Unifying Continual Learning and Machine Unlearning for Expressiveness Aware Models

Aditya Bangar, Siddhant Jakhotiya
Department of Computer Science
Indian Institute of Technology Kanpur
Kanpur, UP 208016
`{adityavb21, siddhantj21}@iitk.ac.in`

Hamza Masood, Yash Chandra
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, UP 208016
`{hamzam21, yashc21}@iitk.ac.in`

Piyush Rai
Department of Computer Science
Indian Institute of Technology Kanpur
Kanpur, UP 208016
`piyush@cse.iitk.ac.in`

Abstract

The increasing ubiquity of generative models has sparked concerns about their potential misuse to generate harmful, inappropriate or malicious content. There is also a growing need for a model to forget certain tasks as the nature of the task or the underlying distribution of the data changes with time. Simultaneously, there is a growing interest in developing intelligent agents capable of adaption in dynamic settings by continually learning new tasks while retaining previously acquired knowledge. Integrating these two frameworks results in a model that can learn new tasks as well as selectively forget tasks based on user signals without having to be retrained from scratch. However, this statically sized model runs into the risk of overfitting when the number of tasks are few and the possibility of saturation as the training distribution becomes increasingly complex with more tasks. To address this challenge, we propose a framework that combines machine unlearning and continual learning, employing a dynamic model size that adjusts according to the number of tasks it manages.

1 Introduction

Deep generative models have made remarkable progress in recent years with exceptional quality of generated content. Unfortunately, these models can also be misused to create realistic-looking images of inappropriate content (Deepfakes). Beyond these concerns, a model operating in a dynamic environment would simply require some task to be unlearned due to irrelevance in the current setting. A naïve solution to the problem is to omit such harmful concepts from the training dataset but this is a challenging task, given that contemporary datasets can contain examples in the order of a billion. Additionally, this would necessitate retraining the model from scratch each time a concept has to be forgotten, which would cost a lot in compute and time.

A similar concern of "catastrophic forgetting" arises in the context of continual learning, where adaptation to a new distribution generally results in a largely reduced ability to capture the old ones. Essentially, both machine unlearning and continual learning deal with modifying model parameters to manage different tasks: in continual learning, we adjust parameters to add a new task B while retaining performance on an existing task A ($\theta_A \rightarrow \theta_{A,B}$), whereas in machine unlearning, the goal is to remove task B and revert to performing only task A ($\theta_{A,B} \rightarrow \theta_A$).

In this report, we present a unified approach to achieve both goals with an additional aspect of a dynamically sized model. This is done to 1) prevent the risk of overfitting on the dataset when the number of tasks are few and 2) capture the increased variation in the data distribution when the number of tasks increases. We focus our scope on Variational Autoencoders (VAEs) trained on the MNIST dataset. Our preliminary results show that our approach maintains performance comparable to a “simple" model trained exclusively for the tasks undertaken after all incremental learning and unlearning phases.

2 Background

2.1 Variational Autoencoders

Variational Auto Encoders are generative models that assume a prior ($p(z|\theta, c)$) over the latent variables z. The posterior ($p(z|\theta, x, c)$) generally being intractable is approximated by $q(z|\phi, x, c)$. The parameters ϕ are learned by maximizing the ELBO, given by

$$\log p(x|\theta, c) \geq \log p(x|\theta, z, c) + D_{KL}(q(z|\phi, x, c)||p(z|\theta, c)) = \text{ELBO}(x|\theta, c).$$

For generation we simply sample z from the prior ($p(z|\theta, c)$) and pass these samples to the decoder to get generated data as output.

2.2 Continual Learning

Continual learning assumes that a learning agent learns continuously over a sequence of tasks, generally without access to previous data when learning new tasks. The primary goal of continual learning is to learn consecutive tasks without forgetting the knowledge learned from earlier tasks, and leverage the previous knowledge to obtain better performance or faster convergence on the new tasks.

Assume K tasks, with the k^{th} task having a loss function ℓ^k, a training dataset D^k and a testing dataset D^k_{test}. Assume the agent adopts a model f_θ parameterized by $\theta \in \mathbb{R}^d$.

On learning task k, the agent loses its access to $D^{<k} = \{D^1, \ldots, D^{k-1}\}$. After learning all K tasks, the agent's objective is to achieve low loss on all test datasets $\{D^k_{\text{test}}\}_{k\in[K]}$. Denote the agent's model after learning task k as f_{θ^k}. Then, the overarching objective of a CL agent is to optimize

$$\min_{\theta^K} \frac{1}{K} \sum_{k\in[K]} \mathbb{E}_{(x,y)\sim D^k_{\text{test}}} \Big[\ell^k(x, y, f_{\theta^K}) \Big].$$

Here θ^K depends on all previous θ^k such that $k < K$. Therefore, this objective is hard to optimize directly. Hence, it is decomposed into K objectives and for the k^{th} task, we wish to achieve low loss on this task, while maintaining good accuracy on previously learned tasks.

This is also not straight-forward, and can lead to catastrophic forgetting of the previous tasks when learning new tasks. There are three mainstream approaches to tackle this problem. They are as follows:

1. *Regularization:* Regularization-based methods assume all information learned from $D^{<k}$ is in θ^{k-1}. Thus they aim to minimize the training loss on D^k while ensuring that θ stays close to θ^{k-1}:
$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim D^k} \Big[\ell(x, y, f_\theta) \Big] + \alpha D(\theta, \theta^{k-1}),$$
where α is a hyperparameter determining the strength of regularization and $D(\theta, \theta^{k-1})$ is a divergence measure that captures how similar θ is to θ^{k-1}.
One of the popular regularization techniques is **Elastic Weight Consolidation** (EWC). EWC approximates the posterior distribution of weights, after training a model θ^{k-1} on $D^{<k}$, by utilizing the Laplace approximation on the initial tasks $D^{<k}$, resulting in a regularization effect.
The posterior is given by $\log p(\theta|D^{<k}, D^k) = \log p(D^k|\theta) - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^{k-1})^2$, where F is the Fisher information matrix (FIM) and λ is a weighting parameter.
For variational models, we modify the F_i to measure the sensitivity of θ_i on the ELBO:
$F_i = \mathbb{E}_{p(x|\theta^*,c)p(c)}[(\frac{\partial}{\partial\theta_i}\text{ELBO}(x|\theta, c))^2]$.

2. *Memory Replay:* We make use of **Generative Replay**, a technique that involves using a generative model to produce data from past tasks, which is used to augment data from the current task to enhance the training of a discriminative model. This helps us to train on the previous datasets without needing to store those explicitly.
3. *Dynamic Architecture:* Static architectures may face the problem of scaling up when the network gets saturated if we begin with a small network, or over-fitting if we begin with a larger network. One approach to address this issue is dynamically expanding the network (**DEN**). The initial network is compact, and whenever the loss for a new task falls below a specified threshold, we can expand the network's capacity.

2.3 Machine Unlearning

Machine unlearning involves removing specific training data to eliminate their impact on a trained model. Approximate unlearning is an approach that bypasses the necessity of retraining the model entirely using the remaining data. Bayes' rule provides an elegant way to formulate unlearning as an inference problem.

Given some data for deletion, D_{del} and the current posterior $p(\theta|D_{del} \cup D_{ret})$, the goal is to find the posterior with respect to the retained data $p(D_{ret})$, i.e.

$$p(\theta|D_{ret}) \propto \frac{p(\theta|D_{del} \cup D_{ret})}{p(D_{del}|\theta)}$$

This update looks fairly simple, but there are a lot of challenges in obtaining a reasonable posterior after unlearning. First, we usually don't have access to the exact posterior of the model $p(\theta|D_{del} \cup D_{ret})$ due to inherent intractabilities and so we need to approximate the posterior. Second, as $p(D_{del}|\theta)$ approaches 0, the required posterior $p(D_{del}|\theta)$ destablises. These problems can even lead to catastrophic forgetting (forgetting of data we wish to retain). Similar problems are faced in continual learning which enables us to define selective forgetting from the perspective of continual learning.

In a typical continual learning scenario, the goal is to utilize task-specific parameters (θ_A) to train a model for a new task (B) while preserving the ability to perform task A independently. This involves transitioning the model's parameters (θ_A) to accommodate both tasks (A and B), denoted as $\theta_A \rightarrow \theta_{A,B}$. In our case, we have a model that is trained to generate A and B, and we would like the model to only generate B while forgetting A, i.e., $\theta_{A,B} \rightarrow \theta_B$. We will discuss methods to accomplish this goal.

2.4 Dropout

Dropout is a regularization technique commonly used in neural networks during training, where randomly selected neurons are ignored, or "dropped out" with a probability specified by the user. This prevents co-adaptation of neurons and effectively reduces overfitting by creating an ensemble of smaller networks during training. Some popular dropout techniques are as follows:

- *Weight dropout:* randomly drops weights at each training step.
- *Unit dropout:* randomly drops full neurons along with the associated weights at each training step.

We shall utilize the unit pruning method for our experiments.

2.5 Pruning

Magnitude-based pruning: These strategies use magnitude weights as the criteria to judge importance. Some popular magnitude-based pruning strategies:

- *Weight pruning:* We keep k weights in each layer having the largest L^2-norm.

$$\mathcal{W}(\boldsymbol{\theta}) = \left\{ \underset{1 \leq i \leq \text{rows}(\mathbf{W})}{\text{argmax-k}} \; |\mathbf{W}_{io}| \;\middle|\; 1 \leq o \leq \text{cols}(\mathbf{W}), \mathbf{W} \in \mathcal{W} \right\}$$

- *Unit pruning:* We keep k neurons in each layer having the largest L^2-norm of column vector comprising of incoming weights to the neuron.

$$\mathcal{W}(\boldsymbol{\theta}) = \left\{ \underset{\substack{\mathbf{w}_o \\ 1 \leq o \leq \text{Ncol}(\mathbf{W})}}{\text{argmax-k}} \; \|\mathbf{w}_o\|_2 \,\middle|\, \mathbf{W} \in \text{weights} \right\}$$

3 Literature Review

3.1 Selective Amnesia: A Continual Learning Approach to Forgetting in Deep Generative Models [1]

The paper builds a technique for forgetting by leveraging continual learning techniques of generative replay and elastic weight consolidation to prevent "catastrophic forgetting".

We partition the dataset D as $D = D_f \cup D_r = \{(x_f^{(n)}, c_f^{(n)})\}_{n=1}^{N_f} \cup \{(x_r^{(m)}, c_r^{(m)})\}_{m=1}^{N_r}$, where D_f and D_r correspond to the data to forget and remember respectively. D is assumed to be generated from the underlying distribution given by $p(x, c) = p(x|c)p(c)$. The distribution over concepts/class labels is defined as $p(c) = \sum_{i \in f,r} \phi_i p_i(c)$ where $\sum_{i \in f,r} \phi_i = 1$. The two concept/class distributions are disjoint such that $p_f(c_r) = 0$ where $c_r \sim p_r(c)$ and vice-versa. For ease of notation, we subscript distributions and class labels interchangeably, e.g., $p_f(c)$ and $p(c_f)$.

We have a trained conditional generative model with parameters θ^*, obtained by maximizing the expected log-likelihood over the joint distribution $p(x, c)$. This maximizes the likelihood of the dataset D. Our goal is to retrain this model in such a way that it retains the ability to generate samples similar to those in D_r for a given condition c_r, while minimizing its ability to generate samples similar to those in D_f for a given condition c_f. Importantly, this retraining process should be conducted without access to the original datasets D. This scenario reflects a common situation where only the model itself is available, not its training data.

The posterior distribution can be represented as

$$\begin{aligned} \log p(\theta|D_f, D_r) &= \log p(D_f|\theta, D_r) + \log p(\theta|D_r) - \log p(D_f|D_r) \\ &= \log p(D_f|\theta) + \log p(\theta|D_r) + C. \end{aligned}$$

On applying EWC we get

$$\begin{aligned} \log p(\theta|D_r) &= -\log p(D_f|\theta) + \log p(\theta|D_f, D_r) + C \\ &= -\log p(x_f|\theta, c_f) - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^*)^2 + C \end{aligned}$$

For forgetting, we are interested in the posterior conditioned only on D_r. Generative replay term is incorporated in the objective to enhance the performance on D_r, the obective becomes

$$\log p(\theta|D_r) = \frac{1}{2}\left[-\log p(x_f|\theta, c_f) - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^*)^2 + \log p(x_r|\theta, c_r) + \log p(\theta)\right] + C.$$

With uniform prior and expectations written down explicitly the objective further reduces to

$$\mathcal{L} = -\mathbb{E}_{p(x|c)p_f(c)}\left[\log p(x|\theta, c)\right] - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^*)^2 + \mathbb{E}_{p(x|c)p_r(c)}\left[\log p(x|\theta, c)\right].$$

Maximizing the likelihood can be achieved by maximizing the Evidence Lower Bound (ELBO). However, in our scenario, where we aim to minimize the likelihood of data being unlearned(D_f) to facilitate unlearning, minimizing the ELBO does not necessarily ensure this.

Consider a surrogate distribution $q(x|c)$ such that $q(x|c_f) \neq p(x|c_f)$. Assume we have access to the MLE optimum for the full dataset $\theta^* = \text{argmax}_\theta \mathbb{E}_{p(x,c)}\left[\log p(x|\theta, c)\right]$ such that $\mathbb{E}_{p(c)}\left[D_{KL}(p(x|c)||p(x|\theta^*, c))\right] = 0$. Define the MLE optimum over the surrogate dataset as $\theta^q = \text{argmax}_\theta \mathbb{E}_{q(x|c)p_f(c)}\left[\log p(x|\theta, c)\right]$. Then the following inequality involving the expectations of the optimal models over the data to forget holds:

$$\mathbb{E}_{p(x|c)p_f(c)}\left[\log p(x|\theta^q, c)\right] \leq \mathbb{E}_{p(x|c)p_f(c)}\left[\log p(x|\theta^*, c)\right].$$

The decrease in the value of log-likelihood is given by

$$\mathbb{E}_{p(x|c)p_f(c)}\left[\log p(x|\theta^q, c) - \log p(x|\theta^*, c)\right] = -\mathbb{E}_{p_f(c)}\left[D_{KL}(p(x|c)||q(x|c))\right].$$

The above equation implies that the greater the difference between the distributions the greater the decrease in the value of the log-likelihood. Finally by replacing the likelihood terms with ELBO we get the objective function of the paper as

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(x|c)p_f(c)}\left[\log p(x|\theta, c)\right] - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^*)^2 + \mathbb{E}_{p(x|c)p_r(c)}\left[\log p(x|\theta, c)\right] \\ &\geq \mathbb{E}_{q(x|c)p_f(c)}\left[\text{ELBO}(x|\theta, c)\right] - \lambda \sum_i \frac{F_i}{2}(\theta_i - \theta_i^*)^2 + \mathbb{E}_{p(x|c)p_r(c)}\left[\text{ELBO}(x|\theta, c)\right]\end{aligned}$$

The final objective function applies EWC and generative replay for data retention and simultaneously maps the data D_r to a surrogate distribution thus lowering its likelihood and facilitating forgetting.

3.2 Continual Learning and Private Unlearning[2]

This paper formalizes the continual learning and private unlearning problem and introduces a straightforward solution that achieves exact unlearning by construction.

In their framework, the queries ρ_t are subdivided into 3 types of learning instructions:

1. **R**: Learn task I_t permanently.
2. **T**: Temporarily learn task I_t, which can be forgotten in the future.
3. **F**: Forget task I_t with exact unlearning.

A dictionary Ψ^t of the learnt tasks' statuses is maintained.

For each task with $\rho_t = \mathbf{T}$, an isolated temporary network with parameters $\hat{\theta}$ is created. Based on the subsequent learning instruction for the same task, the agent either removes this isolated model (**F**) or merges it with the main model (**R**).

The following four cases arise and are evaluated accordingly:

1. If $\rho_t = \mathbf{R}$ and $I_t \notin \Psi^{t-1}$, then this is the first time the task I_t is observed, so the agent then performs conventional CL using Dark Experience Replay++ and updates the main model parameters to θ^t_{main}.
2. If $\rho_t = \mathbf{T}$, in order to benefit from prior learning experience, the agent initializes an isolated model with parameters $\hat{\theta}$ copied from θ^{t-1}_{main} , then directly performs SGD update.
3. If $\rho_t = \mathbf{R}$ and $I_t \in \Psi^{t-1}$, then task I_t has been learnt previously with a temporary network θ^{I_t}, so the knowledge learned in θ^{I_t} is merged into θ^t_{main} using knowledge distillation.
4. If $\rho_t = \mathbf{F}$, the temporary network θ^{I_t} is simply removed.

This method had the following limitations:

- The model lacks flexibility due to its restricted queries. Users cannot forget a data if they have learnt it using the R instruction.
- A new temporary network is generated for each task, leading to significant growth in model size with more tasks, which is not efficient.

3.3 Regularize, Expand and Compress: NonExpansive Continual Learning[3]

REC is a unified framework which explores model expansion and compression in continual learning setting. The final model is non-expansive but the performance is enhanced by network expanding procedure.

The key highlights are:

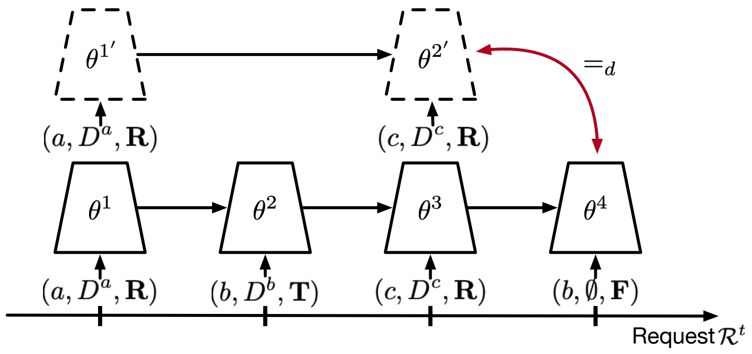
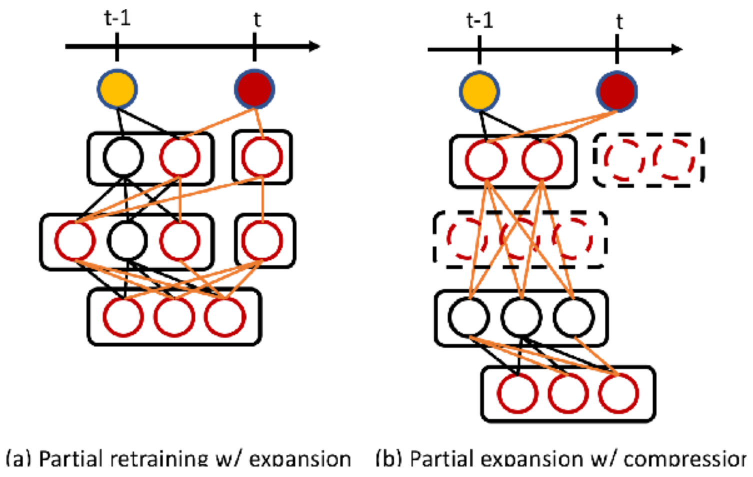


Figure 1: An illustration of the Continual Learning and private unlearning (CLPU) [2] problem setting. After the agent has temporarily learned on task b with data D^b, if the agent is later requested to unlearn task b, the unlearned model parameters θ^4 should be indistinguishable from $\theta^{2'}$ in distribution as if the agent has never learned on task b.

- A regularized weight consolidation algorithm to avoid forgetting. In addition to EWC, it uses the $l_{2,1}$ - norm regularization to capture the common subset of relevant parameters for task $t-1$ and task t and the l_1 sparse norm is imposed to learn the new task-specific parameters.
- An automatic neural architecture search (AutoML) engine to expand the network to increase model capability. After learning the network θ^{t-1} on the data $D^{<t}$, it automatically searches the best child network θ^t for task t among all the generated children networks $\theta^t_1, ..., \theta^t_m$
- Compression of the expanded model to the size of the initial model after a new task is learned. Knowledge distillation helps in accomplishing this task. The soft-labels (the logits) are used as knowledge distillation to train the student model.

This technique is rather intricate and demands computational power. Therefore, we opt for a more simplified and innovative approach to expand and compress our model.



t-1
t
t-1
t
(a) Partial retraining w/ expansion
(b) Partial expansion w/ compressior

Figure 2: Comparison between (a) DEN and (b) REC [3]

3.4 Dynamically Expandable Networks[4]

Dynamically Expandable Networks (DEN) can be used to dynamically decide the network capacity after a learning task.

The following steps describe the expansion algorithm:

1. After learning the t-th task through selective retraining - retraining only the weights that are affected by the new task, we assess whether the model fails to obtain the desired loss ($\mathcal{L}_t$) below a set threshold (τ).

2. If this is the case, we expand the network capacity in a top-down manner, while eliminating any unnecessary neurons using group-sparsity regularization.
3. DEN calculates the drift ρ_t, as the ℓ_2-distance between the incoming weights at $t-1$ and at t, for each unit to identify units that have drifted too much from their original values during training and duplicate them.
4. After duplication, retrain the network. In practice, this secondary training usually converges fast due to the reasonable parameter initialization from the initial training.

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

```
Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_T)$, Thresholds $\tau$, $\sigma$
Output: $\mathbb{W}^T$
for $t = 1, \dots, T$ do
  if $t = 1$ then
    Train the network weights $\mathbb{W}^1$
  else
    $\mathbb{W}^t = SelectiveRetraining(\mathbb{W}^{t-1})$
    if $\mathcal{L}_t > \tau$ then
      $\mathbb{W}^t = DynamicExpansion(\mathbb{W}^t)$
    $\mathbb{W}^t = Split(\mathbb{W}^t)$
```

The three main components of this algorithm are:

Selective Retraining: The initial model is trained with ℓ_1 regularization to promote sparsity in the weights. Throughout the incremental learning procedure, $\mathbb{W}^{t-1}$ is maintained sparse to focus on the subnetwork connected new task.

To learn the t^{th} a sparse linear model is fit to predict task t using topmost hidden units of the neural network via solving the following problem:

$$\min_{\mathbb{W}^t_{L,t}} \mathcal{L}(\mathbb{W}^t_{L,t}; \mathbb{W}^{t-1}_{1:L-1}, \mathcal{D}_t) + \mu \|\mathbb{W}^t_{L,t}\|_1$$

where where $1 \leq l \leq L$ denotes the l_{th} layer of the network and $\mathbb{W}^{t-1}_{1:L-1}$ denotes the set of all other parameters except $\mathbb{W}^t_{L,t}$.

Dynamic Expansion: The capacity of each layer is expanded by k neurons and group sparsity regularization is used to drop the unnecessary hidden units.

Network Split: If the semantic drift ρ_t of a neuron exceeds the set threshold (σ), it means the this neuron has changed significantly. Such neurons are split into two copies and the overall architecture is changed. After this duplication, the network is retrained. This retraining is however faster due to reasonable parameter initialization.

DENs are useful in expanding the network capacity when needed and can be easily integrated in our setting simply by modifying the *Selective Retraining* step. We will discuss this later as one of our contribution.

4 Our Contribution

4.1 Combined Learning and Unlearning

We work with generative models that generate images. During unlearning we supply the model with corrupt/noisy data corresponding to the task being unlearned. This encourages the model to modify the learned distribution towards the noisy distribution from which the corrupt images were generated. After training, the task to be forgotten will be mapped to the corrupt distribution.This is same as discussed earlier in the selective amnesia approach.

Similarly, to retain the previous tasks, we just supply images from the dataset corresponding to the label being learned. This enables the model to learn the distribution corresponding to the label.

The above steps are combined with continual learning technique of generative replay which preserve the performance on the labels the model is required to retain.

4.2 Model Adaptation

We begin with a base model, and as we learn or unlearn, the model size dynamically adjusts to maintain flexibility and expressiveness.

If the model undergoes a process of unlearning certain data categories, it should subsequently demand less computational resources for representation. Hence, in such instances, we compress the model.

We scale up when network capacity gets saturated with more and more new tasks to learn.

We have used innovative and elegant resizing policies to incorporate this idea.

4.2.1 Model Compression

During training we are implementing targeted dropout [5] to make the final model sparse and then pruning the unimportant neurons to yield a compressed model. The algorithm for compression is as follows:

1. We calculate the importance of each neuron by calculating the L2 norm of the column vector containing the incoming weights for that neuron.
2. Then we select bottom γ percent neurons as the candidates for dropout and dropout them with a drop probability α.
3. The above steps are repeated in each epoch.
4. Finally trained model is pruned using unit pruning to keep only the top k percent neurons in each layer.

Targeted dropout induces sparsity and independence among neurons of the same layer, thus enabling compression through pruning without much loss in performance. We can determine the dependence of the model on weights/units being pruned by calculating pruning loss defined as the second degree Taylor expansion of change in loss, $\Delta\mathcal{E} = |\mathcal{E}(\theta - d) - \mathcal{E}(\theta)|$:

$$\Delta\mathcal{E} = | - \nabla_{\theta}\mathcal{E}^{\top} d + \frac{1}{2}\, d^{\top} H d + \mathcal{O}(\|d|^{3})|$$

Where $d_i = \theta_i$ if $\theta_i \in \overline{\mathcal{W}(\boldsymbol{\theta})}$ (the weights to be removed) and 0 otherwise. $\nabla_{\boldsymbol{\theta}}\mathcal{E}$ are the gradients of the loss, and H is the Hessian.

4.2.2 Model Expansion

We have used a simplified idea derived from Dynamically Expandable Networks (DEN) to increase the network capacity. The following steps describe our expansion algorithm:

1. Before learning a new task, we first expand our model to enhance it's representation power.
2. For expansion, we firstly increase the number of neurons in each layer by $k\%$ (a hyperparameter).
3. We then copy the initial weights of the previous model. The other weights are randomly initialized with small values.
4. Then we carry out the continual learning training process as described before.

5 Experiments

5.1 Experimental Setup

We are using a Conditional VAE model, trained to generate the digits similar to the MNIST dataset. To test our model, first we carry out random passes of length 10. At each pass, we randomly decide to learn or forget. For each case we select a random set of numbers that needs to learnt or forgotten (ensuring that the learnt labels are distinct from the current ones and the forget labels labels are a subset of the current ones respectively). We currently choose the number of epochs according to number of labels to retain in the model.For metrics, we choose to train a "simple" model that is directly trained on the labels that are left in the model at the end of all passes. Then we compare the accuracy of this model with ours. Shown in the upcoming slides.We also plot the average accuracy of our model after each pass on all the labels it retains.

5.2 Results

5.2.1 Comparison with "Simple" Model

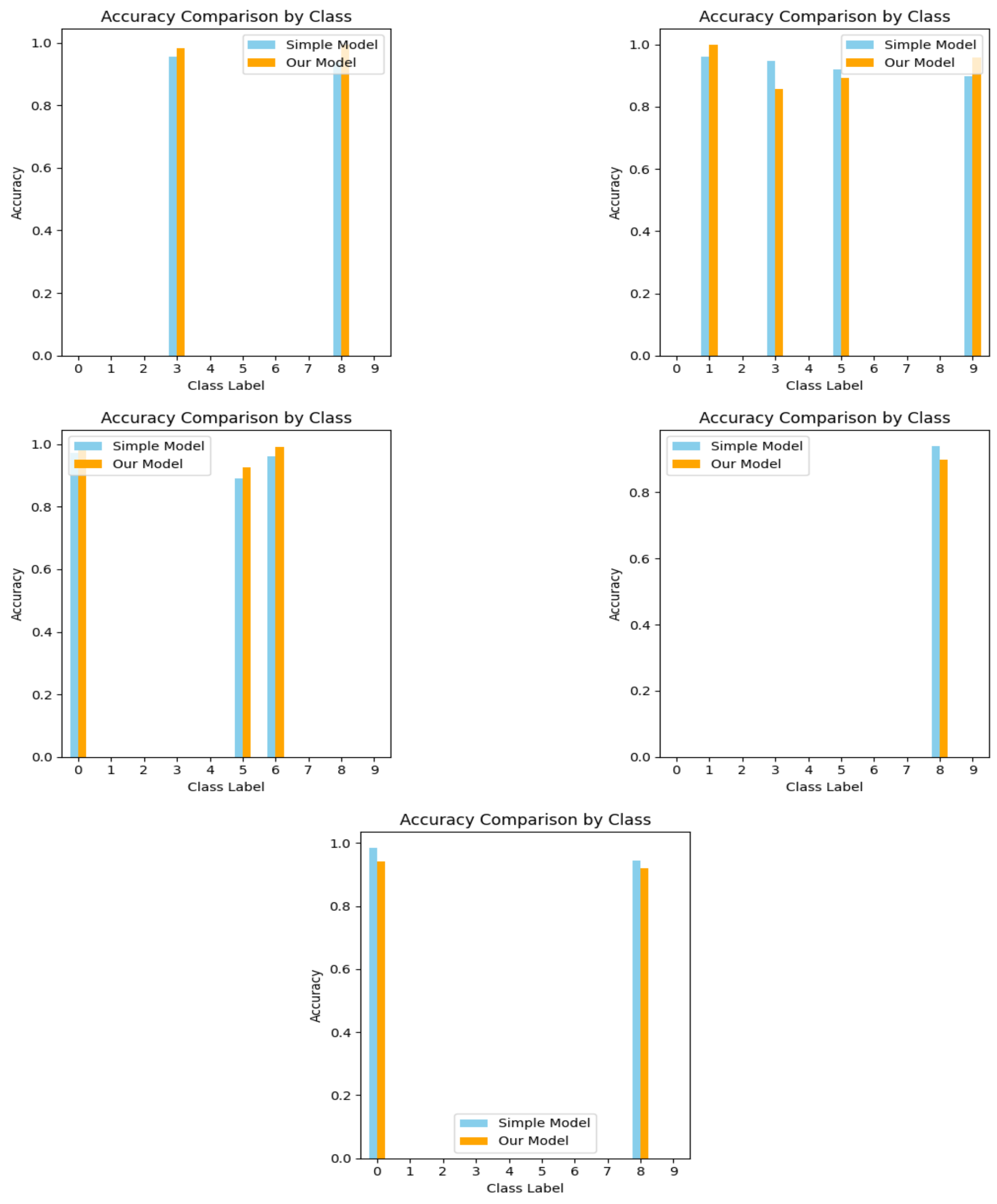


Figure 3: The figures above show the results of our model after 10 random passes between continual learning and unlearning with a “simple" model that was directly trained on the labels that were left at the end. We see that the results of our model are more or less the same as that of the “simple" model.

5.2.2 Accuracy Plots vs Time

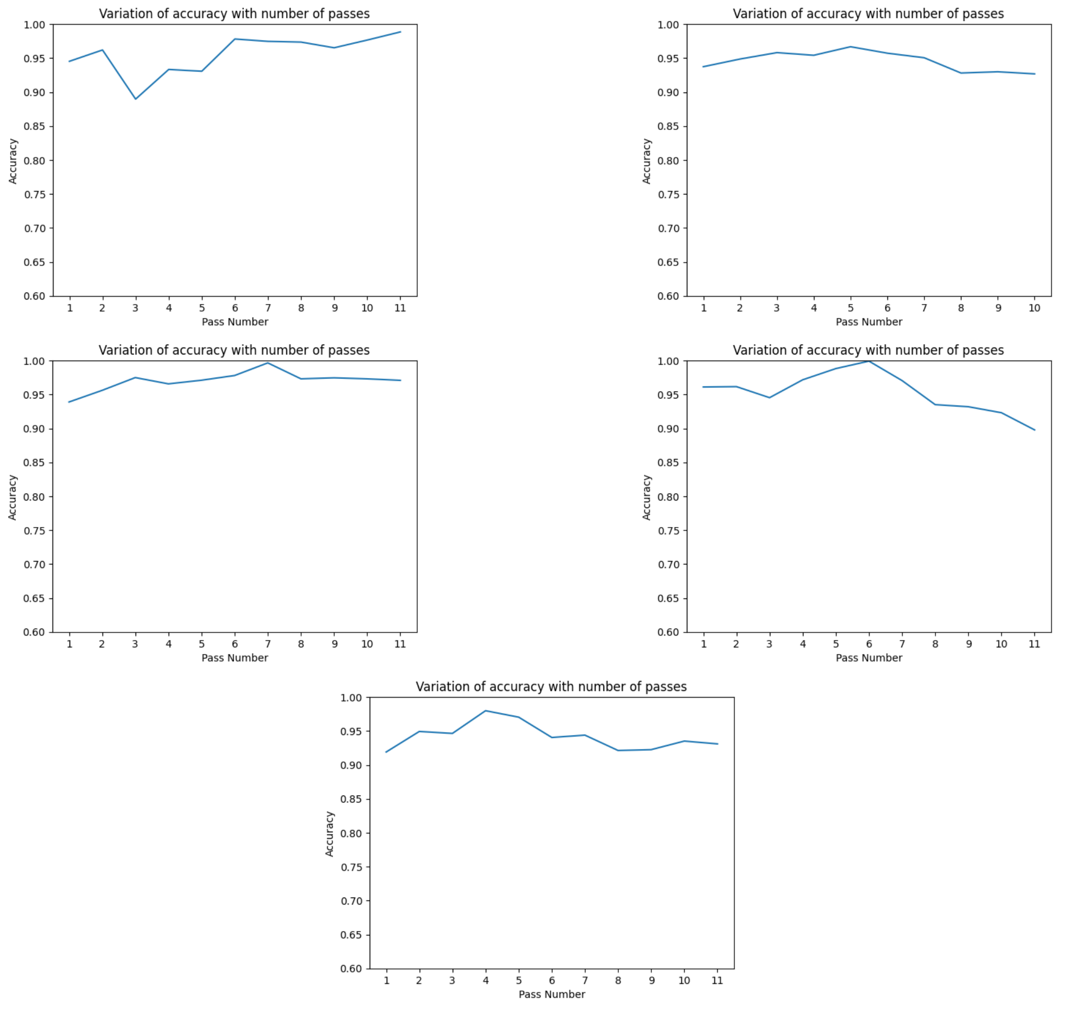


Figure 4: The above figures show the average accuracy of all the labels that are retained in the model after every pass. The accuray is well maintained above 90% in most cases.

5.2.3 Performance Comparison with Model Resizing while Forgetting

Runs	**SA (%)**	**Ours (%)**
Run 1	95.25	96.19
Run 2	95.22	94.86
Run 3	92.84	95.61
Run 4	93.98	96.16

Table 1: Average Accuracy Readings for the original method in Selective-Amnesia Paper vs Our model which encorporates model resizing.

The above readings are taken from random initial digits and forgetting random digits from the model. Values from 4 runs are taken. As seen in the above table, our model not only gives better accuracy on average but is more efficient and should theoretically take less training time as well, although it's yet to be tested.

6 Learnings

Through this project, we were able to explore the paradigms of machine unlearning as well as continual learning, enriching our understanding with the cutting edge practices in these areas through our literature review (generative replay, EWC and DEN being the noteworthy mentions).

Additionally, we gained hands-on experience in working with a VAE model, a deep-generative model whose concept and design was discussed during the coursework. This project also enhanced our proficiency with Python libraries commonly used in machine learning and supporting tasks.

The process of translating theoretical concepts into practical implementations was an engaging aspect of the work. Beyond these, it was a novel experience to come up with something new by creatively combining existing practices and building upon of some of the most recent work in the domain.

7 Conclusions and Future Work

With this framework it becomes possible for models to be flexible in terms of their expressibility with a varying number of tasks. This allows machine learning models to be deployed in dynamic environments where the underlying data distributions pertaining to the task may change however the model can autonomously adapt to the new setting based on the user input. It also reduces cost since predictions are performed using a smaller model in case the task becomes simpler.

At the time of this report, we used a simple metric of contracting/expanding the network by a (fixed) percentage of neurons in the current network. This can be refined to more aware policies that take into account the number of tasks being forgotten/learned and the complexity of the distribution, thus working on building the most optimally sized model after each forgetting/continual training pass. To this effect, incorporating DENs into the framework is a very clear extension to the work presented.

References

[1] Alvin Heng and Harold Soh. Selective amnesia: A continual learning approach to forgetting in deep generative models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[2] Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning. In Sarath Chandar, Razvan Pascanu, and Doina Precup, editors, *Proceedings of The 1st Conference on Lifelong Learning Agents*, volume 199 of *Proceedings of Machine Learning Research*, pages 243–254. PMLR, 22–24 Aug 2022.

[3] Jie Zhang, Junting Zhang, Shalini Ghosh, Dawei Li, Jingwen Zhu, Heming Zhang, and Yalin Wang. Regularize, expand and compress: Nonexpansive continual learning. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 843–851, 2020.

[4] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018.

[5] Aidan N. Gomez, Ivan Zhang, Siddhartha Rao Kamalakara, Divyam Madaan, Kevin Swersky, Yarin Gal, and Geoffrey E. Hinton. Learning sparse networks using targeted dropout, 2019.