Generative Latent Replay for Continual Learning

Jacob Armstrong
Institute of Biomedical Engineering
Oxford University
jacob.armstrong@eng.ox.ac.uk

Anshul Thakur
Institute of Biomedical Engineering
Oxford University
anshul.thakur@eng.ox.ac.uk

David A. Clifton
Institute of Biomedical Engineering
Oxford University
davidc@robots.ox.ac.uk

***Abstract*—Catastrophic forgetting is a pervasive problem in machine learning, where standard models are incapable of adapting to new domains without losing performance on prior ones. Many continual learning methods have been proposed in recent years to tackle this problem, however the most effective methods are generally replay based, which violate one of the core desiderata of continual learning: that past data is not stored. Generative replay has been proposed as an alternative, however building accurate generative models for complex data is computationally prohibitive. Here we introduce Generative Latent Replay, where a model's backbone is frozen after initial training as a feature extractor and generative models are trained on this condensed latent space for replay from each domain. We compare against a range of continual learning methods and show that Generative Latent Replay is a viable choice for data-free scenarios.**

***Index Terms*—Continual learning, domain adaptation, machine learning**

I. Introduction

Continual learning, the ability to learn and adapt to new tasks without forgetting previously learned knowledge, is a crucial requirement for artificial intelligence systems operating in dynamic environments. However, traditional machine learning models often suffer from catastrophic forgetting, where performance on previous tasks deteriorates significantly when learning a new task. This limitation has motivated the development of various continual learning methods, such as rehearsal and regularization techniques, to mitigate the negative impact of forgetting.

While many elegant continual learning methods have been proposed, most regularisation based methods struggle to improve model performance on the challenging Domain Incremental subset of problems [1]. These are perhaps the

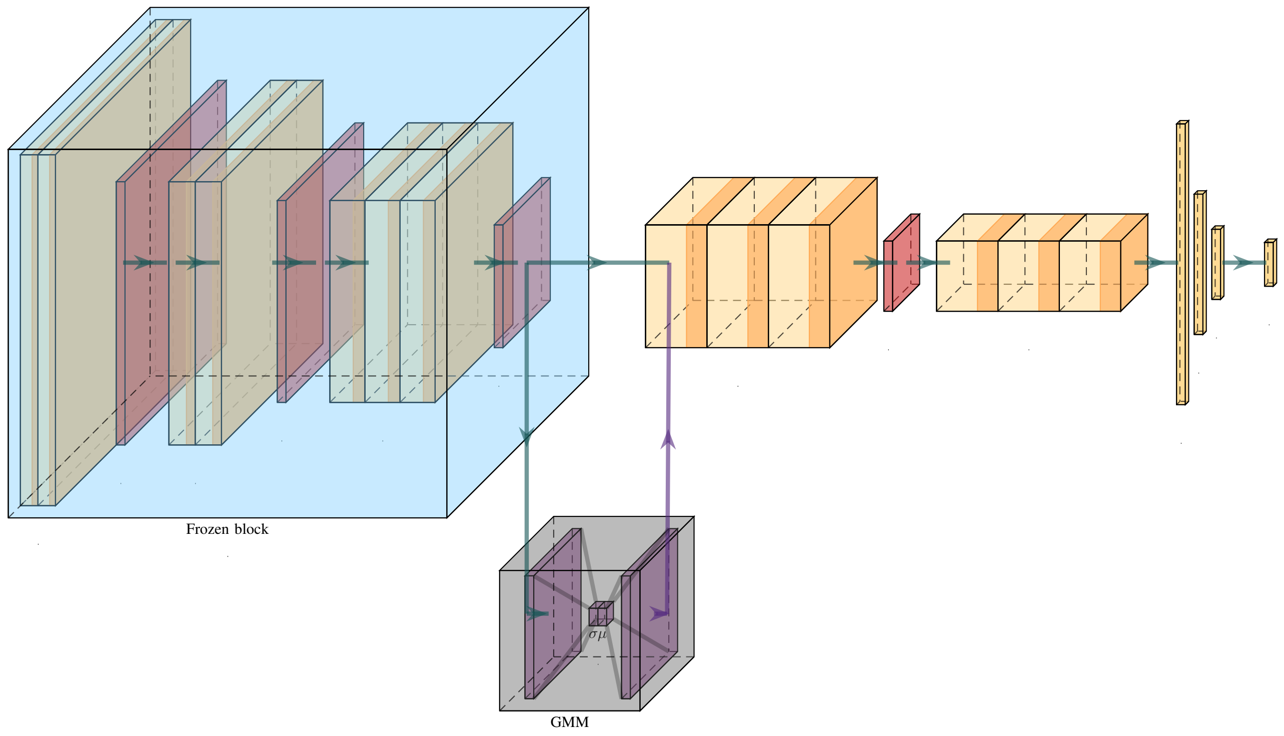

Fig. 1: Illustration of the Generative Latent Replay model. Orange boxes refer to convolutional blocks in the base model. Red blocks refer to pooling layers. Orange columns refer to feedforward layers. Green arrows depict the transfer of data through the network. Purple arrows depict the injection of sampled synthetic data from the latent-space generative model (here represented as a GMM in the grey box). The blue box indicates the portion of the model that is frozen after initial training. Activation functions etc are omitted for readability. Practically the level of freezing can be adjusted as necessary, with the requirement that the generator input dimensionality is congruent.

most natural continual learning scenario in many fields, where a model has a nominally static task to solve, but the distribution of input features it is exposed to varies over time (such as an autonomous vehicle experiencing different physical environments, or a medical diagnosis model encountering different patient populations).

Replay based methods have shown consistently high performance in domain incremental benchmarks [2], but may be inapplicable in many real world scenarios since they require a (potentially unbounded) storage of past examples. Methods have been proposed which reduce the number of required exemplars per domain encountered [3], but in scenarios where the limitation is not space or computational cost, but legal or ethical restrictions on the storage and sharing of the data itself, these methods remain inapplicable.

Generative replay [4] has been proposed as an alternative. By generating synthetic data from past experiences, these methods can effectively "replay" previously encountered tasks for the model to revisit and learn from, without the need to store real data between training experiences. However training of accurate generative models carries considerable computational and time expense, and may be infeasible for complex non-image based data.

In this paper, we propose Generative Latent Replay (GLR), a novel continual learning method which leverages the power of generative models to enable replay of latent sampled data points from past experiences. GLR comprises two components: a frozen backbone network, which is trained on the initial experience, and a generative model, which is trained on the output of data from each subsequent experience through the frozen backbone. The generative model is able to "replay" latent samples from past experiences during current training on the fly, effectively allowing the model to revisit and learn from previously encountered tasks.

We demonstrate the effectiveness of GLR on a variety of domain incremental continual learning benchmarks, and show that it significantly outperforms state-of-the-art data-free methods, making it a promising choice in scenarios where storage of explicit replays is impractical.

Contributions: Our contributions are threefold: 1) we introduce GLR as a novel approach to continual learning, 2) we provide a thorough analysis of the method's performance and ablation studies, and 3) we release the code and trained models for reproducibility and further research (available at https://github.com/iacobo/generative-latent-replay).

Related work: [5] introduce latent replay, and [4] introduce generative replay. We combine these methods but introduce a wider variety of generative models and explore computational caveats and fidelity of the generated data. [6] independently proposed a similar model, but applied to task-incremental problems using a Generative Adversarial Network (GAN) as the generative component, focusing on facial recognition problems. We explore the more challenging (for regularisation based methods) domain incremental subset of continual learning problems, using a GMM as the main generator in our experiments.

Method	**Replay**	**Frozen Backbone**
Naive		
Replay	True	
Generative Replay	Generated	
Latent Replay	True	✓
Generative Latent Replay	Generated	✓

TABLE I: Comparison of replay and latent based methods in terms of functional components.

II. Methods

A. Overview

The driving principle behind GLR is that shallow layers of a neural network typically learn low-level representations of data which are theoretically invariant to domain and covariate shift (for example, edge and shadow detection in images is presumed to be invariant of the subject of the image e.g. images of cats vs images of dogs).

Given these, we can train the model end-to-end on the primary experience in the traditional supervised learning setup. We may then fix the parameters of a sufficiently deep 'backbone' of the network to act as a frozen feature extractor on the latent feature space (i.e. freeze the model's backbone). We may then train a generative model on the latent representations of each dataset as encoded by this backbone. Latent samples may then be generated from the generator on the fly during later experiences for replay to balance current training with training on synthesized prior tasks.

The frozen section of the model remains static acting like a pre-processing step of feature extraction, only the subsequent layers of the network are trained on subsequent steps.

We provide an explicit description in Algorithm 1.

Several broad assumptions are made during training which underly the method's feasibility, and while these are 'soft' considerations they should be considered before exploring such methods over existing continual learning strategies:

1) The initial dataset is broad enough to train a generally useful feature extractor for this datatype independent of covariate shift. If the initial dataset is not large enough, a pretrained model for this type of data may be used.
2) the bottleneck for the frozen layer is sufficiently compressed such that a generator can be trained in reasonable time.

B. Method description

GLR consists of two components: a neural network (itself decomposed into a freezable backbone, and classification head) and a generative model (see Fig. 1). The backbone is trained on the initial experience, while the generative model is trained on the output of data from each experience through the frozen backbone. The generative model is then used to "replay" latent samples from past experiences during current training of the classification head, effectively allowing the model to revisit and learn from previously encountered tasks.

The skeleton for the neural network may be any standard neural network with the requirement that there are no skip connections between the layer delineating the frozen backbone

Algorithm 1 Generative Latent Replay (GLR)

```
Require:
    Datasets $D_1, D_2, \ldots, D_n$
    Trainable neural network $f(x) := f_{\text{classify}}(f_{\text{core}}(x))$
    Memory constraint $M$
    Trainable generative model $p(z)$
 1: Train neural network $f()$ on dataset $D_1$
 2: Freeze parameters of core layers $f_{\text{core}}()$
 3: for $i = 1$ to $n$ do
 4:     Extract feature vectors $Z_i = \{f_{\text{core}}(x_i) | (x_i, y_i) \in D_i\}$
 5:     Train $p(z)$ on feature vectors $z_i \in Z_i$ and their
        corresponding labels $y_i$ to obtain trained generator
        $p_i(z)$
 6:     if $i < n$ then
 7:         Draw $\lfloor \frac{M}{i} \rfloor$ latent samples $Z_j^* = \{(z_j^*, y_j^*)\}$ from
            each generator $p_j(z)$ to obtain latent replay buffer
            $Z^* = \bigcup_{j=1}^{i} Z_j^*$
 8:         Pre-process new dataset $D_{i+1}$ with frozen layers
            $f_{\text{core}}()$ to obtain features $Z_{i+1}$.
 9:         Augment $Z_{i+1}$ with latent samples $Z^*$ to obtain
            combined dataset $\boldsymbol{Z_{i+1}} = Z_{i+1} \cup Z^*$
10:         Fine-tune classification layers $f_{\text{classify}}()$ on $\boldsymbol{Z_{i+1}}$
11:     end if
12: end for
```

and classification head. In the event this is not possible, further classification layers may be prepended to the output layer.

More explicitly, let $D_1 = \{(x_i, y_i) | i \in \{1, ..., N_1\}\}$ be the training dataset for the initial experience, where x_i is the input data and y_i is the corresponding label. We first train a neural network $f(x) := f_{\text{classify}}(f_{\text{core}}(x))$ to minimize the loss[1] on the initial dataset D_1:

$$\mathcal{L}_1 = -\frac{1}{N_1} \sum_{i=1}^{N_1} y_i \log f(x_i)$$

Once the training is completed, we freeze the first K layers of the network (f_{core}), which will act as the frozen backbone for all subsequent learning experiences. We then use the output of the frozen backbone network, in the form of feature vectors $Z_1 = \{f_{\text{core}}(x_i)\}_{i=1}^{N_1}$, to train a generative model $p_1(z)$. This generative model is able to generate latent samples $z_i^* \sim p_1(z)$ that are similar to the feature vectors from past experiences.

For each subsequent learning experience, we collect a dataset $D_i = \{(x_j, y_j)\}_{j=1}^{N_i}$, where x_j is the input data and y_j is the corresponding label.

We then draw a set of latent samples $Z_j^* = \{(z_k^*, y_k^*)\}_{k=1}^{\lfloor \frac{M}{i} \rfloor}$ from each of the previously trained generative models $p_1(z), \ldots, p_i(z)$, and combine these into a latent replay buffer $Z^* = \bigcup_{j=1}^{i-1} Z_j^*$. We then pre-process the current dataset D_i using the frozen layers of the network to obtain a set of feature vectors $Z_i = \{f_{\text{core}}(x_i) | x_i \in D_i\}$, and augment this with the latent samples in the replay buffer to obtain the combined dataset $\boldsymbol{Z_i} = Z_i \cup Z^*$. Finally, we fine-tune the classification layers of the network $f_{\text{classify}}(x)$ on the combined dataset $\boldsymbol{Z_i}$ to minimize the loss:

$$\mathcal{L} = \mathcal{L}_i + \lambda \sum_{j=1}^{i-1} \mathcal{L}_j^*$$

where

$$\mathcal{L}_i = -\frac{1}{N_i} \sum_{(x_k, y_k) \in D_i} y_k \log(f(x_k))$$

is the cross-entropy loss over the current dataset, and

$$\mathcal{L}_j^* = -\frac{1}{\lfloor \frac{M}{i} \rfloor} \sum_{(z_k^*, y_k^*) \in Z_j^*} y_k^* \log(f_{\text{classify}}(z_k^*))$$

is the cross-entropy loss for each of the latent buffers. This loss is minimized over the parameters of the classification layer $f_{\text{classify}}()$, and the hyperparameter λ controls the trade-off between these two loss terms. Setting $\lambda = i - 1$ balances the loss at a given step evenly between all past and current tasks (see Figure 3).

This fine-tuning step forces the classification layers of the model to balance maintaining performance on (synthetic) previous tasks while adapting to the current one, as in Replay and Latent Replay.

C. Choice of generator

The choice of generator is an important consideration in GLR. While recent neural network–based generators, such as Generative Adversarial Networks (GANs) and invertible flow networks, have shown success in certain domains, they may be infeasible in many scenarios. In particular, when working with more complex datasets such models can be computationally expensive and time-consuming to train.

In light of this, we opted to use a simpler Gaussian Mixture Models (GMM) based generator in GLR. GMM's are less computationally expensive and can be trained more quickly, making them a preferable choice for more complex datasets or low volume datasets, and are hence a better indicator of scalability of the method. The reduced complexity of the dataset as exported by the latent layer of the model also favours the use of a simpler model.

Note that while GMM's have been shown to perform well even in scenarios where the underlying assumptions of the method are broken (i.e. the data being modeled is not drawn from a mixture of gaussians), for exceptionally complex data it may perform inadequately, and a more flexible model may be preferable. Ultimately the choice will be down to which method best balances the quality of generated data, training time, and computational and memory cost.

Note that there is much leeway in the details of implementation of the generative component. The generator may be specified to generate a feature target pair, as a conditional model generating samples given a label as input, or the version of the classification head of the model at each training experience may be used to bootstrap target labels.

[1]Without loss of generality, we use the cross entropy loss in our description.

III. Experiments

A. Benchmark experiments

In this paper we explored two classic domain incremental continual learning benchmark experiments, Permuted MNIST and Rotated MNIST, and compared the performance of GLR to a range of extant continual learning methods including Replay, Latent Replay, and Elastic Weight Consolidation (EWC). See Figure 2 for example images from the experimental datasets.

All experiments are implemented in the standard domain incremental setup, i.e. the model is trained and evaluated on a series of nominally identical classification tasks, where the general data structure remains the same but the distribution of features is altered between training experiences.

Permuted MNIST: This is a modification of the traditional MNIST classification problem [7]. It is a 10-class classification problem, where the input data are images of handwritten digits and the labels are the numbers 0 to 9, but in this variant the pixels of the images are randomly permuted between tasks. The labels remain the same.

Rotated MNIST: Another modification of MNIST. In this, images are randomly rotated an angle between 0 and 360 degrees between tasks. The angle of rotation is fixed within tasks.

Full details of the datasets and the training and test data splits can be found in [8].

B. Additional experiments

In addition to these two experiments, we also conducted several additional experiments analysing the effect of varying sub-model choices of GLR. For simplicity, the following are all evaluated only on the Rotated MNIST experiment:

- *Replay buffer size:* GLR performance was compared using models with different replay buffer sizes corresponding to 1%, 10%, 25%, and 50% of the size of the current dataset (60000 exemplars).

⚠Section not yet complete
- *Freeze depth:* We compare freezing different backbone depths of 3, 4, and 5 CNN layers post-initial training.
- *Generator:* We compare the performance of GLR using i) a GMM vs ii) a simple feedforward based autoencoder for the generative component.

C. Experimental Setup

Model definition: In our experiments, we used a simple CNN architecture with feedforward classification layers, as illustrated in Figure 1. Our model consists of a backbone of 5 convolutional layers based off of AlexNet (with weights initialised as in the pretrained version over Imagenet in the PyTorch implementation), followed by a randomly initialised set of 3 feedforward layers of widths 64, 32, 10. For our benchmark experiments involving latent replay, the model was frozen after the final convolutional layer. We chose not to include Batch Normalization in the CNN as it has been shown to exacerbate the problem of catastrophic forgetting [9].

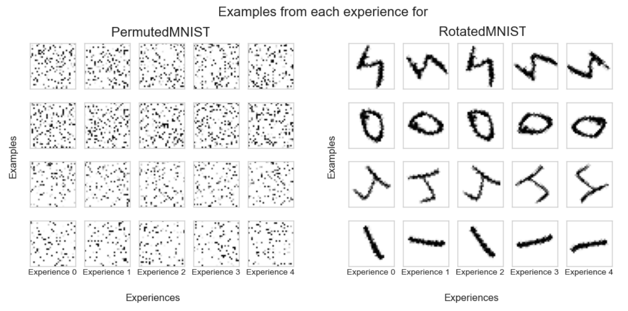

Fig. 2: Visualisation of 4 example images from each experience for 5 sequential experiences on each of the domain incremental continual learning datasets of Permuted MNIST (left), and Rotated MNIST (right).

Hyperparameter selection: Hyperparameters for all models were chosen over a coarse gridsearch of 'reasonable' values as determined from the literature. Due to the nature of continual learning experiments, we assumed no access to future training datasets during hyperparameter selection and tuned purely based on performance of the training data from the first dataset (as is reasonable to expect in a live continual learning scenario).

Experiments: We employed a standard domain-incremental Continual Learning setup, where a dataset is divided into separate experiences, comprising the same nominal input features and targets, but where the distribution of input features shifts between experiences. Models were trained successively on each experience, and evaluated after each training experience on a hold-out test set from all experiences. Early stopping of training based on monotonic training accuracy increases no smaller than 0.03 (with respect to accuracy) between epochs was employed to restrict overfitting.

We evaluated GLR against a range of continual learning methods: i) EWC [10], a regularization-based method, ii) Replay, which involves replay of true past stored raw examples, iii) Latent Replay, which involves replay of latent encoded past stored examples, and iv) Naive Fine-tuning, as a baseline (Table II). This enables us to compare the performance of GLR against data-free methods (EWC), as well as the ideal case of access to past data. Additionally, the comparison with Latent Replay allows us to 1) effectively isolate and evaluate the generative component of GLR and test the fidelity of the latent samples as a means of fortifying the higher levels of the network against catastrophic forgetting, and 2) compare Latent Replay with full Replay to analyse the effect of regularising the backbone of the network as a method in and of itself.

Archetype	**Method**	**Abbreviation**	**Source**
Baseline	Naive fine-tuning	Naive	
Regularization	EWC	EWC	[10]
	Replay	Replay	
Rehearsal	Latent Replay	LR	[5]
	Generative Latent Replay	GLR	[This work]

TABLE II: Continual Learning methods evaluated.

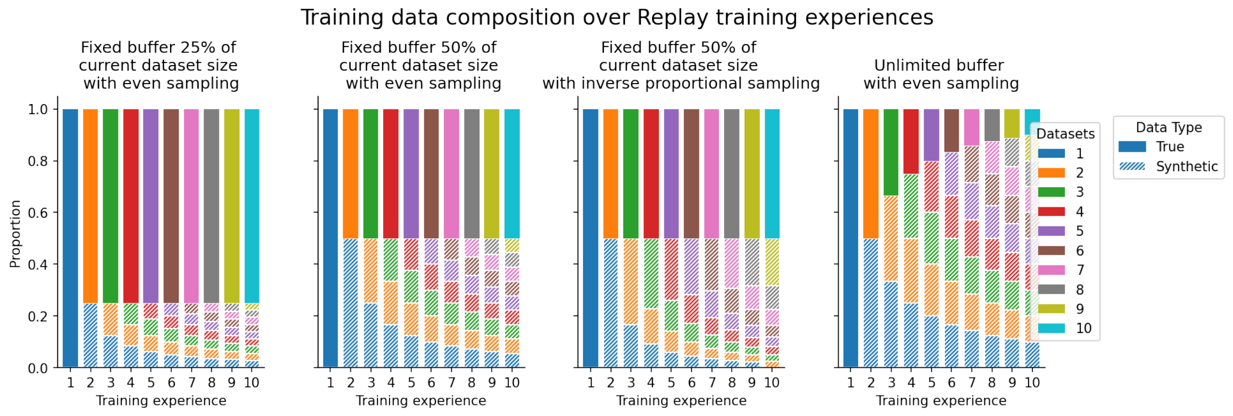

Fig. 3: Visualisation of the proportion of true current data and synthetic latent samples from past experiences used by GLR during each training phase for a continual learning problem on 10 successive experiences. Left-most diagram shows sampling method used in our experiments for a fixed buffer size of $M = 0.25|D_i|$ and even sampling of prior tasks (this is for illustrative purposes only, actual main experiments use $M = 0.1|D_i|$). Second diagram shows same sampling method for a buffer size of $M = 0.5|D_i|$ (also with even prior-task sampling). Third diagram shows a fixed buffer with an alternative sampling method where the number of samples per task decays with time from task introduction. Right-most diagram shows even sampling but with an unlimited adaptive buffer $M = (i-1)|D_i|$ for each experience i. Replay and Latent Replay use sampling strategies to GLR in each experiment, with the synthetic examples replaced by real examples.

All experiments were conducted using the same underlying neural network architecture and general training procedure.

D. Other considerations

Code and data availability: All experimental code is released at https://github.com/iacobo/generative-latent-replay. Our method has been implemented using the PyTorch based Avalanche continual learning library [8]. We hope this will enable other researchers to easily and simply use and modify our methods for further research, as our model (and our implementation of Latent Replay) can be straightforwardly imported in the same manner as any other continual learning strategy in the library. All experiments have been performed on open source standard benchmark problems to aid reproducibility and comparison with extant methods.

Metrics: We evaluate training and test accuracy and loss over each task. We also evaluate the average of these metrics over all ($n = 5$) tasks per experiment to provide a single comparable value of overall performance on the entire set of experiences between methods. As our datasets have an equal number of examples both between tasks and within classes per task, such metrics prove adequate for comparison. However more consideration may be required in unbalanced scenarios [11].

Replay Sampling Strategy: With replay based methods, it is important to consider the balancing strategy between prior tasks and the size of the buffer relative to the current task. To ensure fairness in our comparison of GLR against other replay-based strategies, we have adopted a fixed buffer size M, and allocated a buffer of size $\lfloor \frac{M}{i-1} \rfloor$ for prior task samples at experience i. This approach is a realistic and practical setup, as it allows for a fixed buffer constraint for memory, and can be easily adapted to real-world scenarios by adjusting M such that $|D_i| \leq \frac{M}{i} \forall i$.

For a more in-depth examination of different replay sampling strategies, we refer the reader to previous works such as [12]. To provide an illustration comparing our replay strategy to an alternative strategy assuming an unlimited data buffer and computational resources, we include Figure 3.

For an exceptionally long series of sequential tasks in a continual learning setup (for example in certain medical applications [11]), we provide the following adaptation to Algorithm 1, where there is a fixed number of tasks permitted in the buffer, and oldest tasks are dropped off as the buffer membership limit is exhausted: include membership constraint G in the required, change the formula on line 6 to read $Z^* = \bigcup_{j=\max(1,i-G)}^{i} Z_j^*$.

IV. Results

A. Benchmark results

We evaluated GLR on the Permuted MNIST and Rotated MNIST datasets and compared it to the following baselines: Naive, EWC, Latent Replay and Replay. The results for both experiments are summarized in Table III.

On the Permuted MNIST experiment, GLR outperformed the Naive and EWC baselines in terms of final average accuracy and final average loss, achieving a final average accuracy of 0.6517 and final average loss of 0.9962. While GLR surpassed traditional data-free methods using regularization strategies (EWC), it was not on par with the data-retaining methods Latent Replay and Replay, which were the best performing methods in this experiment.

On the Rotated MNIST experiment, GLR achieved a final average accuracy of 0.8746 and final average loss of 0.4317, and outperformed the Naive and EWC baselines. However, while it achieved much higher performance it was still not on par with Latent Replay and Replay, which were the best performing methods on this task as well.

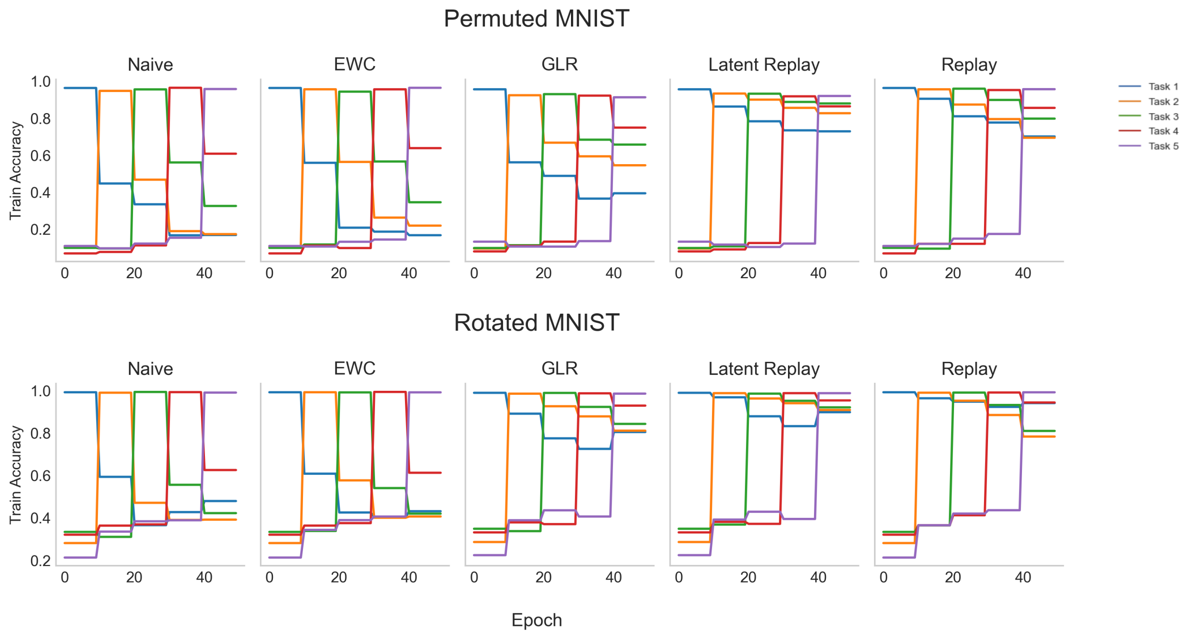

Fig. 4: Visualisation of the results for Permuted MNIST (top), and Rotated MNIST (bottom) experiments. Coloured lines indicate model's accuracy as measured before and after training on each of the 5 sequential tasks in the experiment. Naive and EWC noticeably suffer from significant catastrophic forgetting. GLR, Replay, and Latent Replay mitigate this to increasing degrees.

Permuted MNIST

Method	Final Avg Acc	Final Avg Loss
Naive	0.4467	1.6653
EWC	0.4670	1.7722
GLR	0.6517	0.9962
Latent Replay	0.8437	0.6207
Replay	0.8010	0.6715

Rotated MNIST

Method	Final Avg Acc	Final Avg Loss
Naive	0.5819	2.1731
EWC	0.5488	2.4301
GLR	0.8746	0.4317
Latent Replay	0.9338	0.2121
Replay	0.8939	0.4122

TABLE III: Final average Accuracy and Loss over all (5) tasks for each method on the Permuted MNIST (top) and Rotated MNIST experiments (bottom).

To better visualize the results, we provide a graph in Figure 4 illustrating the per-task performance of all methods in terms of accuracy and loss throughout the training process for each task.

B. Replay buffer size

We also compared variants of GLR with the replay buffer size hyper-parameter varied across the sample space. Results can be observed in Figure 5. Accuracy (and loss) across all tasks increased (resp. decreased) monotonically as the buffer size was increased, however the rate of change is approximately logistic (R^2 goodness of fit 0.9972), slowing with each increment. In our benchmark experiments we used a buffer of 10% the size of the current dataset (highlighted orange), balancing buffer size with performance, and is also in line with sampling procedures used by replay based methods in the literature.

Rotated MNIST (GLR Buffer Size)

Method	Final Avg Acc	Final Avg Loss
GLR 600 (1%)	0.7372	1.0784
GLR 6000 (10%)	0.8746	0.4317
GLR 15000 (25%)	0.9019	0.3350
GLR 30000 (50%)	0.9150	0.3065

TABLE IV: Final average Accuracy and Loss over all (5) tasks for GLR with different replay buffer sizes on the Rotated MNIST experiments.

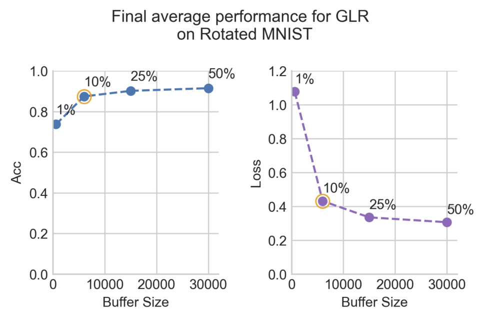

Fig. 5: Final average accuracy and loss over all tasks for the Rotated MNIST experiment for GLR models using different replay buffer sizes of 600, 6000, 15000, and 30000 synthetic examples (1%, 10%, 25%, and 50% of the current dataset's size (60000) respectively). Values for model used in comparative experiments with other methods (10%) highlighted in orange.

⚠Section not yet complete

C. Freeze depth:

D. Generator:

V. Discussion

In this paper, we proposed Generative Latent Replay (GLR), a novel approach for Continual Learning that utilizes generative models and strict regularisation of low level representations to enable the replay of latent samples from past experiences. Our experiments demonstrated that GLR significantly outperforms state-of-the-art data-free methods in terms of retaining performance on previous tasks while learning new ones on the Permuted MNIST and Rotated MNIST tasks, and is even comparable in performance to pure replay on the Rotated MNIST experiment.

GLR performed significantly better on the Rotated MNIST task compared to the Permuted MNIST task. This is expected as the Rotated MNIST task exemplifies a more natural domain shift between datasets, where the angle of rotation of the images varies between tasks. As expected, high level features learned by the CNN layers during initial training on the first dataset (e.g. edges, corners, crossings) remained a useful basis of input features for training on subsequent tasks.

In contrast, the Permuted MNIST task is a pathological example where the input features are effectively scrambled between tasks, making it more challenging for the model to retain performance on previous tasks, as the frozen backbone will output essentially ‘random’ features on subsequent tasks, limiting the benefit of regularizing these layers. The results on the Rotated MNIST task demonstrate the potential of GLR to handle more realistic domain shifts and its promise for handling Continual Learning in real-world domain-incremental scenarios where the underlying distribution of input features between experiences varies but does not fundamentally alter the nature of the input feature space.

Our ablation studies (i.e. comparing Naive, LR, GLR, and Replay, see Table I) confirmed the significance of incorporating a frozen backbone network and latent replay in the GLR method. However, we note that in more challenging scenarios such as Permuted MNIST GLR did not perform on par with LR, and as such explicit replay based methods (e.g. LR and replay) may be preferable in situations where access to and storage of past data is feasible and/or a generative model cannot be reliably constructed over a sufficiently condensed latent space.

We note also that the superior performance of Latent Replay compared to full Replay suggests that strictly freezing the backbone of the network after initial representative training may be a useful tool in mitigating catastrophic forgetting per se. It may be worthwhile investigating its performance in combination with existing regularisation techniques such as EWC, Synaptic Intelligence, or Learning without Forgetting. It also may be worth considering the use of LR with an uninvertible network backbone in the case where raw data cannot be stored, but a generative model is infeasible to train, using the backbone itself as an ‘encoder’ for the raw data.

In terms of future work, we plan to investigate more complex generative models, such as autoencoders or generative adversarial networks, as generators for GLR. We also plan to investigate further hybrid methods incorporating regularisation strategies on the mid-upper layers of the network. Additionally, we aim to study the application of GLR to real-world structured medical data, a scenario exemplifying both the inability to store sensitive data between tasks, and sensible domain shifts between datasets (as patient populations vary across sites). We also plan to investigate the impact of bespoke replay strategies on the performance of GLR.

In conclusion, we have presented a novel approach for Continual Learning that leverages the power of generative models to enable the replay of latent samples from past experiences. Our results demonstrate the potential of GLR on a variety of continual learning benchmarks and show that it significantly outperforms state-of-the-art data-free methods in terms of retaining performance on previous tasks while learning new ones, and is comparable with data-retaining methods without compromising data storage limitations. We believe that GLR represents a promising direction for addressing the challenge of catastrophic forgetting in artificial intelligence systems operating in dynamic environments where data cannot be stored indefinitely.

VI. Acknowledgements

Jacob Armstrong is supported by the EPSRC Center for Doctoral Training in Health Data Science (EP/S02428X/1).

References

[1] Nikhil Churamani, Ozgur Kara, and Hatice Gunes. Domain-incremental continual learning for mitigating bias in facial expression and action unit recognition. *arXiv preprint arXiv:2103.08637*, 2021.

[2] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[3] Fei Mi, Xiaoyu Lin, and Boi Faltings. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 408–413, 2020.

[4] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[5] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[6] Samuil Stoychev, Nikhil Churamani, and Hatice Gunes. Latent generative replay for resource-efficient continual learning of facial expressions. 2022.

[7] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[8] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M van de Ven, et al. Avalanche: an end-to-end library for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3600–3610, 2021.

[9] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, pages 989–998, 2020.

[10] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[11] Jacob Armstrong and D Clifton. Continual learning of longitudinal health records. *arXiv preprint arXiv:2112.11944*, 2021.

[12] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2180–2187. IEEE, 2021.