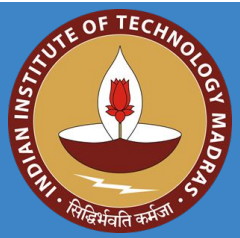


Analysis of Learned Features Using Interpretable Neural Nets under Memory Loss Settings

For Phase-2 Project
Of
Masters of Technology

Presented by:-
Sahil Bafna
CS21M056

Guide:
Dr. Harish Guruprasad

Panel Member:
Dr. Arun Rajkumar

Continual Learning

- **Continual Learning (CL)** is a technique in machine learning that helps machines learn new things while not forgetting what they have already learned. It's like a human brain that can learn new things without forgetting what it already knows.

- In traditional machine learning, a neural network is trained on a single task and its weights are optimized for that task only. If the network is trained on a new task, it may overwrite the knowledge it has acquired from the previous task, leading to the problem of **catastrophic forgetting**. Ex:- Modified School Learning.

- Continual learning has many practical applications, such as in autonomous driving, robotics, natural language processing, and image recognition, where a system must be able to learn and adapt to new situations while retaining knowledge from previous experiences

- Moving forward we will deal with **Strict- Continual Learning paradigms** only**.**

Strict-Cl Paradigms

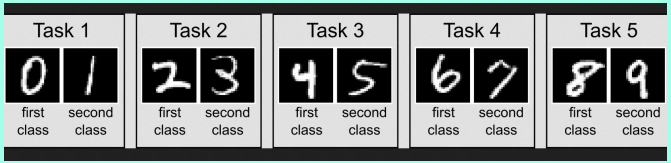
- **Task Incremental Learning**: In this paradigm, the network learns a sequence of tasks one at a time, with no access to previous tasks once they have been learned. The network must selectively retain important weights related to previous tasks while allowing other weights to be updated for the new task.

- **Memory Constrained Learning**: In this paradigm, the network has limited memory resources and must use these resources to selectively store and retrieve information related to previous tasks. The network must manage its memory efficiently to avoid catastrophic forgetting and to ensure that it can learn new tasks without losing knowledge from previous tasks.

- **No Negative Feedbacks**: In this paradigm, future (or past) data samples cannot be provided to the network with the current data/task in any way.

Continual Learning

Tasks:

- **Similar tasks**: contains same type of information in different permutation and have same number of class to infer
  - Ex: PMNIST, SplitMNIST
  -
  - 

- **Syntactically similar tasks:** contains different type of data but have same number of classes to infer
  - Ex: **task 1** : Amazon review with Neg/Pos feedback class and **task 2**: MNIST with only datapoints of first 2 classes

- **Dissimilar task:** contains different type of data and have different number of classes to infer too.
  - Ex: **Task 1**: CIFAR-100 **Task 2**: MNIST , we use **multiheaded** CNN.

Observing Catastrophic Forgetting

- Table Below shows the result of experiments done on simple Relu network to observe catastrophic forgetting

- Dataset:- used permuted-MNIST dataset and partitioned it into 5 tasks, each containing 10 classes that corresponded to distinct permutations of the original MNIST dataset

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	98.25				
Train-2	30.36	86.47			
Train-3	27.30	29.02	88.22		
Train-4	22.36	21.26	32.38	81.36	
Train-5	16.60	17.91	17.89	35.23	86.32

Observing Catastrophic Forgetting

Three Scenarios of CL

Three continual learning scenarios based on whether at test time task identity is provided and—in case it is not—whether it must be inferred.

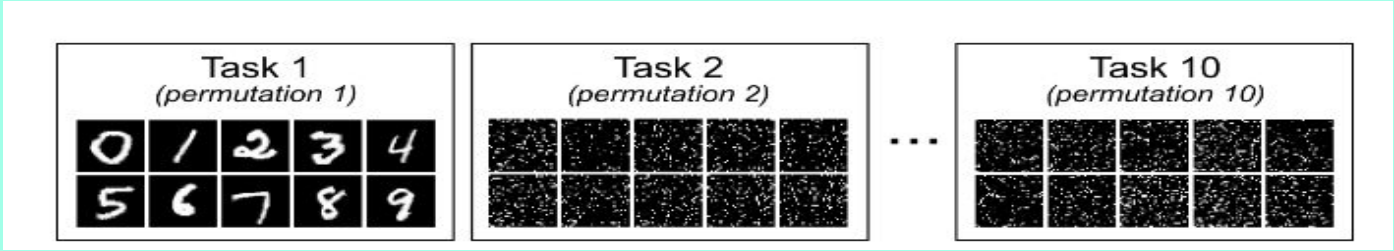
Task-IL : Solve tasks so far, task-ID provided | With task given, is it the 1st or 2nd class? (e.g., 0 or 1)

Domain-IL: Solve tasks so far, task-ID not provided | With task unknown, is it a 1st or 2nd class?

(e.g., in [0, 2, 4, 6, 8] or in [1, 3, 5, 7, 9])

Class-IL: Solve tasks so far and infer task-ID | With task unknown, which digit is it?

(i.e., choice from 0 to 9)



SOTA Methods to overcome CF

- **Data Replay:** Usage of old or future task data in any way to train the neural network
- **Multi-Heads:** refers to the usage of last layer for each task
- **Generative Replay:** refers to the usage of generative modelling for the dataset or task at the hand and using these generated samples for some form of data replay.
- **Multi-Models:** refers to usage of a neural network model other than the original model to help .

But most of the CL Methods relax the constraints of **Strict-CL i.e.,** Not using any negative exemplars and largely preserving the base neural network model.

Below Table Shows the results of **Data Replay Method** for **PMNIST.**

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	94.52				
Train-2	96.31	85.39			
Train-3	97.52	90.36	88.22		
Train-4	97.21	94.12	90.27	89.23	
Train-5	97.63	95.21	94.88	94.12	92.01

Elastic Weight Consolidation (EWC)

- This approach remembers old tasks by selectively slowing down learning on the weights important for those tasks.

- The EWC methodology alleviates catastrophic forgetting by regularizing parameters of a network trained on previous tasks by penalizing any change in them according to their importance

- This importance is indicated by the Fisher information matrix i.e. after a network is trained on one task, fine-tuning on the next task is performed on according to

- The Fisher information matrix can be used to measure the importance of each weight in a neural network for previous tasks, and the matrix can be used to define a regularization term that constrains the network's weights to retain information from previous tasks while learning new ones

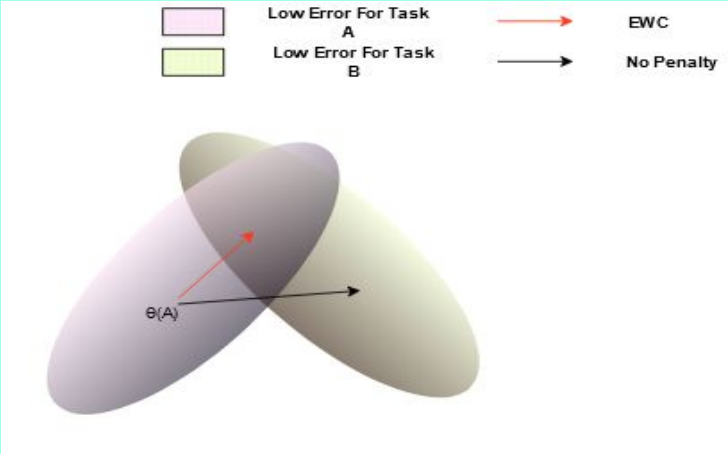
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

EWC Experiments and Results

- **Dataset**: used permuted-MNIST and created 5 task of 10 class each being different permutation of MNIST.

- **Architecture**: we have used a CNN with 2 convolution layers of size (1, 10), (10, 20) respectively followed by a maxpool layer and 2 fully connected layer for 10 class classification using relu activation

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	98.91				
Train-2	95.31	98.23			
Train-3	95.22	96.36	97.29		
Train-4	95.01	95.63	96.87	98.95	
Train-5	94.91	95.08	95.48	95.23	97.12



EWC allows retention of knowledge learned from task A while training on task B. It accomplishes this by identifying the parameters responsible for good performance on task A (pink parameter region) and on task B (green parameter region) in a parameter space. EWC then uses this information to update the parameters to a new solution for task B (red arrow) that does not cause significant deterioration of performance on task A.

EWC's Limitation

- **Limited scalability**: EWC can become computationally expensive as the number of tasks increases. This is because it requires storing and computing the Fisher information matrix for each task, which can be computationally intensive.

- **Strong assumptions about task similarity**: EWC assumes that the tasks are similar, and that the optimal weights for each task are similar to the optimal weights for previous tasks. This assumption may not hold in all situations, leading to suboptimal performance or even catastrophic forgetting.

- **Lack of flexibility**: EWC is a fixed method that assumes a specific regularization strength for all tasks. This may not be optimal for all tasks, as some tasks may require stronger regularization than others.

- **Difficulty in determining the number of tasks**: EWC requires knowing the number of tasks in advance, which may not always be possible in real-world scenarios.

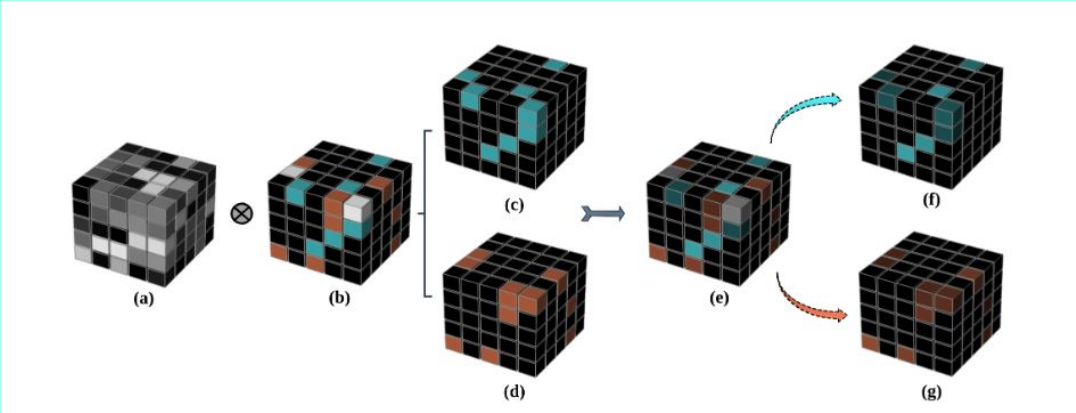
- **Limited effectiveness for non-stationary tasks**: EWC may not be effective for tasks that are non-stationary, meaning that the data distribution changes over time. In such cases, EWC may not be able to prevent catastrophic forgetting as the network may need to drastically change its weights to adapt to the new data distribution.

Relevance Mapping Network

- **Relevance Mapping**, which is a method inspired by the Optimal Overlap Hypothesis, that aims to learn an optimal representational overlap, such that unrelated tasks use different network parameters, while allowing similar tasks to have a representational overlap
- RMNs are a type of neural network that focus on identifying the most important input features for a given task, while ignoring irrelevant or redundant features
- RMNs use a mapping function to map the input features to a low-dimensional relevance space, where the relevance of each feature is estimated based on its importance for the task at hand
- Once the input features are mapped to the relevance space, the relevance scores can be used to weight the input features and determine their contribution to the output

$$\mathbb{M}_{\mathbb{P}i} \approx \prod_i \mathcal{L}_R(\mathcal{N}(\mu_i, \sigma_i^2)) \quad \text{and} \quad \mathcal{L}_R(x_i; \beta) = \frac{1}{1 + \exp(-(\beta(x_i - 0.5)))}$$

we can think of the RMNs as replacing the weights of a network with the product of the weights and a binary relevance mixture.



Algorithm 1: RMN Supervised Continual Learning

Input: The input for the algorithm consists of the following: data x and corresponding ground truth labels y for n tasks, a prune parameter μ, and task labels i paired with all x.

Given : The algorithm requires two inputs: parameters W and initialized relevance mappings M_P.

for *each task* i **do**
  $f(x_i; W, M_{P_i}) \rightarrow \hat{y}_i = \sigma((W \cdot MP_i) \cdot x_i)$
  Compute Loss: $L(\hat{y}_i, y_i)$
  Optional: Add Sparsity Loss: $L(\hat{y}_i, y_i) + (MP_i)_{l_0}$
  Backpropagate and optimize
  Prune $M_{P_i} \leq \mu$ only
  Stabilize parameters in f where $M_{P_i} = 1$
end
for *data* x *and ground-truth task label* i **do**
  Output: $f(x, i; W)$
end

- Sparsity loss is a regularization term that is added to the loss function of a machine learning model to encourage sparsity in the model's weights or activations. The sparsity loss penalizes the model for having non-zero weights or activations, which encourages the model to use only a small subset of the available features or units

- For a sufficiently simple task, only a subset of the parameters in `W` are often required
- if we could learn the importance or relevance of each weight node, we could apply a zero-mask to the non-essential parameters without pruning or modifying them and still successfully learn the ground-truth.
- A set of mappings can explicitly be representing the neuron-to-neuron connections of the network
- Any network f with weight tensors `W` can have such corresponding sets of neuron connection representations Mp for T tasks/mappings, where each set Mpi activates a subnetwork mapping in f that could be used for various purposes for a task i.

RMNs Experiments

- **Dataset**: used permuted-MNIST dataset and partitioned it into 5 tasks, each containing 10 classes that corresponded to distinct permutations of the original MNIST dataset. You also employed the Split-MNIST dataset and created 5 tasks,with each task consisting of 2 classes. Additionally, used the CIFAR-100 dataset and divided it into 20 tasks, each containing 5 classes

- **Architecture**: Used Multi Layer Preceptron(MLP) for both P-MNIST and S-MNIST that have 3 fully connected layer of size 100 used batch normalization and relu, and softmax for optimizer. For CIFAR-100 it uses 3 convolution block having 2 layers each of size (32, 64, 128) each resp. followed by max-pool and relu and 2 fully connected layer.

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	98.91				
Train-2	95.31	98.23			
Train-3	95.22	96.36	97.29		
Train-4	95.01	95.63	96.87	98.95	
Train-5	94.91	95.08	95.48	95.23	97.12

RMNs Limitation

- **Complexity**: RMN can be computationally expensive to train and evaluate, especially for large datasets with high-dimensional input features. The model involves many hyperparameters that need to be tuned to achieve optimal performance, which can also increase the computational burden.

- **Interpretability**: While RMN can provide insights into the relevance of features in a dataset, it may not provide a complete explanation of the underlying relationships between features. Additionally, the interpretability of the model may be limited by its complexity and the large number of parameters.

- **Generalization**: RMN may not generalize well to new datasets with different characteristics than the training data. The model may be overfitting to the training data, which can result in poor performance on new data.

- **Scalability**: RMN may not scale well to extremely large datasets. As the size of the dataset grows, the computational cost of training and evaluating the model may become prohibitively expensive.

Catastrophic Remembering

Capability of the network to discriminate between old and new inputs. This is referred to as Catastrophic Remembering.

Unlike CF , CR i s not the result of learning new data after having already learned an initial set of patterns but rather the result of the network’s learning a function too well in some sense.

Incremental Moment Matching

Incremental Moment Matching (IMM) is a technique used to reduce catastrophic forgetting in continual learning. The basic idea behind IMM is to align the statistical moments of the current task with those of the previous tasks, in order to minimize the differences between the weight distributions learned for each task.

In incremental moment matching (IMM), the moments of posterior distributions are matched in an incremental way. In our work, we use a Gaussian distribution to approximate the posterior distri-bution of parameters

Mean-IMM

- In Mean Incremental Moment Matching (Mean-iMM), the two networks are trained on different tasks sequentially. When training a neural network on multiple tasks, catastrophic forgetting can occur, meaning that the network may forget previously learned information when training on a new task.

- To address this issue, Mean-iMM trains separate neural networks for each task and averages their parameters to obtain a solution that is a mixture of the parameters of all previous tasks. In each layer, the mixing ratios αk determine how much weight to give to each network.

- The mixing ratios αk are typically set based on the number of tasks seen so far, and they are normalized so that their sum is equal to one. By averaging the parameters of multiple networks trained on different tasks, Mean-iMM can reduce the amount of catastrophic forgetting that occurs when learning new tasks.

Mode-IMM

- Mode Incremental Moment Matching (Mode-iMM) is a variant of Mean-iMM that uses the covariance information of the posterior distribution of a Gaussian approximation to the weight distribution for each task.

- In Mode-iMM, instead of simply averaging the parameters of two networks in each layer using mixing ratios, the algorithm computes the mode (peak) of the posterior distribution of each network's weights. Then, the algorithm uses these modes as the parameters for a new network. To incorporate the covariance information, the algorithm also calculates the average of the covariances of the posterior distributions of each network's weights, and adds this to the diagonal of the covariance matrix of the new network's weights.

- By incorporating the covariance information of the posterior distributions, Mode-iMM can capture the uncertainty in the weights and make more informed decisions about which network parameters to use. This can lead to better performance and less catastrophic forgetting when learning new tasks.

Experiments and Results

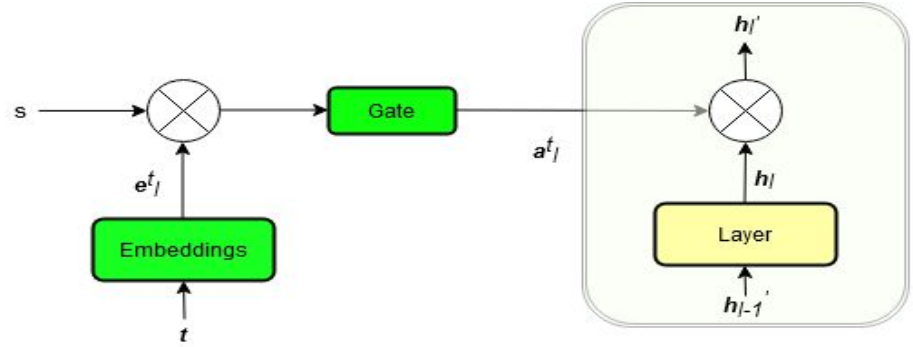
Dataset: Here we have used Disjoint MNIST with 2 task containing 5 classes each. i.e., Task 1 -{0, 1, 2, 3, 4} and Task 2 - {5, 6, 7, 8, 9}

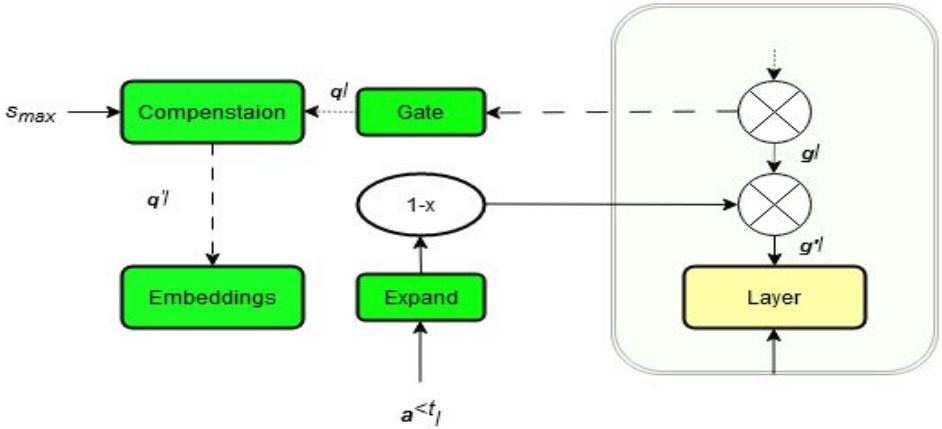
	Mean-IMM	Mode-IMM
Mean-Accuracy	95.12	98.38

Table : Mean Accuracy Disjoint MNIST

Hard Attention to Task

- HAT uses almost-binary attention vectors that are learned using gated task embedding and backpropagation.
- The mask allows a portion of the weights to remain fixed, while the rest adapt to the new task
- HAT uses a layer-wise attention mechanism that conditions the neural network to the current task by element-wise multiplication of the output of the units of layer .
- Unlike typical attention mechanisms that generate a probability distribution, the attention mechanism used in HAT is a gated version of a single-layer task embedding





- HAT achieves path learning and network learning together using backpropagation and stochastic gradient descent, without the need for a separate path learning stage using genetic algorithms

- The attention mechanism used in HAT is a gated version of a single-layer task embedding.

- The gating mechanism presented in HAT creates hard attention masks, which can be binary and act as "inhibitory synapses" to activate or deactivate the output of units in every layer dynamically, preserving paths across layers.

- With HAT, individual-unit paths are learned and dimensioned automatically, affecting the individual layer weights.

HATs Experiments and Results

- **Dataset**: Used CIFAR-10 with 5 task 2 classes each.

- **Architecture**: a convolutional neural network with three layers of size 64, 128, 256 followed by max-pooling layer and relu as activation followed by 2 fully connected layers with softmax as output.

- **Evaluation**: The forgetting ratio is a metric used to compare task accuracies and determine the level of forgetting that has occurred. This ratio provides a general measurement of the extent to which previous knowledge has been lost

$$\rho^{\tau \leq t} = \frac{A^{\tau \leq t} - A_R^{\tau}}{A_J^{\tau \leq t} - A_R^{\tau}} - 1$$

	$\rho \leq 1$	$\rho \leq 2$	$\rho \leq 3$	$\rho \leq 4$	$\rho \leq 5$
Average forgetting ratio	-0.02	-0.04	-0.04	-0.05	-0.06

Continual Learning Using DLGNs

- DLGNs are a type of neural network that is more interpretable than traditional DNNs and DGNs.

- DLGNs learn basic linear mappings of features, which are then used to enhance the input to a more complex, higher-dimensional representation.

- The gate network in DLGNs serves as a deep linear network, while the weight network has a constant input of 1 to make it easier to learn a linear function using weights. This allows DLGNs to perform both linear feature learning and linear weight training in a lifted space, providing both the power of non-linearity and interpretability.

- Catastrophic Forgetting occurs when a neural network forgets previously learned information when learning new tasks. DLGNs can alleviate this by separating the weights and gates through NPVs and NPFs, respectively.

- For similar tasks, the change in NPVs upon encountering multiple tasks can be minimized, helping to preserve previously learned information.

Experiments on DLGNs

- **First**, we try to have the hidden layers of a neural network the NPVs can be shared across all tasks, while still having multiple task-specific output layers. Also Known as hard parameter sharing

- **Second**, used different learning rate for layers and neurons throughout the network. To preserve the general features learned by layers closer to the input, it is advisable to minimize the extent of changes made to them.

- **Third** we try to have task specific layers by freezing or slow down learning significantly for the layer or neurons that we would like to reserve for that task. Despite its effectiveness, the approach of assigning a pre-defined structure for parameter sharing still has limitations. While it may be suitable for established computer vision problems, it can be prone to errors when applied to novel or less-studied tasks.

- **Fourth*,** Implement the above studied **Elastic Weight Consolidation (EWC)** Method over NPVs to reduce Catastrophic forgetting.'

Will Start our Experiments By observing Catastrophic forgetting in DLGNs.

Experiment Details

- **Dataset**: used permuted-MNIST and created 5 task of 10 class each being different permutation of MNIST. Further we used Office Home dataset with 4 task each being Art, Clipart, Product, Real-World with 65 categories.

- **Architecture**: we use 2 separate VGGs for Weight and Gates , Weight network is Called Neural Path Value (NPV), Gates network is Called Neural Path Feature (NPF), Identity Function is used instead of ReLU. NPVs are provided with constant 1 as input, NPFs are provided with actual image as input, 13 hidden convolution layer. Global Average Pooling is used at the end of 5 blocks, at last we have 3 FC Dense Layer at last we have Softmax activation

Office-Home Dataset

Results

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	97.04				
Train-2	20.32	94.74			
Train-3	10.21	15.32	93.17		
Train-4	10.56	13.71	20.56	90.19	
Train-5	10.12	10.38	16.21	19.28	93.21

Observing Catastrophic Forgetting on DLGN on P-MNIST

Results

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	97.52				
Train-2	86.23	94.04			
Train-3	78.35	82.23	89.71		
Train-4	76.01	80.63	85.87	88.95	
Train-5	72.08	78.48	83.23	86.23	89.30

Accuracy for Selective Layer on DLGN for P-MNIST

Results

	Art	Clipart	Product	Real World
Art	70.71			
Clipart	51.28	67.24		
Product	52.01	67.21	72.34	
Real World	53.21	58.21	62.21	71.12

Accuracy for Selective Layer on DLGN for Office-Home Dataset

Results

	Task-1	Task-2	Task-3	Task-4	Task-5
Train-1	98.39				
Train-2	96.74	95.39			
Train-3	89.00	91.50	95.79		
Train-4	84.20	83.42	92.54	95.57	
Train-5	79.76	69.94	81.74	90.14	95.83

Accuracy for EWC over DLGN for PMNIST

Conclusion

- **DLGNs** provide a promising solution to the problem of catastrophic forgetting in continual learning.

- By utilizing hard parameter sharing, varying learning rates, and task-specific layers, along with NPFs and NPVs, we can optimize the network to learn multiple tasks while minimizing the risk of forgetting

- Additionally, the use of probabilistic methods can further enhance the effectiveness of DLGNs in multitasking scenarios. Among the three approaches discussed earlier, the third approach of task-specific layers with frozen or slow learning and Using EWC worked better than the other two, highlighting the importance of adaptability in continual learning

Future Work

- Techniques such as Relevance Mapping, and probabilistic methods can be used to overcome these challenges by preserving existing knowledge while accommodating new information

- In addition, the use of posteriors provides a powerful tool for improving the efficiency and adaptability of DLGNs in multi-task settings, by enabling shared priors and better uncertainty quantification.

- By using Bayesian inference to compute these posteriors, the model can perform both parameter estimation and uncertainty quantification simultaneously, resulting in more robust and adaptive learning

Thank You