randomly initialized models were tested to compare the difference in training stability and overall effectiveness of the strategy when the latent space had strong organization to start. Models with frozen weights in the earlier structures were also tested to show effectiveness when the latent space was highly stable.

The experiments in this thesis were of an iterative nature and used to inform the next tests performed. Initial investigations using CIFAR-10 used heavy class imbalanced partitions to understand whether MoE had an effect in that kind of system. Later tests used the information from CIFAR-10 and early CoRE-50 tests to design variant 1's algorithm and to explore domain incremental learning performance. Variant 1's performance on tests done on CoRE-50 was then used to design variant 2. Office-home tests were done to understand the differences in behavior of an MoE with different routers under heavy domain shift.

Experiments that are presented in the results chapter are those that were found to be most informative of the large variety of experiments performed. These were selected to motivate the design choices made in the variants, and to best explain the answers to the guiding research questions.

Dataset	Experiment (short name)	Experiment setting (brief)
CIFAR-10	Class imbalance: CIFAR-10	5 partitions with class imbalance; no scheduler; weighted cross-entropy used; no pre-training.
CoRE-50	Long with baselines	Long runs (10 epochs per domain) without pre-training; five model types compared (MoE variants, ViT, ConvNeXt).
CoRE-50	Mini with reduced MLP ratio	Mini ViT and MoE with reduced MLP ratio (MoE MLPs = 1/4) to match total params; variant 1 vs standard MoE.
CoRE-50	Short pre-train with variant 1	Short pre-training on subset removed from domain partitions; compare variant 1, standard MoE, and ViT to study early–mid continual learning.
Office-Home	Long with variant 1	Long training without pre-training on ordered domains (art, clip art, product, real); compare variant 1 and baselines.
Office-Home	Model transfer with both variants	Transfer pretrained ViT weights into MoE variants 1 & 2; some early layers frozen for variant 2; evaluate post-transfer behavior.

Table 3.1: Explanations of the experiments used for the results section.

4
Results

The following is divided by dataset and experiment. First, class-imbalance with the CIFAR-10 dataset will be presented. The easy domain shifts of CoRE-50 with MoE variant 1 will then be shown. Then the hard domain shift case of Office-Home will be shown.

Experiments without pre-train show early continual learning training dynamics while those with pre-train are split into two types. The majority are a subset pre-train that are removed from the domain partitions to allow for general representations to begin to be built in the latent space. This best represents early-mid continual learning dynamics. The other is a model transfer technique to take a pre-trained ViT and create an MoE version of it by copying weights into individual experts. The pre-trained model is trained on ImageNet [5] to give strong semantic meaning within the latent spaces. This best represents late training dynamics.

There are a few experiments that tested mini models instead of small models to see if model capacity had a significant effect. Mini models had 5-10 million active parameters, whereas small models had 20-30 million active parameters. The mini model experiments also used MoE where they are approximately the same number of total parameters. Small model experiments used MoE with the same number of active parameters as compared to the dense model.

Notes for understanding the graphs:

- A reminder that general trends and domain-by-domain changes are more important than final values for continual learning.
- Vertical dotted lines on graphs are the partition splits, so large changes in values are to be expected at these points.
- The modified methods for calculating continual learning values by epoch are used except for forward transfer.
- "EMA router" model label is variant 1 in the graphs. "Prototype router" is variant 2.

4.1 Class imbalance: CIFAR-10

CIFAR-10 is a small dataset with few patches for tokens to be generated from it. When performing with class imbalance, the weighted cross entropy appeared to mostly handle the issue. The greater accuracy of ConvNeXt as in Figure 4.1 shows that the inductive bias of convolutional networks wins out in small, low resolution datasets. MoE does slightly help a transformer architecture, but does not make up for the inherent need of a large amount of data. The forgetting is also only slightly

helped as in Figure 4.2. This is most likely due to the dataset not being large enough to warrant the heavy MoE structure.

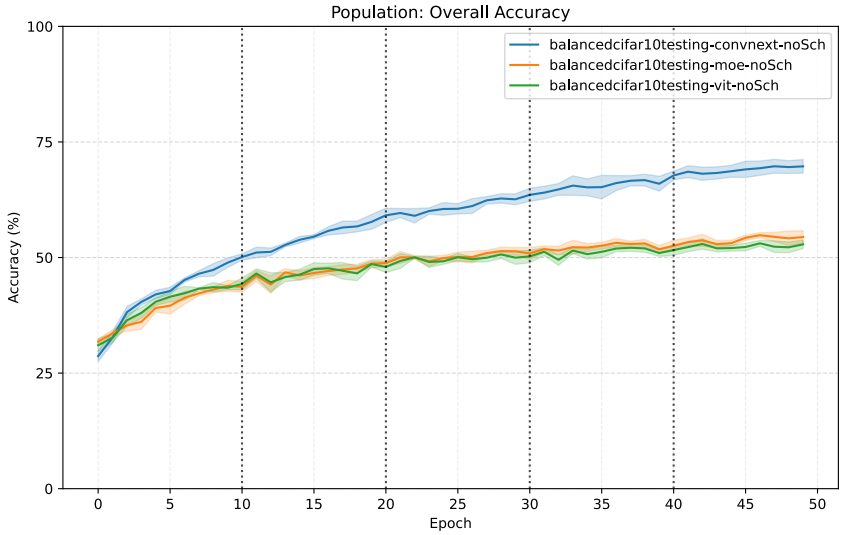


Figure 4.1: Overall accuracy scores for CIFAR-10 testing with 5 partitions with class imbalance. Shadow is the standard deviation over 5 separate runs. No scheduler was used in this instance.

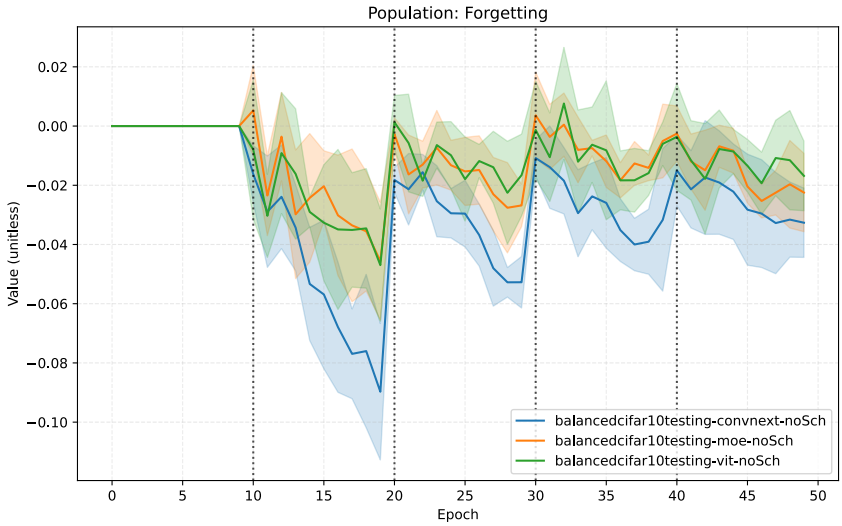


Figure 4.2: Forgetting measure of the models in CIFAR-10 testing. First interval cannot perform forgetting since there is no prior information to forget.

4.2 DIL, easy domain shift: CoRE-50

All CoRE-50 tests were done with the same order of domain to match with other research using this dataset. There are 11 domains demarcated by vertical dotted lines in the results graphs.

4.2.1 Long with baselines

Five models are used. MoE_ensemble_LR_scale is a test that was done as a simple test for a different architecture that was shortly explored during the thesis experimentation where there are 4 experts used as an ensemble, each with a different learning rate. MoE-many is an 8 expert, top-2 MoE method to test the effect of additional experts. MoE has 4 experts, top-1 routing. ViT is identical to MoE but is a simple dense model. ConvNeXt is the baseline alternative architecture to compare against.

An initial long testing where each domain had 10 epochs, no pre-training for the models, and variants are not tested was performed. This level of complexity in the dataset showed that vision transformer architectures tend to perform better than convolutional architectures, as can be seen in Figure 4.3. Interestingly, ConvNeXt did perform slightly better when considering class balanced metrics, however it did not retain old information nor reuse the information gained in prior domains as well as the ViT architectures as shown in Figure 4.5 and 4.6. MoE architectures performed only slightly better than ViT in the tasks, which shows poor parameter efficiency in this case.

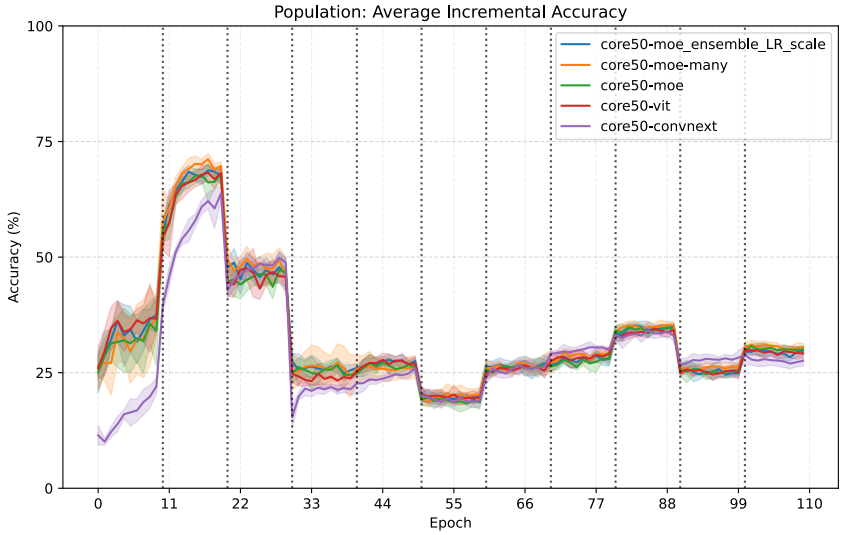


Figure 4.3: Average incremental accuracy on CoRE-50. Heavy overlap of performance when random initialization is used for the models, however ConvNeXt tends to lag slightly compared to ViT structures. ViT has similar accuracy to the MoE models.

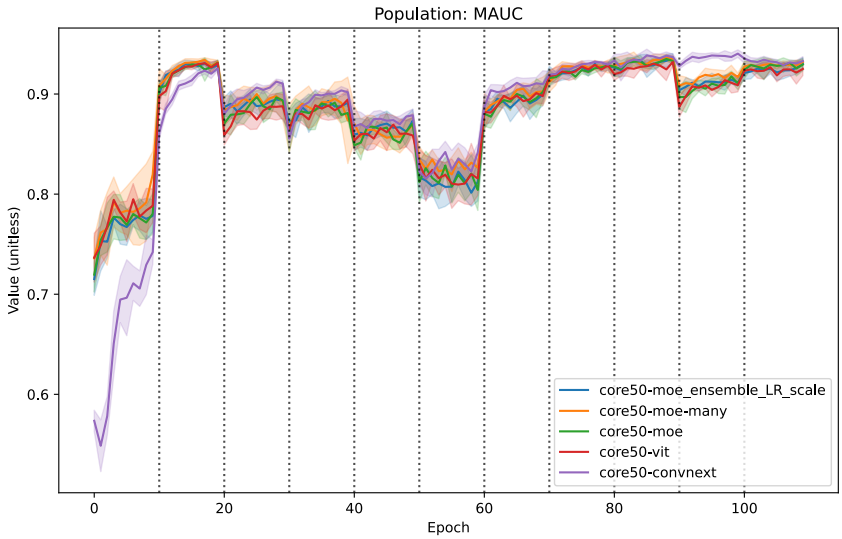


Figure 4.4: CoRE-50 MAUC score. With less class imbalance, it is shown that MoE with top-2 routing and double the experts of the standard MoE performs best of the ViT architectures.

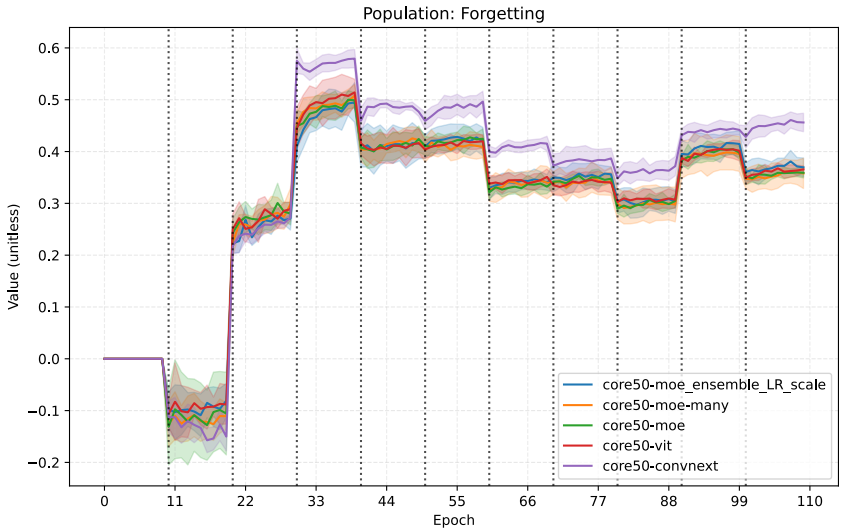


Figure 4.5: CoRE-50 forgetting measure without pre-training. The ViT has similar forgetting to the MoE structures, while ConvNeXt consistently performs worse on old partitions. An initial negative forgetting is seen only in the second partition for all models.

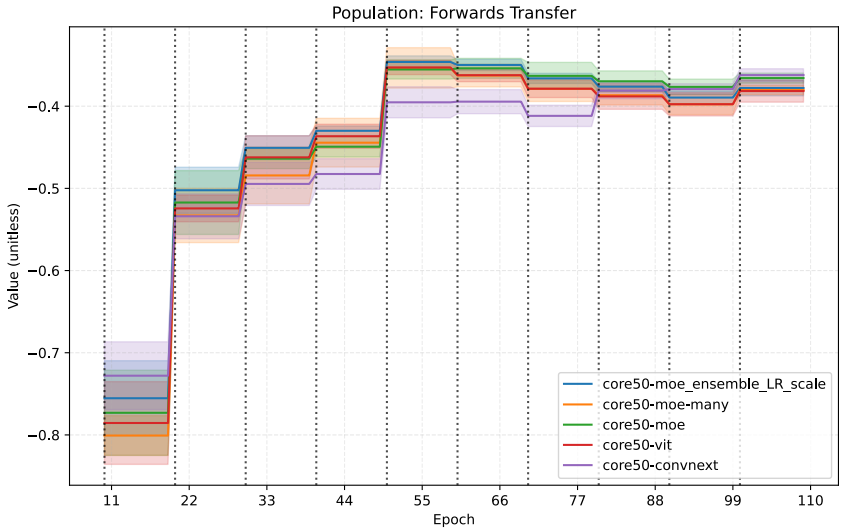


Figure 4.6: CoRE-50 forward transfer without pre-training. No positive forwards transfer is seen in any model, with ViT having the worse FWT and the standard MoE having the best as learning stabilizes.

4.2.2 Mini with reduced MLP ratio

This experiment was done in order to test the limits of MoE when model capacity is less than optimal and to present the power of sparse activation. Mini versions of ViT are used and variant 1 is tested against the standard MoE with 4 experts and switch routing. The MoE in this case have MLP ratios that are one-quarter of the ViT in order to have similar numbers of total parameters. This means at run-time the MoE models have fewer parameters than the ViT.

Results show that variant 1 outperforms both the standard MoE and ViT in A_IA until the tenth domain as shown in Figure 4.7. However, it is not due to better retention after the second domain as in Figure 4.8. Surprisingly, it is adapting faster, which is unusual for MoE. This is probably because of the fast specialization of the experts that variant 1's router allows alongside the reduced MLP ratio leading to more aggressive learning per expert. However, the structural reuse of underlying patterns in the data is done very well until the tenth domain, where significant rewriting of it's parameters occur, as in Figure 4.9.

A comparison with a pre-trained MoE is shown in Figure 4.10 to show how the EMA router's specialization per domain is with the MoE that was tested alongside it and an MoE that has pre-training. The MoE with pre-training is not shown with other graphs as it is not a fair comparison except as to understand the expert usage. This shows that the EMA router performs better domain specialization than even a pre-trained MoE and still avoids expert collapse.

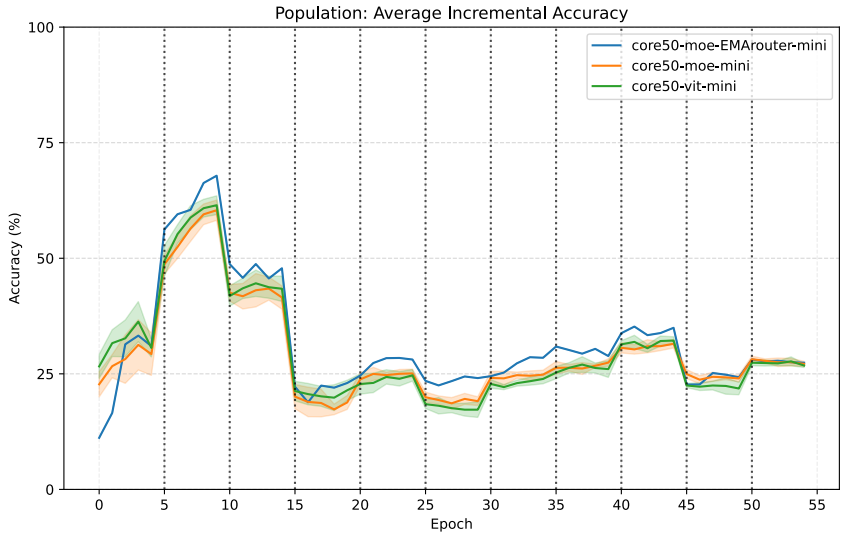


Figure 4.7: When there are fewer partitions, the EMA router consistently outperforms the standard MoE and ViT. However at the tenth partition, this behavior is lost.

Figure 4.8: CoRE-50 forgetting measure with mini MoE. The EMA router forgets the least until a critical threshold of domains is reached, where forgetting becomes significantly worse than the other models.

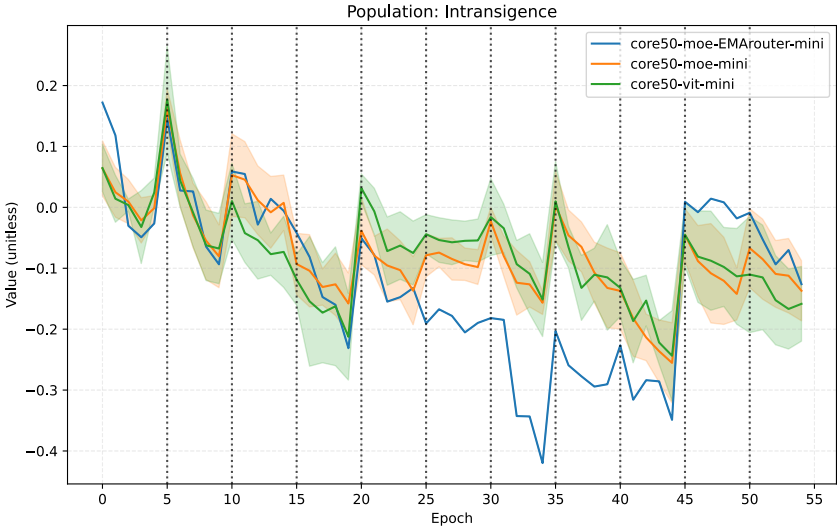
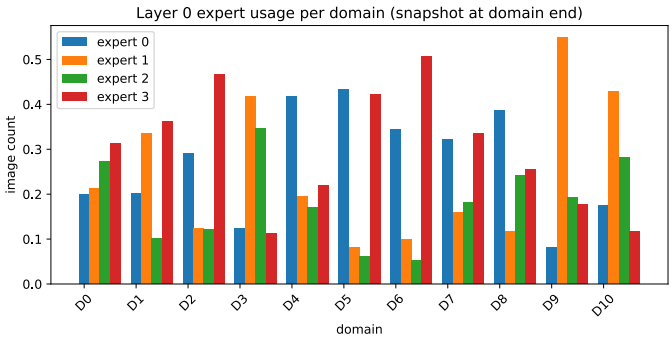
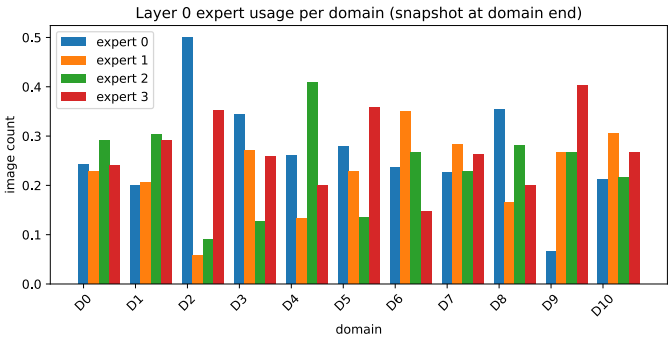


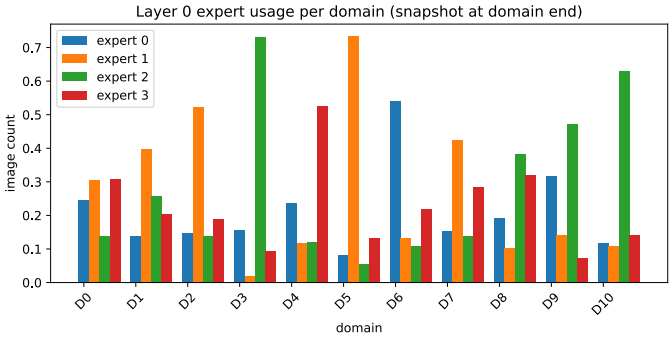
Figure 4.9: Intransigence measure for CoRE-50 with mini MoE. The EMA router effectively reuses structures until the tenth domain is reached, where overwriting occurs.



(a) Standard MoE



(b) Standard MoE with pre-training



(c) Variant 1: EMA routed MoE with loss and repulsion

Figure 4.10: Expert utilization fractions per domain at the first MoE layer (block 8 in depth). The pre-trained case shows the most balanced expert usage, while no pre-train leads to faster expert specialization. The EMA router shows rapid and clear specialization quickly, primarily divided by domain.

4.2.3 Short pre-train with EMA variant

The short pre-train is done in order to compare performance of the routers in early-mid continual learning training dynamics. The pre-train allows for the latent spaces to be better organized prior to having a variety of domains given that would normally require restructuring of the latent space organization if the domain has samples that are severely out of distribution. However the results show in Figure 4.11 that the pre-training does not effectively simulate this. The geometric mean dropping to zero implies that there is at least one class that has not been classified correctly whatsoever. A complete collapse of the classes for multiple domains after pre-training is shown in all models, but the least in a standard MoE. The effectiveness of the MoE in recovering compared to the other two models shows that class or feature-based MoE experts handle larger class representation distributions better, at least in the early-mid training. Variant 1 shows the least negative effect on old knowledge based on Figure 4.12 and ties with the standard MoE for having new representations learn well with the additional domains prior as in Figure 4.13. The overall performance of variant 1 with a reduced MLP shows a strong ability for enabling better generalization ability when a somewhat structured latent space exists.

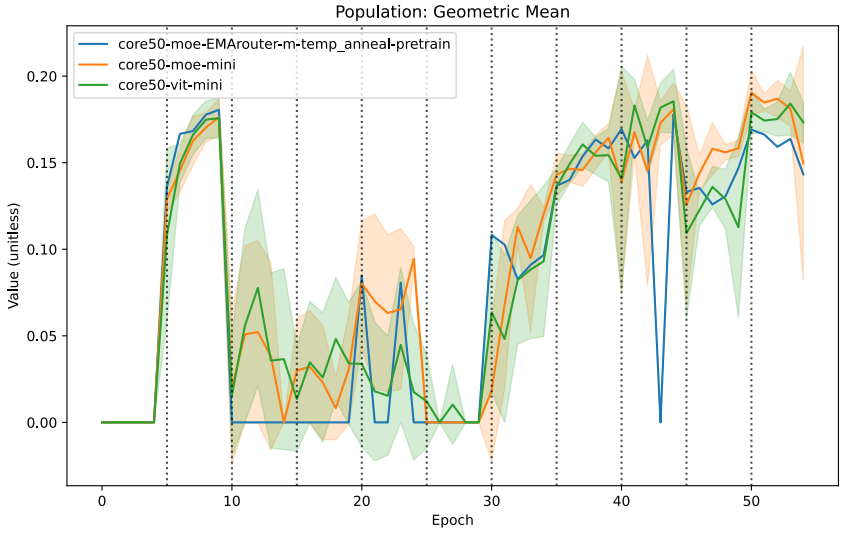


Figure 4.11: Geometric mean on CoRE-50 with a short pre-train for the models. Shows that the pre-train does not help in stabilizing equal improvement by class. The sharp downward spikes are due to an inability to perform at all in at least one class. The standard MoE tends to best hold consistency between classes

In the short pre-train test of CoRE-50, it can be seen in Figure 4.11 that a standard MoE performs most consistently among classes. This follows the design since experts will become class group experts, so class-by-class accuracy should degrade the least. In the EMA router case, some experts are starting nearly new so they do not know semantic structures of the classes yet.

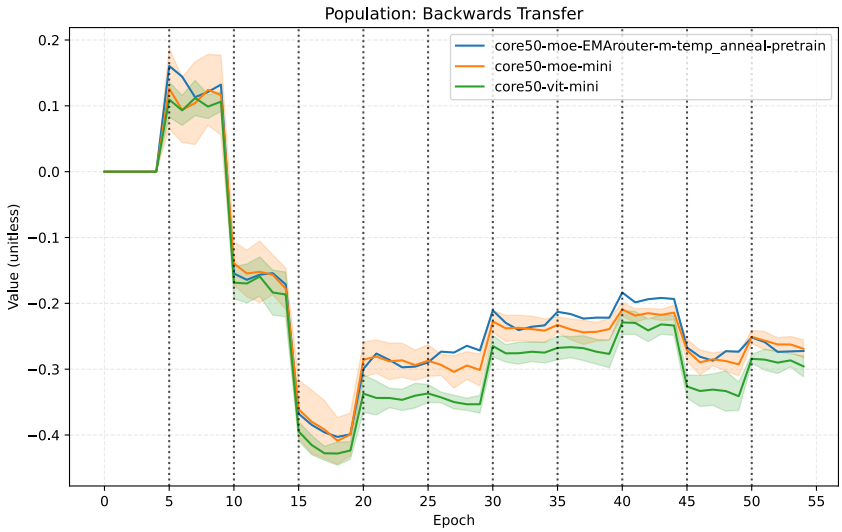


Figure 4.12: CORE-50 pre-trained setting backwards transfer. The EMA router and MoE show better performance in reducing the degradation of old tasks as more domains are learned.

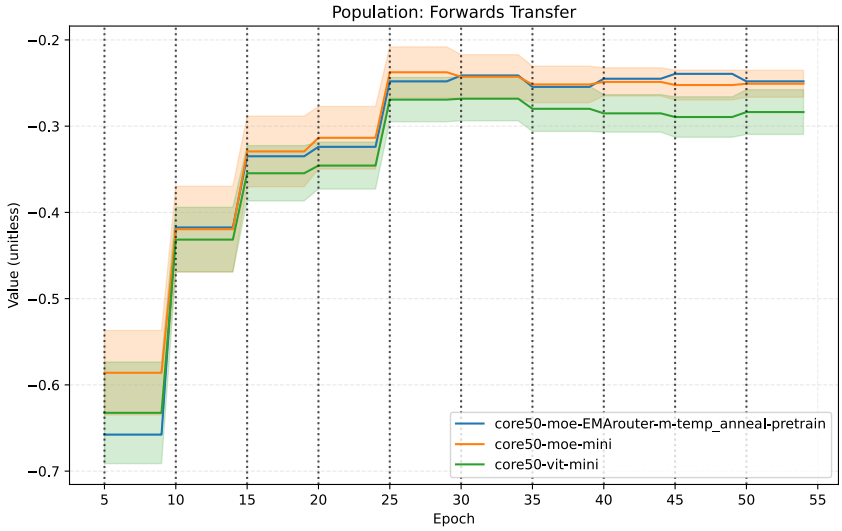


Figure 4.13: CORE-50 pre-trained setting forwards transfer. The prior domains hurt the training of the model on newer domains less for MoE models than the dense ViT.

4.3 DIL, hard domain shift: Office-Home

Office-Home experiments all use the same domain order to follow the standard practice of the dataset. The order is art, clip art, product images, and then real-life images. The "EMArouter" is variant 1 and the "prototyperouter" is variant 2 in the experiments.

4.3.1 Long with EMA variant

Long tests were initially done to better understand the behavior on Office-Home when significant training time was given without any pre-trained weights or pre-training phase. The dense model shows strong performance as the long training allows for the training sample dividing effect of MoE to become a driving factor in overall accuracy as in Figure 4.14. Variant 1 and the standard MoE show similar accuracy and show better stability in old representations than the ViT as shown in Figure 4.15. In Figure 4.16, variant 1 outperforms a standard MoE in forwards transfer, but does not outperform the ViT. The problem of latent space organization time is apparent in this early continual learning simulation experiment, leading the ViT to perform better over the MoE's.

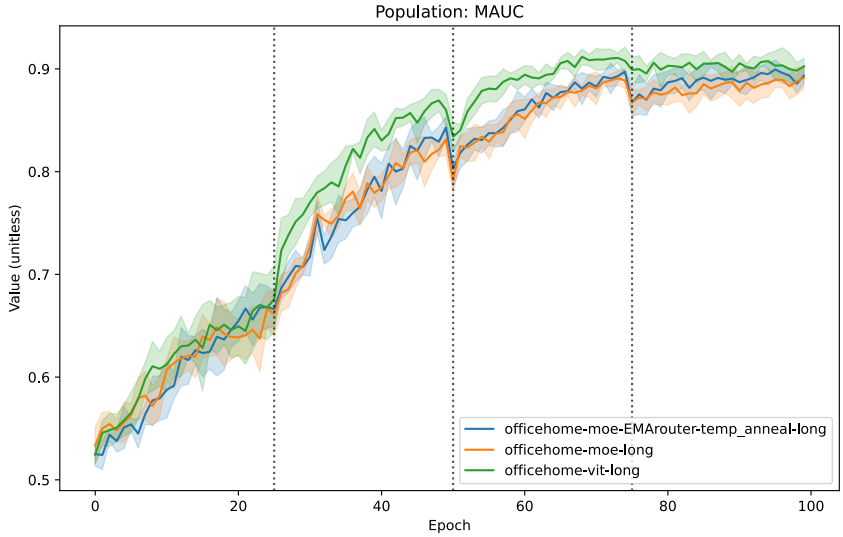


Figure 4.14: When long training time is given for Office-Home without pre-training, the dense variant does better in class balanced performance and MoE variant 1 is slightly better than the standard MoE

Figure 4.15: Forgetting measure for long training time on Office-Home dataset. The MoE models forget significantly less than the dense variant as more domains are presented.

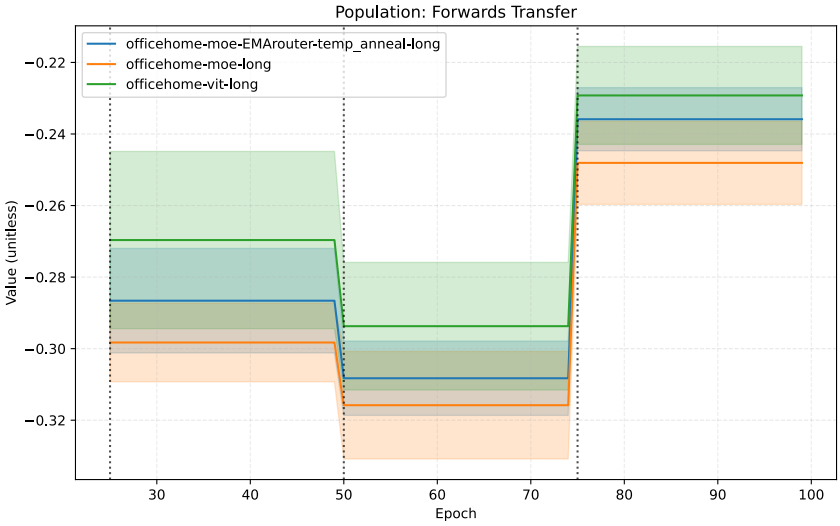


Figure 4.16: Forwards transfer on Office-Home with no pre-training. The ViT re-uses the representations best compared to the MoE for learning new domains.

4.3.2 Model transfer with EMA variants

In the model transfer testing, variant 2 had been created based on the results of the prior experimentation. This uses transfer of weights from a pretrained ViT to the MoE variants prior to training. Variant 1 is named "EMArouter" and variant 2 is named "prototyperouter" in the resultant graphs. The strong latent space representations inherited allow showing how the routers can perform once good stability is reached within the latent space. Variant 2 had the early layers frozen since the later layers use the representations for a single routing selection to ensure strong routing stability.

The results showed very different behavior between all model variants other than in balanced accuracy as shown in Figure 4.17. Variant 1 manages to keep similar accuracy to the ViT and standard MoE, however variant 2 takes longer to close the gap in accuracy. Interestingly, the standard MoE, ViT, and variant 1 all have reductions in geometric mean at later domains, but variant 2 continues to have positive change throughout the entire training period.

In Figure 4.18 variant 2 shows strong reduction in forgetting, even having negative forgetting in domain 2. This shows that the domain based routing with clustering like domains can have a good effect, even though not all experts were properly used by variant 2 as shown in Figure 4.21. It can also be seen that the balanced standard MoE usage can work well for good accuracy, but not for continuing to store old information as effectively as domain routing when the domain changes themselves are strong. Variant 1 showed some domain based routing, but not as effectively as variant 2. But given that variant 1 performed similarly to the normal MoE and ViT while having better backwards transfer as in Figure 4.20 is it's own benefit. Variant 2 had poor forward transfer, which was also apparent due to the time it took to catch up in geometric mean. This is hopefully due to creating better generalized structures.

Both variants had better backwards transfer in Figure 4.19 than the standard ViT and MoE, with variant 2 managing to have positive transfer during the second domain. This means that the centroid routing in both variants has improved the ability of the model to harm the representations of the old domains less than baselines.

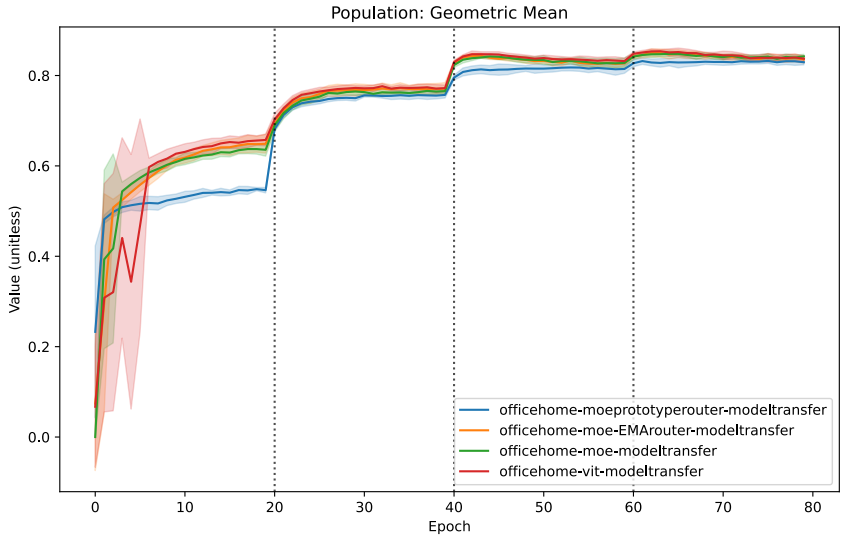


Figure 4.17: Office-Home geometric mean using a model-transfer pretrained backbone. The ViT performs best but also takes the longest to stabilize. All models but variant 2 have drops in G-mean, which continues to improve. Initial training stability is reverse of comparative final overall performance. Variant 1 performs similarly to a standard MoE and dense ViT.

Figure 4.18: Office-Home forgetting measure using a model-transfer pre-trained backbone. The variants have the least forgetting initially. Variant 1 begins overwriting as domain 3 enters. Variant 2 shows consistently the least loss of performance on old domains.

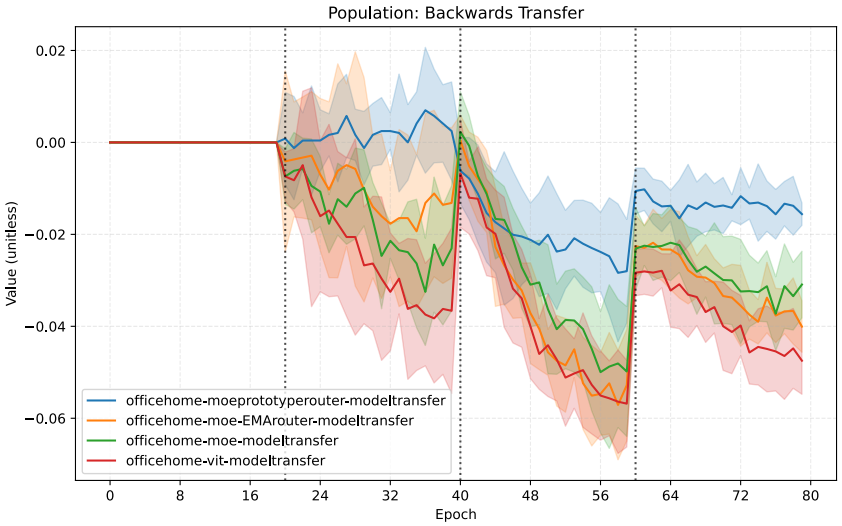


Figure 4.19: Office-Home backwards transfer using a model-transfer pre-trained backbone. New learning harms the old domains the least for variant 2. Variant 1 performs well until domain 3 is reached. Dense ViT consistently harms old representations the most.

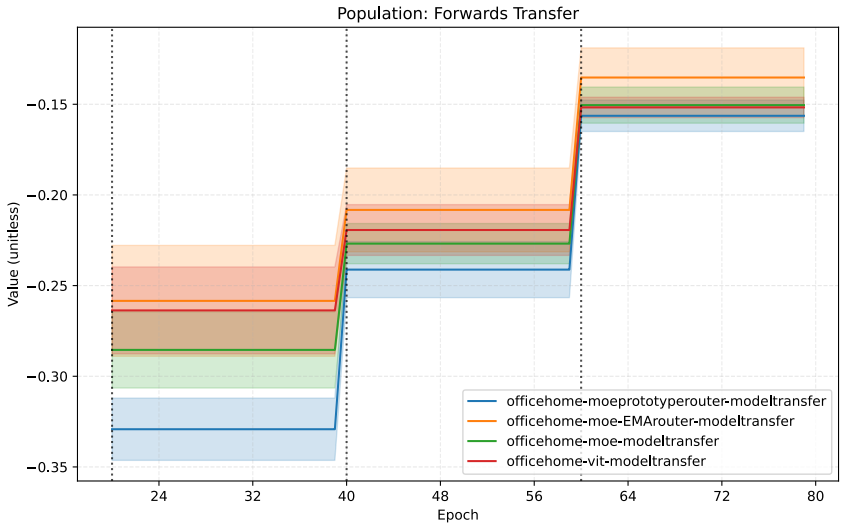
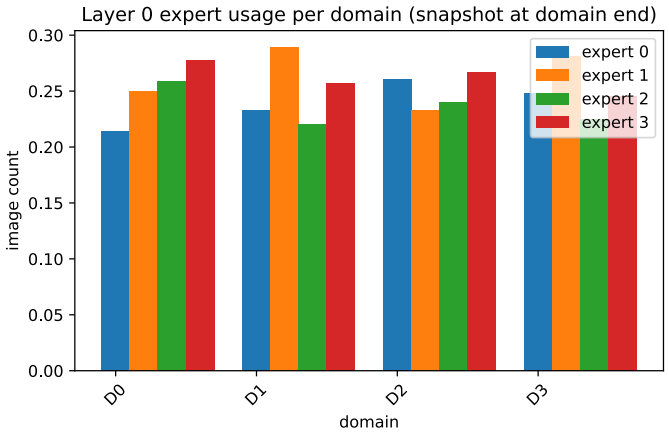
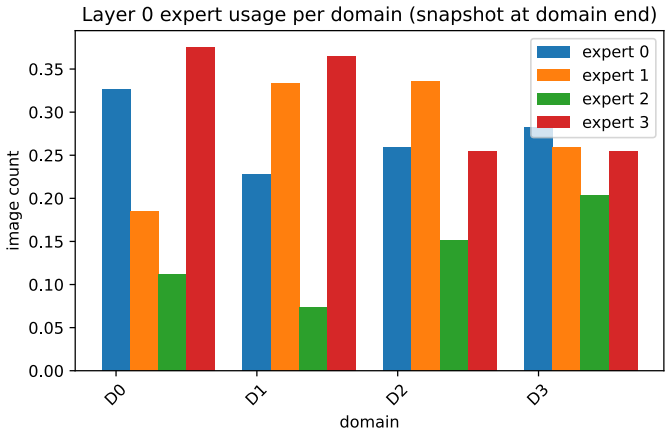


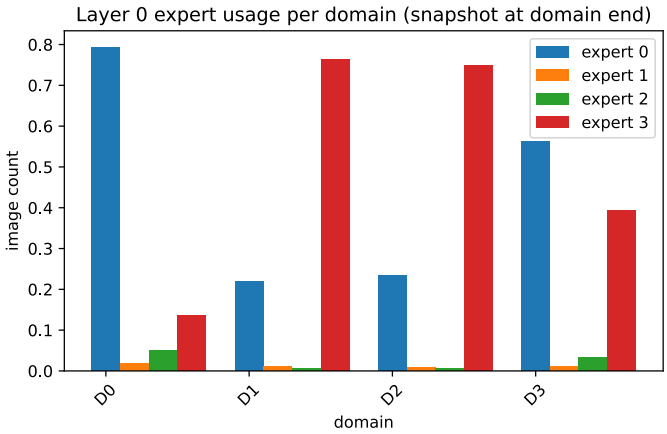
Figure 4.20: Office-Home forwards transfer using a model-transfer pre-trained backbone. Variant 2 has the worst utilization of old domains for new domain learning. Variant 1 reuses old domains the best.



(a) Standard MoE



(b) Variant 1: EMA routed MoE with loss and repulsion



(c) Variant 2: EMA routed MoE with mean and standard deviation, full split of structure

Figure 4.21: Expert utilization fractions per domain at the first MoE layer (block 8 in depth). The imbalance is more severe as the split in the latent space is more stable. However, semantic stability is better in this setup. D0 is art, D1 is clipart, D2 is product, and D3 is real-life object. Variant 1 shows some specialization by domain. Variant 2 shows strong domain specialization with some experts being heavily underutilized.

4.4 Additional results, data and code availability

Additional results of other experiments and plots for the above experiments are available in the Appendix C. The code and setups used can be found in the public GitHub repository [63]. The exact experimental data used can be requested from the writer of this report.

5

Discussion

5.1 Comparison of model behaviors in continual learning

In the setting of static compute memory and no replay storage, it is shown that MoE under-perform or are about equal against dense baselines in CL. The positives of an MoE in standard learning is that it has the ability to split up a harder problem into smaller problems and to segment the knowledge. The impact of this is that each expert gets a fraction of the samples and interacts with the updates of the rest of the model significantly less. In Figure 4.1, it is shown that MoE fails to match the ViT baseline and ConvNeXt in CIFAR-10. The lack of inductive bias in ViT leads to worse results against the convolutional network that does in this low resolution, small dataset. ViT require large, complex datasets before they are able to outperform convolutional networks in a fair training time setting. MoE networks then split up this data-hungry structure farther, making the training slower than before. This stacking effect leads to a lacking performance. Even in forgetting, the ViT and ViT-MoE perform similarly and worse than ConvNeXt as in Figure 4.2. The common use-case of MoE in extremely large models with I.I.D. datasets and with additional training time show that forgetting is a concern only with heavier complexity. For simple datasets, even with heavy class imbalance, MoE do not show improvements over baseline.

When the datasets are complex the effectiveness of ViT structures becomes apparent as can be seen in Figures 4.3 and 4.5. With complexity comes a greater propensity for forgetting. However the complexity in CoRE-50 is shown to not be great enough for model partitioning to have any significant effect. This may be due to the object representations that the model is trained to detect do not change enough, thus domains are not dominant or discernible in the latent space structure. Once in the Office-Home dataset, the effect of the domains causing forgetting even in a ViT structure becomes evident. In Figure 4.15, the ViT begins to forget more quickly than the MoE variants. This reveals that the domains must force high representational variance of a class before the additional capacity becomes useful.

In Figures 4.12 and 4.13, an interesting interaction is notable. Both the FWT and BWT of the MoE is better than that of the ViT. This means that the data altogether is being leveraged better than in a standard ViT. So the MoE is providing better stability but less plasticity. Since a small pre-training was given in these cases, it shows that long-term data storage is more effective in an MoE with less ability to quickly adapt to new data. In the case when no pre-training was given, as in Figures

4.5 and 4.6, there is not a significant difference in ViT and ViTMoE performance. This may be due to the lack of a well-defined latent space leading to worse expert specialization.

The overwriting of the experts can be seen in the intransigence measure in Figure 4.9. The less overwrite that occurs, the less that there will be a large intransigence jump. In the case of the centroid routing, there is a long period where there is far less overwriting than in both the ViT and standard ViTMoE builds. The router will be discussed further in Section 5.3, but it should be mentioned since it shows that there can be better use of the space in an MoE, but it is not stable or effective.

General MoE behavior in continual learning shows that when there is not availability of data replay to allow for clustering of domains, the MoE holds a similar accuracy with more stability but less plasticity. ViT are not helpful for simple datasets and the complexity must be even greater to make MoE worthwhile. Behavior when using router balancing acts like a distributed dense model. pre-training does have a better effect for MoE than a dense model since specialization can develop sooner from latent space stability.

5.2 Inherent MoE advantages

MoE benefits include subtle improvements due to the balancing that continual learning necessitates between plasticity and stability. Promise is shown in reducing forgetting and lower runtime operations.

Forgetting has been discussed in the prior section and weighed against its drawbacks. This improvement is given automatically by the use of a standard MoE in CL. Other router variants, discussed in the next section, can be used as an amplifier for this inherent effect.

In Section 4.2.2 the MoE were structured to have a similar total number of parameters, meaning that the number of parameters at runtime are reduced compared to the dense model. While the accuracy is similar, the runtime number of parameters per FFN is one-fourth the number of parameters and still less forgetting occurs. The number of parameters per expert is inversely proportional to the number of experts. The fact that the model still runs similarly shows that when the experts are trained with router balancing, it can perform similar to a dense model while having far less compute, aligning with the results shown in [45] in this setting. The advantage in reducing runtime computations is important for real-time systems and for resource constrained systems. Drawbacks as discussed in the prior section still apply, but this is still viable. Other research has found methods to modify dense models to be MoE with little additional training, which this research bolsters by showing that similar performance can be reached either in the case of no pre-training with CoRE-50 and corroborated with other experiments in Appendix C. While not a direct benefit of MoE in continual learning, it enables dense models to be altered into sparse models and make it faster at runtime.

When using a model transfer from a ViT with pre-trained weights, MoE were also shown to stabilize faster in training and to remember less represented classes better than the standard dense ViT as can be seen in Figures 4.11 and 4.17 in

population behavior.

5.3 Improving MoE in DIL

There were two variants of the centroid routing tested. The version with the fully unfrozen, untrained model that had routing decisions at each expert layer and had auxiliary contrastive loss applied to the model did alter expert utilization against a standard MoE in both DIL datasets as in Figures 4.10 and 4.21. Expert usage was less balanced and some domain clustering and specialization by domain was observed, especially in CoRE-50. Initial conditions of the prototypes appear to have a large effect in how well the experts themselves are utilized and how they specialize.

Improvement was seen when pre-trained backbones were utilized as in sections 4.2.3 and 4.3.2. The most likely cause of this is the use of the latent space prototypes. With an initially unstable latent space or restructuring of the latent space as new training data and domains are received, the router collapses to using only a subset of the experts or to not specialize in a meaningful way. If the latent space has little change, such as when the earlier model is frozen or the latent space is already structured, the prototype definition stays stable and thus allows specialization. Forgetting was reduced when using pre-trained models and this routing method. In the case of Office-Home, the forgetting reduced significantly as compared to the baseline when using the second variant where the model splits completely at the eighth layer depth. In Figure 4.21(c), the routing is shown to be distinctly peaked based on the domain. Only two experts did not collapse of the four, but less forgetting occurred and the semantic separation of the domains appeared to follow. With additional testing in good methodologies to initialize the centroids or to force utilization of the dead experts, the method shows that forgetting purely by alteration of routing algorithm better domain clustering and experts that gain expertise in a domain without domain label provided are possible. The issue of the domains being separable in different layers could explain the behavior seen, so methods to search for domain clusters within multiple latent layers may be necessary. Capacity measures with softer constraints could potentially improve the expert usage as well.

Accuracy with the second variant of the new router method tended to lag that of the standard models, but would close the gap with time as in Figure 4.17. The stability is greater, so the plasticity of the model does suffer. A result of greater separation of the model is that the FWT is worse since partitioned knowledge is not reused as in Figure 4.20. The first variant where the model is splitting via prototypes at each MoE layer showed behavior more alike the dense model, but actually was adapting to new domains the most quickly. The most likely reason is that the specialization was less by domain but allowed for better specialization for features than that of a standard MoE balancing method. The lower performance of the first variant on the CoRE50 dataset also shows that when the domain representation is not great enough to be encoded in the latent space, the router fails to have domain-aligned expertise emerge.

While time constraints limited testing of the new algorithms, more analysis is needed to identify which pieces of the algorithm yield improvements. Additional testing with how the centroids are initialized could reveal insight into the new spe-

cialization behavior. Parameter sweeps using the balance between mean and standard deviation, separation loss, contrastive loss, and learning rate could also assist in finding the limits for improvement of using experts to better align to domain. The additional loss elements improvement in model performance could be found using ablation tests. Larger datasets with more domains or testing with cyclical domain reuse could reveal how well the experts hold their domain specialization and domain clustering.

These novel routing methods presented show that exact routing methodology for MoE in CL is a significant driver in how stable or plastic an MoE is. While other papers recommend freezing the router for stable CL performance [64], this shows that this is unnecessary when routing signals do not come from the classification itself. These variants, especially the variant using model transfer and full later-depth model splitting, also show that in complex datasets, forgetting can be improved using this algorithm.

5.4 Existing CL methods viable for MoE

Throughout the experience of developing and experimenting with novel MoE routing methods, several best practices have become clear. The following recommendations can be used for more effective future MoE development.

A strong benefit of using MoE structures as a base is that the large majority of methods used in CL today can easily be used with MoE structures with little to no modification. The router can be separated from the general gradient flow for optimization methods, replay massively improves specialization and routing, regularization per expert can help retention of expert identity and data, and representation like model weight transfer or teacher distillation can massively improve stabilization of the model. The MoE structure also helps to improve representation methods, as shown in the better stability of the MoE models compared to baseline in Section 5.2.

Weight freezing strategies and MoE are well-matched, but this research shows that in resource constrained environments it is not necessary for stable training. The primary fix that allowed even a standard MoE model to properly stabilize was to have a reduced router learning rate if learned. When using switch routing with an implicitly learned router such as the new algorithms proposed in this research, router freezing is entirely unnecessary. Also for continual learning to be continual, freezing appears to be the antithesis of the final goal of a model that can continue to learn.

For the setting that this research focuses on, the primary recommendation would be to use generative feature replay on top of the routing algorithm proposed. Gradient clipping and strategies like AR1* [8] when applying the loss to the update would also be recommended to improve model retention of class representations, especially in settings where the number of experts is fewer than the number of domains. Finally, pre-training is essential. A well-organized latent space can improve training massively in both accuracy and convergence.

Given the benefits that the MoE structure give in the resource constrained setting it is a natural fit even if pure classification accuracy does not improve sig-

nificantly. In long term, massive dataset settings, the gains should only improve, as shown in the improvement of MoE accuracy with dataset difficulty.

5.5 Outlook of MoE in CL

Recent MoE research has been leaning towards geometric routing using latent space partitioning [52], [54], [56], [57]. This research also supports the direction, especially in the case of continual learning. Hierarchical MoE also show promise for defining experts by both domain and class clustering [22], [23], [47]. The use of larger datasets more realistic to CL as the topic gains interest will improve understanding of model advantages and behavior in CL. There are also strong methods to get the advantages of MoE after stable representation spaces have been learned by weight transfer of a dense model into a MoE structure. Adapter heads with MoE routing will also allow for more efficient adapter usage in CL, which is easy to implement when already using adapter heads.

6
Conclusion

In this paper, MoE models were compared against standard models in continual learning challenge and a new routing strategy was presented to better behave in DIL without domain labels. While large gains were not found in the results, this research helps to identify how to use MoE in this largely under-researched field and to explain the reasons why to use MoE and when MoE are unnecessary.

The results of this work shows that MoE performance in CL is dependent on router design and that MoE models have inherent advantages that are not directly accuracy improvements parameter efficiency, forgetting reduction, and explainability. Dataset complexity is essential in whether an MoE will have strong gains - when the dataset is too small or simple in representation breadth, MoE will struggle to match or improve on dense model raw accuracy performance. Future developments on this work would be the addition of larger datasets, parameter sweeps for algorithm optimization, prototype initialization and update improvements, and the exploration of cyclical domain presentation. This work adds to the recent growing field of geometric routing for mixture-of-experts models and presents that they are viable in continual learning given smart training strategies.

Bibliography

[1] A. Behrouz, M. Razaviyayn, P. Zhong, and V. Mirrokni, *Nested Learning: The Illusion of Deep Learning Architectures*, 2025. DOI: 10.48550/ARXIV.2512.24695.

[2] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, Mar. 1991, ISSN: 0899-7667. DOI: 10.1162/neco.1991.3.1.79.

[3] L. Zheng, Y. Yang, and Q. Tian, *SIFT Meets CNN: A Decade Survey of Instance Retrieval*, arXiv:1608.01807, May 2017. DOI: 10.48550/arXiv.1608.01807.

[4] D. Patterson et al., *Carbon Emissions and Large Neural Network Training*, 2021. DOI: 10.48550/ARXIV.2104.10350.

[5] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," en, *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y.

[6] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[7] V. Lomonaco and D. Maltoni, "CORe50: A New Dataset and Benchmark for Continuous Object Recognition," en, in *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, Oct. 2017, pp. 17–26.

[8] V. Lomonaco, D. Maltoni, and L. Pellegrini, *Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches*, 2019. DOI: 10.48550/ARXIV.1907.03799.

[9] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5018–5027.

[10] E. P. a. C. of the European Union, *Artificial Intelligence Act, article 6, annex 1*, Jun. 2024.

[11] E. P. a. C. of the European Union, *Artificial Intelligence Act, article 12*, Jun. 2024.

[12] A. Vaswani et al., *Attention Is All You Need*, arXiv:1706.03762, Aug. 2023. DOI: 10.48550/arXiv.1706.03762.

[13] A. Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929, Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[14] N. Shazeer et al., *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*, arXiv:1701.06538, Jan. 2017. DOI: 10.48550/arXiv.1701.06538.

[15] E. Garmash and C. Monz, "Ensemble Learning for Multi-Source Neural Machine Translation," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Y. Matsumoto and R. Prasad, Eds., Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1409–1418.

[16] J. Puigcerver, C. Riquelme, B. Mustafa, and N. Houlsby, *From Sparse to Soft Mixtures of Experts*, arXiv:2308.00951, May 2024. DOI: 10.48550/arXiv.2308.00951.

[17] W. Fedus, B. Zoph, and N. Shazeer, *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*, arXiv:2101.03961, Jun. 2022. DOI: 10.48550/arXiv.2101.03961.

[18] Y. Zhou et al., *Mixture-of-Experts with Expert Choice Routing*, arXiv:2202.09368, Oct. 2022. DOI: 10.48550/arXiv.2202.09368.

[19] D. Dai et al., *DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models*, arXiv:2401.06066, Jan. 2024. DOI: 10.48550/arXiv.2401.06066.

[20] M. Collier, E. Kokiopoulou, A. Gesmundo, and J. Berent, *Routing Networks with Co-training for Continual Learning*, 2020. DOI: 10.48550/ARXIV.2009.04381.

[21] D. Dai et al., *StableMoE: Stable Routing Strategy for Mixture of Experts*, arXiv:2204.08396, Apr. 2022. DOI: 10.48550/arXiv.2204.08396.

[22] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 2, Oct. 1993, 1339–1344 vol.2. DOI: 10.1109/IJCNN.1993.716791.

[23] W. Zhao, Y. Gao, S. A. Memon, B. Raj, and R. Singh, *Hierarchical Routing Mixture of Experts*, arXiv:1903.07756, Mar. 2019. DOI: 10.48550/arXiv.1903.07756.

[24] B. Liu et al., *Diversifying the Mixture-of-Experts Representation for Language Models with Orthogonal Optimizer*, 2023. DOI: 10.48550/ARXIV.2310.09762.

[25] B. Zoph et al., *ST-MoE: Designing Stable and Transferable Sparse Expert Models*, 2022. DOI: 10.48550/ARXIV.2202.08906.

[26] L. Wang, X. Zhang, H. Su, and J. Zhu, *A Comprehensive Survey of Continual Learning: Theory, Method and Application*, 2023. DOI: 10.48550/ARXIV.2302.00487.

[27] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, *Dark Experience for General Continual Learning: A Strong, Simple Baseline*, 2020. DOI: 10.48550/ARXIV.2004.07211.

[28] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, *iCaRL: Incremental Classifier and Representation Learning*, 2016. DOI: 10.48550/ARXIV.1611.07725.

[29] A. Prabhu, P. H. S. Torr, and P. K. Dokania, "GDumb: A Simple Approach that Questions Our Progress in Continual Learning," en, in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 524–540, ISBN: 9783030585365. DOI: 10.1007/978-3-030-58536-5_31.

[30] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent Replay for Real-Time Continual Learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 10 203–10 209, ISBN: 9781728162126. DOI: 10.1109/IROS45743.2020.9341460.

[31] H. Shin, J. K. Lee, J. Kim, and J. Kim, *Continual Learning with Deep Generative Replay*, 2017. DOI: 10.48550/ARXIV.1705.08690.

[32] X. Liu et al., *Generative Feature Replay For Class-Incremental Learning*, 2020. DOI: 10.48550/ARXIV.2004.09199.

[33] Z. Li and D. Hoiem, *Learning without Forgetting*, 2016. DOI: 10.48550/ARXIV.1606.09282.

[34] J. Kirkpatrick et al., *Overcoming catastrophic forgetting in neural networks*, arXiv:1612.00796, Jan. 2017. DOI: 10.48550/arXiv.1612.00796.

[35] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, *Memory Aware Synapses: Learning what (not) to forget*, arXiv:1711.09601, Oct. 2018. DOI: 10.48550/arXiv.1711.09601.

[36] F. Zenke, B. Poole, and S. Ganguli, *Continual Learning Through Synaptic Intelligence*, 2017. DOI: 10.48550/ARXIV.1703.04200.

[37] D. Maltoni and V. Lomonaco, *Continuous Learning in Single-Incremental-Task Scenarios*, 2018. DOI: 10.48550/ARXIV.1806.08568.

[38] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, arXiv:2002.05709, Jul. 2020. DOI: 10.48550/arXiv.2002.05709.

[39] N. Houlsby et al., *Parameter-Efficient Transfer Learning for NLP*, 2019. DOI: 10.48550/ARXIV.1902.00751.

[40] R. Aljundi, P. Chakravarty, and T. Tuytelaars, *Expert Gate: Lifelong Learning with a Network of Experts*, 2016. DOI: 10.48550/ARXIV.1611.06194.

[41] T. Veniat, L. Denoyer, and M. Ranzato, *Efficient Continual Learning with Modular Networks and Task-Driven Priors*, 2020. DOI: 10.48550/ARXIV.2012.12631.

[42] A. A. Rusu et al., *Progressive Neural Networks*, 2016. DOI: 10.48550/ARXIV.1606.04671.

[43] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou, *Overcoming catastrophic forgetting with hard attention to the task*, 2018. DOI: 10.48550/ARXIV.1801.01423.

[44] A. Mallya and S. Lazebnik, *PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning*, 2017. DOI: 10.48550/ARXIV.1711.05769.

[45] L. Wang, X. Zhang, Q. Li, J. Zhu, and Y. Zhong, "CoSCL: Cooperation of Small Continual Learners is Stronger than a Big One," 2022. DOI: 10.48550/ARXIV.2207.06543.

[46] S. Lee, J. Ha, D. Zhang, and G. Kim, *A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning*, 2020. DOI: 10.48550/ARXIV.2001.00689.

[47] H. Hihn and D. A. Braun, *Hierarchically Structured Task-Agnostic Continual Learning*, 2022. DOI: 10.48550/ARXIV.2211.07725.

[48] J. Zhou et al., *Separation and Collaboration: Two-Level Routing Grouped Mixture-of-Experts for Multi-Domain Continual Learning*, 2025. DOI: 10.48550/ARXIV.2508.07738.

[49] M. Le et al., *Mixture of Experts Meets Prompt-Based Continual Learning*, 2024. DOI: 10.48550/ARXIV.2405.14124.

[50] M. Le, B.-N. Dao, H. Nguyen, Q. Tran, A. Nguyen, and N. Ho, *One-Prompt Strikes Back: Sparse Mixture of Experts for Prompt-based Continual Learning*, 2025. DOI: 10.48550/ARXIV.2509.24483.

[51] A. Cheng et al., *EMoE: Eigenbasis-Guided Routing for Mixture-of-Experts*, 2026. DOI: 10.48550/ARXIV.2601.12137.

[52] I. F. Shihab, S. Akter, and A. Sharma, *Grassmannian Mixture-of-Experts: Concentration-Controlled Routing on Subspace Manifolds*, 2026. DOI: 10.48550/ARXIV.2602.17798.

[53] B. Lyu, S. Murakami, H. Kamigaito, and P. Zhang, *Routing by Analogy: kNN-Augmented Expert Assignment for Mixture-of-Experts*, 2026. DOI: 10.48550/ARXIV.2601.02144.

[54] J. Yang, *Latent Prototype Routing: Achieving Near-Perfect Load Balancing in Mixture-of-Experts*, 2025. DOI: 10.48550/ARXIV.2506.21328.

[55] S. Wang, M. Lu, N. Moshkov, J. C. Caicedo, and B. A. Plummer, *Anchoring to Exemplars for Training Mixture-of-Expert Cell Embeddings*, 2021. DOI: 10.48550/ARXIV.2112.03208.

[56] I. Ternovtsii and Y. Bilak, *Geometric Routing Enables Causal Expert Control in Mixture of Experts*, 2026. DOI: 10.48550/ARXIV.2604.14434.

[57] S. Ahrac, N. Hochwald, and M. Geva, *Routers Learn the Geometry of Their Experts: Geometric Coupling in Sparse Mixture-of-Experts*, 2026. DOI: 10.48550/ARXIV.2605.12476.

[58] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, *Don't forget, there is more than forgetting: New metrics for Continual Learning*, 2018. DOI: 10.48550/ARXIV.1810.13166.

[59] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence," 2018. DOI: 10.48550/ARXIV.1801.10112.

[60] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, "A survey on imbalanced learning: Latest research, applications and future directions," en, *Artificial Intelligence Review*, vol. 57, no. 6, p. 137, May 2024, ISSN: 1573-7462. DOI: 10.1007/s10462-024-10759-6.

[61] T.-M. H. Hsu, H. Qi, and M. Brown, *Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification*, arXiv:1909.06335, Sep. 2019. DOI: 10.48550/arXiv.1909.06335.

[62] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A ConvNet for the 2020s*, 2022. DOI: 10.48550/ARXIV.2201.03545.

[63] I. MacLeod, *Scaleoutsystems/continous-MoE*, original-date: 2026-01-27T10:43:24Z, May 20, 2026. Accessed: Jun. 5, 2026. [Online]. Available: https://github.com/scaleoutsystems/continous-MoE.

[64] H. Li, S. Lin, L. Duan, Y. Liang, and N. B. Shroff, *Theory on Mixture-of-Experts in Continual Learning*, 2024. DOI: 10.48550/ARXIV.2406.16437.

[65] E. Arani, F. Sarfraz, and B. Zonooz, *Learning Fast, Learning Slow: A General Continual Learning Method based on Complementary Learning System*, 2022. DOI: 10.48550/ARXIV.2201.12604.

[66] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, *Revisiting Batch Normalization For Practical Domain Adaptation*, 2016. DOI: 10.48550/ARXIV.1603.04779.

A

Appendix 1: Detailed Description of Evaluation Metrics

This appendix provides detailed descriptions of all evaluation metrics used throughout this project. The selected metrics originate from three complementary research domains:

- **Continual Learning (CL)** metrics, which evaluate knowledge retention, forgetting, transfer, and adaptation across sequentially learned domains.
- **Imbalanced Learning** metrics, which assess classification performance under class imbalance.
- **Mixture-of-Experts (MoE)** metrics, which quantify expert specialization and routing behavior.

Together, these metrics provide a comprehensive view of model performance, robustness, knowledge transfer, and expert utilization.

A.1 Performance Matrix (R-Matrix)

The performance matrix is the fundamental structure from which most continual learning metrics are derived.

$$R = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,D} \\ 0 & a_{2,2} & \cdots & a_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{D,D} \end{bmatrix} \tag{A.1}$$

where ($a_{i,j}$) denotes the performance on domain (j) after training has been completed on domain (i).

A.1.1 Variable Definitions

(D) Total number of domains or tasks.
($a_{i,j}$) Performance on domain (j) after learning domain (i).
($a_{d,d}$) Performance immediately after learning domain (d).
($a_{d,D}$) Final performance on domain (d) after all domains have been learned.

The diagonal elements represent the performance achieved immediately after training each domain. The final column represents the retained performance after completion of the entire continual learning sequence.

A.2 Average Incremental Accuracy (A_{IA})

Average Incremental Accuracy measures the average performance over all observed domains.

$$A_{IA} = \frac{1}{D} \sum_{d=1}^{D} a_d \tag{A.2}$$

where (a_d) denotes the evaluation score obtained on domain (d).

A.2.1 Variable Definitions

[(D)] Number of domains encountered.
[(a_d)] Accuracy or evaluation score on domain (d).

This metric summarizes overall continual learning performance in a single value. Higher values indicate better average performance across all domains.

A.3 Forgetting Measure (FM)

The Forgetting Measure quantifies the average loss of performance on previously learned domains after additional training.

$$FM = \frac{1}{D-1} \sum_{d=1}^{D-1} \left(\max_{l \in 1,\ldots,D} a_{d,l} a_{d,D} \right) \tag{A.3}$$

A.3.1 Variable Definitions

($a_{d,l}$) Performance on domain (d) after learning domain (l).
($\max_l a_{d,l}$) Best performance ever achieved on domain (d).
($a_{d,D}$) Final performance on domain (d).

For each domain, the metric computes the difference between the best historical performance and the final retained performance. An FM value of zero indicates perfect retention, while larger values indicate greater forgetting.

A.4 Backward Transfer (BWT)

Backward Transfer measures how learning later domains influences performance on previously learned domains.

$$BWT = \frac{1}{D-1} \sum_{d=1}^{D-1} (a_{d,D} a_{d,d}) \tag{A.4}$$

A.4.1 Variable Definitions

($a_{d,d}$) Performance immediately after learning domain (d).
($a_{d,D}$) Final performance on domain (d).

Positive BWT values indicate that learning subsequent domains improved earlier domains. Negative values indicate degradation of previous knowledge and are commonly associated with catastrophic forgetting.

A.5 Forward Transfer (FWT)

Forward Transfer evaluates whether previously learned knowledge improves performance on future unseen domains before training occurs on those domains.

$$FWT = \frac{1}{D-1} \sum_{d=2}^{D} (a_{d,d-1} a_d^*) \tag{A.5}$$

A.5.1 Variable Definitions

($a_{d,d-1}$) Performance on domain (d) before training domain (d).
(a_d^*) Baseline performance on domain (d) without prior continual learning.

Positive values indicate beneficial transfer of previously acquired knowledge, while negative values suggest that prior learning interferes with future tasks.

A.6 Intransigence Measure (IM)

The Intransigence Measure quantifies the inability of a continual learning system to effectively learn new domains due to constraints imposed by retaining previously acquired knowledge.

$$IM = a_D^* a_{D,D} \tag{A.6}$$

A.6.1 Variable Definitions

(a_D^*) Performance achieved by a reference model trained solely on domain (D).
($a_{D,D}$) Performance achieved on domain (D) within the continual learning framework.

Lower values indicate better adaptability to new domains. High intransigence suggests that mechanisms designed to preserve old knowledge may be restricting the acquisition of new knowledge.

A.7 Geometric Mean (G-Mean)

The Geometric Mean is commonly used in imbalanced classification settings to evaluate balanced performance across all classes.

$$G\text{-Mean} = \left(\prod_{c=1}^{K} \text{Recall}_c \right)^{\frac{1}{K}} \tag{A.7}$$

where recall for class (c) is defined as

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \tag{A.8}$$

A.7.1 Variable Definitions

(K) Number of classes.
(TP_c) True positives for class (c).
(FN_c) False negatives for class (c).
(Recall_c) Recall for class (c).

The geometric mean penalizes poor performance on any individual class and therefore provides a more balanced assessment than overall accuracy when class distributions are skewed.

A.8 Multi-Class Area Under the ROC Curve (MAUC)

The Multi-Class Area Under the ROC Curve extends the binary AUC metric to multi-class classification problems.

$$MAUC = \frac{2}{K(K-1)} \sum_{i<j} AUC(i,j) \tag{A.9}$$

A.8.1 Variable Definitions

(K) Number of classes.
($AUC(i,j)$) Pairwise AUC between classes (i) and (j).

MAUC measures the ability of a classifier to distinguish between classes independently of the selected decision threshold. Values closer to one indicate superior discriminative ability.

A.9 Per-Domain Expert Utilization

Per-domain expert utilization measures the fraction of samples from a given domain that are routed to a particular expert in a Mixture-of-Experts architecture.

$$U_{d,i} = \frac{N_{d,i}}{N_{d,\text{total}}} \tag{A.10}$$

A.9.1 Variable Definitions

($N_{d,i}$) Number of samples from domain (d) routed to expert (i).
($N_{d,\text{total}}$) Total number of samples from domain (d).
($U_{d,i}$) Utilization of expert (i) for domain (d).

This metric provides insight into expert specialization and routing behavior. High utilization values indicate that an expert is strongly associated with a particular domain, while more uniform distributions suggest shared expertise across domains.

A.10 Summary

The selected metrics characterize four major aspects of system behavior:

1. **Overall predictive performance**
   - Average Incremental Accuracy
   - G-Mean
   - MAUC
2. **Knowledge retention**
   - Forgetting Measure
   - Backward Transfer
3. **Knowledge transfer**
   - Forward Transfer
   - Backward Transfer
4. **Adaptation and expert specialization**
   - Intransigence Measure
   - Per-domain Expert Utilization

Collectively, these metrics provide a comprehensive assessment of continual learning performance, class-balanced predictive ability, transfer dynamics, and expert allocation behavior.

A.11 Intermediate-Domain Metric Computation

The continual learning metrics described in this appendix are traditionally computed only after completion of training on an entire domain. However, for the purposes of monitoring training dynamics, intermediate metric values were also computed during training epochs within a domain.

Assume the model is currently training on domain (d), and let (e) denote the current epoch within that domain. Rather than waiting until domain (d) is fully trained, the corresponding row of the performance matrix is populated using the model's current evaluation results at epoch (e).

An intermediate performance matrix can therefore be defined as

$$R^{(e)} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,D} \\ 0 & a_{2,2} & \cdots & a_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{d,j}^{(e)} \end{bmatrix}$$

where

$[a_{d,j}^{(e)}]$

denotes the performance on domain (j) after epoch (e) of training on the currently active domain (d).

All previously completed domains retain their final values in the matrix, while only the row corresponding to the currently trained domain is replaced by its intermediate epoch measurements. The standard continual learning metrics (Average Incremental Accuracy, Forgetting Measure, Backward Transfer, Forward Transfer, and Intransigence) are then computed using ($R^{(e)}$) in exactly the same manner as the completed-domain formulations presented earlier in this appendix.

Once training on domain (d) is complete, the intermediate values ($a_{d,j}^{(e)}$) are replaced with their final values ($a_{d,j}$), permanently fixing the corresponding row of the performance matrix. Training then proceeds to domain (d+1), whose row becomes the new source of intermediate epoch measurements.

This procedure provides a continuous estimate of continual learning behavior throughout training while remaining fully consistent with the final metric definitions. Consequently, the metric values converge to the standard continual learning metrics at the completion of each domain and ultimately match the final reported values after the last domain has been trained.

B

Appendix 2: Experimental setup details

Meta-config Summary

CIFAR-10 meta-configs

Property	cifar10_easyDirichlet	cifar10_easyflat	balancedcifar10testing-*
Dataset	cifar10	cifar10	cifar10
Configs	tinyvit, vitmoe, convnext	balanced convnext/vit/moe	balanced convnext/vit/moe variants
Batch size	128	128	128
Optimizer	adamw/adam	adamw	adamw
LR (loss.lr)	0.001	0.001	0.001
Loss	weighted_cross_entropy	weighted_cross_entropy	weighted_cross_entropy
Pretrain enabled	yes	yes	yes
Pretrain epochs	7	7	3
Epochs per domain	3	3	10
Partition type	dirichlet	dirichlet	dirichlet
Augmentations	flip,color-jitter,blur,noise	flip,color-jitter,blur,noise	random crop,flip,color-jitter,blur,noise
Model families	vit_moe / convnext	vit_moe/convnext	vit_moe/convnext/convnext-noSch
MoE experts (examples)	unshared=4, shared=1	unshared=4, shared=1	varied (4 unshared typical)

Notes: 'balancedcifar10testing-*' contains multiple configs (convnext vs moe) with similar hyperparams; convnext pretrain epochs listed as 3 in that file.

CORe50 meta-configs

Property	core50_DIL	core50_DIL_m_pretrain	core50_DIL_m_singleseed	core50_DIL_mini
Dataset	core50	core50	core50	core50
Configs	moe / vit / convnext variants	moe-EMA-pretrain, moe-mini, vit-mini	moe-EMA, moe-m-noPre, vit-m-noPre	moe-EMA-mini, moe-mini, vit-mini
Batch size	128	128	128	128
Optimizer	adamw	adamw	adamw	adamw
LR	0.0001	0.0001	0.0001	0.0001
Loss	weighted_cross_entropy (alpha=auto)	weighted_cross_entropy	weighted_cross_entropy	weighted_cross_entropy
Pretrain	sometimes enabled (varies)	enabled (epochs=7)	enabled	enabled
Epochs per domain	10 (default)	10/5 (mini)	5	5
Partition type	domainIncremental	domainIncremental	domainIncremental	domainIncremental
MoE experts (examples)	varied: unshared 0–8, shared 1	unshared 4 / num_experts 4	num_experts 4 (imagelevel)	unshared=4 in mini
Routing / anneal	many configs use routing anneal, prototype EMA in pretrained heads	yes (anneal, moe_expert_lr_mult)	yes	yes

Notes: Several CORe50 configs differ in 'moe_num_unshared_experts' and routing anneal; see per-meta-config tables below for details. ConvNeXt differed in some parameters due to tuning for better performance for it.

OfficeHome meta-configs

Property	officehome_DIL_full	officehome_DIL_modeltransfer
Dataset	officehome	officehome
Configs	EMArouter pretrain + mini + vit-mini	moe-modeltransfer + vit-modeltransfer
Batch size	128 (mini) / 64 (modeltransfer,long)	64
Optimizer	adamw	adamw
LR	0.0001	0.0001
Loss	weighted_cross_entropy	weighted_cross_entropy
Pretrain	enabled (EMA pretrain)	often disabled
Epochs per domain	5–25 (varies: mini=5, long=25)	20
Partition type	domainIncremental	domainIncremental
MoE experts	commonly 4 experts	1 unshared expert (modeltransfer)
Special params	routing anneal, prototype EMA, moe_expert_lr_mult	pretrained_vit flags

Property	officehome_DIL_long-*
Dataset	officehome
Configs	long/mini/long-mini variants
Batch size	64 / 128 (mini-long)
Optimizer	adamw
LR	0.0001–0.0003 (prototyperouter uses 0.0003)
Loss	weighted_cross_entropy
Pretrain	disabled for long in some configs
Epochs per domain	15–25
Partition type	domainIncremental
MoE experts	4 experts (moe, prototyperouter)
Special params	routing anneal, moe lambdas, moe_anneal_epochs

Notes: 'officehome-moeprototyperouter-modeltransfer' uses 'vit_switch_moe' family with 'num_experts=4' and 'routing_alpha=0.3'; 'officehome-moeprototyperouter-modeltransfer' loss LR=0.0003. This is because there is a reduction of learning for the model due to the frozen layers.

Per-meta-config model and parameter tables

cifar10_easyDirichlet

Field	Value
Configs included	configs/tinyvitmoe_cifar10_easyDirichlet.jsonc, configs/vit_moe_cifar10_easyDirichlet.jsonc, configs/convnext_cifar10_easyDirichlet
Dataset	cifar10
Batch size	128
Loss	weighted_cross_entropy, lr=0.001, weighted=true
Optimizer	adamw
Pretrain	enabled, num_samples=2000, epochs=7, scheduler=linear (0.1->1.0)
Epochs per domain	3
Partition	dirichlet, min_size=1400, balanced=true
Augmentations	random_flip, color_jitter, blur, noise
Model variants	tinyvitmoe: embed=384, depth=8, heads=6, moe unshared=4, shared=1; vitmoe: alpha=0.5 partition; convnext tiny, patch_size=8
Experts per model	tinyvitmoe/vitmoe: 4 unshared + 1 shared (moe config)

cifar10_easyflat

Field	Value
Configs	balancedcifar10testing-convnext-noSch.jsonc, balancedcifar10testing-moe-noSch.jsonc, balancedcifar10testing-vit-noSch.jsonc
Batch size	128
Loss lr	0.001
Partition	dirichlet, balanced=true
Experts	moe configs: unshared=4, shared=1

core50_DIL

Field	Value
Configs	core50-moe*, core50-vit, core50-convnext variants
Batch size	128
Loss lr	0.0001
Partition	domainIncremental (num_partitions=11)
Epochs per domain	10
Experts	moe-many: unshared up to 8, moe-top-k up to 2

moe-ensemble uses shared experts multiplier list (1.0, 0.5, 0.1, 0.01)

officehome_DIL_full

Field	Value
Configs	officehome-moe-EMArouter-m-temp_anneal-pretrain.jsonc, officehome-moe-mini.jsonc, officehome-vit-mini.jsonc
Batch size	128 (mini)
Loss lr	0.0001
Pretrain	enabled for EMArouter pretrain (epochs=7)
Epochs per domain	5
Experts	vit_moe imagelevel: num_experts=4, embed=256, depth=10, heads=4

Seeds used and what they affect

Seed key	Effect in code
global	Default base seed if specific seeds missing; normalized by setup; used rarely (DIL/batch parsing)
dataset	Used in 'create_dataloaders' to seed dataset-level selection (e.g., 'make_mini_cifar') and initial RNG before dataset construction (see `dataset_fcns/dataset_utils.py`).
partition	Passed to 'dirichlet_split''static_split' to control partitioning randomness (see 'dirichlet_split' seed argument).
loader	Used when creating DataLoader objects ('create_train_test_loaders') as 'seed' to shuffle and to seed splitshuffle operations.
model	Influences model-level randomness if factories use RNG (e.g., expert init); it's stored in config seeds but applied by 'setup' when building model.
pretrain	Applied before pretraining stage in 'DIL_Experiment.py' (sets 'torch', 'numpy', 'random' seeds) so pretraining sampling reproducible.
training	Applied before main continual training stage in 'DIL_Experiment.py' (sets RNGs).
baseline	Applied before baseline training stage; sets RNGs for baseline runs so results reproducible.
replay	If replay buffer enabled, would control sampling determinism from the buffer (not used currently).

Where seeds are applied in code

Seeds are parsed and normalized in 'DIL_Experiment.py' (see 'parse_cli_seeds' and 'setup' call). The 'create_dataloaders' function in 'dataset_fcns/dataset_utils.py' reads 'cfg['seeds']' and sets 'np.random.seed', 'random.seed', and 'torch.manual_seed' for 'dataset_seed'; 'partition_seed' is passed into 'dirichlet_split'. In 'DIL_Experiment.py', pretrain/baseline/training seeds are applied immediately before those stages using 'torch.manual_seed', 'np.random.seed', and 'random.seed'.

Stars and mismatches

If two configs inside a meta-config do not match on a major field (e.g., 'batch_size' or 'epochs_per_domain'), the meta-config entry above includes a note with a star. See specific per-config tables for exact values. Examples found:

- 'balancedcifar10testing-*': convnext pretrain epochs set to 3 in that file, while other CIFAR configs use 7. Marked in notes above.
- OfficeHome mixes batch sizes between mini (128) and modeltransfer/long (64) - table shows both.

Generated from repository configs and experiment scripts on June 9, 2026. Future updates to the codebase will not be included.

C

Appendix 3: Additional results

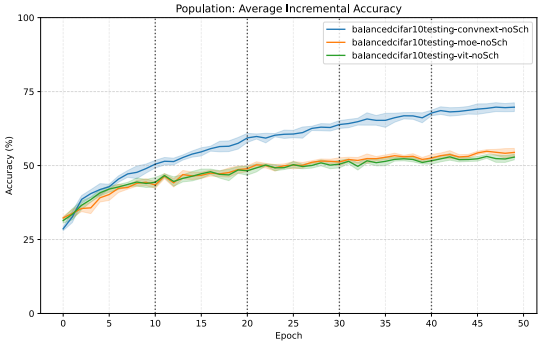
Additional Experiments

For additional experiment information, the results can be reproduced or old results can be viewed in the thesis repository [63]. The readme.txt includes steps for running experiments, how to set up new experiments, and how to explore the experimental results. Additional experiments and neural networks that did not end up in the final thesis can also be viewed.

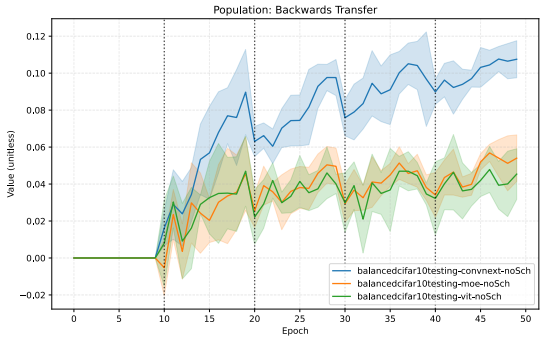
Additional graphs of selected experiments

The following graphs are from the experiments that were in the discussion. This includes all the graphs from the continual learning metrics per experiment.

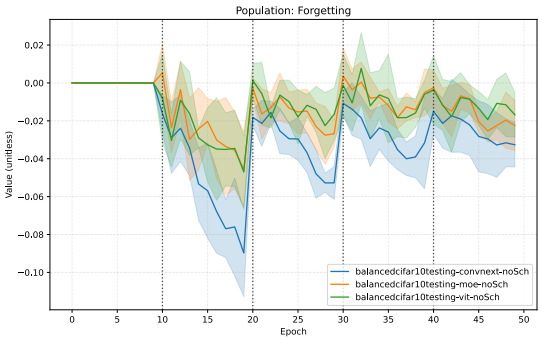
C.1 CIFAR-10



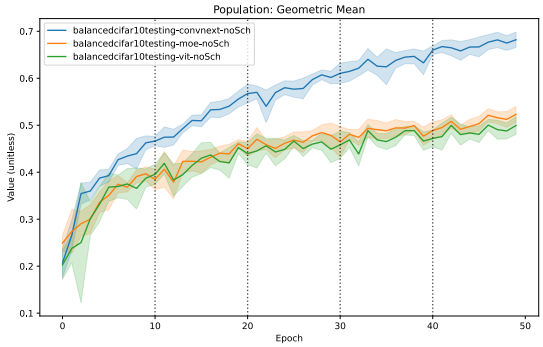
(a) AIA



(b) BWT

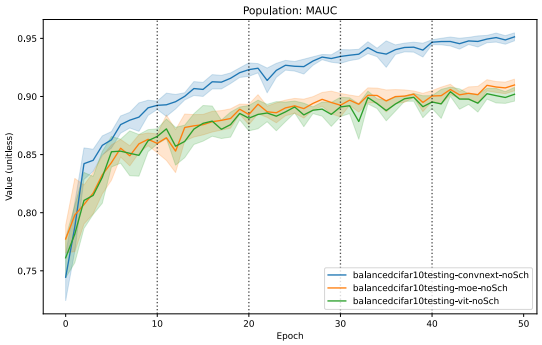


(c) FM

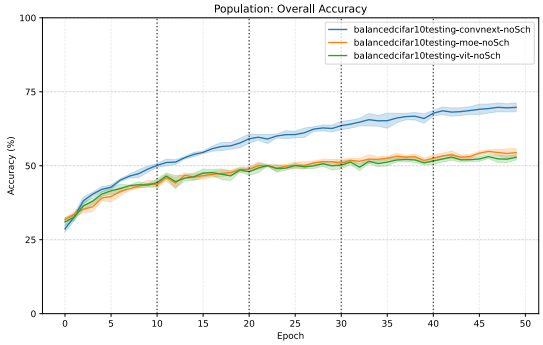


(d) G-MEAN

Figure C.1: CIFAR-10 metrics (1/2)



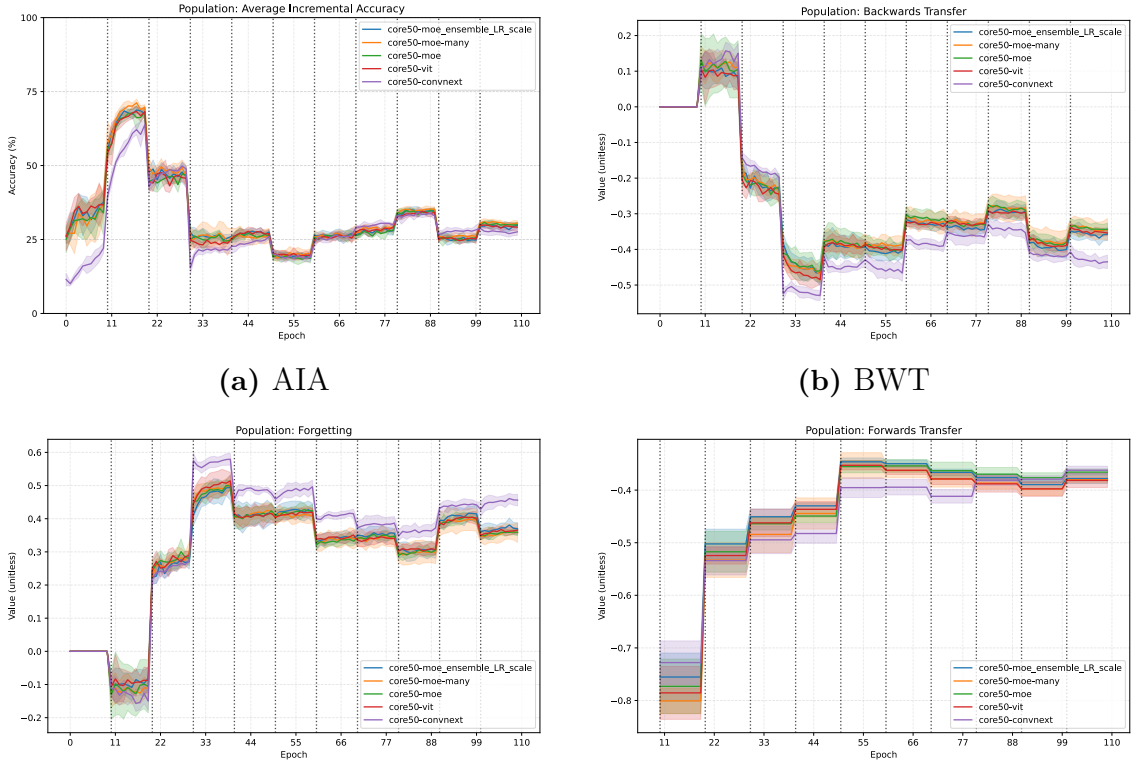
(a) MAUC



(b) OA

Figure C.2: CIFAR-10 metrics (2/2)

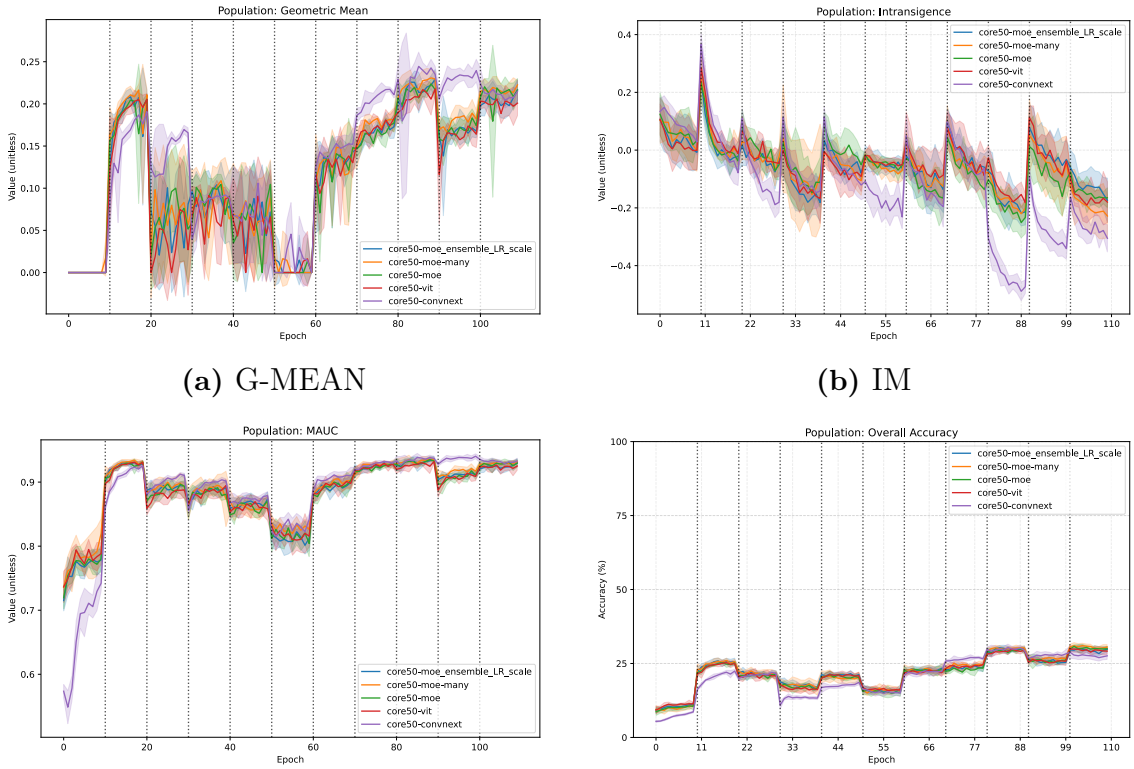
C.2 CoRE-50



(a) AIA **(b)** BWT

(c) FM **(d)** FWT

Figure C.3: CoRE-50 metrics (1/2)

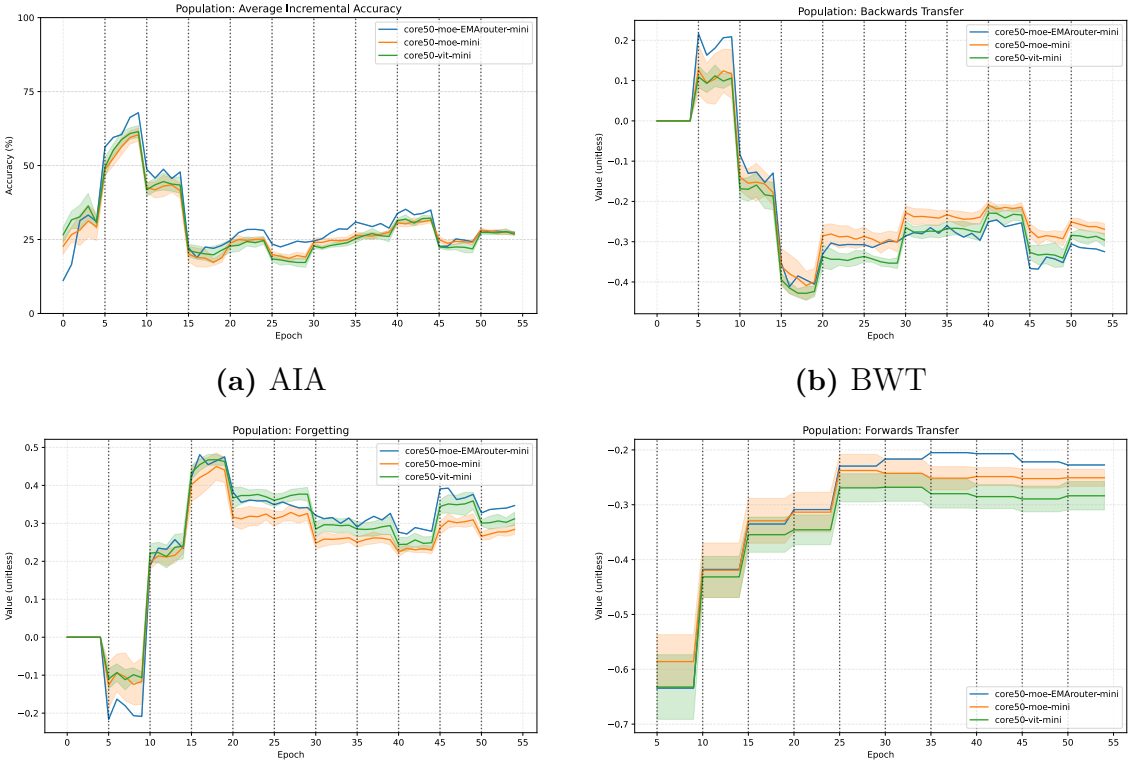


(a) G-MEAN **(b)** IM

(c) MAUC **(d)** OA

Figure C.4: CoRE-50 metrics (2/2)

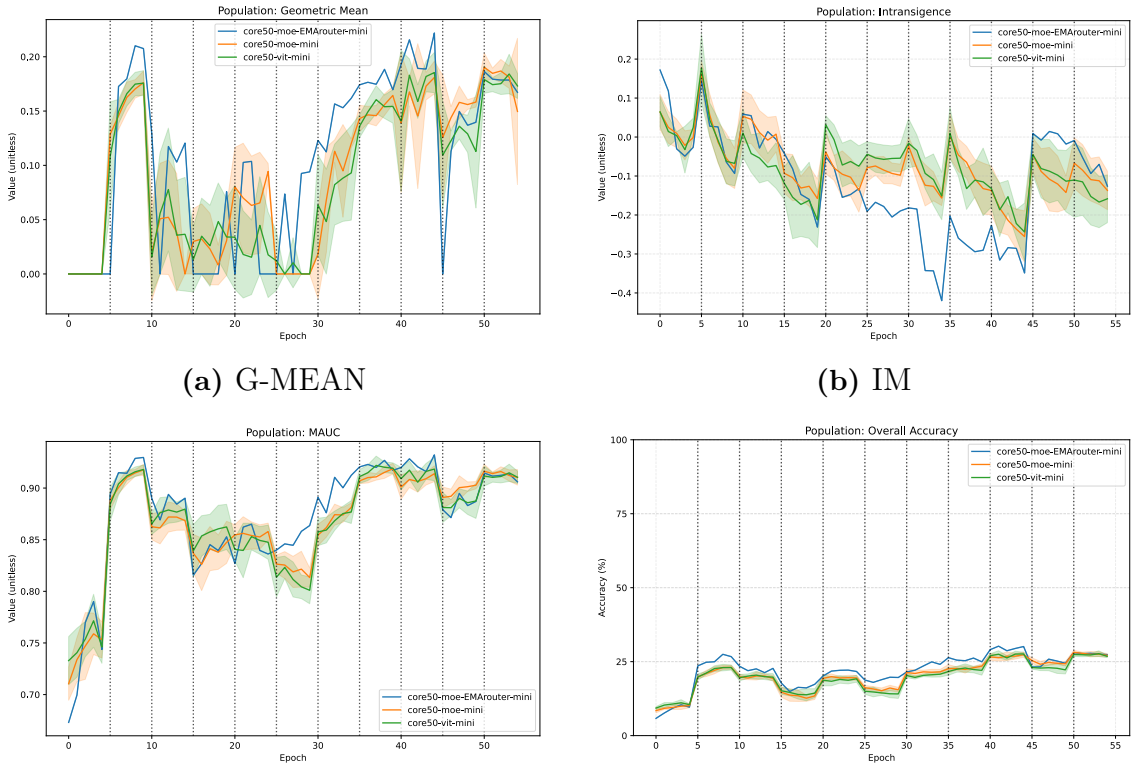
C.3 CoRE-50_mini



(a) AIA **(b)** BWT

(c) FM **(d)** FWT

Figure C.5: CoRE-50_mini metrics (1/2)

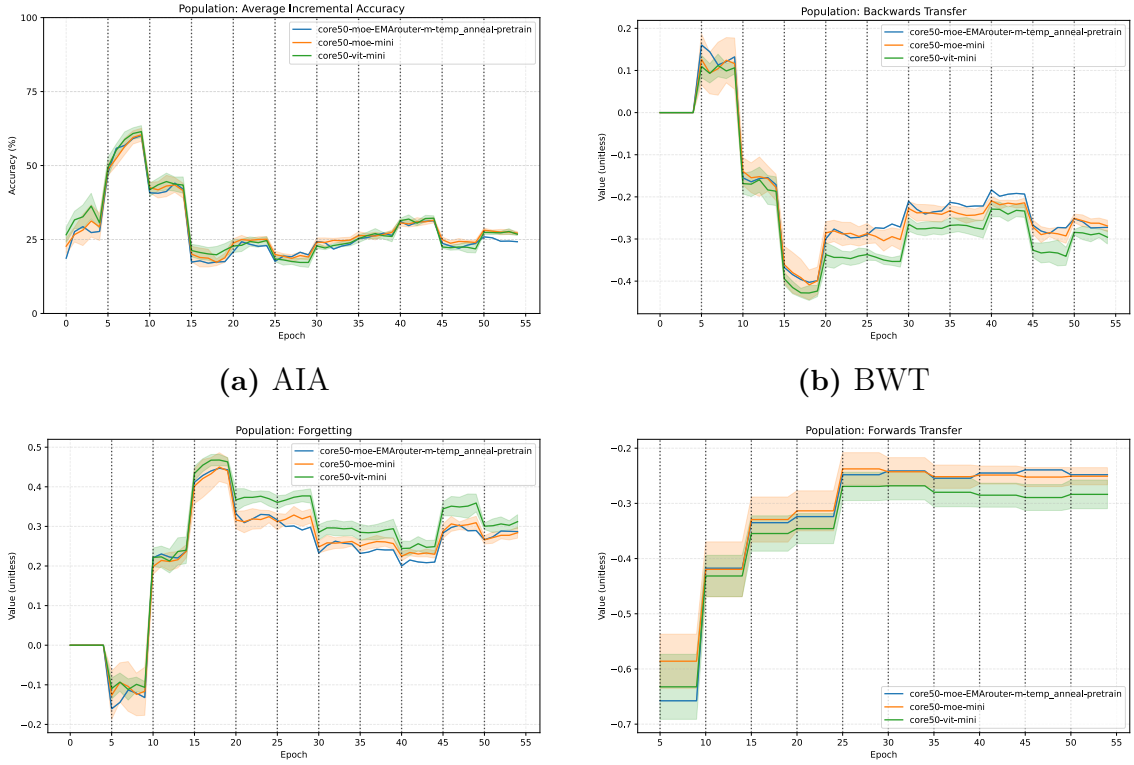


(a) G-MEAN **(b)** IM

(c) MAUC **(d)** OA

Figure C.6: CoRE-50_mini metrics (2/2)

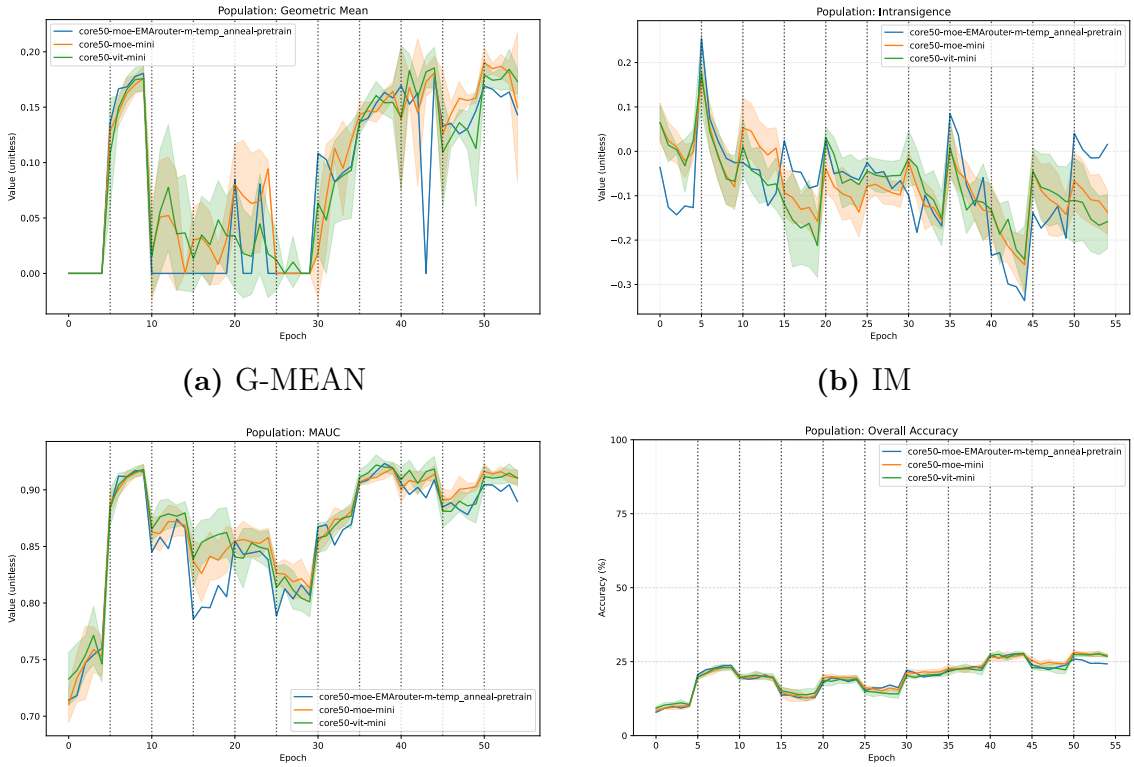
C.4 CoRE-50_pre



(a) AIA **(b)** BWT

(c) FM **(d)** FWT

Figure C.7: CoRE-50_pre metrics (1/2)

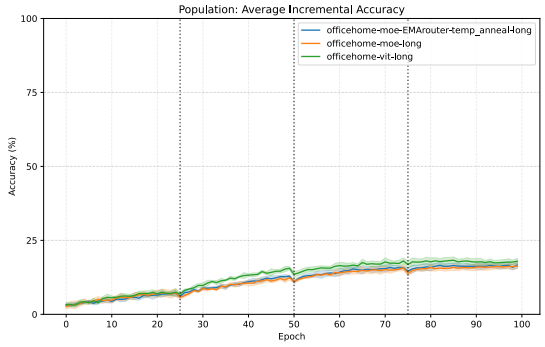


(a) G-MEAN **(b)** IM

(c) MAUC **(d)** OA

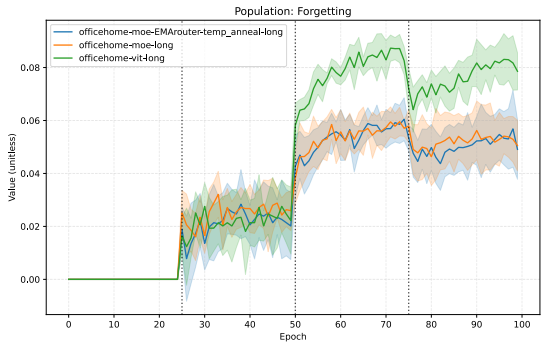
Figure C.8: CoRE-50_pre metrics (2/2)

C.5 OfficeHome_long-s

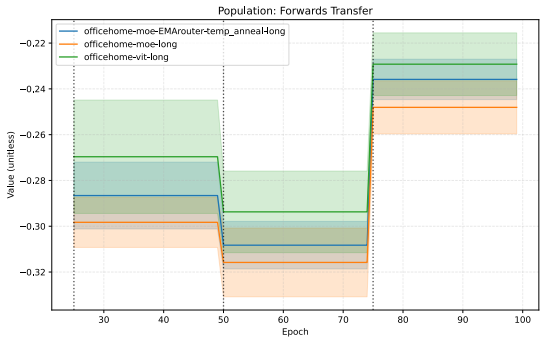


(a) AIA

(b) BWT

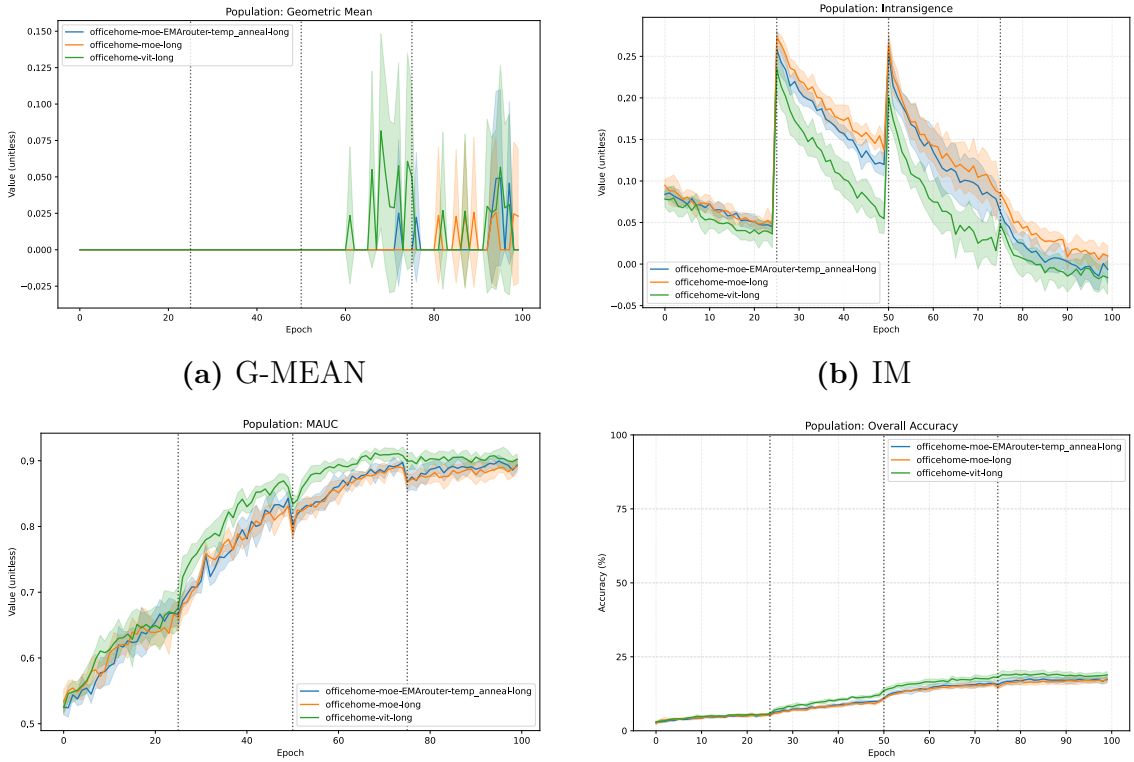


(c) FM



(d) FWT

Figure C.9: OfficeHome_long-s metrics (1/2)

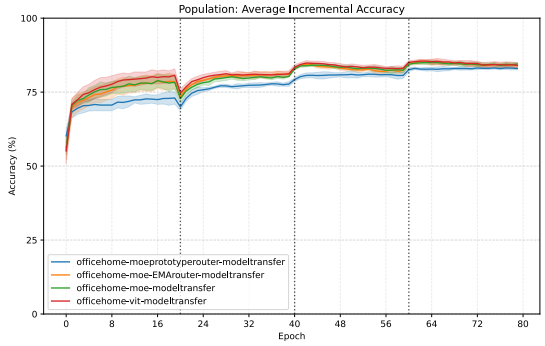


(a) G-MEAN **(b)** IM

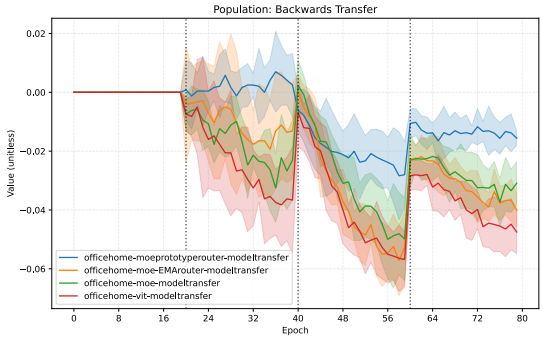
(c) MAUC **(d)** OA

Figure C.10: OfficeHome_long-s metrics (2/2)

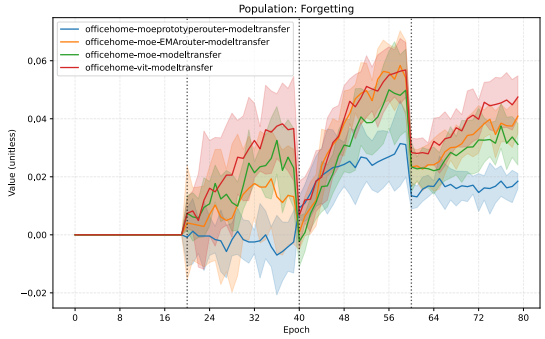
C.6 OfficeHome_pre



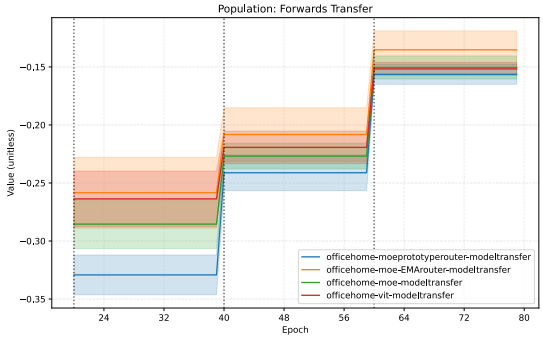
(a) AIA



(b) BWT

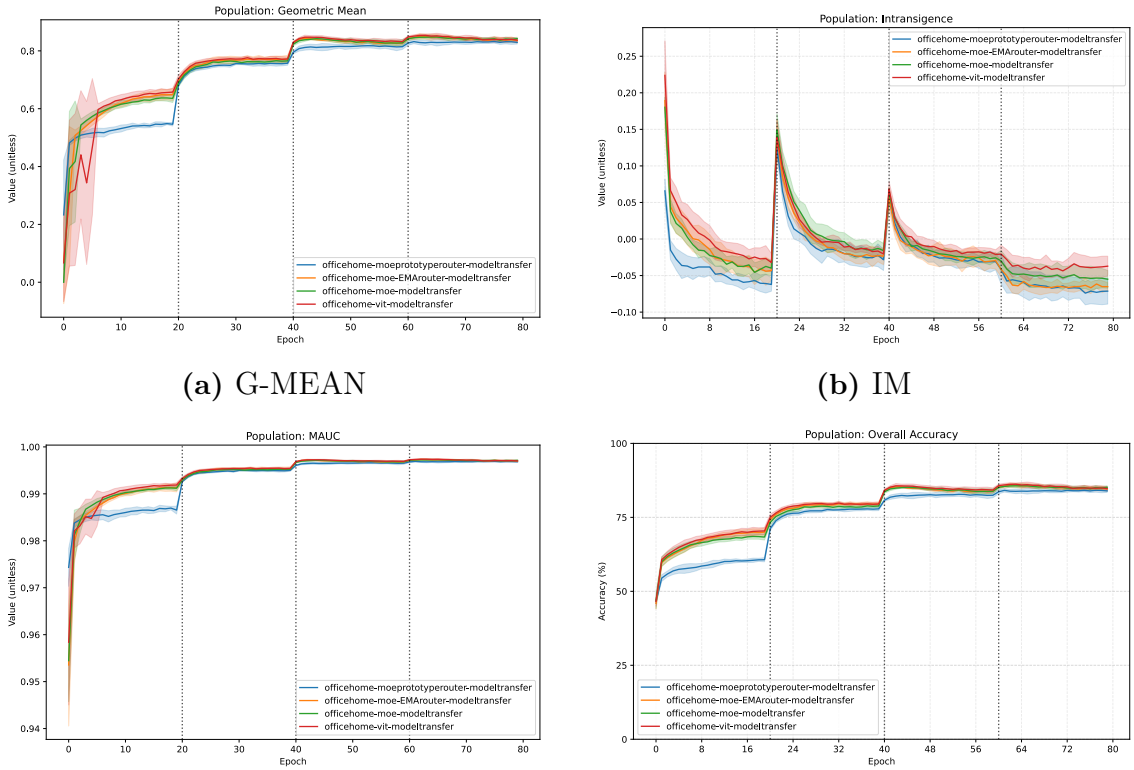


(c) FM



(d) FWT

Figure C.11: OfficeHome_pre metrics (1/2)



(a) G-MEAN **(b)** IM

(c) MAUC **(d)** OA

Figure C.12: OfficeHome_pre metrics (2/2)

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY