Journal of Industrial Engineering and Applied Science
Journal Home: jieas.suaspress.org | CODEN: JIEAAE
Vol. 2, No. 5, 2024 | ISSN 3005-6071 (Print) | ISSN 3005-608X (Online)

When Federated Learning Meets Machine Unlearning

BARTRA, Mary j [1*]

[1] TOMS Bloomberg machine learning and ai lab, USA

**** BARTRA, Mary j is the corresponding author, E-mail: maryjbartra@outlook.com***

Abstract: This research paper explores the intersection of Incremental Learning and Unlearning in the context of machine learning, with a particular emphasis on dynamic environments where data evolves rapidly, and models must continuously adapt. Incremental learning allows machine learning models to efficiently update their knowledge without retraining from scratch, while incremental unlearning ensures compliance with privacy regulations or user requests by selectively removing specific data points from the model. The paper discusses several key techniques for balancing learning and unlearning, including Elastic Weight Consolidation (EWC), gradient-based unlearning, fine-tuning, and memory-based methods.

Keywords: Heterogeneous Machine Learning, Federated Learning, Machine Unlearning, Graph Neural Network, Gradient-based Learning, Elastic Weight Consolidation.

DOI: https://doi.org/10.5281/zenodo.13854241 **ARK:** https://n2t.net/ark:/40704/JIEAS.v2n5a07

1 INTRODUCTION

Federated Learning (FL) emphasizes decentralized data processing across multiple devices or clients without transferring raw data to a central server. By preserving data privacy and reducing communication costs, FL enables organizations to collaborate on training machine learning models while maintaining control over their proprietary data. However, as privacy and data governance regulations evolve, there is an increasing demand for mechanisms that allow entities to have their data removed from these trained models. In heterogeneous network, it means we need an opposite mechanism to the distributed learning process to systematically eliminate certain part of data and models, where machine unlearning comes into play.

Machine Unlearning on the other end of the table, focuses on erasing the influence of specific data points from trained models, ensuring compliance with data removal requests, such as those stipulated by regulations like the. When federated learning meets machine unlearning, it creates a new frontier in ensuring that privacy concerns are adequately addressed in distributed systems. The process could be data-driven, or model driven. The two processes are not totally opposite, where individual components may even be shared or re-usable. The synergy between these two approaches brings unique challenges and opportunities, especially in terms of balancing efficiency, accuracy, and compliance with privacy requirements.

2 REVIEW OF PAST LITERATURE

Before we work on federated learning and machine unlearning applications, let's review the relevant advances in heterogeneous machine learning, federated learning, machine unlearning and federated unlearning in sequence.

2.1 HETEROGENEOUS MACHINE LEARNING

Heterogeneous networks ([1] and [2]) refer to systems with varying connectivity, device capabilities, and computational resources, which is a critical issue in federated learning. FL systems often involve a wide variety of devices, ranging from high-powered servers to low-energy mobile phones, connected over different network conditions. This heterogeneity affects model updates: In federated learning, some devices may have limited bandwidth or lower computational capabilities, which means they can only contribute partial or less frequent updates. This impacts the aggregation of model updates and the efficiency of the overall learning process. The heterogeneous network environment can introduce vulnerabilities during communication between clients and the central server, which can have implications for both federated learning and unlearning processes, especially in environments where secure data transmission is essential. This is further illustrated in [3].

Heterogeneous networks and graph neural networks (GNNs) are advanced concepts within the broader field of machine learning, particularly within graph-based learning. Both are essential for modeling and analyzing complex data structures, where relationships between data points are not uniform. Their relevance to machine learning lies in their ability to handle structured data that goes beyond traditional grid-like data used in conventional ML techniques (such as images or tabular data). In the past, we have seen so much success in using Graph Neural Network (GNN) in resolving

image and tabular data problems as shown in [4,5,6] and [7]. The key idea behind GNNs is to aggregate information from a node's neighbors to update the node's representation. This aggregation allows the model to learn from both local and global structures within the graph. A typical GNN follows the message-passing paradigm, where each node iteratively updates its feature vector by combining its own features with those of its neighbors.

In a social network, the nodes might represent individuals, and the edges might represent friendships or connections. A GNN can learn to predict behaviors like influence or detect community structures by considering the relationships (edges) and node features (such as user interests or demographics).

GNNs have broad applications in areas like drug discovery, social network analysis, and recommendation systems, where data is inherently structured as a graph. GNNs are increasingly used in areas requiring an understanding of relationships between entities, which goes beyond the capabilities of traditional machine learning methods, for example Wang et al. [8] proposed GNN for sports analytics.

Standard GNNs are typically designed for homogeneous graphs, where nodes and edges are of a single type. However, heterogeneous networks require GNNs to be extended to account for multiple types of nodes and edges. This extension leads to Heterogeneous Graph Neural Networks (HetGNNs), which are specifically tailored to heterogeneous graphs.

In HetGNNs, the model must process different types of nodes and edges, capturing the distinct relationships and interactions between them. To achieve this, HetGNNs use different message-passing rules for each type of node and edge, ensuring that the diversity in the graph's structure is properly reflected in the learned representations according to [9] and [10]. Xu et al. [11] showed very promising results in detecting financial risk using heterogeneous GNN. Li et al. [9] constructed LSTM recurrent neural networks to simply the heterogeneous network representation.

2.2 Federated Learning

Federated Learning (FL) operates by distributing the model training process across multiple decentralized clients, often edge devices, while keeping their data localized. Each client independently computes model updates based on its data and shares these updates with a central server, which aggregates them to improve the global model. This process allows organizations to create models without aggregating the raw data centrally, thereby reducing the risk of privacy violations or data breaches [12].

Federate learning features privacy-preserving. Since the data remains on the clients, FL reduces the risk of data breaches and complies with privacy regulations like GDPR [13]. It also features decentralized training: FL leverages distributed data from various devices, making it more scalable than traditional centralized learning. In federate learning, instead of transmitting data, clients send model updates (e.g., gradients) to a central server, which aggregates them into a global model.

There are three common types of Federated Learning [14]: 1. Horizontal Federated Learning: Data across different clients shares similar features but differs in the samples (e.g., mobile users using the same application). 2. Vertical Federated Learning: Different clients have complementary features about the same set of instances (e.g., different institutions sharing data about the same individuals). 3. Federated Transfer Learning: Combines knowledge from different datasets to improve performance in settings where clients have different feature spaces and sample sets.

2.3 Machine Unlearning

Machine unlearning refers to the process of selectively removing the influence of specific data points from a machine learning model after it has been trained. This concept is especially important when it comes to handling sensitive data, such as in cases where a user requests that their data be deleted for privacy reasons (e.g., to comply with data regulations [15]. The goal of machine unlearning is to ensure that once data is deleted, it has no lingering effects on the model's behavior, essentially making the model behave as if the data was never included during training.

Traditional machine learning models, once trained, integrate patterns and relationships from the data and are generally not designed to "forget" specific data points [16]. Retraining the model from scratch after removing a few data points is computationally expensive and often impractical for large-scale models. Machine unlearning aims to address this problem by enabling efficient removal of specific data without the need for complete retraining.

The process of machine unlearning involves modifying the machine learning model to reverse or undo the effects of certain data points. Various techniques are used, such as Exact Unlearning and Approximate Unlearning [17]. Exact unlearning directly removing the contribution of specific data points from the model, maintaining the model's accuracy while ensuring that the specified data no longer influences the model. Approximate unlearning implements methods that approximate the removal of data points but may not fully guarantee that all influences of the data are completely eliminated. This method can be faster but less precise. Adversarial training introduced in [18] and [19] can be used as an effective method for machine unlearning. Attacks to machine unlearning also occurs frequently as according to [20,21] and [22]. Some de-centralized approach opens our door to the federated unlearning.

2.4 Federated Unlearning

For heterogeneous learning, Wu et al. [23] initiates a federated unlearning way that leverages knowledge distillation to remove the influence of specific clients from a

global model. As indicated by Shaik et al. [24], the federated unlearning approach treat the global model as a teacher to guide the training of an unlearning model. The approach reduces client-side computational pressure and enhanced the model generalization. Zhu et al. [25] proposes a similar federated unlearning framework called FedLU, which is for heterogeneous knowledge graph embedding learning. The model enables di-directional transfer of knowledge between local and global levels. Due to security attack in the unlearning process, Li et al. [26] uses a fine-tuning process guided by a teacher network to achieve unlearning at the student network.

In machine unlearning within heterogeneous networks, some additional challenges are mainly in unlearning efficiency and coordination in unlearning. Devices with lower computational power or limited network connectivity might not be able to perform unlearning requests promptly. This delays compliance with privacy regulations and can introduce inconsistencies in how different clients handle unlearning.

Federated unlearning within a heterogeneous network requires coordination across diverse devices, ensuring that the impact of unlearning a specific data point is reflected across the entire system.

In federated learning, heterogeneity manifests in the form of different data distributions and varying device capabilities, making it crucial to develop strategies that accommodate these differences. When integrating machine unlearning into such systems, this heterogeneity adds complexity to the unlearning process, requiring specialized methods for handling both data and model diversity efficiently. As the fields of FL and unlearning continue to evolve, addressing the challenges of heterogeneity will be essential to maintaining system performance and ensuring privacy compliance.

3 METHDOLOGIES FOR INTEGRATING FEDERATED LEARNING AND MACHINE UNLEARNING

3.1 MOTIVATION

The intersection of federated learning and machine unlearning presents several unique challenges, which arise from the distributed nature of the former and the stringent privacy requirements of the latter.

Data Distribution and Decentralization: In federated learning, data remains distributed across multiple clients, making it difficult to trace back which data points influenced the global model. This decentralization complicates the process of identifying and removing specific data points when an unlearning request is made. As a result, federated unlearning methods must be designed to handle such decentralized environments effectively.

Communication and Computational Overhead: Implementing machine unlearning in a federated system can significantly increase communication and computational costs. For example, when an unlearning request is received, a federated system may need to communicate with multiple clients to adjust their local models, which can be resource intensive. Balancing the trade-off between computational efficiency and the accuracy of the unlearning process is a key challenge in federated learning systems.

Privacy and Security Risks: Both federated learning and machine unlearning are designed to enhance privacy. However, the unlearning process itself can introduce new privacy risks. For instance, unlearning requests could be exploited by adversaries to infer sensitive information about the training data or model behavior. Additionally, ensuring that unlearning does not leak information about the removed data point is critical, as failure to do so could compromise user privacy and trust.

Model Performance and Utility: Machine unlearning can degrade model performance if not carefully managed. In a federated learning context, ensuring that the global model retains its predictive capabilities after removing specific data points is particularly challenging. As data points are removed, the model might require significant retraining to maintain its accuracy, leading to potential trade-offs between privacy compliance and model utility.

3.2 DIFFERENTIAL PRIVACY WITH UNLEARNING

A bank uses a machine learning model to assess customer creditworthiness. By applying DP with Unlearning, if a customer requests the removal of their data, the model's predictions remain valid while complying with privacy and legal standards.

ALGORITHM 1. DIFFERENTIAL PRIVACY UNLEARNING

```
Input: N clients, global model M, privacy budget ε, noise variance σ^2
Initialize: M_0 (initial global model), privacy budget ε

For each round t in 1 to T:
   For each client k in parallel:
      // Local training with differential privacy
      Train local model M_k on client k's data
      Compute gradient G_k
      Add noise: G_k' = G_k + Noise(σ^2)  // Add noise for differential privacy

      Send the noisy gradient G_k' to the server

   // Server aggregation
   Aggregate global model: M_(t+1) = M_t + η * Σ(G_k')

   // Unlearning request for client c
   If unlearning request received from client c:
      Compute impact of client c's data on M_(t+1) based on differential privacy constraints
      Adjust model parameters: M_(t+1) -= η * Gradient(c)
```

Journal of Industrial Engineering and Applied Science
Journal Home: jieas.suaspress.org | CODEN: JIEAAE
Vol. 2, No. 5, 2024 | ISSN 3005-6071 (Print) | ISSN 3005-608X (Online)

Update privacy budget ε

Return: Final global model M_T

Differential Privacy with Unlearning (Algorithm 1) adds noise to the model updates from each client to obscure the contribution of individual data points. It ensures privacy while facilitating unlearning by limiting the impact of any single data point. This differential privacy approach can potentially apply to studies in financial market treasury trading as illustrated in Li et al. [9]. The LSTM network in [9] would be an excellent benchmark for differential privacy with unlearning.

3.3 Client-Side Unlearning

Client-side unlearning refers to the process of removing specific data or user contributions directly at the client's side, without requiring the central server to retrain or modify the global model extensively. This approach is particularly relevant in decentralized systems like Federated Learning (FL), where training occurs locally on clients' devices (e.g., smartphones, edge devices, or organizational servers).

Instead of removing the influence of data on the central server, the clients independently handle the removal of their data from the locally trained model and send updates reflecting the unlearned data to the central server. This distributed approach helps to preserve privacy, ensure minimal retraining costs, and reduce communication overhead, making it both computationally efficient and privacy compliant. This idea is referred to the same as presented by Sun et al. [10].

Algorithm 2. Client-side unlearning

Input: N clients, global model M, local models M_k, unlearning request from client c

For each round t in 1 to T:
  For each client k in parallel:
    // Local training
    Train local model M_k on client k's data
    Compute local gradient G_k
    Send G_k to the server

  // Server aggregation
  Aggregate global model: M_(t+1) = M_t + η * Σ(G_k)

  // Unlearning request for client c
  If unlearning request from client c:
    // Client-side unlearning
    Remove client c's data from local model M_c
    Recompute gradient G_c' on M_c without unlearned data
    Send unlearned gradient G_c' to the server
    Server updates global model: M_(t+1) -= η * (G_c - G_c')

Return: Final global model M_T

Here's a step-by-step explanation of how Client-Side Unlearning (Algorithm 2) function in a Federated Learning environment:

During the **Initial Training Process**: In the Federated Learning setup, clients (e.g., smartphones, organizational servers) download the initial global model from the central server. Each client trains the global model on its local dataset, computing gradients based on its data, which reflect the contribution of the local data to the global model. The clients then send the gradients or model updates back to the central server, which aggregates these updates to form a new global model.

When **Unlearning Request** arrives: A client might request to unlearn a portion of its local data, either because the data was erroneously included, the user has requested to remove their data (e.g., "right to be forgotten"), or due to legal and privacy regulations. Upon receiving the unlearning request, the client recalculates its local model without the specified data. This step is done without sharing the raw data with the central server, ensuring privacy.

In the stage of **Recalculation of Gradients**: The client re-trains its local model based on the remaining data after the unlearning request is processed. It calculates new gradients or updates (representing the influence of the modified local data) and sends the "unlearned" gradients to the central server. The server can then subtract the old gradients (which included the unlearned data) and add the updated gradients, thereby ensuring that the global model no longer reflects the contribution of the unlearned data.

For the last **Model Aggregation** step: The central server aggregates these updated gradients, removing the influence of the unlearned data on the global model. Importantly, this method avoids the need for the server to retrain the entire global model from scratch, making it an efficient approach for large-scale systems with many clients.

While Client-Side Unlearning offers many benefits, it also faces some challenges: 1. Data Heterogeneity: Clients may have diverse, non-IID (independent and identically distributed) data, which can make it more difficult to effectively reverse the impact of unlearned data on the global model. 2. Communication Overhead: Although client-side unlearning reduces retraining on the server, it still requires communication between clients and the server to transmit updates. In scenarios with frequent unlearning requests, this could lead to increased communication overhead. 3. Model Accuracy: Removing data from local models and recomputing gradients could slightly impact global model accuracy, especially if the unlearned data had a significant contribution to the model's performance.

3.4 Gradient-Based Unlearning

Gradient-Based Unlearning is a technique for removing the influence of specific data points from a machine learning model by leveraging the model's gradients. In this approach, the contributions made by a particular data point (or set of data points) to the trained model are reversed by modifying or rolling back the gradients that were computed during training. This method allows for the efficient removal of data's impact without requiring complete retraining from

Journal of Industrial Engineering and Applied Science
Journal Home: jieas.suaspress.org | CODEN: JIEAAE
Vol. 2, No. 5, 2024 | ISSN 3005-6071 (Print) | ISSN 3005-608X (Online)

scratch.

The core idea behind gradient-based unlearning is that the model's training process can be viewed as a sequence of updates applied to the model's parameters. These updates are computed using gradients derived from each training sample, which represent the direction and magnitude by which the model's weights need to change to reduce prediction error. By systematically removing or adjusting the gradients associated with the data points that need to be unlearned, the model can revert to a state as if the specific data were never included in training.

ALGORITHM 3. GRADIENT-BASE UNLEARNING ALGORITHM

```
Input: N clients, global model M, local models M_k, learning rate η

For each round t in 1 to T:
  For each client k in parallel:
    // Local training
    Train local model M_k on client k's data
    Compute gradient G_k
    Send G_k to the server

  // Server aggregation
  Aggregate global model: M_(t+1) = M_t + η * Σ(G_k)

  // Unlearning request for client c's data point d
  If unlearning request from client c's data point d:
    // Gradient-based unlearning
    Identify gradient G_d associated with data point d
    Reverse its contribution to the model:
    M_(t+1) = M_(t+1) - η * G_d  // Reverse gradient of d

Return: Final global model M_T
```

The gradient-based unlearning in Algorithm 3 can be broken down into the following steps:

1. Gradient Computation During Training: During the normal training process, the machine learning model learns by updating its parameters (weights) using the gradients computed for each training data point. These gradients reflect the contribution of each data point to the overall model.

   For each data point x_i, the gradient of the loss function $L(x_i)$ with respect to the model's parameters θ is computed, denoted as $\nabla_\theta L(x_i)$

2. Tracking and Storing Gradients: To enable efficient unlearning, the system stores the gradients associated with each data point (or mini batch) during training. These gradients can be saved as a history of updates applied to the model's parameters.

3. Unlearning Request: When a data point x_i (or a set of data points) needs to be unlearned, the system identifies the gradients $\nabla_\theta L(x_i)$ that were computed during the training phase. The system calculates the negative of these gradients to reverse the effect of the specific data on the model's parameters. This step is essential to "unlearn" the contribution made by the data.

4. Gradient Reversal or Subtraction: The system applies the reverse gradients $-\nabla_\theta L(x_i)$ to the model's parameters to undo the effect of the training on that particular data point. This essentially rolls back the model's parameters to a state as if the data point x_i had not been included in the training process.

5. Model Adjustment: After adjusting the parameters using reverse gradients, the model is updated accordingly. The resulting model behaves as if the unlearned data never influenced its training, allowing for effective unlearning without the need for complete retraining.

The major benefit of Gradient-Based Unlearning is efficiency. Gradient-based unlearning is highly efficient compared to full retraining, as it targets specific updates related to the unlearned data. This reduces the computational cost, especially in large models where retraining would be prohibitively expensive. By focusing on reversing specific gradients, this approach can scale to large datasets and complex models without needing to reconstruct the entire model from scratch.

Since only the contributions of the unlearned data are removed, the overall structure and performance of the model remain largely intact. This ensures that the model continues to perform well without significant degradation after unlearning.

During the implementation of Challenges of Gradient-Based Unlearning, we are faced with gradient storage: One challenge is the need to store gradients associated with each data point or batch of data points during training. In large-scale applications, this can result in significant memory overhead, which needs to be managed efficiently.

Some model architectures, particularly those with highly complex layers like deep neural networks, may not lend themselves as easily to gradient-based unlearning due to the complexity of their gradient calculations.

In some cases, gradient-based unlearning may only provide an approximate reversal of the data's impact, especially when multiple unlearning requests accumulate. This could result in minor inaccuracies in the unlearned model, which we will not cover in the scope of current study

3.5 INCREMENTAL LEARNING AND UNLEARNING

Incremental Learning and Unlearning are two complementary processes that enable a machine learning model to evolve over time by continually learning from new data while selectively forgetting previously learned data upon request. The goal of incremental learning is to allow the model to adapt to changes in the data distribution without needing to retrain from scratch, while unlearning is the mechanism to remove specific data points or knowledge that should no longer contribute to the model.

Incremental learning allows a model to continuously update its knowledge base with new information without

Journal of Industrial Engineering and Applied Science
Journal Home: jieas.suaspress.org | CODEN: JIEAAE
Vol. 2, No. 5, 2024 | ISSN 3005-6071 (Print) | ISSN 3005-608X (Online)

discarding or retraining from the entire dataset. This is crucial in dynamic environments where data evolves, and the model must remain adaptable and scalable. In this approach:

The model keeps learning incrementally from new data points or batches of data over time. It avoids "catastrophic forgetting" by retaining useful knowledge from past data while integrating new information. This approach is ideal for systems where continuous learning is needed (e.g., online learning or dynamic environments with streaming data).

On the other hand, incremental unlearning deals with the systematic and efficient removal of specific knowledge learned from data points. This may be needed due to privacy regulations (like GDPR or CCPA), error correction, or user requests to have their data forgotten. In incremental unlearning: Instead of retraining the entire model when data needs to be unlearned, the model is adjusted to reverse the effect of the specific data points. The model effectively "forgets" the contributions of data that need to be removed while retaining the rest of the knowledge. This is particularly challenging in dynamic systems because the model must adaptively handle both learning and unlearning without performance degradation.

ALGORITHM 4. INCREMENTAL LEARNING-UNLEARNING ALGORITHM

```
Input: N clients, global model M, local models M_k, learning rate
η, incremental batch size B

For each round t in 1 to T:
   For each client k in parallel:
      // Incremental learning
      Divide client k's data into batches B_1, B_2, ..., B_n
      For each batch B_i:
         Train local model M_k on batch B_i
         Compute gradient G_k_i on batch B_i
         Send G_k_i to the server

      // Server-side aggregation
      Aggregate model incrementally:
      M_(t+1) = M_t + η * Σ(G_k_i)

   // Unlearning request for client c
   If unlearning request from client c:
      Identify batches where data from client c is present
      Recompute local model for these batches without client c's
data
      Send unlearned gradients G_c' to the server
      Server updates global model: M_(t+1) -= η * (Old gradients
from c - G_c')

Return: Final global model M_T
```

Algorithm 4 involves adding new data or knowledge to the model in small, manageable portions without retraining on the entire dataset:

Learning from New Data: The model is updated using the gradients from new data, while keeping track of previous data contributions.

Adaptive Integration: Techniques such as regularization or specific architectures (like elastic weight consolidation) are often used to prevent catastrophic forgetting.

Efficient Updating: The system efficiently learns from new data without needing access to the full historical data.

Incremental Unlearning ensures that the model can "forget" certain data while retaining other relevant knowledge:

Identify Data to Be Forgotten: The model identifies which data needs to be unlearned based on user requests or privacy obligations.

Reverse the Effect: The model reverts the impact of the data to be forgotten, either by subtracting the gradients associated with that data or using fine-tuning strategies to remove its influence.

Model Adjustment: The model is adjusted accordingly so that the specific data no longer affects its predictions, while other data contributions remain intact.

We also summarized the techniques used in Incremental Learning and Unlearning in Algorithm 4.

Firstly, Elastic Weight Consolidation (EWC) is method where important parameters of the model, learned from earlier tasks, are consolidated to prevent catastrophic forgetting. It helps balance learning and unlearning by regularizing new updates. Secondly, regularization penalizes large updates to certain parameters to prevent the model from drifting too far from its previous knowledge during incremental updates. Fine-tuning can be used for unlearning specific data, where a model is slightly adjusted to minimize the influence of the forgotten data while retaining general patterns.

Incremental learning systems often use memory-based techniques to store a small subset of old data that is periodically retrained with the new data to retain past knowledge. This can help facilitate both learning and unlearning. Gradient Reversal for Unlearning is used by reversing the gradients of the data to be unlearned, the system can remove their contributions incrementally. The integration of Incremental Learning and Unlearning offers significant advancements for machine learning models operating in dynamic, data-driven environments.

4 CONCLUSION AND FUTURE RESEARCH DIRECTION

The integration of Incremental Learning and Unlearning offers significant advancements for machine learning models operating in dynamic, data-driven environments. Incremental learning ensures that models can adapt and evolve by incorporating new data without retraining from scratch, making them ideal for real-time applications such as financial markets, healthcare, and autonomous systems. Conversely, incremental unlearning provides a critical mechanism for complying with privacy regulations, allowing users to request

the removal of their data from models without compromising the system's overall performance. Together, these techniques ensure that machine learning systems remain adaptable, scalable, and compliant in a rapidly changing world.

By leveraging approaches like Elastic Weight Consolidation (EWC), gradient-based unlearning, and fine-tuning strategies, it is possible to maintain high model accuracy while efficiently integrating or forgetting specific data points. These techniques prevent catastrophic forgetting in incremental learning scenarios and allow for selective removal of data without necessitating expensive retraining processes. The application areas for incremental learning and unlearning are broad, ranging from online learning systems and autonomous vehicles to healthcare and financial fraud detection, showing immense potential for deployment across diverse fields.

Despite the promising developments, several challenges remain in the integration of incremental learning and unlearning, opening the door for future research:

Scalability in Large-Scale Models: As models grow in complexity and size, developing efficient algorithms for incremental unlearning that can scale to large datasets without affecting performance remains a critical research area. Techniques that allow for scalable gradient reversal or selective parameter adjustments would be particularly valuable.

Privacy Preservation in Federated Learning: Federated learning systems that incorporate both incremental learning and unlearning present unique challenges, especially in terms of balancing model accuracy with privacy constraints. Research could focus on integrating differential privacy techniques with unlearning to ensure that users' contributions are securely removed while maintaining model performance.

Handling Concept Drift: Incremental learning models must deal with concept drift — a situation where the underlying data distribution changes over time. Future work could focus on refining techniques for detecting and adapting to concept drift, while ensuring that older, irrelevant data is incrementally unlearned to avoid skewing model predictions.

Real-Time Learning and Unlearning: In real-time systems such as financial trading or autonomous driving, the model must simultaneously learn from new data and unlearn old data without delay. Future research could explore the development of lightweight, real-time algorithms that can handle both learning and unlearning without compromising on efficiency or accuracy.

Joint Learning and Unlearning Frameworks: Another area of interest lies in developing unified frameworks that seamlessly integrate both incremental learning and unlearning processes. These frameworks would provide an end-to-end solution for continuous learning while ensuring that unlearning requests are handled in an optimized and efficient manner.

Adversarial Attacks and Robustness: As unlearning techniques become more prevalent, they may be targeted by adversarial attacks aimed at forcing models to forget critical data. Future research could focus on improving the robustness of unlearning algorithms against such attacks, ensuring that malicious actors cannot exploit these mechanisms to degrade model performance.

In conclusion, the combination of incremental learning and unlearning holds significant potential for enhancing the flexibility, scalability, and privacy of machine learning systems. However, there remain many open research questions, particularly in terms of scalability, efficiency, privacy preservation, and robustness against adversarial threats. Addressing these challenges will unlock the full potential of these techniques and pave the way for their broader adoption in real-world applications.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to the incredible contributors of the GitHub machine learning community. Your unwavering dedication, innovative ideas, and generous sharing of knowledge have not only accelerated the progress of machine learning but also made it more accessible to everyone. The repositories, libraries, tutorials, and insightful discussions you maintain have been invaluable to countless researchers, practitioners, and learners like me. Your contributions continue to inspire and empower the next generation of machine learning enthusiasts, and for that, I am sincerely thankful.

FUNDING

Not applicable.

INSTITUTIONAL REVIEW BOARD STATEMENT

Not applicable.

INFORMED CONSENT STATEMENT

Not applicable.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

CONFLICT OF INTEREST

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

PUBLISHER'S NOTE

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

AUTHOR CONTRIBUTIONS

Not applicable.

ABOUT THE AUTHORS

BARTRA, Mary j

TOMS Bloomberg machine learning and ai lab, USA.

REFERENCES

[1] Z. Wu, "MPGAAN: Effective and Efficient Heterogeneous Information Network Classification," Journal of Computer Science and Technology Studies, vol. 6, p. 08–16, 2024.

[2] Z. Wu, "Deep Learning with Improved Metaheuristic Optimization for Traffic Flow Prediction," Journal of Computer Science and Technology Studies, vol. 6, p. 47–53, 2024.

[3] Z. Wang, Y. Chen, F. Wang and Q. Bao, Improved Unet model for brain tumor image segmentation based on ASPP-coordinate attention mechanism, 2024.

[4] X. Liu and Z. Wang, "Deep learning in medical image classification from mri-based brain tumor images," arXiv preprint arXiv:2408.00636, 2024.

[5] X. Liu, H. Qiu, M. Li, Z. Yu, Y. Yang and Y. Yan, "Application of Multimodal Fusion Deep Learning Model in Disease Recognition," arXiv preprint arXiv:2406.18546, 2024.

[6] X. Liu, Z. Yu and L. Tan, "Deep Learning for Lung Disease Classification Using Transfer Learning and a Customized CNN Architecture with Attention," arXiv preprint arXiv:2408.13180, 2024.

[7] X. Liu, Z. Yu, L. Tan, Y. Yan and G. Shi, "Enhancing Skin Lesion Diagnosis with Ensemble Learning," arXiv preprint arXiv:2409.04381, 2024.

[8] Z. Wang, Y. Zhu, Z. Li, Z. Wang, H. Qin and X. Liu, "Graph neural network recommendation system for football formation," Applied Science and Biotechnology Journal for Advanced Research, vol. 3, p. 33–39, 2024.

[9] Z. Li, B. Wang and Y. Chen, "Incorporating economic indicators and market sentiment effect into US Treasury bond yield prediction with machine learning," Journal of Infrastructure, Policy and Development, vol. 8, p. 7671, 2024.

[10] M. Sun, Z. Feng, Z. Li, W. Gu and X. Gu, "Enhancing financial risk management through lstm and extreme value theory: A high-frequency trading volume approach," Journal of Computer Technology and Software, vol. 3, 2024.

[11] K. Xu, Y. Wu, Z. Li, R. Zhang and Z. Feng, "Investigating financial risk behavior prediction using deep learning and big data," International Journal of Innovative Research in Engineering and Management, vol. 11, p. 77–81, 2024.

[12] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated machine learning: Concept and applications," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, p. 1–19, 2019.

[13] K. Bonawitz, "Towards federated learning at scale: Syste m design," arXiv preprint arXiv:1902.01046, 2019.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Artificial intelligence and statistics, 2017.

[15] Y. Cao, A. F. Yu, A. Aday, E. Stahl, J. Merwine and J. Yang, "Efficient repair of polluted machine learning systems via causal unlearning," in Proceedings of the 2018 on Asia conference on computer and communications security, 2018.

[16] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie and N. Papernot, "Machine unlearning," in 2021 IEEE Symposium on Security and Privacy (SP), 2021.

[17] S. Schelter, "amnesia–towards machine learning models that can forget user data very fast," in 1st International Workshop on Applied AI for Database Systems and Applications (AIDB19), 2019.

[18] Y. Tao, "Meta Learning Enabled Adversarial Defense," in 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE), 2023.

[19] L. Tan, S. Liu, J. Gao, X. Liu, L. Chu and H. Jiang, "Enhanced self-checkout system for retail based on improved YOLOv10," arXiv preprint arXiv:2407.21308, 2024.

[20] Y. Tao, "SQBA: sequential query-based attack," in Fifth International Conference on Artificial Intelligence and Computer Science (AICS 2023), 2017.

[21] Y. Tao, Y. Jia, N. Wang and H. Wang, "The FacT: Taming Latent Factor Models for Explainability with Factorization Trees," in Proceedings of the 42nd International ACM SIGIR Conference on Research and

Development in Information Retrieval, New York, NY, USA, 2019.

[22] Y. Tao, Z. Wang, H. Zhang and L. Wang, "NEVLP: Noise-Robust Framework for Efficient Vision-Language Pre-training," arXiv preprint arXiv:2409.09582, 2024.

[23] C. Wu, S. Zhu and P. Mitra, "Federated unlearning with knowledge distillation," arXiv preprint arXiv:2201.09441, 2022.

[24] T. Shaik, X. Tao, H. Xie, L. Li, X. Zhu and Q. Li, "Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy," arXiv preprint arXiv:2305.06360, 2023.

[25] X. Zhu, G. Li and W. Hu, "Heterogeneous federated knowledge graph embedding learning and unlearning," in Proceedings of the ACM web conference 2023, 2023.

[26] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," arXiv preprint arXiv:2101.05930, 2021.