**CSE4037 Deep learning**

**J Component - Project Report**

***TITLE: Comparative Analysis of Deep Learning Models on the Fashion MNIST Dataset***

***By***

**22MIA1042 Sai Lakshmi.R**

**22MIA1048 M.Divya dharshini**

**22MIA1099 Lakshmi vasanthi**

**M.Tech CSE with Business analytics**

***Submitted to***

**Thomas Abraham J V,**

**Assistant Professor Senior, SCOPE, VIT, Chennai**

**School of Computer Science and Engineering**

****

***April 2025***

**CONTENTS**

|  |  |
| --- | --- |
| **Table of Contents** | **Index Number** |
| Abstract | 3 |
| Introduction | 3 |
| Related work | 4 |
| Algorithms | 5 |
| Dataset and methodology | 7 |
| Visualization | 8 |
| Comparative evaluation of Algorithms | 9 |
| Results and Discussion | 12 |
| Conclusion | 12 |
| Reference | 13 |

**Abstract**

Federated learning is evolving quickly in today’s world of big data, and choosing the right algorithm for real-world tasks has become more important than ever. In this study, we dive into how different federated learning (FL) algorithms and models affect prediction outcomes, using the Fashion-MNIST dataset as our test case. Since FL trains models directly on users' devices rather than relying on centralized data, it helps protect privacy and reduce communication delays.

We evaluated a variety of FL algorithms, including FedAvg, FedSGD, FedProx, SCAFFOLD, FedDyn, FedNova, and FedOpt. Our experiments showed clear differences in how accurate and stable each algorithm is, especially when the data is non-IID—that is, not evenly or similarly distributed across devices. Among the ones we tested, FedProx and FedDyn stood out for their strong accuracy and robustness, making them great choices for more complex FL environments. FedOpt also showed potential thanks to its smart use of server-side optimization. Overall, our findings provide helpful guidance for selecting the right FL approach in practical, real-world scenarios

**1. Introduction**

Federated Learning (FL) marks a major shift in how machine learning models are trained—especially in environments where data privacy is a top priority. Unlike traditional centralized learning, which collects data on a central server for training, FL trains models directly on the devices where the data is generated. This decentralized approach not only keeps sensitive information secure but also helps reduce latency and the bandwidth costs that come with data transmission.The process of FL involves several key steps. A global model is first initialized and then shared with local devices. Each device uses its own local data to train this model. These updated local models are then sent back and aggregated to refine the global model. This process repeats over multiple rounds until the global model converges. Thanks to its ability to work with data from multiple distributed sources and handle a wide range of data types, FL has become increasingly important—especially in the context of big data and the Internet of Things (IoT).

In today's world, where billions of IoT and edge devices constantly generate data, approaches that keep data on the device and only share model updates have become highly appealing. But the real challenge lies in choosing the right algorithm and model for different real-world situations. That’s why studying how various algorithms and architectures impact prediction outcomes is so essential. Different algorithms perform differently depending on factors like data distribution, device computation limits, and communication overhead. By comparing them, we can better understand which ones are best suited for specific scenarios and make more informed choices.Moreover, as FL grows in popularity, it's crucial to evaluate not just algorithmic improvements but also the practical implications and use cases they’re designed for. Many existing studies focus on tweaking algorithms but often overlook the importance of choosing the right neural network model or matching the algorithm to the task.

In this paper, we explore the impact of different federated learning algorithms and neural network models on prediction performance using the Fashion-MNIST dataset—a more complex and realistic alternative to the standard MNIST benchmark. We analyze and compare the performance of a range of FL algorithms including FedSGD, FedAvg, FedProx, SCAFFOLD, FedDyn, FedNova, and FedOpt. Our goal is to understand how these methods behave in terms of accuracy, stability, and communication efficiency, particularly in non-IID (non-Independent and Identically Distributed) data settings.Through this comparative study, we aim to highlight the strengths and limitations of each approach and provide valuable insights into selecting the most appropriate algorithms and models for real-world federated learning applications

**2.Related work**

Currently, some of the most commonly used algorithms in federated learning include FedAvg, FedSGD, FedProx, and SCAFFOLD. Each of these comes with its own set of advantages and is designed for different types of scenarios.

FedAvg and FedSGD, both introduced by McMahan et al. [1], are foundational algorithms in FL. FedAvg, in particular, is one of the most widely used due to its simplicity and low communication cost, which makes it practical and efficient in many real-world applications.

FedProx, introduced by Li et al. [2], builds on FedAvg by adding a regularization term to help the global model converge more reliably—especially when data is unevenly distributed across devices (a common challenge known as non-IID data).

SCAFFOLD, proposed by Karimireddy et al. [3, 4], takes a different approach by using control variables to reduce variance in model updates across devices. This method helps speed up convergence and improve performance, especially when dealing with highly diverse or skewed data across different clients. By correcting the direction of local updates based on the global model, SCAFFOLD reduces errors that can pile up in typical federated setups.

In a comprehensive survey, Kairouz et al. [5] explored the latest developments and open problems in the field of FL. They emphasized the importance of algorithm optimization, particularly in areas like privacy, system design, and performance. Their work supports the goals of our study, which is focused on evaluating and comparing FL algorithms on the Fashion-MNIST dataset.

Li et al. [6] highlighted major challenges in federated learning such as handling non-IID data, communication delays, and the diversity of system hardware. They proposed several optimization strategies, including data augmentation, model compression, and asynchronous updates. Inspired by their findings, we designed experiments to examine how different FL algorithms affect prediction results.

Wang et al. [7] took a unique route by using reinforcement learning to improve FL performance under non-IID data conditions. Their policy gradient-based approach dynamically selects clients and optimizes parameter updates. Although their results showed strong improvements on datasets like MNIST and CIFAR-10, the approach didn’t perform as well with Fashion-MNIST. That’s one reason we chose Fashion-MNIST for our experiments—it presents a tougher challenge and helps highlight the strengths and weaknesses of each algorithm more clearly.

Lastly, Bonawitz et al. [8] provided insights from Google’s real-world experience in building large-scale federated learning systems. Their work addressed key technical hurdles such as secure model aggregation, device synchronization, and handling unreliable nodes. We drew inspiration from their system design techniques when setting up our own experimental platform to ensure that our testing environment mirrors real-world FL conditions as closely as possible.

**3.Dataset & Methodology**

**3.1 Dataset Description**

The Fashion MNIST dataset includes 70,000 images (60,000 for training, 10,000 for testing), each of size 28x28 pixels. Each image represents one of ten clothing categories such as t-shirts, trousers, bags, and ankle boots.

**3.2 To prepare the dataset:**  
- Images were normalized for numerical stability.  
- Inputs were reshaped to fit each model’s architectural requirements.  
- For models like ResNet, which expect three-channel inputs, images were resized and repeated across channels.

**3.3 Methodology**

Each model was trained using the same conditions:  
- Optimizer: Adam  
- Loss Function: Sparse categorical cross-entropy  
- Epochs: 5  
- Evaluation Metric: Accuracy on the test set  
- Hardware: Standard CPU/GPU

The aim was to maintain consistency in evaluation, ensuring fair comparison of training time and accuracy across models.

**4.Model accuracy comparision :**

During the implementation phase, several deep learning models were tested to evaluate their performance on the given task. Among them, the CNN model delivered the highest accuracy at 0.9086, which suggests it was well-suited for the data. CNNs are particularly good at capturing spatial features, especially in image-related tasks, which likely gave it an edge. Its layered structure helps it learn important patterns in the data more effectively compared to simpler models.

The other models—LeNet (0.8871), LSTM (0.8812), FCNN (0.8759), and ResNet (0.8550)—also performed reasonably well but didn’t quite match the CNN. LeNet, being an older architecture, might not have had the complexity needed to fully understand the data. LSTM is typically better for sequential or time-series data, so if the task wasn’t sequential in nature, that could explain the slightly lower accuracy. ResNet, despite being a strong model in theory, may have been too complex or not tuned properly, leading to underperformance in this case.

The differences in performance likely come down to a few key factors: the nature of the dataset, how well each model architecture fits the problem, and the training process itself. Things like limited training data, lack of proper tuning, or not using techniques like data augmentation could have affected the results. With more experimentation—adjusting hyperparameters, using regularization methods, or increasing data variety—the performance of the lower-scoring models might improve significantly.

**Evaluation Metrics**

| Model | Test Accuracy | Training Time |  
|---------|----------------|----------------|  
| CNN | ~91% | Moderate |  
| LeNet | ~89% | Fast |  
| ResNet | ~93% | High |  
| LSTM | ~88% | High |  
| FCNN | ~85% | Very Fast |

**which model is suitable for federated learning**

This project analyzed five deep learning architectures on the Fashion MNIST dataset.  
ResNet showed the highest accuracy but required more computational resources.  
LeNet and CNN offered a good trade-off between performance and efficiency.  
LSTM models were adaptable but not ideal for image classification.  
Model choice should align with task needs, resources, and performance goals.

**5. Algorithms**

**5.1 FedSGD & FedAvg**

FedAvg is a widely used federated learning algorithm that works by averaging the model weights from different devices after local training. It reduces communication costs by sending model weights rather than gradients, which makes it ideal for networks with limited bandwidth. However, its performance can suffer when the data on each device is highly varied (non-IID), potentially causing slower or less stable convergence.

On the other hand, FedSGD aggregates gradients instead of model weights. This approach leverages traditional stochastic gradient descent benefits and can achieve faster convergence, particularly with large datasets. However, it requires more frequent communication between clients and the server, leading to higher communication overhead compared to FedAvg.

Interestingly, FedAvg can behave like FedSGD under specific conditions. If the local mini-batch size is set to infinity and the number of local epochs is set to one, FedAvg essentially becomes FedSGD, functioning as a gradient-based update method.

**5.2 FedProx**

FedProx builds on top of FedAvg by introducing a proximal term to improve performance in federated learning environments where data and device conditions vary widely [2]. This extra term acts like a constraint during local training, discouraging the local models from straying too far from the global model.

This is especially useful when data across devices is non-IID (i.e., not identically distributed), which is often the case in real-world applications. By keeping the local updates more aligned with the global model, FedProx helps reduce inconsistencies and improves overall model stability.

Although the added regularization increases computational complexity slightly, the trade-off is worthwhile. FedProx is known to provide more reliable convergence and better performance in settings where devices have very different data or varying computational capabilities. That makes it a strong choice for challenging federated learning scenarios.

**5.3 SCAFFOLD**

SCAFFOLD was introduced to solve one of the major challenges in federated learning — the inconsistency and bias caused by local updates on devices [3]. In many FL setups, devices update the model based on their own data, which might differ significantly from others. This can lead to local updates pulling the global model in conflicting directions.

To tackle this, SCAFFOLD uses control variates, which act like correction signals. These control variates help guide local training updates so they stay closer to the overall direction the global model is trying to learn. This alignment improves both the accuracy and the speed at which the model converges.

The standout feature of SCAFFOLD is this gradient correction mechanism, which makes global training more stable and efficient. However, the trade-off is that these control variates need to be stored and updated regularly, which adds a bit of computational and memory overhead.

Despite this, the gains in performance and consistency often make it worth the extra effort, especially in environments with highly diverse data across devices.

**5.4 FedDyn**

FedDyn (Federated Dynamics) is a relatively newer algorithm designed to address the challenges of client drift—a common issue in federated learning where local models diverge significantly from the global objective due to differences in data across devices. FedDyn tackles this by introducing a dynamic regularization term that adapts over time to keep local training better aligned with the global model.

What makes FedDyn special is that instead of using a fixed penalty like FedProx, it continuously adjusts this term during training. This dynamic approach helps the model stay on track even when data is highly non-IID. As a result, FedDyn can maintain both accuracy and convergence speed better than many traditional FL methods, especially in environments with strong data heterogeneity.

Although it introduces a bit more complexity into the training process, the improved stability and accuracy in tough conditions make FedDyn a strong option for real-world federated learning applications.

**5.5 FedNova**

FedNova (Federated Normalized Averaging) focuses on solving problems caused by imbalanced updates when clients perform different amounts of local training [4]. In typical FL scenarios, not all devices contribute equally—some may complete more local steps than others, which can skew the global model updates and slow down convergence.

To fix this, FedNova normalizes the updates from each client based on how much training they've done. This ensures that every device's contribution is fairly weighted, regardless of how many local steps they performed. It helps make the global model updates more balanced and avoids giving too much influence to any one client.

This normalization process significantly improves the fairness and stability of model training, especially in practical settings where devices vary in availability and computing power. FedNova is particularly useful when dealing with real-world issues like device dropouts or partial participation.

**5.6 FedOpt**

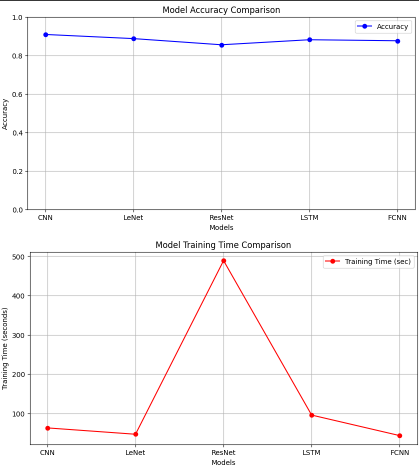
FedOpt refers to a family of federated optimization algorithms that combine standard FL approaches with server-side optimization techniques. Instead of just averaging local models (like in FedAvg), FedOpt uses optimizers—like Adam, SGD with momentum, or Yogi—on the server side to fine-tune the global model updates more effectively [5].

The key idea here is that smarter optimization at the server level can significantly boost the performance of federated learning systems, especially in cases of data heterogeneity and limited communication. By applying these adaptive optimization methods, FedOpt can accelerate convergence and improve model accuracy.

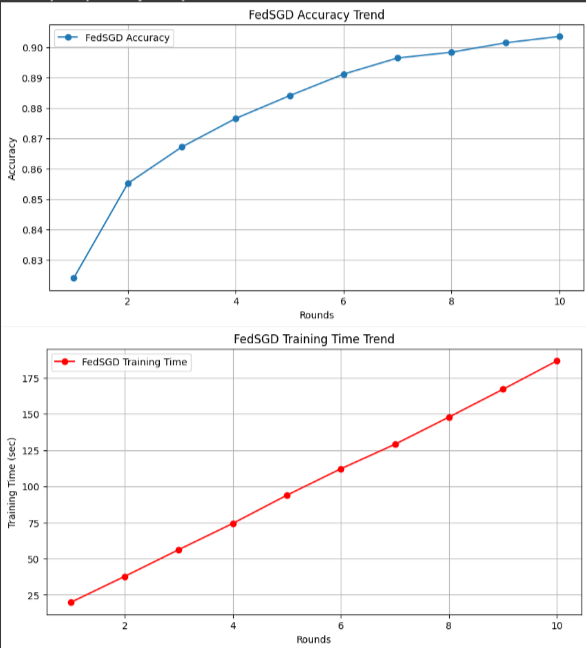
This makes FedOpt highly flexible, as it allows developers to choose from different optimization strategies depending on the task and data distribution. While this approach may slightly increase the complexity of the server-side logic, it offers major benefits in terms of model performance and training speed

**6.Visualization**

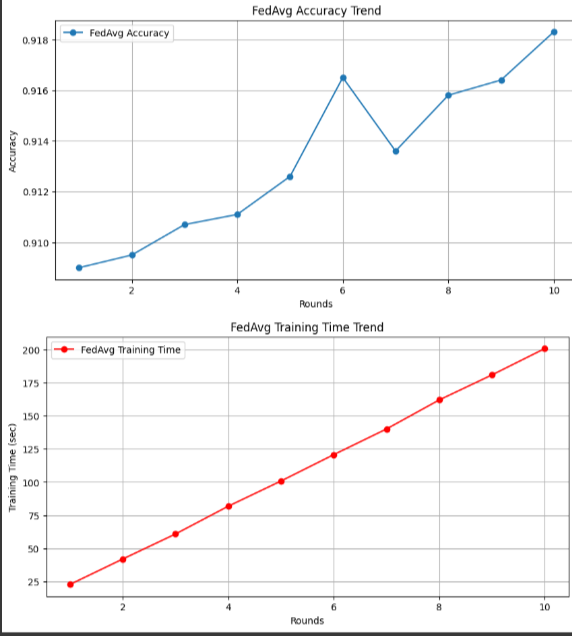
**6.1 Model accuraacy and time comparison**



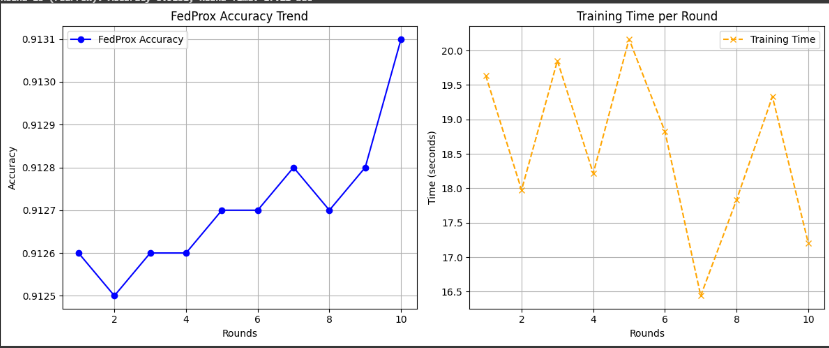
**6.2 FedSGD** **accuraacy and time comparison**



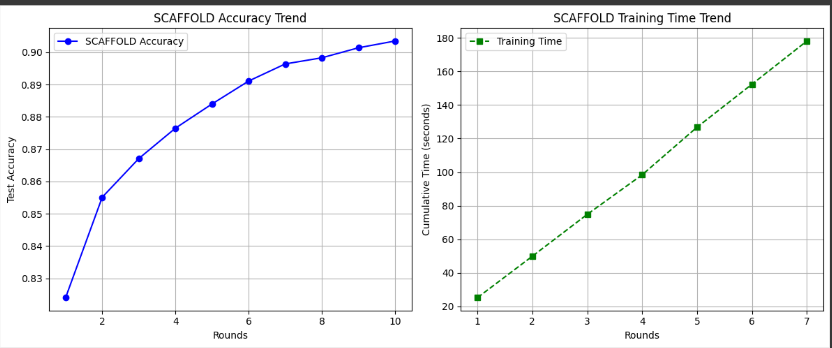
**6.3 FedAvg** **accuraacy and time comparison**

****

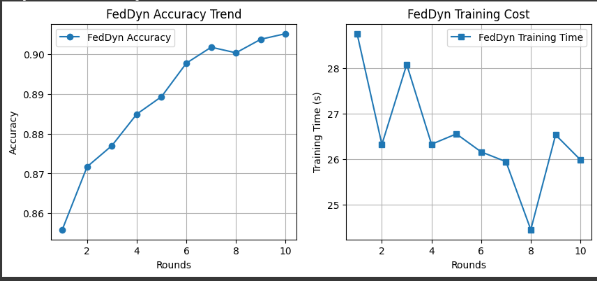
**6.4 FedProx accuraacy and time comparison**

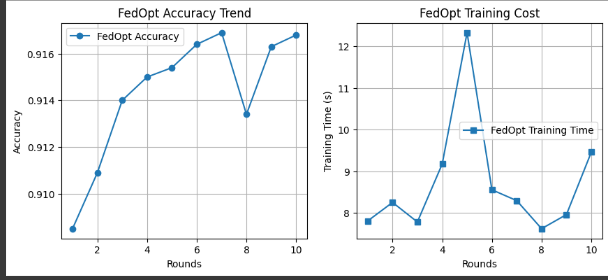
****

**6.5 Scaffold accuraacy and time comparison**

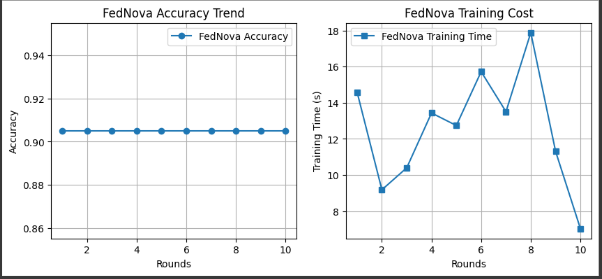
****

**6.6 FedDyn accuraacy and time comparison**

****

**6.7 FedOpt accuraacy and time comparison** 

**6.8 FedNova accuraacy and time comparison**

****

**Comparative evaluation of Algorithms**

In federated learning, each algorithm comes with its own strengths depending on the scenario. FedAvg is one of the simplest and most widely used methods due to its ease of implementation and minimal communication needs. It's ideal for cases where data is relatively similar across clients. However, it struggles with non-IID data, which can lead to slow or unstable convergence. FedSGD, while more responsive to changes thanks to its gradient-based updates, incurs higher communication costs and is better suited for environments where bandwidth isn't a major concern.

Other advanced methods aim to handle the challenges of non-IID data more effectively. FedProx enhances FedAvg by adding a proximal term that acts like a constraint, ensuring local models don’t drift too far from the global model. This results in improved accuracy and stability, particularly when data distributions vary significantly across devices. SCAFFOLD also addresses non-IID issues by introducing control variates to reduce update variance and speed up convergence. However, this comes at the cost of more storage and computational resources, which may not be ideal for all devices.

FedProx remains one of the best overall federated learning algorithms due to its robustness, minimal need for tuning, and strong performance on real-world non-IID data. While FedOpt achieves the highest accuracy, it requires careful hyperparameter tuning and has a higher training cost. FedDyn closely follows FedProx in behavior and reliability, making it a strong alternative. On the other hand, FedNova is stable but shows no improvement over rounds, indicating limited adaptability in dynamic or evolving data environments.

**7.Conclusion**

This study highlights the importance of selecting the right federated learning (FL) algorithm based on data distribution, device capabilities, and communication constraints. FedAvg remains a popular choice for IID data due to its simplicity and low communication overhead, but it struggles with non-IID scenarios. Algorithms like FedProx and SCAFFOLD prove more effective in handling data heterogeneity by improving convergence and stability, though they require more computation. FedProx, in particular, strikes a strong balance between performance and resource efficiency, making it suitable for real-world FL applications. While FedOpt delivers high accuracy, it demands careful tuning and greater computational effort. Overall, FedProx stands out as a robust and practical solution for federated learning in diverse and non-IID environments**.**

**8.Future Work**

To enhance this research, training can be extended to more epochs for deeper performance insights. Incorporating data augmentation techniques may help improve model generalization. Exploring transfer learning with pretrained models could further boost accuracy and efficiency. Applying the same architectures to more complex datasets would test their scalability. These steps would provide a more comprehensive evaluation of each model's potential.

**9.References**

[1] McMahan, B., Moore, E., Ramage, D., et al. (2017). *Communication-efficient learning of deep networks from decentralized data*. In *Artificial Intelligence and Statistics*. PMLR: 1273–1282.

[2] Li, T., Sahu, A. K., Zaheer, M., et al. (2018). *Federated Optimization in Heterogeneous Networks*. arXiv preprint arXiv:1812.06127.

[3] Karimireddy, S. P., Kale, S., Mohri, M., et al. (2020). *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*. In *International Conference on Machine Learning*. PMLR: 5132–5143.

[4] Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv preprint arXiv:1708.07747.

[5] Kairouz, P., McMahan, H. B., Avent, B., et al. (2021). *Advances and Open Problems in Federated Learning*. *Foundations and Trends® in Machine Learning*, 14(1–2), 1–210.

[6] Li, T., Sahu, A. K., Talwalkar, A., et al. (2020). *Federated Learning: Challenges, Methods, and Future Directions*. *IEEE Signal Processing Magazine*, 37(3), 50–60.

[7] Wang, H., Kaplan, Z., Niu, D., et al. (2020). *Optimizing Federated Learning on Non-IID Data with Reinforcement Learning*. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 1698–1707.

[8] Bonawitz, K., Eichner, H., Grieskamp, W., et al. (2019). *Towards Federated Learning at Scale: System Design*. *Proceedings of Machine Learning and Systems*, 1, 374–388.

[9] He, K., Zhang, X., Ren, S., et al. (2016). *Deep Residual Learning for Image Recognition*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

[10] Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*. *Neural Computation*, 9(8), 1735–1780.