Learn together over time

Distributed Multi-frequency time series framework

IN5000: Masters Thesis

Arnob Chowdhury

TU Delft

Distributed Multi-frequency time series framework

by

Arnob Chowdhury

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday August 26, 2025 at 12:30 PM.

Student number:	5977045	
Project duration:	August 1, 2024 – August 26, 2025	
Thesis committee:	Dr. Lydia Chen,	Supervisor, Chair, TU Delft
	Dr. Thiago Guzella,	Company Supervisor, ASML
	Aditya Shankar,	Daily Supervisor, TU Delft
	Dr. Cynthia Liem,	Committee Member, TU Delft

Cover: AI Generated, DALLE, OpenAI

TU Delft

Preface

I would like to express my sincere gratitude to all those who supported me throughout this thesis. This work would not have been possible without the guidance and encouragement of my supervisors, Dr. Lydia Chen, Dr. Thiago Guzella, and Aditya Shankar. Their expertise, patience, and constructive feedback were invaluable to the development of this research.

I am especially grateful to Dr. Lydia Chen for giving me the opportunity to collaborate with ASML, and to Dr. Thiago Guzella for his guidance and constructive criticism during this collaboration. Working in such a professional environment has been a transformative experience that greatly contributed to my academic and personal growth.

Finally, I owe my deepest thanks to my family and friends for their constant support and encouragement. Their belief in me gave me strength during challenging moments. This accomplishment is as much theirs as it is mine.

Arnob Chowdhury
Delft, August 2025

Contents

1 Abstract

Modern industrial systems, from wind-farm monitoring to economic indicators like GDP generate vast amounts of time series data from diverse sources. These data streams are sampled at varying and often inconsistent frequencies, presenting challenges for accurate forecasting. Furthermore, in many real-world scenarios, data are distributed across nodes or tasks, introducing complications due to heterogeneity across tasks. Existing forecasting approaches typically address frequency misalignment and decentralized learning as separate problems, limiting their ability to model real-world deployments effectively.We propose CrossFreqNet, a unified multi-task encoder–decoder architecture that addresses both challenges: (i.) integrating multi-frequency data streams without the need of up or down sampling to match frequency, preserving signal integrity and (ii.) introducing GradBal, a gradient-balancing mechanism that mitigates learning conflicts caused by task heterogeneity and promoting stable convergence across tasks in a distributed learning environment. Across four public benchmarks and one industrial dataset, our model reduces forecasting errors by up to 72% over the best multi-task baseline (UniTS) and up to 48% over PCGrad, a SOTA gradient conflict mitigation method. We do this by answering following research questions:

- **RQ1:** How effective is it to leverage high-frequency time series data to forecast low-frequency time series data compared to resampling them to have a single frequency?
- **RQ2:** Is collaborative learning(MTL) better than non collaborative learning
- **RQ3:** Does performing Gradient Attenuation by using GradBal as a conflict resolution method in time series data from multiple sources help in getting a better model utility?
- **RQ4:** How effectively can we replicate Multi-Task baselines' performance in a truly decentralized setting, both without confidentiality and under confidentiality constraints, using federated algorithms?

Code is made available at `https://github.com/arc-arnob/TS-MTL/`.

2

Research Paper

Learn Together Over Time: Distributed Multi-frequency Time Series Framework

Arnob Chowdhury
TU Delft
Delft, Netherlands

Aditya Shankar
TU Delft
Delft, Netherlands

Thiago Guzella
ASML
Eindhoven, Netherlands

Lydia Chen
TU Delft
Delft, Netherlands

Abstract

Modern industrial systems, from wind-farm monitoring to economic indicators like GDP generate vast amounts of time series data from diverse sources. These data streams are sampled at varying and often inconsistent frequencies, presenting challenges for accurate forecasting. Furthermore, in many real-world scenarios, data are distributed across nodes or tasks, introducing complications due to heterogeneity across tasks. Existing forecasting approaches typically address frequency misalignment and decentralized learning as separate problems, limiting their ability to model real-world deployments effectively. We propose CrossFreqNet, a unified multi-task encoder–decoder architecture that addresses both challenges: (i.) integrating multi-frequency data streams without the need of up or down sampling to match frequency, preserving signal integrity and (ii.) introducing GradBal, a gradient-balancing mechanism that mitigates learning conflicts caused by task heterogeneity and promoting stable convergence across tasks in a distributed learning environment. Across four public benchmarks and one industrial dataset, our model reduces forecasting errors by up to 72% over the best multi-task baseline (UniTS) and up to 48% over PCGrad, a SOTA gradient conflict mitigation method. Code is made available at https://github.com/arc-arnob/TS-MTL/.

1 Introduction

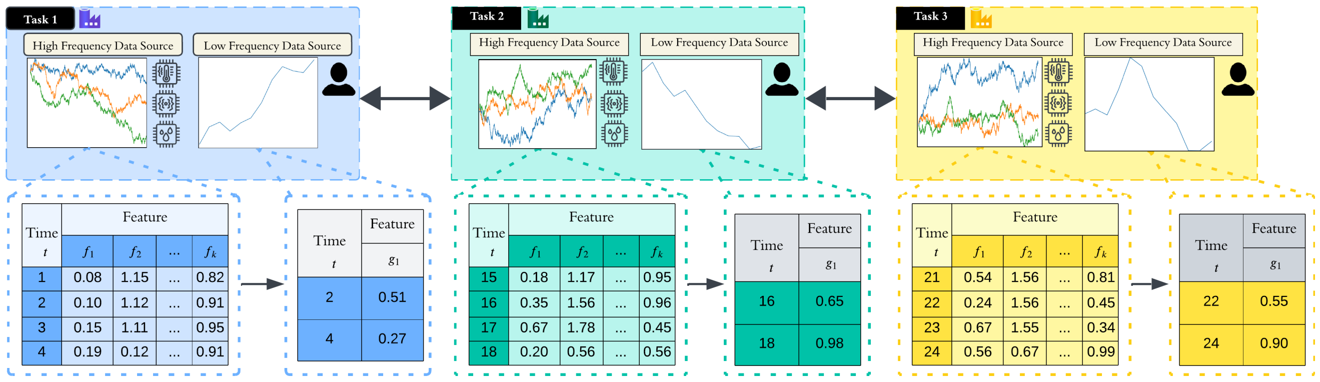
Time series forecasting plays a critical role in several domains, such as industrial monitoring [36, 39], healthcare [15], energy systems [47] and country GDP [24, 37]. However, real-world deployments often have data that is distributed across multiple sites[27, 55] with heterogeneous environments [52, 66], and may be collected at irregular intervals [37, 40, 51, 56]. Consider the scenarios shown in Figure 1 and Example 1.1, involving multiple sites. We treat each site as a *task* that generates related time series data over time t. Each task consists of multi-variate high-frequency input signals $f_1, f_2, \ldots, f_k$, and a low-frequency target signal g_1. This setup defines a *multi-frequency*, *multi-task* forecasting problem, where the objective is to forecast low-frequency outcomes g_1 using historical observations and the high-frequency inputs $f_1, f_2, \ldots, f_k$, while sharing knowledge across tasks. The high-frequency signals capture fine-grained patterns linked to the observed low-frequency targets. Hence, incorporating the high-frequency inputs can help refine the forecasts of the low-frequency signals, as demonstrated in existing work [37]. However, developing such a multi-frequency multi-task forecasting framework is not straightforward due to the following challenges.

Example 1.1

Consider a forecasting scenario involving electricity load management across multiple regions. Each region operates identical monitoring systems equipped with sensors that collect high-frequency data, such as *hourly* weather conditions (e.g., temperature, wind speed) and real-time grid metrics. Additionally, each region periodically reports a low-frequency metric, such as the *daily* peak load demand, reflecting the overall grid stability and depends on the high-frequency inputs. The objective is to forecast the low-frequency (*daily*) peak loads using historical observations, and the high-frequency (*hourly*) sensor data. Since regions experience diverse weather patterns and usage behaviors, a *multi-task* learning approach enables knowledge sharing across sites, improving prediction accuracy.

Challenges.

(i). The first challenge is to extend multi-frequency forecasting from a single-task to a multi-task setting. While encoder-decoder architectures have shown strong performance in handling mixed frequencies, without requiring up- or down-sampling, they are limited to the single-task setting [21, 37, 65]. However, real-world deployments often require handling multiple tasks simultaneously [18, 22]. Two straightforward approaches for multi-task extension are: *data pooling*, by combining all tasks into one large dataset, and *non-collaborative* learning (one model per task). However, each has its flaws: pooling fails to capture task-specific patterns[57], while non-collaborative learning avoids sharing knowledge across tasks.
(ii). The second challenge arises when we implement *collaborative* learning through *multi-task* learning to overcome flaws with data-pooling and non-collaborative learning. It introduces an optimization hurdle which existing multi-task models like UniTS[23] & FATHOM[14] overlook: *negative transfer*, where the model's performance on one task is degraded by the learning process of another[57]. This phenomenon often stems from *gradient conflicts*, where the parameter updates required for different tasks directly oppose each other, leading to unstable convergence and poor generalization [45, 66]. These conflicts are particularly severe when


Task 1
High Frequency Data Source
Low Frequency Data Source
Task 2
High Frequency Data Source
Low Frequency Data Source
Task 3
High Frequency Data Source
Low Frequency Data Source

Time	Feature			
t	f_1	f_2	...	f_k
1	0.08	1.15	...	0.82
2	0.10	1.12	...	0.91
3	0.15	1.11	...	0.95
4	0.19	0.12	...	0.91

Time	Feature
t	g_1
2	0.51
4	0.27

Time	Feature			
t	f_1	f_2	...	f_k
15	0.18	1.17	...	0.95
16	0.35	1.56	...	0.96
17	0.67	1.78	...	0.45
18	0.20	0.56	...	0.56

Time	Feature
t	g_1
16	0.65
18	0.98

Time	Feature			
t	f_1	f_2	...	f_k
21	0.54	1.56	...	0.81
22	0.24	1.56	...	0.45
23	0.67	1.55	...	0.34
24	0.56	0.67	...	0.99

Time	Feature
t	g_1
22	0.55
24	0.90

Figure 1: Mixed-frequency, multi-task forecasting setup with high-frequency inputs ($f_1, f_2, \dots, f_k$), where k is the number of features and low-frequency targets (g_1) across multiple tasks over time t.

tasks are heterogeneous or their data distributions are non-i.i.d.; a common scenario in real-world time series applications [26].

Contributions. To tackle *multi-frequency, multi-task* time series forecasting, we extend the LSTM-based model from [37] with the following key contributions:

Multi-Frequency Multi-Task Forecasting Model. We develop a multi-task model that fuses high- and low-frequency time series across related tasks by extending the single-task encoder–decoder architectures [37] with task-specific heads, unlike simple foundational models that train common shared parameters for all tasks. Extending to multi-task learning, it jointly learns shared and task-specific representations and sets a strong benchmark under full data access. Hence, addressing the flaws with naive data pooling and non-collaborative learning.

Gradient Balancing to Mitigate Task Conflicts. To mitigate *negative transfer* from task conflict in multi-task training, we introduce *GradBal*. This gradient balancing method scales task contributions using pairwise gradient cosine similarity and loss magnitudes, while preserving gradient direction. Unlike methods that consider conflicts detrimental [38, 66], our method considers conflicts informative. This improves convergence and accuracy in non-i.i.d. multi-task forecasting, outperforming strong baselines.

Evaluations. We conduct extensive experiments to validate the effectiveness of our multi-task modeling strategy. First, our results demonstrate the clear benefits of leveraging high-frequency signals rather than scaling them to have the same frequency as a low-frequency signal, preventing data fabrication or loss. Then we highlight the advantages of collaborative multi-task learning over non-collaborative learning, and confirm that gradient conflict mitigation via our proposed GradBal method significantly improves forecasting accuracy. Furthermore, we explore a truly confidential, distributed approach and show that it maintains robust forecasting performance compared to confidentiality-unconstrained multi-task methods.

2 Background and Related Work

Time Series Forecasting. Classical forecasting methods, including ARIMA, ARIMAX, and Vector AutoRegression (VAR), are widely used for their interpretability and efficiency on linear and stationary data [31, 35]. However, they often struggle with asynchronous, nonlinear, and non-stationary time series commonly observed in real world applications [36, 39]. Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) excel at capturing sequential dependencies even with limited data[10, 20, 33]. Transformer-based models, such as PatchTST [50], iTransformer [41], and foundational models like Chronos [7], are powerful in modeling intricate temporal dynamics but typically require large datasets to avoid overfitting [55].

Multi-Frequency Forecasting. Multi-frequency forecasting addresses scenarios with data sampled at multiple intervals [37]. Multi-Time Attention Networks (mTAN) [56] use continuous-time attention with interpolation to handle irregular sampling for classification tasks. Transformer-based methods, such as Scaleformer [54] and Multirate-former [40], capture patterns at multiple temporal resolutions for forecasting. An LSTM-based encoder–decoder framework[37] has been proposed for mixed-frequency prediction, managing different sampling rates but limited to single-task settings. These limitations motivate the need for frameworks integrating multi-frequency capabilities into multi-task environments.

Multi-Task Learning. In Multi-task learning (MTL)[13] we are given $N \geq 2$ tasks each with its own loss function $L_i(\theta)$ where θ is shared parameters and the aim is to find the θ such that:

$$\tilde{\theta} = \arg\min_{\theta \in \mathbb{R}^m} \left\{ L_{global}(\theta) \triangleq \frac{1}{N} \sum_{i=1}^{N} L_i(\theta) \right\}.$$

Recent MTL frameworks, such as Shared-Private Attention Networks [17], FATHOM [14], and UniTS [23], employ explicit MTL by using the concept of shared and task-specific representations with LSTM and transformers. In contrast, implicit MTL pools data from all tasks into a single model, jointly optimizing shared parameters to capture multi-task relationships. These approaches effectively enhance learning outcomes by utilizing shared patterns across tasks, yet they generally do not address multi-frequency data scenarios or mitigate conflicting gradients.

Conflicting tasks. In MTL, a major optimization problem is conflicting tasks caused by conflicting gradients during parameter update. In multi-task learning the parameter update is typically

Table 1: Comparison of State-of-the-art models addressing multi-task and federated settings.

Model/Paper	**Multi-task learning**	**Mixed-Frequency**	**Task Conflict Mitigation**
FATHOM [14]	✓	×	×
Shared-Private Attention [17]	✓	×	×
Scaleformer[54]	×	✓	×
UniTS [23]	✓	×	×
TSDiff [30]	×	×	×
Encoder-Dual-Decoder [37]	×	✓	×
Proposed Model (This Thesis)	✓	✓	✓

Legend: ✓ = Fully supported, × = Not supported

driven by the *mean* gradient of N tasks $g_0 = N^{-1} \sum_{i=1}^{N} g_i$, where $g_i = \nabla L_i(\theta)$. When two tasks disagree, i.e. $\langle g_i, g_0 \rangle < 0$, the step can *harm* task i and cause *negative transfer.* Several algorithms(see Figure 3) adjust g_0 before applying the update. GradNorm rescales individual losses so that all tasks progress at comparable rates while leaving gradient directions unchanged [16]. PCGrad resolves head-on clashes by projecting every task gradient onto the subspace that does not oppose the others, thereby removing direct conflict [66]. PCGrad's complete removal of conflicts may erase shared temporal information, disrupting the LSTM's sequence learning. Finally, CA-Grad frames the update as a constrained optimisation and selects the shortest non-negative combination of task gradients that yields descent for every objective [38]. CAGrad's insistence on simultaneous improvement can constrain the model's flexibility to handle phased or lagged task relationships. While these techniques curb strong conflict, they may also erase subtle lead-lag cues valuable in time series forecasting, motivating softer balancing approaches. Our contributions are summarized in Table 1.

Federated Learning & Confidentiality. In many real world cases raw data cannot be shared hence federated learning (FL) setup trains a global model θ while raw data remain local, making it suitable for confidentiality-sensitive, settings [63].During each round, clients update θ on their data, send the *model update* to a central server, and receive the aggregated model for the next round. Model updates can be sent either as weight deltas i.e. the incremental differences between the locally updated parameters and the previous global parameters after local training (FedAvg) or as raw gradients (FedSGD); although the two are mathematically equivalent, gradient sharing is rarely used because it heightens privacy leakage [68], increases communication bandwidth as gradients has to be sent per batch, and also destabilizes training when those gradients are averaged [44]. When updates may leak information, FL is commonly combined with *differential privacy* (DP). DP-SGD [6] clips per-sample gradients and injects Gaussian noise, yielding an (ε, δ) guarantee that any single record has little influence on the released update but degrades model utility. To maintain model utility, secure aggregation (SecAgg) [11, 61] is used, which ensures the server observes only the sum of client updates, not the individual contributions. Each client masks its update w_i with pair-wise random pads r_{ij} that cancel out when the server computes $W = \sum_{i=1}^{N} w_i$.

3 Methodology

We propose CrossFreqNet, a multi-task forecasting framework for, multi-frequency time series data. Each task is distributed and involves high-frequency inputs and low-frequency targets. Our model extends an encoder-decoder architecture[37], which forms the backbone for all our strategies, with a gradient-balancing mechanism, GradBal, to address inter-task conflicts[1].

Problem Definition. We consider *multi-frequency, multi-task* time series forecasting with N *tasks.* For each task $n \in [N]$ we have high-frequency data $X_n^{HF} \in \mathbb{R}^{D_{HF} \times W_{HF}}$ and low-frequency (LF) data $X_n^{LF} \in \mathbb{R}^{D_{LF} \times W_{LF}}$, where D represents feature dimensions and W represents time window. The HF and LF data streams share the same feature space across all tasks but differ in their sampling frequencies, resulting in a fixed frequency ratio $r = W_{HF}/W_{LF}$. Each task n predicts future LF quality metrics $Y_n \in \mathbb{R}^P$ using both historical LF data X_n^{LF} and the corresponding HF sensor readings X_n^{HF}, where P is the prediction horizon.

CrossFreqNet: A Multi-Task Multi-frequency Encoder-Decoder Model with Gradient Balancing.

Our approach, *multi-frequency, multi-task* encoder-decoder with Gradient Balancing architecture, illustrated in Figure 2, extends the LSTM-based framework[37] to a multi-task setup to address the issue with naive *data pooling* and *collaborative learning* and incorporates a gradient balancing mechanism we call *GradBal* to mitigate *task conflicts.* The model uses a hard parameter-sharing architecture: a shared LSTM-based encoder-decoder is trained across all tasks, while task-specific MLP output layers capture unique task-specific characteristics.

The key innovation in our model is the gradient balancing component(GradBal, Algorithm 2.). Inspired by the gradient surgery technique used for images [16, 38, 66], we adapt the method to the time series forecasting setting to address the gradient conflicts inherent in multi-task learning. In hard parameter sharing architectures, naively averaging gradients from diverse task distributions can lead to destructive interference and hinder convergence. Our proposed centralized approach with GradBal fills this gap by softly attenuating gradient conflicts rather than eliminating them, thus allowing mild negative interactions as implicit regularization. Designed explicitly for temporal scenarios, GradBal leverages periodic conflicts inherent in time series data, treating them as informative rather than detrimental. Algorithms 1 and 2 detail the forecasting model structure, with f[37] as backbone, and our gradient balancing procedure, respectively.

Our gradient balancing strategy scales each task's gradient contribution based on the pairwise cosine similarity of task gradients and their corresponding loss magnitudes. This approach explicitly addresses the issue of gradient conflicts, which can lead to negative transfer during multi-task optimization. Given a set of gradients $\mathcal{G} = \{g_1, \ldots, g_n\}$ and their corresponding task losses $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, the GradBal operates as follows:

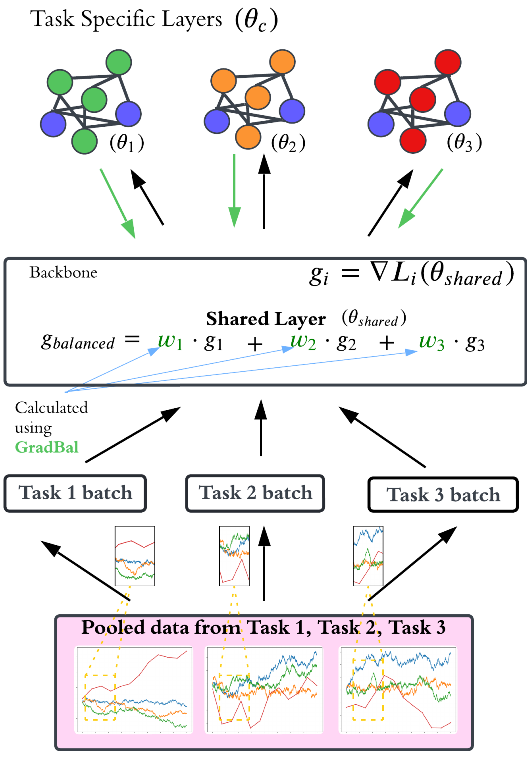
First, initial task weights are computed proportionally to each task's loss magnitude:

$$w_n \propto L_n \quad \forall n \in 1, \ldots, N \tag{1}$$

Next, for every pair of distinct tasks i, j , the cosine similarity between their gradients is calculated:

$$\rho_{i,j} = \cos(g_i, g_j) \tag{2}$$

[1]This architecture was the result of multiple modeling iterations, most prominent of which are presented in the Appendix 8.1



Task Specific Layers (θ_c)
(θ_1)
(θ_2)
(θ_3)
Backbone
$g_i = \nabla L_i(\theta_{shared})$
Shared Layer (θ_{shared})
$g_{balanced} = w_1 \cdot g_1 + w_2 \cdot g_2 + w_3 \cdot g_3$
Calculated using GradBal
Task 1 batch
Task 2 batch
Task 3 batch
Pooled data from Task 1, Task 2, Task 3

Figure 2: Gradient Balancing in CrossFreqNet where backbone(f in Algorithm 1) is a multi-frequency encoder-decoder LSTM-based model by Lin et al. [37]

If two tasks have negatively aligned gradients, their respective weights are scaled down to attenuate the negative impact on the shared parameters, controlled by a hyperparameter :

$$w_i \leftarrow w_i \cdot (1 - \alpha \cdot |\rho_{i,j}|) \tag{3}$$

After adjusting all conflicting task pairs, the weights are normalized to ensure a proper convex combination:

$$w_n \leftarrow \frac{w_n}{\sum_{j=1}^{N} w_j} \tag{4}$$

Finally, the balanced gradient is computed as a weighted sum of the adjusted task gradients:

$$g^{\text{balanced}} = \sum_{n=1}^{N} w_n \cdot g_n \tag{5}$$

This procedure ensures mild gradient conflicts are preserved to respect the lead-lag relationship[67] in time series and as implicit regularization, promoting stable optimization and improved generalization. Unlike approaches such as PCGrad, which entirely remove conflicts by projecting gradients orthogonally, our method softly attenuates conflicts, preserving valuable temporal interactions[58] inherent in multi-task multi-frequency time series. Figure 3 illustrates the conceptual difference between gradient adjustments in GradNorm[16], PCGrad[62], CAGrad[38] and our Gradient Balancing (GradBal) method. In summary, our gradient balancing strategy dynamically scales each task's gradient contribution based on pairwise cosine similarity and task loss magnitudes, mitigating negative transfer and promoting stable optimization. It is parameterized by a hyperparameter α where a higher α causes a stronger downscaling of the task's contribution, effectively reducing the impact of conflicting tasks on the shared model.

Algorithm 1 CrossFreqNet: Multi-Task Forecasting Model with Gradient Balancing

```
Require: Pooled dataset $\mathcal{D} = \bigcup_{n=1}^{N} \mathcal{D}_n$, model $\mathcal{M}_\theta$, learning rate
    $\eta$, number of epochs $T$ and $N$ is number of tasks.
 1: Initialize shared parameters $\theta_{\text{shared}}$, task-specific output layers
    $\{\theta_n^{\text{out}}\}_{n=1}^{N}$
 2: for epoch = 1 to $T$ do
 3:     Initialize gradient buffer $\mathcal{G} \leftarrow \emptyset$, loss buffer $\mathcal{L} \leftarrow \emptyset$
 4:     for each task $n \in \{1, \dots, N\}$ do
 5:         Sample mini-batch $\mathcal{B}_n$ from $\mathcal{D}_n$
 6:         Compute loss $\mathcal{L}_n = \ell(f(\mathcal{B}_n; \theta_{\text{shared}}, \theta_n^{\text{out}}))$
 7:         Backpropagate $\nabla_{\theta_{\text{shared}}} \mathcal{L}_n$ and store as $g_n$
 8:         Save $g_n$ in $\mathcal{G}$ and $\mathcal{L}_n$ in $\mathcal{L}$
 9:     end for
10:     $g^{\text{balanced}} \leftarrow$ GradientBalance($\mathcal{G}, \mathcal{L}$)   // See Algorithm 2
11:     Update shared parameters: $\theta_{\text{shared}} \leftarrow \theta_{\text{shared}} - \eta \cdot g^{\text{balanced}}$
12:     for each task $n$ do
13:         Backpropagate $\nabla_{\theta_n^{\text{out}}} \mathcal{L}_n$
14:         Update: $\theta_n^{\text{out}} \leftarrow \theta_n^{\text{out}} - \eta \cdot \nabla_{\theta_c^{\text{out}}} \mathcal{L}_n$
15:     end for
16: end for
17: return Trained model $\mathcal{M}_\theta$
```

Algorithm 2 GradientBalance($\mathcal{G}, \mathcal{L}$) (GradBal)

```
Require: Gradients $\mathcal{G} = \{g_1, \dots, g_n\}$, losses $\mathcal{L} = \{\mathcal{L}_1, \dots, \mathcal{L}_C\}$
 1: Compute initial weights $w_c \propto \mathcal{L}_n$ for all tasks $n$
 2: for each pair $(i, j)$ with $i \neq j$ do
 3:     Compute cosine similarity $\rho_{i,j} = \cos(g_i, g_j)$
 4:     if $\rho_{i,j} < 0$ then
 5:         Adjust $w_i \leftarrow w_i \cdot (1 - \alpha \cdot |\rho_{i,j}|)$
 6:     end if
 7: end for
 8: Normalize weights: $w_n \leftarrow \frac{w_n}{\sum_j w_j}$
 9: Compute balanced gradient: $g^{\text{balanced}} = \sum_{c=1}^{C} w_n \cdot g_n$
10: return $g^{\text{balanced}}$
```

GradBal stands out as an intuitive solution for achieving strong performance in multi-task time series forecasting. Unlike PCGrad, which relies on computationally expensive sequential gradient projections to eliminate pairwise conflicts, or CAGrad, which solves constrained optimization problems to balance worst-case task performance, GradBal employs a straightforward conflict-aware reweighting approach. This eliminates the need for complex projection operations or iterative suboptimizations, making it significantly easier to implement and integrate into existing frameworks. Moreover, using direct similarity-based weighting instead of geometric projections, GradBal converges 2.1x faster than PCGrad(See Figure 5) while providing interpretable insights into task conflicts.

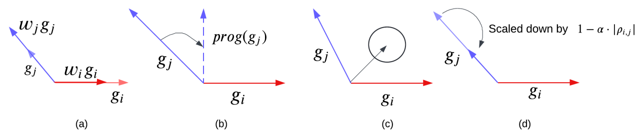


Figure 3: Gradient–conflict strategies. (a) GradNorm, (b) PC-Grad, (c) CAGrad, (d) GradBal.

4 Experiment

We do a comprehensive set of experiments to evaluate how well our multi-task modeling strategy works to address the following research questions.

- **RQ1:** How effective is it to leverage high-frequency time series data to forecast low-frequency time series data compared to resampling them to have a single frequency?
- **RQ2:** Is collaborative learning(MTL) better than non collaborative learning
- **RQ3:** Does performing Gradient Attenuation by using GradBal as a conflict resolution method in time series data from multiple sources help in getting a better model utility?
- **RQ4:** How effectively can we replicate Multi-Task baselines' performance in a truly decentralized setting, both without confidentiality and under confidentiality constraints, using federated algorithms?

Baselines

In this section, we compare our *multi-frequency multi-task* model with two types of baseline approaches: *non-collaborative* and *collaborative*. We also evaluate our model in a confidentiality-preserving federated learning environment.

For non-collaborative and collaborative baselines, we selected three representative models based on empirical experiments detailed in Appendix 8.1. These models demonstrated the highest accuracy compared to other strategies tested: *ARIMAX* [12], a traditional time series model; *UniTS* [23], a transformer-based multi-task model; and *TSDiff* [30], a diffusion-based model. We excluded Scaleformer since other baselines perform better already. Additional baseline experiments are detailed in the Appendix. These selected baselines operate at a single frequency, assuming alignment between exogenous and endogenous variables.

Non-Collaborative baselines. Each task is trained separately without any information shared between tasks.

Collaborative baselines. all task data are pooled in this setting to enable shared learning. We distinguish two types: implicit collaboration or simple data pooling, where task relationships are not modeled (e.g., ARIMAX and TSDiff trained on concatenated data or random batches), and explicit collaboration, where architectures or optimization explicitly encode task structure (e.g., UniTS with task-specific heads, CrossFreqNet with gradient balancing). This distinction also allows us to assess the benefit of naive data pooling versus structured multi-task learning.

The above two baselines allow us to quantify two distinct contributions: the benefit of cross-task knowledge-sharing, Figure **??**, and the advantage of explicitly modeling multiple data frequencies.

Subsequently, we also evaluate different task conflict resolution techniques, including our proposed Gradient Balancing (GradBal) as well to assess their effectiveness in stabilizing training and improving generalization in multi-task, mixed-frequency forecasting scenarios.. Further, this also allows us to set an empirical upper bound on the performance of multi-task learning for when we introduce confidentiality constraints on information sharing

Federated Baselines. Finally, we also experiment with three federated setting and compare performance of the multi-frequency encoder-decoder model, the best performing one, in a confidentiality-preserving and non-confidentiality -preserving setting. Data remains on each site, and only model updates are exchanged with a central server. We benchmark three widely used FL optimizers, using multi-frequency encoder-decoder(MF-ED[37]) as backbone, as backbone as it performs the best in non federated setting: (i) FedAvg [44], which averages local weight deltas; (ii) FedProx [34], which adds a proximal term to stabilize training under client heterogeneity; and (iii) SCAFFOLD [28], which employs control variates to reduce gradient drift. Details of how we implement a confidentiality-preserving federated learning environment are presented in the Appendix 8.2 & 8.4, and a more detailed confidentiality analysis is presented in Appendix 9.3.

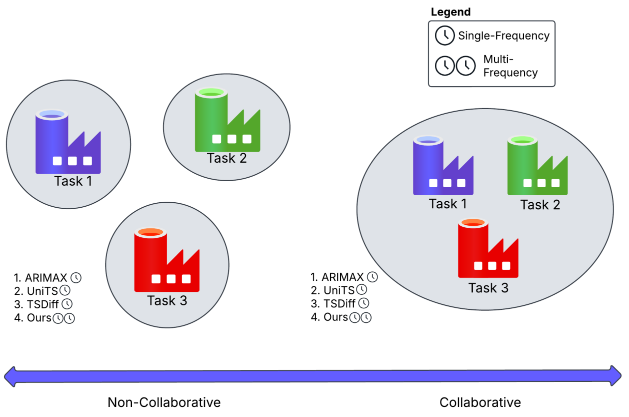


Figure 4: A Schematic comparison of two approaches for time series forecasting

Data.

We evaluate our framework on five environmental, energy, and industrial datasets. The *Air Quality Dataset*[4] includes hourly pollutant and weather readings from March 2013 to February 2017. To simulate a mixed-frequency setting, we interpolate the target variable-carbon monoxide (CO)-to 15-minute intervals while retaining hourly resolution for all input features. The *Wind Forecast Dataset*[2] contains hourly wind power generation and hourly pressure readings for seven wind farms. We use a 2-month subset to mirror the data volume typical of industrial deployments. We interpolate wind power generation to be 15 minutes. The *Load Forecast Dataset*[1] includes hourly electricity load values along with 15-minute temperature readings across 20 U.S. grid zones, providing a naturally multi-rate input structure. The *Spain Electricity Shortfall Dataset*[5] comprises 3-hourly electricity shortfall records and weather data from five Spanish cities over 2015–2017. We interpolate the weather inputs to 6-hour intervals while keeping the shortfall targets at their original resolution. The dataset represents a multi-source, single-target setting, as city-level weather features correspond to a single national shortfall measurement. Finally, we include a proprietary real-world dataset from a leading semiconductor manufacturing equipment manufacturer, containing device-level sensor and quality metrics collected at varying sampling rates.

Training Approach.

We do *sliding-window sampling* where each raw time series is first split once chronologically into an 80% training portion and a 20% hold-out test portion.

Inside the training segment, we generate many encoder–decoder pairs:

- a high-frequency window of length L^{HF} (e.g., 64 points),
- a low-frequency window of length L^{LF} (e.g., 32 points), and
- a target at horizon H steps ahead.

Model Hyperparameter. All deep models use the Adam optimizer. Below, we summarize the few key hyperparameters required to reproduce our results.

ARIMAX. For each site, we run the Augmented Dickey–Fuller test (significance level 0.05). If the series is non-stationary, we differ once ($d = 1$); otherwise, we keep $d = 0$. The autoregressive order is 33 ($p = 32$), with one moving-average term ($q = 1$).

UniTS. We use the default settings from the official repository, modifying only a few values: three encoder layers, model dimension $d_{\text{model}} = 64$, patch size of 1, and stride of 1. The learning rate is 1e-3 with weight decay 5e-6. We train for 30 epochs using a batch size of 16 and gradient accumulation of 32 steps; gradient clipping is set to 100.

TSDiff. Our diffusion baseline uses a hidden size of 64, sinusoidal time embeddings of size 128, and 100 denoising steps with $\beta \in [10^{-4}, 0.1]$. We use guidance scale 1.0, learning rate 1e-3, batch size 16, and train for 30 epochs.

Conflict-resolution variants. PCGrad requires no extra hyperparameters. CAGrad introduces a single hyper-parameter, the simplex radius. We sweep over 0.1, 0.2, 0.3, 0.4 and report results with the most stable value (0.4). GradBal (our proposal) uses one factor α to attenuate noisy gradients. We use $\alpha = 0.5$ throughout.

Our model. We use a two-layer LSTM encoder with 128 units and dual decoders (one for HF and one for LF), each with 64 units. Dropout is set to 0.2. We use batch size 32, learning rate 1e-3, and train for 30 epochs. The input windows L_{HF}, L_{LF} are dataset dependent, and forecasting horizon $H \in 1$. HF: LF ratio is also dataset-specific but fixed during training and should be an integer.

Federated Learning. In a federated setting, each task owns a model described in the previous paragraph, and training is done over 10 rounds and 30 epochs of local training. We assume 80% trusted tasks for our hybrid differential confidentiality and secured aggregation method. For Differential confidentiality, we use $\sigma_e = 0.5$ and $\sigma_g = 0.2$ for air quality, load, wind, and Spain datasets, and $\sigma_e = 0.4$ and $\sigma_g = 0.5$ for the industry dataset. They are calculated using the RDP accountant library provided by Opacus[46]. It calculates the optimal noise needed to the desired ϵ and δ parameters using parameters like number of rounds, number of local epochs, dataset size, and batch size. The above chosen values of $\sigma\epsilon = 3$ and $\delta < 1/|d|$ where d is the size of the dataset[6] for both encoder level noise and DP-SGD, respectively.

Evaluation Metric.

We report the **Mean Absolute Error (MAE)** as our primary accuracy metric, accompanied by the **Standard Error of the Mean (SEM)**. Formally, the SEM is computed as:

$$\text{SEM} = \frac{s}{\sqrt{n}} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}}{\sqrt{n}},$$

where s is the sample standard deviation, n is the number of repeated experiments or folds, x_i denotes the MAE for the i^{th} experiment, and $\bar{x}$ is the mean MAE. SEM provides a measure of uncertainty and indicates how precisely the reported MAE estimates the true population error. Smaller SEM values indicate more consistent results across repetitions and thus higher reliability of our evaluation.

Evaluation.

Table 2 benchmarks our approach against non-collaborative and collaborative baselines. Table 3 evaluates our multi-task optimization against existing methods, while Table 4 assesses multi-frequency model performance in a federated setting. Experiments span five datasets with forecast horizon $H = 1$ & $H = 2$; longer horizons are reported in Appendix 9.2.

Comparison for Single Frequency vs Multi Frequency Modeling(RQ1). In the *non-collaborative* learning strategy shown in Table 2, our model consistently outperforms all single-frequency baselines: ARIMAX, UniTS, and TSDiff, confirming the benefit of using multi-frequency inputs. For Load dataset at $H = 1$ we see 64% reduction in MAE compared to the best performing baseline and 20% reduction in MAE for real-world industry dataset in non-collaborative setting. Similar gains are observed for $H = 2$ as well. These improvements primarily stem from directly leveraging high-frequency inputs without up- or down-sampling, thereby preserving short-term patterns often lost in baselines due to temporal smoothing. While ARIMAX and LSTM-based models capture temporal dependencies via explicit sequential modeling, architectures such as UniTS's transformer encoder and TSDiff's non-autoregressive diffusion process employ parallel processing mechanisms. In data-limited, high-frequency settings, these mechanisms may be less effective at modeling fine-grained temporal patterns, contributing to the observed performance gap. The Spain dataset's poor performance in non-collaborative training reflects the limitation of independent task learning for multi-site scenarios. Without multi-task optimization, our model cannot exploit the beneficial knowledge sharing across related sites, which explains why the collaborative variant shows dramatic improvement (60% MAE reduction)

Comparison for collaborative vs non-collaborative approach (RQ2). Table 2 also shows that in the *collaborative* learning strategy our method without any task conflict resolution(*Ours*$^{-}$) does not perform better consistently, hence the performance results remain inconclusive. However, with *GradBal*(*Ours*$^{+}$) our method

Strategy	Model	Air Quality		Load		Wind		Spain		Industry	
		H1	H2	H1	H2	H1	H2	H1	H2	H1	H2
Non-Collaborative	ARIMAX	0.33±0.002	0.35±0.002	0.47±0.001	0.56±0.001	0.82±0.003	0.83±0.002	1.01±0.001	1.05±0.001	0.24±0.001	0.34±0.002
	TSDiff	1.89±0.040	1.78±0.050	1.26±0.020	0.96±0.020	1.07±0.020	1.03±0.040	1.37±0.040	1.17±0.030	2.14±0.020	1.98±0.040
	UniTS	0.30±0.030	0.35±0.020	0.47±0.030	0.55±0.020	0.70±0.015	0.75±0.010	**0.23±0.018**	**0.28±0.020**	0.51±0.020	0.52±0.040
	Ours^{-}	**0.20**±0.004	**0.29**±0.004	**0.17**±0.003	**0.19**±0.003	**0.53**±0.004	**0.54**±0.004	0.27±0.005	0.43±0.006	**0.19**±0.002	**0.30**±0.004
Collaborative	ARIMAX	0.35±0.003	0.31±0.003	0.43±0.002	0.57±0.003	0.74±0.003	0.76±0.003	0.83±0.004	0.94±0.005	0.28±0.003	0.35±0.003
	TSDiff [21]	1.05±0.040	0.95±0.050	1.26±0.020	0.96±0.020	0.96±0.020	0.94±0.040	1.18±0.040	1.05±0.030	1.75±0.040	1.31±0.040
	UniTS [15]	0.29±0.030	0.39±0.040	0.40±0.040	0.50±0.040	0.77±0.020	0.79±0.050	0.21±0.040	0.25±0.050	0.30±0.020	0.37±0.040
	Ours^{-}	0.20±0.003	0.33±0.003	0.12±0.001	0.14±0.002	0.82±0.002	0.94±0.004	0.42±0.003	0.40±0.003	0.17±0.003	0.36±0.003
	Ours^{+}	**0.18**±0.002	**0.26**±0.002	**0.11**±0.001	**0.16**±0.002	**0.63**±0.002	**0.72**±0.003	**0.11**±0.003	**0.14**±0.004	**0.09**±0.002	**0.17**±0.004

Table 2: Mean Absolute Error (↓) with standard errors (SEM). Bold indicates best performance per dataset-horizon; underlined shows runner-up within each strategy group (Non-Collaborative vs Collaborative). "Ours^{+}" is CrossFreqNet; "Ours^{-}" is CrossFreqNet without conflict resolution.

performs best consistently. For example, on the Load dataset at $H = 1$, we reduce the MAE by up to 71% compared to UniTS in *Ours*$^{-}$, which further improves with *Ours*$^{+}$. The improvement is more pronounced in the Industry dataset, where we see an improvement of 52% compared to the best-performing model in *Ours*$^{+}$, which is an improvement over *Ours*$^{-}$ by 69%. This shows that a naive multi-task learning (MTL) approach, which does not consider the heterogeneity of tasks, fails to yield optimal results. We achieve consistently superior performance only by adding our novel task conflict mitigation technique in *Ours*$^{+}$. The results also demonstrate that a multi-task learning framework can outperform both non-collaborative (one model per task) and naive data pooling approaches.

Impact of Multitask Optimization(RQ3). Table 3 shows com-

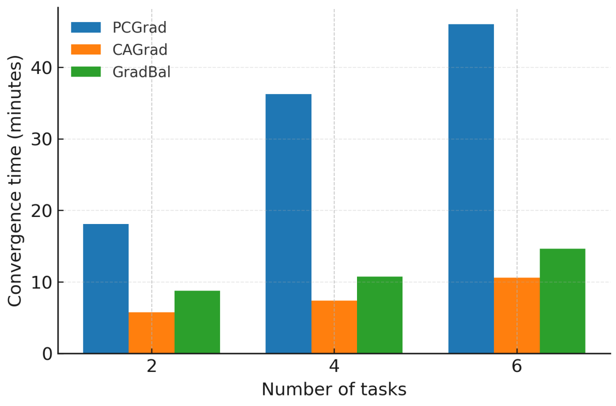


Figure 5: Convergence time (in minutes) for PCGrad, CAGrad, and GradBal across different numbers of tasks.

parison of different multi-task optimization techniques compared to ours. We see that gradient balancing(Ours^{+}) does improves performance compared to no gradient balancing(Ours^{-}). Our method also beats PCGrad(second best). In multi-task learning for time series forecasting with an LSTM model, attenuating conflicting gradients rather than fully deconflicting them (as in PCGrad) leads to better performance by preserving beneficial shared information across tasks On the Spain dataset at horizon $H = 1$, GradBal achieves an MAE improvement of 48% compared to best performing Units-PCGrad. GradBal is particularly effective for time series data, where temporal dependencies create gradient interactions that are less detrimental and potentially more informative than those in image-based tasks. By reducing the effect of conflicts instead of eliminating them, the model maintains a flexible balance between task-specific and shared learning. Attenuation is also known to acts as a form of regularization, similar to gradient noise, helping avoid overfitting and improving generalization, especially with multiple tasks[49]. However, GradBal's advantages diminish with transformer-based models like UniTS and diffusion-based models like TSDiff. Unlike LSTMs, which sequentially process data and explicitly preserve temporal dependencies through their recurrent structure, transformers rely on parallel processing and positional encodings, while diffusion models employ a non-autoregressive denoising process. These architectural differences alter gradient dynamics, reducing the effectiveness of attenuation tailored to LSTM-based sequential learning. Consequently, GradBal excels primarily with LSTMs, where its gradient balancing aligns closely with the temporal nature of time series forecasting.

As shown in Figure 5, GradBal achieves convergence speeds competitive with CAGrad, despite operating at $O(N^2)$ complexity compared to CAGrad's $O(N)$ implementation. While CAGrad maintains a speed advantage through its constrained optimization formulation, GradBal delivers an optimal balance of computational efficiency, algorithmic simplicity and model utility, making it particularly well-suited for practical multi-task scenarios where ease of implementation is important as well.

Ablation: Confidentiality Analysis(RQ4). We investigate the impact of confidentiality-preserving federated learning (FL) methods on forecasting performance across heterogeneous, mixed-frequency tasks, using FedAvg, FedProx, and SCAFFOLD under conditions with and without differential privacy (DP). Table 4 provides comparative results.

Analyzing the results, we observe that introducing differential privacy consistently increases MAE values compared to the multi-task model across almost all scenarios due to the inherent trade-off between confidentiality and model accuracy. Specifically, FedAvg shows modest degradation when incorporating DP, with MAE increasing on average by approximately 24%, which is indeed the cost of confidentiality. Interestingly, on the real-world industry dataset

Model	Technique	Air Quality		Load		Wind		Spain		Industry	
		H=1	H=2	H=1	H=2	H=1	H=2	H=1	H=2	H=1	H=2
TSDiff	wo conflict res.	1.05±0.040	0.95±0.05	1.26±0.020	0.96±0.02	0.96±0.020	0.94±0.04	1.18±0.040	1.05±0.03	1.75±0.040	1.31±0.02
	PCGrad	1.18±0.040	0.95±0.05	1.25±0.020	0.80±0.02	0.96±0.020	0.92±0.04	1.15±0.040	1.03±0.03	2.11±0.040	1.98±0.04
	CAGrad	0.94±0.010	0.91±0.05	0.71±0.020	0.71±0.02	1.01±0.030	1.01±0.03	1.09±0.040	1.09±0.03	0.98±0.040	0.99±0.04
	GradBal	1.03±0.030	0.94±0.04	1.26±0.030	0.92±0.03	0.96±0.040	0.92±0.04	0.22±0.040	0.26±0.05	1.95±0.030	1.98±0.03
UniTS	wo conflict res.	0.29±0.030	0.39±0.04	0.40±0.040	0.50±0.04	0.77±0.020	0.79±0.05	0.21±0.040	0.25±0.05	0.30±0.020	0.37±0.04
	PCGrad	0.22±0.030	0.26±0.04	0.37±0.030	0.46±0.04	0.73±0.040	0.72±0.06	0.19±0.030	0.24±0.05	0.28±0.020	0.36±0.04
	CAGrad	0.26±0.030	0.30±0.04	0.41±0.040	0.50±0.04	0.81±0.050	0.77±0.05	0.55±0.040	0.55±0.06	0.17±0.030	0.25±0.04
	GradBal	0.23±0.030	0.25±0.04	0.40±0.040	0.48±0.04	0.71±0.050	0.74±0.06	0.22±0.040	0.26±0.05	0.25±0.030	0.32±0.04
Ours$^{+/-}$	wo conflict res(Ours^{-}).	0.20±0.003	0.33±0.003	0.12±0.001	**0.14±0.002**	0.82±0.002	0.94±0.004	0.42±0.003	0.60±0.003	0.29±0.003	0.32±0.003
	PCGrad	0.18±0.002	0.28±0.002	0.12±0.002	**0.14±0.003**	0.73±0.003	0.82±0.004	0.40±0.003	0.69±0.003	0.21±0.002	0.38±0.002
	CAGrad	0.78±0.002	0.78±0.002	0.80±0.003	0.80±0.003	0.80±0.003	0.80±0.002	0.77±0.003	0.78±0.003	0.55±0.002	0.55±0.002
	GradBal(Ours^{+})	**0.18±0.002**	**0.25±0.002**	**0.11±0.001**	0.16±0.002	**0.63±0.002**	**0.72±0.003**	**0.11±0.003**	**0.14±0.004**	**0.09±0.002**	**0.17±0.004**

Table 3: Mean Absolute Error (↓), standard error of mean (SEM), for horizons H=1 and H=2. Bold marks the lowest MAE and underline the second-lowest within each column, where Ours^{+} is CrossFreqNet with GradBal and Ours^{-} CrossFreqNet without any conflict resolution technique.

Algorithm	confidentiality	Air Quality		Load		Wind		Spain		Industry	
		H=1	H=2	H=1	H=2	H=1	H=2	H=1	H=2	H=1	H=2
FedAvg	No Confidentiality	0.19±0.001	0.30±0.003	0.12±0.002	0.14±0.002	0.71±0.003	0.72±0.004	0.41±0.003	0.75±0.004	0.28±0.001	0.50±0.003
	Confidentiality	0.20±0.001	0.32±0.003	0.17±0.002	0.28±0.003	0.74±0.003	**0.75**±0.002	0.45±0.002	0.62±0.002	**0.16**±0.001	**0.33**±0.003
FedProx	No Confidentiality	0.19±0.002	0.28±0.002	0.11±0.001	0.20±0.002	0.70±0.003	0.84±0.004	0.32±0.002	0.55±0.003	0.14±0.002	0.32±0.003
	Confidentiality	**0.19**±0.002	**0.31**±0.005	**0.16**±0.002	**0.21**±0.004	**0.72**±0.003	0.88±0.005	**0.44**±0.002	**0.61**±0.005	0.18±0.003	0.34±0.004
SCAFFOLD	No Confidentiality	0.45±0.004	0.50±0.005	0.32±0.003	0.45±0.004	0.83±0.004	0.88±0.005	0.76±0.004	0.78±0.005	1.59±0.004	1.87±0.003
	Confidentiality	0.50±0.004	0.54±0.005	0.35±0.003	0.49±0.004	0.88±0.004	0.91±0.005	0.78±0.004	0.80±0.005	2.27±0.003	4.21±0.004

Table 4: Federated learning on heterogeneous mixed-frequency tasks. We compare FedAvg, FedProx and SCAFFOLD with and without differential-confidentiality noise (ϵ_{enc} = 3 and ϵ_{grad} = 3) for horizon 1 (H = 1) and horizon 2 (H = 2)

at $H = 1$ and $H = 2$, MAE decreases by 42% and 34% respectively, indicating that the calibrated noise added acts as a regularizer[49] in FedAvg.

FedProx exhibits similar trends, with a slightly narrower gap between confidentiality and non-confidentiality settings. For example, on the Industry dataset at horizon 1, MAE worsens from 0.14 (confidentiality) to 0.18 (non-confidentiality), demonstrating a 28% deterioration when confidentiality measures are applied. Interestingly, FedProx maintains generally lower MAE values than FedAvg in some cases. In contrast, it is the opposite in others, suggesting its proximal optimization mechanism provides additional robustness to the confidentiality-induced noise, but not so in others.

SCAFFOLD consistently shows the highest MAE values among the evaluated methods, particularly when introducing DP. For instance, on the Industry dataset, horizon 1, MAE deteriorates significantly from 1.59 (no confidentiality) to 2.27 (confidentiality), a 42% increase, clearly highlighting its sensitivity to confidentiality-preserving noise.

FedAvg and FedProx show comparable performance, though FedAvg struggles with task drift in heterogeneous data. Its convergence remains solid, and added noise can act as a regularizer, boosting model performance in some cases, such as on the Industry dataset. FedProx edges out FedAvg thanks to its proximal regularization, which effectively reduces the impact of heterogeneity. SCAFFOLD lags, hindered by unstable convergence on non-IID data due to the added complexity of control variates [9], requiring extensive fine-tuning for stability. Similar trends are noted in existing studies [9, 60], though these focus solely on image data like CIFAR-10.

5 Conclusion and Future Work

We introduce *CrossFreqNet*, a multi-frequency multi-task framework that extends encoder-decoder architectures to share knowledge across tasks while learning task-specific patterns. Our key contribution, *GradBal*, reduces gradient conflicts without changing gradient directions, showing that preserving mild interactions regularizes temporal data effectively. CrossFreqNet achieves up to 72% MAE reduction over the strongest multi-task baseline and 48% over PCGrad across four public benchmarks and one industrial dataset.

We also evaluate federated learning for privacy-sensitive scenarios using differential privacy and secure aggregation. Despite added noise and masking, our federated variant retains 50%–90% of centralized accuracy while providing formal (ϵ,δ)-DP guarantees. Among federated optimizers, FedProx handles heterogeneous task distributions best, balancing privacy and performance effectively.

Future directions include: (1) conflict-aware gradient methods for temporal data with phase shifts and lead-lag relationships, (2) federated learning approaches tailored for time series forecasting using FedAvg, FedProx, and SCAFFOLD to handle mixed-frequency heterogeneous data, and (3) federated frameworks enabling gradient-level sharing with privacy preservation, allowing methods like GradBal to operate in decentralized settings.

References

[1] 2012. Global Energy Forecasting Competition 2012 - Load Forecasting. https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting. Kaggle Dataset.

[2] 2012. Global Energy Forecasting Competition 2012 - Wind Forecasting. https://www.kaggle.com/competitions/GEF2012-wind-forecasting. Kaggle Dataset.

[3] 2015. Rossmann Store Sales. https://www.kaggle.com/competitions/rossmann-store-sales. Kaggle Dataset.

[4] 2017. Beijing Multi Site Air Quality Data. https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data. UCI Machine Learning Repository Dataset.

[5] 2023. Spain Electricity Shortfall Challenge 2023–2024. https://www.kaggle.com/competitions/spain-electricity-shortfall-challenge-2023-2024/data. Kaggle Dataset.

[6] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[7] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. 2024. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815* (2024).

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[9] Gustav A Baumgart, Jaemin Shin, Ali Payani, Myungjin Lee, and Ramana Rao Kompella. 2024. Not all federated learning algorithms are created equal: A performance evaluation study. *arXiv preprint arXiv:2403.17287* (2024).

[10] Paul Bilokon and Yitao Qiu. 2023. Transformers versus LSTMs for electronic trading. *arXiv preprint arXiv:2309.11400* (2023).

[11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[12] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[13] R. Caruana. 1997. Multitask Learning. *Machine Learning* 28, 1 (1997), 41–75. doi:10.1023/a:1007379606734

[14] Yujing Chen, Yue Ning, Zheng Chai, and Huzefa Rangwala. 2020. Federated Multi-task Learning with Hierarchical Attention for Sensor Data Analytics. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8. doi:10.1109/IJCNN48605.2020.9207508

[15] Yujie Chen, Xuping Qin, Ji Wang, Chunyang Yu, and Wei Gao. 2020. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *IEEE Intelligent Systems* 35, 4 (2020), 83–93.

[16] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*. PMLR, 794–803.

[17] Z. Chen, E. Jiaze, X. Zhang, H. Sheng, and X. Cheng. 2020. Multi-task Time Series Forecasting with Shared Attention. In *Proceedings of the 2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 917–925. doi:10.1109/ICDMW51353.2020.00128

[18] Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W Tsang. 2022. A multi-view multi-task learning framework for multi-variate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7665–7680.

[19] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[20] Aysu Ezen-Can. 2020. A Comparison of LSTM and BERT for Small Corpus. *arXiv preprint arXiv:2009.05451* (2020).

[21] Claudia Foroni, Massimiliano Marcellino, and Christian Schumacher. 2015. Unrestricted mixed data sampling (MIDAS): MIDAS regressions with unrestricted lag polynomials. *Journal of the Royal Statistical Society Series A: Statistics in Society* 178, 1 (2015), 57–82.

[22] Dashan Gao, Xin Yao, and Qiang Yang. 2022. A survey on heterogeneous federated learning. *arXiv preprint arXiv:2210.04505* (2022).

[23] S. Gao, T. Koker, O. Queen, T. Hartvigsen, T. Tsiligkaridis, and M. Zitnik. 2024. UniTS: A Unified Multi-Task Time Series Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37. 140589–140631.

[24] E. Ghysels, A. Sinko, and R. Valkanov. 2007. MIDAS Regressions: Further Results and New Directions. *Econometric Reviews* 26, 1 (2007), 53–90. doi:10.1080/07474930600972467

[25] Soma Hansel, Erich Kobler, and Alexander Effland. [n. d.]. FedPCE: Federated Personalized Client Embeddings. ([n. d.]).

[26] Xiaoyu He, Suixiang Shi, Xiulin Geng, Jie Yu, and Lingyu Xu. 2023. Multi-step forecasting of multivariate time series using multi-attention collaborative network. *Expert Systems with Applications* 211 (2023), 118516.

[27] Mohammad Ameen Husnoo, Ahsan Anwar, Nima Hosseinzadeh, S. Nazrul Islam, Abdun Naser Mahmood, and Rana Doss. 2023. A Secure Federated Learning Framework for Residential Short-Term Load Forecasting. *IEEE Transactions on Smart Grid* 15, 2 (2023), 2044–2055.

[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.

[29] Jien Kim, Gunryeong Park, Miseung Kim, and Soyoung Park. 2023. Cluster-based secure aggregation for federated learning. *Electronics* 12, 4 (2023), 870.

[30] Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Bernie Wang. 2023. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems* 36 (2023), 28341–28364.

[31] V. I. Kontopoulou, A. D. Panagopoulos, I. Kakkos, and G. K. Matsopoulos. 2023. A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks. *Future Internet* 15, 8 (2023), 255. doi:10.3390/fi15080255

[32] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl. 2017. Time-Series Extreme Event Forecasting with Neural Networks at Uber. In *International Conference on Machine Learning (ICML)*, Vol. 34. SN, 1–5.

[33] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. 2021. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492* (2021).

[34] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems (MLSys)*, Vol. 2. 429–450.

[35] B. Lim and S. Zohren. 2021. Time-Series Forecasting with Deep Learning: A Survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379, 2194 (2021), 20200209. doi:10.1098/rsta.2020.0209

[36] Chin-Yi Lin, Yu-Ming Hsieh, Fan-Tien Cheng, Hsien-Cheng Huang, and Muhammad Adnan. 2019. Time Series Prediction Algorithm for Intelligent Predictive Maintenance. *IEEE Robotics and Automation Letters* 4, 3 (2019), 2807–2814. doi:10.1109/LRA.2019.2918684

[37] J. Lin and G. Michailidis. 2024. A Multi-Task Encoder-Dual-Decoder Framework for Mixed Frequency Data Prediction. *International Journal of Forecasting* 40, 3 (2024), 942–957. doi:10.1016/j.ijforecast.2024.01.012

[38] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems* 34 (2021), 18878–18890.

[39] D. Liu, Y. Wang, C. Liu, K. Wang, X. Yuan, and C. Yang. 2024. Blackout Missing Data Recovery in Industrial Time Series Based on Masked-Former Hierarchical Imputation Framework. *IEEE Transactions on Automation Science and Engineering* (2024), 1–13. doi:10.1109/tase.2023.3287895

[40] Diju Liu, Yalin Wang, Chenliang Liu, Xiaofeng Yuan, and Chunhua Yang. 2023. Multirate-former: An efficient transformer-based hierarchical network for multi-step prediction of multirate industrial processes. *IEEE Transactions on Instrumentation and Measurement* 73 (2023), 1–13.

[41] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625* (2023).

[42] Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing* 415 (2020), 106–113.

[43] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[45] Ze Meng, Xin Yao, and Lifeng Sun. 2021. Multi-task distillation: Towards mitigating the negative transfer in multi-task learning. In *2021 IEEE international conference on image processing (ICIP)*. IEEE, 389–393.

[46] Meta AI. 2025. Opacus accounting utilities. https://opacus.ai/api/accounting/utils.html. Accessed: 2025-06-24.

[47] Thomas Morstyn, Niall Farrell, Sarah J. Darby, and Malcolm D. McCulloch. 2018. Using Peer-to-Peer Energy-Trading Platforms to Incentivize Prosumers to Form Federated Power Plants. *Nature Energy* 3, 2 (2018), 94–101.

[48] National Renewable Energy Laboratory (NREL). 2006. Solar Power Generation. https://www.nrel.gov/grid/solar-power-data.html. Dataset.

[49] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807* (2015).

[50] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[51] Waqas Saeed. 2022. Frequency-Based Ensemble Forecasting Model for Time Series Forecasting. *Computational and Applied Mathematics* 41, 2 (2022), 66.

[52] Ozan Sener and Vladlen Koltun. 2018. Multi-task Learning as Multi-objective Optimization. In *Advances in Neural Information Processing Systems*, Vol. 31.

[53] Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. 2022. 16 federated knowledge distillation. *Machine Learning and Wireless Communications* 457 (2022).
[54] A. Shabani, A. Abdi, L. Meng, and T. Sylvain. 2022. Scaleformer: Iterative Multi-Scale Refining Transformers for Time Series Forecasting. *arXiv preprint* (2022). arXiv:2206.04038 [cs.LG]
[55] A. Shankar, L. Y. Chen, J. Decouchant, D. Gkorou, and R. Hai. 2024. Share Your Secrets for Privacy! Confidential Forecasting with Vertical Federated Learning. *arXiv preprint* (2024). arXiv:2405.20761 [cs.LG]
[56] S. N. Shukla and B. M. Marlin. 2021. Multi-Time Attention Networks for Irregularly Sampled Time Series. *arXiv preprint* (2021). arXiv:2101.10318 [cs.LG]
[57] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning?. In *International conference on machine learning*. PMLR, 9120–9132.
[58] Johannes Stübinger and Katharina Adler. 2020. How to identify varying lead–lag effects in time series data: Implementation, validation, and application of the generalized causality algorithm. *Algorithms* 13, 4 (2020), 95.
[59] J. et al. Sun. 2020. Crypto Dataset. https://kaggle.com/competitions/g-research-crypto-forecasting. Dataset referenced in paper.
[60] Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xin He, Bo Han, and Xiaowen Chu. 2022. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *International Conference on Machine Learning*. PMLR, 21111–21132.
[61] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 1–11.
[62] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2020. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874* (2020).
[63] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security* 15 (2020), 3454–3469.
[64] Y. Yaron, L. Reyzin, D. Zhang, and D. Papadopoulos. 2023. Verifiable Random Functions (VRFs). Internet-Draft draft-irtf-cfrg-vrf-13, IETF. https://www.ietf.org/archive/id/draft-irtf-cfrg-vrf-13.html Work in Progress.
[65] Liang Yu, Lai Tu, and Xiang Bai. 2025. MFRS: A Multi-Frequency Reference Series Approach to Scalable and Accurate Time-Series Forecasting. *arXiv preprint arXiv:2503.08328* (2025).
[66] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 5824–5836.
[67] Yichi Zhang, Mihai Cucuringu, Alexander Y Shestopaloff, and Stefan Zohren. 2023. Dynamic Time Warping for Lead-Lag Relationships in Lagged Multi-Factor Models. *arXiv preprint arXiv:2309.08800* (2023).
[68] L. Zhu, Z. Liu, and S. Han. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32.

6 Appendix

The appendix provides additional strategies and results from the research.

7 Experimental Details.

7.1 Datasets

Table 5 provides detailed information about the public datasets used in our experiments. For the main experiments, we primarily evaluate our proposed model on the *AirQuality*, *Load*, *Wind*, and *Spain* datasets. During the model prototyping stage and to refine the final selected model configuration, we additionally utilized the *Sales*, *Crypto*, and *Solar* datasets.

To properly evaluate our model in a multi-frequency setting, we preprocess these datasets by explicitly differentiating sampling frequencies: the *target or endogenous variables* are downsampled (aggregated) to represent the lower-frequency signals, while the *exogenous variables* are upsampled (via interpolation) to represent higher-frequency signals. This approach ensures compatibility with our model architecture designed for handling mixed-frequency time series inputs.

7.2 Evaluation Metrics

We report the **Mean Absolute Error (MAE)** as the primary accuracy metric, accompanied by the **Standard Error of the Mean (SEM)** to quantify the reliability of reported results.

Mean Absolute Error (MAE). MAE measures the average magnitude of prediction errors without considering their direction, providing an interpretable and scale-sensitive measure of forecasting accuracy:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|,$$

where $\hat{y}_i$ is the model prediction, y_i is the true value, and n is the number of observations. MAE is robust to outliers compared to squared error metrics and directly reflects how far, on average, predictions deviate from the ground truth.

Standard Error of the Mean (SEM).. While MAE summarizes model accuracy, it does not convey how much this accuracy varies across repeated experiments or folds. For that, we compute the SEM:

$$\text{SEM} = \frac{s}{\sqrt{n}} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}}{\sqrt{n}},$$

where s is the sample standard deviation, n is the number of repetitions, x_i denotes the MAE for the i^{th} run, and $\bar{x}$ is the mean MAE. SEM quantifies the precision of the MAE estimate: a smaller SEM implies greater consistency across runs, indicating that the reported MAE is a stable and reliable indicator of true model performance.

Why SEM?. In our time series forecasting experiments, we vary the random seed to alter model initialization and stochastic training behavior, which leads to slight fluctuations in performance across runs. Reporting the Standard Error of the Mean (SEM) captures this variability and quantifies the precision of the reported MAE. SEM indicates how closely the reported MAE would approximate the true population MAE if the experiment were repeated infinitely many times under different initializations.

Reporting Protocol. For all tables and figures, we report MAE alongside SEM as "MAE ± SEM". This conveys both the central performance tendency and the uncertainty in a compact form, enabling fairer comparison between models especially when performance differences are small and could otherwise be within the range of experimental noise.

7.3 Baselines.

ARIMAX. We include the Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX) model as a classical and widely used baseline for time series forecasting. ARIMAX extends the traditional ARIMA framework by incorporating external covariates to improve predictions. It is parameterized by (p, d, q) for the autoregressive order, differencing order, and moving average order, respectively, along with (P, D, Q, s) for potential seasonal components. ARIMAX assumes the underlying time series is (or can be transformed to be) stationary and models temporal dependencies

Table 5: Details of datasets used in experiments.

Dataset	Number of Rows	Number of Features	Target Variable	Source
AirQuality	20,900	11	CO	[4]
Load	157,000	5	Load	[1]
Wind	4,300	5	wp	[2]
Spain	8,700	7	load	[5]
Sales	943	6	Sales	[3]
Crypto	4,300	5	Closing Price	[59]
Solar	105,121	10	Solar Output (kW)	[48]

through a linear combination of past values, differenced terms, and lagged forecast errors, while exogenous variables enter the model linearly.

Although ARIMAX provides a strong statistical benchmark and often performs well for univariate[32] or single-source forecasting tasks, it is inherently limited in multi-source, multi-frequency contexts. In particular, its linear structure cannot effectively capture nonlinear interactions or share information across multiple related series, making it less suitable for complex multi-task forecasting scenarios.

UniTS. UniTS is a versatile model designed to handle various time series tasks, including forecasting, classification, imputation, and anomaly detection, by transforming time series data and task instructions into a shared "token" format. It employs a modified transformer architecture that adapts to different time series lengths and variable counts, using self-attention across time and variables, a dynamic linear operator for dependency capture, and a gating module for diverse data types. For forecasting UniTS uses a "GEN tower" shared across generative tasks, trained simultaneously on diverse datasets to learn broad patterns without requiring separate models. Key hyperparameters include patch size and stride: the patch size defines the length of non-overlapping segments into which the time series is divided, while the stride, equal to the patch size, ensures each segment follows the previous one directly without overlap.

FATHOM. is a federated multi-task learning framework. It combines task-specific attention to identify important local features of each task with a global temporal attention layer that learns shared patterns across devices. These attention layers feed into LSTMs, enabling the model to capture both feature-level and time-dependent relationships. It is made of task specific layers and a shared layer like a standard MTL model[13].

Private-Shared Attention. is a multi-task time series forecasting framework that combines the Transformer's self-attention with a novel shared-private attention scheme. Instead of training each task in isolation, MTL-Trans enables multiple related time series tasks to learn jointly, improving performance when individual tasks have limited data. The model uses *private task-specific Transformer encoders* for local feature extraction and a *shared multi-head attention layer* that aggregates attention patterns across tasks, acting like a global memory of temporal dependencies. Two variants are proposed: (i) a *global shared attention* approach, where all tasks feed into and draw from the same attention pool, and (ii) a *hybrid local-global attention* scheme, which allows tasks to retain private representations while periodically updating and refining the shared attention state. Experiments on real-world datasets show that these architectures outperform single-task baselines and RNN-based multi-task models by leveraging attention to capture long-range dependencies and shared patterns across tasks.

TSDiff. TSDiff [30] is an unconditional diffusion model for time series forecasting that learns a general data distribution through a forward–reverse denoising process. Instead of training for a specific task, TSDiff applies Gaussian noise stepwise to time series data (the forward diffusion) and trains a neural network to reverse this process by predicting the noise (the reverse diffusion). During inference, a self-guidance mechanism conditions the unconditional model on observed time points to generate forecasts, imputations, or synthetic series without retraining. We adapted TSDiff to our multi-task mixed-frequency setting by combining high-frequency (HF) and low-frequency (LF) features into a unified multivariate representation and interpolating HF signals to align with LF targets. Because no diffusion-based multi-task model exists, we extended TSDiff's architecture with client/task identifiers and multivariate inputs, enabling simultaneous forecasting across tasks while preserving the original diffusion objective.

8 Extended Experiments.

8.1 Modeling Strategies

In this section, we outline our approach to developing the most optimal *multi-task* model. To start with our goal we start with the assumption that there is no confidentiality constraint and hence try to build a upper bound baseline. Later we introduce confidentiality constraints with Federated setting. Our methodology consists of three primary modeling strategies.

(1) **Non-Collaborative Models:** Each task is assigned a separate model, with no shared learning across tasks. This approach ensures data isolation but fails to leverage inter-customer similarities.
(2) **Implicit Collaboration, Global Model:** A single model is trained on all available raw data without task-specific adaptations. This approach benefits from large-scale data aggregation but may not generalize well to task-specific variations.
(3) **Explicit Collaboration, Multi-Task Learning (MTL):** We employ MTL ([13], [17], [23]), incorporating task-specific layers on top of a shared representation to allow both collaborative learning and customer-specific fine-tuning.

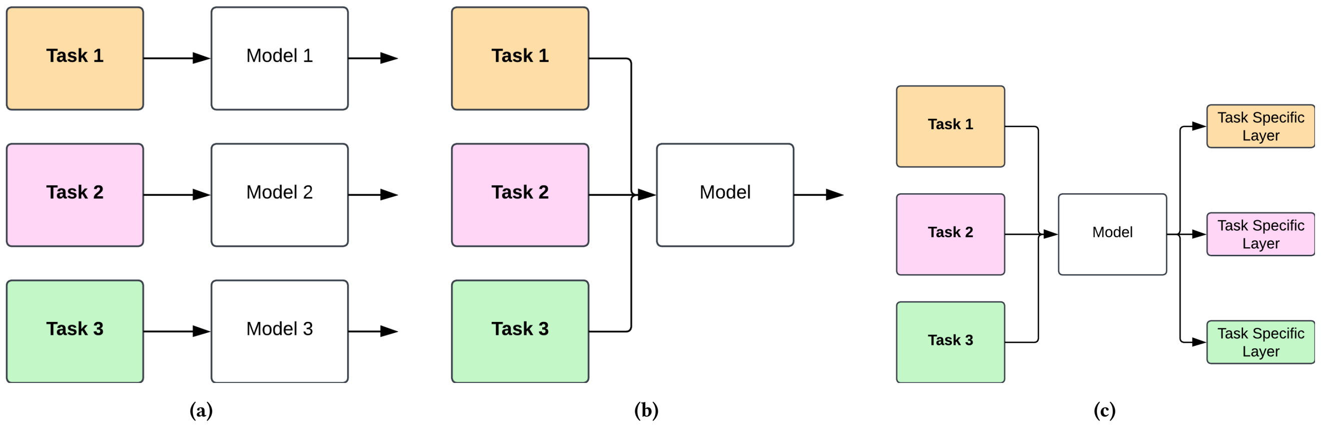


Figure 6: Different Model Training Setup to evaluate which gives the best accuracy where (a) Non-Collaborative Model setup, (b) Implicit-Collaborative Model Setup, (c) Explicit-Collaborative: Multi Task Model Setup

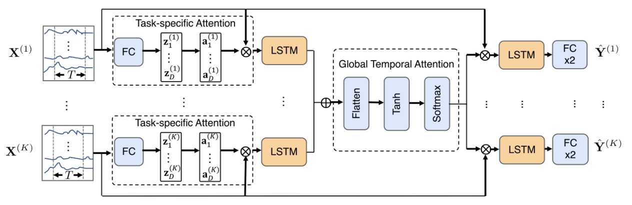


Figure 7: source: FATHOM [14]

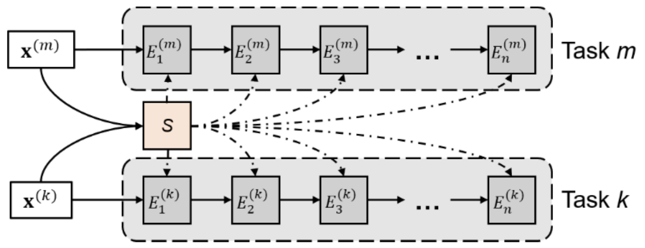


Figure 8: Source: Shared-Private Attention [17]

For each of these modeling strategies, we implement multiple architectures, including ARIMAX, LSTM, and Transformer-based models wherever applicable. Notably, vanilla ARIMAX cannot be directly applied in MTL or Foundational Model setups due to its lack of shared representation capabilities.

Each LSTM-based and Transformer-based model is evaluated under two settings:

(1) Implicit Collaboration Setting: Standard architecture without specialized multi-task adaptations where we train vanilla LSTMs and Transformer based models.
(2) Explicit Collaboration Setting: Incorporates state-of-the-art (SOTA) MTL setups, enhancing the model's ability to capture task-specific variations as shown in Figure 7 and Figure 8.

Given the asynchronous nature of time series data in our setup, we extend our experimentation by incorporating variations of multi-frequency time series model to make it Multi-Task model as discusses in sections below. We realized simplest models prefer to work the best, Figure 10, keeping task specific heads as simple multi layer perceptrons(MLP) rather than added complex decoders.

8.1.1 ***Shared High Frequency(HF) Encoder, Low Frequency(LF) Decoder with private LF Decoder forecasting method.*** We extended the state-of-the-art approach by [37] to a multi-task architecture, as shown in Figure 9 and detailed in Algorithm 3. This model follows the principle of hard parameter sharing, where data from all clients is used to train a centralized shared encoder, followed by task-specific decoders (heads) to fine-tune the model for each client individually.

However, training this model was computationally slow, the performance was not satisfactory. This was likely due to the added complexity of the architecture, which made optimization more difficult.

Algorithm 3 Shared Encoder-Decoder with Attention Mechanism and Client-Specific Forecasting

```
1: Input: Mixed-frequency datasets $\mathcal{D}_c$ for each client $c \in \{1, \dots, C\}$
2: Initialize: Shared parameters $\theta_s$, attention parameters $\theta_a$, client-specific parameters $\theta_c$ for each $c$
3: for each epoch $e = 1$ to $E$ do
4:     for all batches $(X_{hf}, X_{lf}, Y, c)$ in dataset do
5:         Shared HF encoding: $H_{hf} \leftarrow \mathrm{SharedHFEncoder}(X_{hf})$
6:         Attention mechanism: $A \leftarrow \mathrm{Attention}(H_{hf})$ {Attention module [8]}
7:         Shared LF decoding: $H_{lf} \leftarrow \mathrm{SharedLFDecoder}(X_{lf}, A)$
8:         Group by clients: $\{H_{lf}^{(1)}, H_{lf}^{(2)}, \dots, H_{lf}^{(C)}\} \leftarrow \mathrm{GroupByClientID}(H_{lf}, c)$
9:         for all clients $c \in \{1, \dots, C\}$ do
10:            Client-specific LF decoding: $Z_c \leftarrow \mathrm{ClientLFDecoder}_c(H_{lf}^{(c)})$
11:            Client forecasting: $\hat{Y}_c \leftarrow \mathrm{FC}_{128}(\mathrm{FC}_{64}(Z_c))$
12:            Compute loss: $\mathcal{L}_c \leftarrow \mathrm{MSE}(\hat{Y}_c, Y_c)$
13:        end for
14:        Total loss: $\mathcal{L} \leftarrow \sum_{c=1}^{C} \mathcal{L}_c$
15:        Backpropagate $\mathcal{L}$ to update $\theta_s$, $\theta_a$, and $\{\theta_c\}_{c=1}^{C}$
16:    end for
17: end for
```

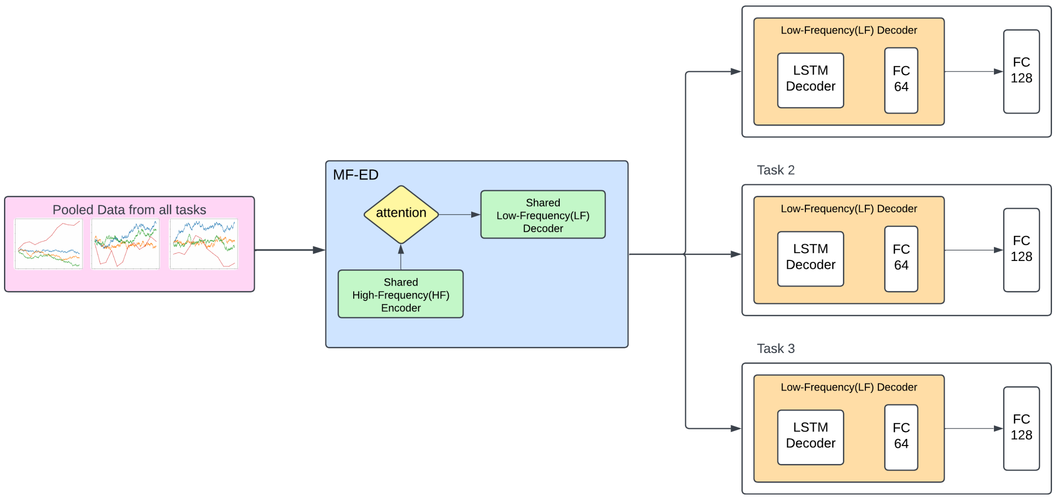

Pooled Data from all tasks
MF-ED
attention
Shared Low-Frequency(LF) Decoder
Shared High-Frequency(HF) Encoder
Low-Frequency(LF) Decoder
LSTM Decoder
FC 64
FC 128
Task 2
Task 3

Figure 9: Shared High Frequency(HF) Encoder, Low Frequency(LF) Decoder with private LF Decoder forecasting method

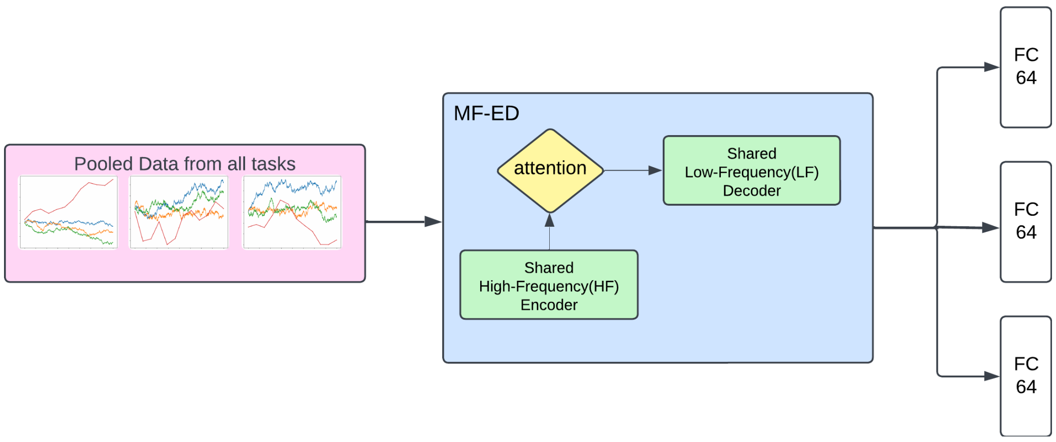

Pooled Data from all tasks
MF-ED
attention
Shared Low-Frequency(LF) Decoder
Shared High-Frequency(HF) Encoder
FC 64
FC 64
FC 64

Figure 10: Shared High Frequency(HF) Encoder, Low Frequency(LF) Decoder with private MLP forecasting method

8.1.2 ***Shared HF-LF: Shared High Frequency(HF) Encoder, Low Frequency(LF) Decoder with private MLP forecasting method.*** Looking at the competitive results of ARIMAX, we decided to simplify the previous architecture by replacing private decoders with simple MLP layers as shown in Figure 10. This was quicker to train and gave promising results was also sets a baseline in out experiment section as (*Ours w/o conflict resolution*) and algorithm for this remains exactly same as in Algorithm 1 but without the *GradBal.*

8.1.3 ***Secured HF-LF: Task-specific High Frequency(HF) Encoder and Low Frequency(LF) Decoder with Shared HF Encoder.*** In the original formulation, task data is held by a central layer, which raises significant confidentiality concerns. To address this, we introduced a new iteration which is similar to FATHOM[14], as shown in Figure 7, where each client retains its own high-frequency (HF) encoder and low-frequency (LF) decoder, while only sharing encoded features with the central shared parameters, as illustrated in Figure 11. This setup follows from FATHOM as shown in Figure 7 and tries to mitigates the risk to confidentiality to some extent, as it becomes harder though not impossible[68] to infer the raw data from aggregated representations.

8.2 Federated Learning: Enforcing Confidentiality.

In Multi-Task Learning (MTL), methods like GradBal successfully remove conflicts between task gradients and improve the model's overall performance. However, applying these methods in Federated Learning (FL) is quite difficult. In FL, the server must protect each tasks's data, so it cannot see the task-specific gradients that GradBal needs to work. Because of this strict confidentiality rule, we cannot directly use gradient-based techniques from MTL in FL[68]. In the future, we might use secure technologies (for example, noising, encryption or secret sharing) so that FL systems can share gradients without breaking privacy. Then we could bring

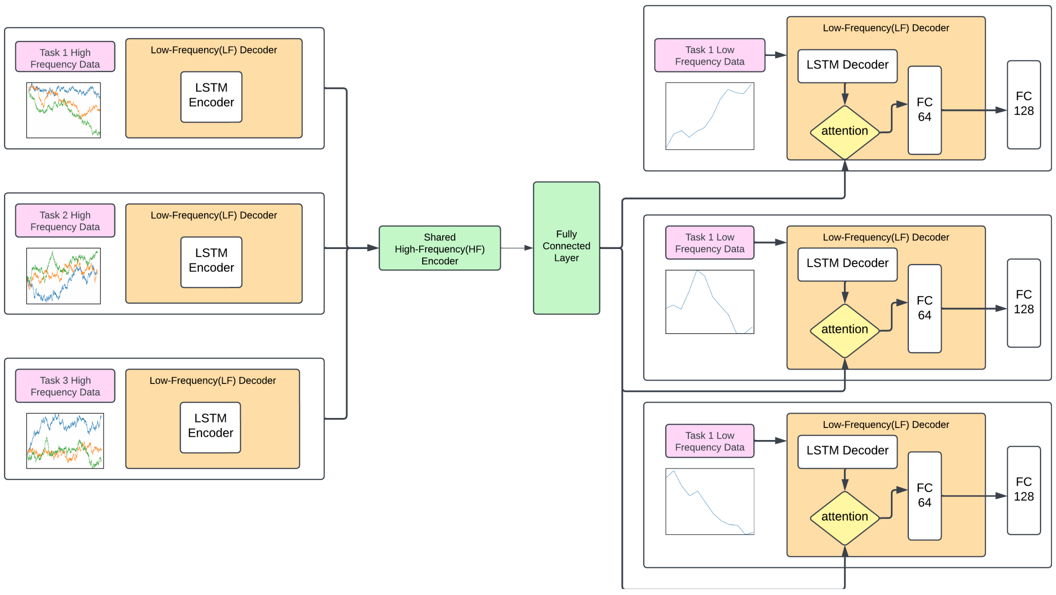

Task 1 High Frequency Data
Task 2 High Frequency Data
Task 3 High Frequency Data
Low-Frequency(LF) Decoder
LSTM Encoder
Shared High-Frequency(HF) Encoder
Fully Connected Layer
Task 1 Low Frequency Data
LSTM Decoder
attention
FC 64
FC 128

Figure 11: Private High Frequency(HF) Encoder and Low Frequency(LF) Decoder with Shared HF Encoder

Algorithm 4 Task-specific High Frequency(HF) Encoder and Low Frequency(LF) Decoder with Shared HF Encoder.

```
1:  Input: Mixed-frequency datasets $\mathcal{D}_c$ for each client $c \in \{1, \ldots, C\}$
2:  Initialize: Shared parameters $\theta_s$, client-specific parameters $\theta_c$ for each $c$
3:  for each epoch $e = 1$ to $E$ do
4:      for all batches $(X_{hf}, X_{lf}, Y, c)$ in dataset do
5:          Client-specific encoding: $H_c \leftarrow \text{ClientEncoder}_c(X_{hf})$
6:          Shared representation: $Z_c \leftarrow \text{SharedEncoder}(H_c)$
7:          Client fusion: $Z_c' \leftarrow \text{Fuse}(Z_c, \text{Embed}(c))$
8:          Client decoding: $\hat{Y}_c \leftarrow \text{ClientDecoder}_c(X_{lf}, Z_c')$
9:          Compute loss: $\mathcal{L}_c \leftarrow \text{MSE}(\hat{Y}_c, Y)$
10:         Backpropagate $\mathcal{L}_c$ to update $\theta_s$ and $\theta_c$
11:     end for
12: end for
```

methods like GradBal into federated settings. For now, though, FL must balance strong data confidentiality against the benefits of advanced optimization methods.

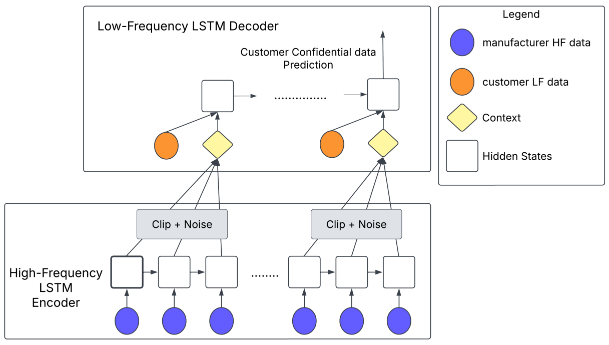
For this study, our primary objective is to assess confidentiality. Therefore, we move our multi-frequency encoder-decoder model into an FL environment, employing established federated optimization techniques like FedProx [34] and SCAFFOLD [28], which inherently handle task heterogeneity through proximal and control variates terms, respectively. Our federated framework includes two key components: (1) local training with differential confidentiality and (2) secure aggregation of model updates an finally (3) Personalization. Figure 12 illustrates our federated setup, with further methodological details provided in next section Appendix 8.2.1.

Local Training with Dual Differential confidentiality. The first component of our federated framework involves local training at each task, corresponding to a HF-LF pair, using a modified version of the LSTM-based encoder-decoder architecture proposed by Lin et al. [37]. Each task trains on their local dataset $\mathcal{D}_c$, which includes high-frequency data and low-frequency data, to forecast task-specific outcomes. To ensure confidentiality, we implement a dual differential confidentiality mechanism:

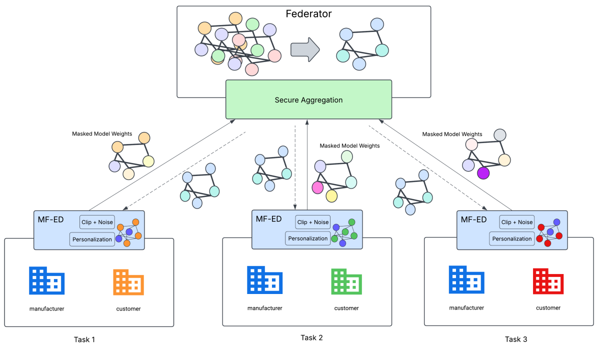
- **Vertical confidentiality (Frequency-Level)**: Calibrated gaussian noise σ_{enc} is added to the clipped encoder outputs, protecting the intermediate representations of high-frequency inputs ensuring $(\epsilon_{enc}, \delta_{enc})$-DP(Alg. 7, line 6). This mitigates risks of inference attacks on frequency-specific data, as intermediate states may leak sensitive patterns [68].
- **Horizontal confidentiality (Task-Level)**: We apply Differentially Private Stochastic Gradient Descent (DP-SGD) [6] during local training. DP-SGD clips gradients to bound the influence of individual data points and adds calibrated Gaussian noise σ_{grad}, ensuring $(\epsilon_{grad}, \delta_{grad})$-DP for task-specific model updates(Alg. 7, line 11).

Building on the dual differential confidentiality mechanisms, since no downstream component directly accesses raw HF data, the gradient mechanism acts solely as a post-processing step[19] on already privatized encoder embeddings. Consequently, the two confidentiality budgets protect disjoint subset of data(encoder outputs and gradients updates, respectively). Hence, we compose the vertical and horizontal confidentiality guarantees to achieve system-level (ϵ, δ)-DP for both high-frequency and low-frequency datasets, as formalized below,

$$\text{System DP guarantee} \begin{cases} (\epsilon_{enc}, \delta_{enc})\text{-DP for high-frequency data,} \\ (\epsilon_{grad}, \delta_{grad})\text{-DP for task-level updates.} \end{cases}$$


Low-Frequency LSTM Decoder
Customer Confidential data Prediction
Legend
manufacturer HF data
customer LF data
Context
Hidden States
Clip + Noise
Clip + Noise
High-Frequency LSTM Encoder

(a) Local training at task level with DP noise at encoder output. The multi-frequency encoder-decoder(MF-ED) in this model is trained jointly using DP-SGD



(b) Federated Setting with secured aggregation where MF-ED is multi-frequency encoder-decoder[37].

Figure 12: Overview of the Federated framework, illustrating local training with dual DP and secure aggregation for multi-task time series forecasting. Detailed in Appendix 8.2.1

Importantly, differential confidentiality serves as the final line of defense when secure aggregation fails or is compromised and individual model updates are exposed to adversaries. The local training process is formalized in Appendix 8.2.1.

Secure Aggregation of Model Updates. The second component of federated framework is Secure Aggregation, which ensures that the central federator can only access aggregated model updates, not individual task contributions. We implement secured aggregation protocol[11], which uses *Diffie-Hellman key exchange* and *t-out-of-n secret sharing* to securely mask individual updates as shown in Algorithm 8. This scheme protects against a *semi-honest federator* and *colluding clients*[43]., ensuring confidentiality as long as at least $t > 1$[29] tasks remain honest and active. Combined SecAgg with DP-SGD reduces noise requirements while maintaining formal confidentiality guarantees, as supported by Truex et al.[61]. Hence,

$$\sigma_{\text{eff}} = \frac{\sigma_{grad}}{\sqrt{t-1}}$$

, where σ_{grad} is the original standard deviation required for the Gaussian mechanism under DP-SGD and σ_{eff} is the actual noise applied. A concise implementation for federated setting is shown in Appendix 8.2.1. The reason we chose to apply noise reduction just on σ_{grad} because we found out that this noise caused most harm to the model utility and experiments on those are presented in Appendix 9.3. The detailed working is presented in Appendix 8.2.1. Algorithm 7 pinpoints where calibrated noise is injected, while Algorithm 8 details the secure aggregation protocol that protects the tasks' model updates.

Personalization Phase. In our federated learning setup, we first complete 10 rounds of standard federated training to produce a final global model. To mimic the task-specific heads of our centralized model, each client then personalizes this global model by fine-tuning it on their local data with a lower learning rate. This personalized model is used for local inference, effectively tailoring the shared global knowledge to each client's specific task or data distribution.

8.2.1 *Confidentiality preserving federated Training Procedure.*

(i.)The Local Pass. Our model trains locally on task-specific high-frequency(HF), and low-frequency(LF), time series data, extracting essential forecasting information through a carefully designed encoder-decoder neural network architecture. The architecture, inspired by Lin (2024)[37], employs Long Short-Term Memory (LSTM) layers, but also crucially integrates privacy-preserving mechanisms to ensure confidentiality at task level.

Each entity within the task $c \in C$, where C is the total number of tasks, owns two time series:

(1) High Frequency inputs: $X_C \in R^{T \times d_X}$
(2) Low Frequency inputs: $Y_C \in R^{T^{'}} \times d_Y$

The forecasting task is to predict $Y_{c,t+h}$ using past h_{HF} window from X_c and h_{LF} window from Y_c, where $T >> T^{'}$ and $r = T/T^{'}$ is the frequency ratio. Hence $h_{HF} = h_{LF} \times r$.

Concretely we define the model as follows:

(1) **High Frequency Encoder $\mathcal{E}$:**

$$H_{c,t} = \mathcal{E}_c(X_{c,t-h_{HF}:t}) \in \mathbb{R}^{h_{HF} \times d} \tag{6}$$

where d is the hidden size of the encoder output and h_{HF} is the high-frequency window length and ϵ_c is realized via an LSTM.

(2) **Local Encoding of HF with Differential Privacy in the latent representation:** High Frequency encoder outputs are independently encodes into latent representations through a LSTM-based encoder, ensuring that raw data remains confidential. Let $H_{c,t}$ be the output of the LSTM based encoder-decoder ϵ_c. We enforce element wise clipping[6]:

$$\tilde{H}_{c,t} = \min(1, B/\|H_{c,t}\|_2) \cdot H_{c,t} \tag{7}$$

where B is the clipping bound, i.e. maximum allowed norm before scaling down.

Then we add calibrated Gaussian noise:

$$H^{priv}_{c,t} = \tilde{H}_{c,t} + \mathcal{N}(0, \|\sigma_{enc}\| B^2 I) \tag{8}$$

where σ_{enc} is the noise scale or standard deviation of the Gaussian noise and I is the identity matrix. This mechanism provides $(\epsilon_{enc}, \delta_{enc})$ - DP and deters honest but curious party from making any inferences when latent is shared across entities in the task.

(3) **Context Attention $\mathcal{A}$:** The decoder uses additive attention[Bahdanau, 2014] to dynamically weigh the differentially private encoder output states at each LF decoding step. At decoder step i , the context vector $c_(c, t, i)$ is computed as weighted sum of the j encoder output steps :

$$c_{c,t,i} = \sum_{j=1}^{h_{HF}} \alpha_{i,j} \, H_{c,t}^{\text{priv},(j)} \tag{9}$$

The attention weights $\alpha_{i,j}$ measure the alignment between the previous decoder hidden state $s_{c,t,i-1}$ and the encoder states $H_{c,t}^{\text{priv},(j)}$. The weights are computed as:

$$\alpha_{i,j} = \frac{\exp\left(v^\top \tanh\left(W_q \, s_{c,t,i-1} + W_k \, H_{c,t}^{\text{priv},(j)}\right)\right)}{\sum_{l=1}^{h_{HF}} \exp\left(v^\top \tanh\left(W_q \, s_{c,t,i-1} + W_k \, H_{c,t}^{\text{priv},(j)}\right)\right)} \tag{10}$$

where learnable parameters are:
(a) Query Projection: $W_q \in R^{d_\alpha \times d}$
(b) Key Projection: $W_k \in R^{d_\alpha \times d}$
(c) Attention Vector: $v \in R^{d_\alpha}$ where d_α denotes the dimensionality of the attention mechanism.

(4) **Decoder $\mathcal{D}_c$**: The decoder predicts LF data using the previously observed LF data and the attention-derived context vector. At decoding step i, the decoder input $z_{c,t,i}$ concatenates the LF data from the previous time step and the context vector:

$$z_{c,t,i} = \left[Y_{c,t-h_{LF}+i} \parallel c_{c,t,i}\right] \in \mathbb{R}^{d_Y+d} \tag{11}$$

The decoder's LSTM updates the hidden state $s_{c,t,i}$ as follows:

$$s_{c,t,i}, h_{c,t,i} = \text{LSTM}_{\text{dec}}(z_{c,t,i}, s_{c,t,i-1}) \tag{12}$$

ith initial hidden state $s_{c,t,0}$, typically initialized as zeros or as learned parameters.

(5) **Output Layer: LF Prediction**: The forecasted LF output at horizon h is generated from the final decoder hidden state:

$$\hat{Y}_{c,t+h}^{LF} = W_o \, s_{c,t,h_{LF}} + b_o \tag{13}$$

where:
- Output projection: $W_o \in \mathbb{R}^{d_Y \times d}$
- Bias term: $b_o \in \mathbb{R}^{d_Y}$

(6) **Differentially Private Stochastic Gradient Descent for LF Decoder** To protect sensitive information in both local and global model updates, our framework incorporates a differentially private stochastic gradient descent algorithm (DP-SGD) [6] at the decoder.
At the task level, gradients are clipped to a fixed norm, and calibrated noise is added during each local update to bound the privacy loss. Following [6], we apply per-example gradient clipping:

$$\tilde{g}_i = \frac{g_i}{\max\left(1, \frac{\|g_i\|_2}{B}\right)} \tag{14}$$

and add Gaussian noise:

$$\tilde{g} = \frac{1}{N} \sum_i \left(\tilde{g}_i + \mathcal{N}(0, \sigma_{grad}^2 B^2 I)\right) \tag{15}$$

This enforces $(\epsilon_{grad}, \delta_{grad})$-DP guarantees for the updates, with noise scale σ_{grad}.

Detailed Algorithm is presented in Algorithm 7.

(ii.)The Aggregator Pass. Complementing the task-level model, our framework includes a **Secure Federated Aggregation Protocol** in addition to the Differential Privacy (DP) already applied during the local pass. While DP protects individual updates, relying on it alone can reduce model accuracy, especially when the number of tasks is large and each task has relatively little data. To address this, our protocol securely aggregates masked model without revealing raw contributions. Our approach is inspired by secure multi-party computation methods from Bonawitz et al. (2017) and Truex et al. (2019).

Each task $c \in C$ computes a local model update:

$$\hat{\Delta}_{c,t} = \theta_t^c - \theta_{t-1}^{\text{global}} \tag{16}$$

then,

(1) **Masking Mechanism for Confidentiality [11]:** To ensure confidentiality during aggregation, we adopt a secure aggregation masking protocol adapted for both semi-decentralized and peer-to-peer setups.
At the beginning of each round, each task establishes pairwise shared secrets with every other task using Diffie-Hellman (DH) key exchange. Each task generates a DH key pair (public and private keys) and exchanges public keys.
Based on these public keys, every pair of tasks c and c' computes a shared secret $s_{c,c'}$ using DH key agreement. To ensure consistent pairing of perturbation masks, tasks are ordered lexicographically by their public keys. This ordering determines roles in their perturbation:
- The task with the smaller key ($c < c'$) adds the PRG-generated mask.
- The task with the larger key ($c > c'$) subtracts the same mask.

These pairwise perturbations are derived from $s_{c,c'}$ using a cryptographically secure pseudorandom generator (PRG).
In addition to pairwise masking, each task samples an independent random vector (a self-mask) and distributes secret shares of it using a (t, n) threshold secret-sharing scheme.
Thus, the masked update sent by each task is:

$$\tilde{\Delta}_c = \Delta_c + m_c + \sum_{c>c'} \text{PRG}(s_{c,c'}) - \sum_{c<c'} \text{PRG}(s_{c,c'}) \tag{17}$$

where m_c is the additional random mask shared via secret sharing.
Upon receiving masked updates, federator aggregate them. Due to the careful construction of masks, all perturbation terms cancel, leaving:

$$S = \sum_{c \in C} \tilde{\Delta}_c = \sum_{c \in C} \Delta_c \tag{18}$$

During aggregation, random masks m_c from alive tasks are collected and canceled during unmasking, ensuring that only the sum of true model updates is revealed.
The federator then applies the FedAvg algorithm:

$$\Delta_{\text{global}} = \sum_{c \in C} w_c \Delta_c \tag{19}$$

$$\theta_{t+1}^{\text{global}} = \theta_t^{\text{global}} + \Delta_{\text{global}} \tag{20}$$

and redistributes the updated global model to all tasks for further personalization.
This DH-based Secure Aggregation scheme provides strong confidentiality even against honest-but-curious federators while maintaining model utility.

(2) **Reduced DP Noise and Privacy Guarantee [61]**: Our fedreated framework achieves robust confidentiality by combining Differential Privacy and Secure Aggregation [61]. Truex et al. formally show that combining DP with Secure Multi-Party Computation (SMC) reduces the required noise for DP while maintaining privacy guarantees.
If each task adds Gaussian noise with standard deviation σ to ensure (ϵ, δ)-DP locally, the total variance across N tasks scales as $N\sigma^2$. However, by securely aggregating obfuscated responses using a trust threshold t (the number of trusted tasks), the required noise reduces to:

$$\sigma_{\text{eff}} = \frac{\sigma^2}{\sqrt{t-1}} \tag{21}$$

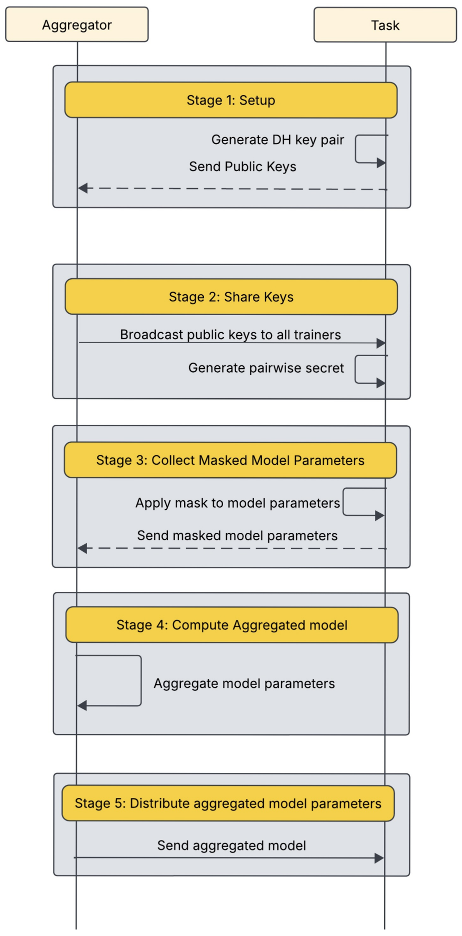
resulting in noise magnitude reduction by $\sqrt{t-1}$ on the server side [61].
In a federated round, if at least t out of N parties are honest and non-colluding, adding independent noise $\mathcal{N}(0, \frac{B^2\sigma^2}{t-1})$ still ensures (ϵ, δ)-DP, even if the remaining $N - t$ tasks collude. This aligns with the theoretical results of Dwork and Roth [19], showing that the coalition's view is reduced to a sum over t honest responses, preserving privacy.

(3) **Cumulative Privacy Across Rounds:** Over multiple rounds, privacy loss accumulates according to the composition theorem: running a (ϵ, δ)-DP mechanism T times results in approximately $(T\epsilon, T\delta)$ privacy loss. This total noise can degrade utility. By employing Secure Aggregation, we reduce the required per-round noise, ensuring both improved utility and bounded cumulative privacy loss.
As shown in prior works [11, 61], Secure Aggregation relies on a trusted parties threshold. In our framework, Secure Aggregation breaks in the worst case when only one trusted party remains, requiring more DP noise to compensate. However, in typical and average cases, Secure Aggregation preserves confidentiality with reduced noise, maintaining both privacy and utility.

Detailed Algorithm is presented in Algorithm 8 and illustrated in Figure 13.

8.3 Other Federated Learning Techniques.

This appendix summarises every federated-learning variant we evaluated before choosing FedAvg.



Aggregator
Task
Stage 1: Setup
Generate DH key pair
Send Public Keys
Stage 2: Share Keys
Broadcast public keys to all trainers
Generate pairwise secret
Stage 3: Collect Masked Model Parameters
Apply mask to model parameters
Send masked model parameters
Stage 4: Compute Aggregated model
Aggregate model parameters
Stage 5: Distribute aggregated model parameters
Send aggregated model

Figure 13: Secure Aggeragation Protocol.

Standard Federated Knowledge Distillation. Algorithm 5 implements the classical Federated Knowledge Distillation (FKD) workflow [53]. Each client trains a local model, converts its outputs to a knowledge vector (typically the logits), and the server averages these vectors to build a single global teacher that is distilled back to every client.

FKD with ID Embedding and Similarity Fusion. Algorithm 6 extends FKD for heterogeneous data. Inspired by similarity-weighted multi-teacher distillation [42] and client embeddings in federated learning [25], it enriches the knowledge vector with a client-ID embedding and fuses the vectors by similarity. This personalised teacher is expected to improve performance when client distributions differ greatly.

8.4 Towards decentralized Federated setting.

What we presented in our methodology is called centralized federated learning setup and forms the foundation of confidentiality-preserving machine learning systems but they inherently rely on a trusted aggregator. In our real world industry use case this centralized trust assumption does not hold. Instead, we further aimed at

Algorithm 5 Federated Multi-Teacher Knowledge Distillation

```
1: Input: Local data $\mathcal{D}_i$ for each client $i$, total rounds $R$, loss weights $\alpha, \beta$
2: Initialize: Model parameters $\theta_i$ for each client $i$
3: for $r = 1$ to $R$ do
4:     for all clients $i = 1$ to $N$ in parallel do
5:         Train local model on $\mathcal{D}_i$ using prediction loss $\mathcal{L}_{\text{pred}}$
6:         Extract knowledge vector $k_i$ from local encoder
7:     end for
8:     Compute average knowledge: $\bar{k} \leftarrow \frac{1}{N}\sum_{i=1}^{N} k_i$
9:     Fuse knowledge: $k_{\text{fused}} \leftarrow F_{\text{agg}}(\bar{k})$
10:    for all clients $i = 1$ to $N$ do
11:        Adapt knowledge: $\hat{k}_i \leftarrow A_i(k_{\text{fused}})$
12:    end for
13:    for all clients $i = 1$ to $N$ in parallel do
14:        Receive $\hat{k}_i$ from aggregator
15:        Compute distillation loss: $\mathcal{L}_{\text{KD}} \leftarrow \left\| \frac{k_i}{\|k_i\|} - \frac{\hat{k}_i}{\|\hat{k}_i\|} \right\|^2$
16:        Update local model with combined loss: $\mathcal{L}_i \leftarrow \alpha \cdot \mathcal{L}_{\text{pred}} + \beta \cdot \mathcal{L}_{\text{KD}}$
17:    end for
18: end for
```

Algorithm 6 Federated Knowledge Distillation with ID Embedding and Similarity-Based Fusion

```
1: Input: Local data $\mathcal{D}_i$, client ID $c_i$, number of clients $N$, rounds $R$, weights $\alpha, \beta$
2: Initialize: Model components $\theta_i = \{\text{Encoder}, \text{Decoder}, P_i, A_i, E(c_i)\}$ for each client $i$
3: for $r = 1$ to $R$ do
4:     for all clients $i = 1$ to $N$ in parallel do
5:         Train local model on $\mathcal{D}_i$ with prediction loss $\mathcal{L}_{\text{pred}}$
6:         Generate knowledge vector: $k_i = P_i(h_i + E(c_i))$
7:     end for
8:     Stack $k_i \in \mathbb{R}^{B\times H}$ for all clients $\to \mathcal{K} \in \mathbb{R}^{N\times B\times H}$
9:     Compute average: $\bar{k} = \frac{1}{N}\sum_i k_i$
10:    for all clients $i$ do
11:        Compute similarity weights $w_i = \text{softmax}(\cos(k_i, \bar{k}))$
12:    end for
13:    Fuse: $k_{\text{fused}} = \sum_i w_i \cdot k_i$
14:    Pass through fusion MLP: $k_{\text{global}} = F_{\text{agg}}(k_{\text{fused}})$
15:    for all clients $i$ in parallel do
16:        Adapt: $\hat{k}_i = A_i(k_{\text{global}})$
17:        Distill loss: $\mathcal{L}_{\text{KD}} = \|\text{norm}(k_i) - \text{norm}(\hat{k}_i)\|^2$
18:        Total loss: $\mathcal{L}_i = \alpha \cdot \mathcal{L}_{\text{pred}} + \beta \cdot \mathcal{L}_{\text{KD}}$
19:        Update model parameters with $\mathcal{L}_i$
20:    end for
21: end for
```

progressively decentralized the federation process to reflect a more distributed nature of trust and data ownership in a real-world scenario. Please note that we make an assumption that moving to these distributed setting would not change the model utility in terms of accuracy but will only cause changes in terms of communication time which is not looked at in this study.

Algorithm 7 Local Training with Dual-Layer DP in Mixed-Frequency Federated Learning

```
Require: Local HF data $\mathcal{D}_c^{HF}$, LF data $\mathcal{D}_c^{LF}$, initial model $\theta_g$, learning rate $\eta$, epochs $E$, encoder noise scale $\sigma_{enc}$, DP-SGD noise scale $\sigma_{grad}$, clipping bounds $B$, $C$
Ensure: Local model update $\Delta\theta_c$
1: Initialize local model: $\theta_c \leftarrow \theta_g$
2: for epoch = 1 to $E$ do
3:     for each mini-batch $(X_c^{HF}, X_c^{LF}, Y_c^{LF})$ do
4:         Encode HF data: $H_c \leftarrow \text{LSTMEncoder}(X_c^{HF})$
5:         Clip encoder outputs: $\tilde{H}_c \leftarrow \min(1, B/\|H_c\|_2) \cdot H_c$
6:         Add Gaussian noise: $H_c^{priv} \leftarrow \tilde{H}_c + \mathcal{N}(0, \sigma_{enc}^2 B^2 I)$
7:         LF predictions: $\hat{Y}_c^{LF} \leftarrow \text{LFDecoder}(X_c^{LF}, H_c^{priv})$
8:         Compute loss: $\mathcal{L}_c \leftarrow \text{MSELoss}(\hat{Y}_c^{LF}, Y_c^{LF})$
9:         Compute gradients: $g_c \leftarrow \nabla_{\theta_c}\mathcal{L}_c$
10:        Clip gradients: $\tilde{g}_c \leftarrow g_c / \max(1, \|g_c\|_2 / C)$
11:        Add Gaussian noise: $g_c^{priv} \leftarrow \tilde{g}_c + \mathcal{N}(0, \sigma_{grad}^2 C^2 I)$
12:        Update model: $\theta_c \leftarrow \theta_c - \eta \cdot g_c^{priv}$
13:    end for
14: end for
15: Compute update: $\Delta\theta_c \leftarrow \theta_c - \theta_g$
16: return $\Delta\theta_c$
```

Algorithm 8 Secure Aggregation Protocol

```
Require: Each client $u$ has local model update $x_u \in \mathbb{R}^d$
Require: Federator coordinates key sharing and aggregation
1: EACH client $u$ generates Diffie-Hellman keypair $(sk_u, pk_u)$
2: FEDERATOR collects and broadcasts all public keys $\{pk_u\}_{u\in[N]}$
3: for EACH client $u$ do
4:     COMPUTE pairwise shared keys $s_{u,v}$ using Diffie-Hellman with all $v \neq u$
5:     GENERATE $t$-out-of-$n$ secret shares for $s_{u,v}$ and random mask $r_u$
6:     COMPUTE masked update: $m_u = x_u + r_u$
7:     SEND $m_u$ and encrypted secret shares to federator
8: end for
9: FEDERATOR aggregates masked updates: $\bar{x} = \sum_u m_u$
10: if client dropout detected then
11:     RECONSTRUCT missing masks via $t$-out-of-$n$ scheme using received shares
12: end if
13: FEDERATOR cancels masks to reveal $\bar{x}$, the aggregated model update
14: return Aggregated model update $\bar{x}$
```

8.4.1 Semi-decentralized Setting: Random-Rotating Federator. To achieve semi-decentralization of federator as shown in Figure 14, our architecture employs a leader election mechanism called Verifiable Leader Election Protocol(VRF)[64] where a Task is randomly selected as the federator for each training round and every other

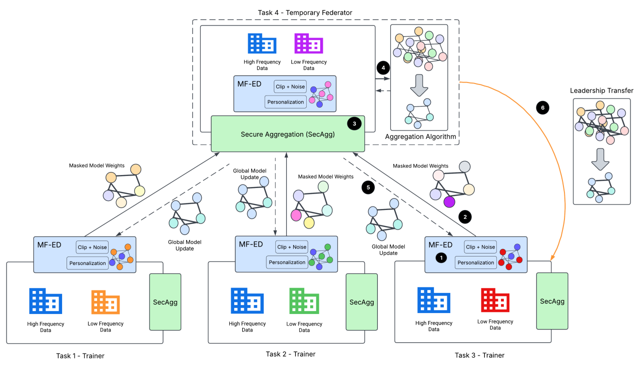


Figure 14: Confidentiality-preserving Semi Decentralize Federated Forecasting Framework, where MF-ED is multi-frequency encoder-decoder[37]

task can verify if election protocol was fair or not. When a task is elected as federator, it assumes aggregation responsibilities i.e. collecting masked model updates from all other tasks, securely aggregating them using Bonawitz-style SecAgg[11], and redistributing the global model. The VRF mechanism ensures the selection process is both random and publicly verifiable. Leadership rotates each round to build fairness and prevent centralization.

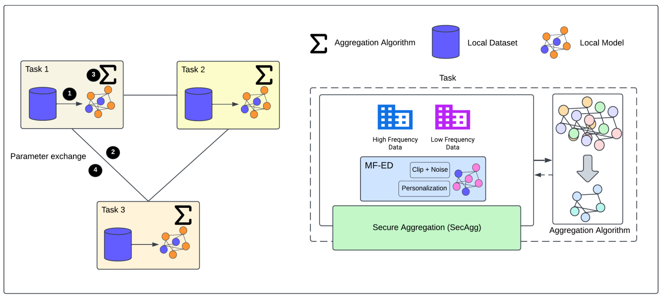


Figure 15: Peer-to-Peer fully decentralized federated learning, where MF-ED is multi-frequency encoder-decoder[37]

8.4.2 Peer-to-Peer: Fully Decentralized Collaboration. While our federated framework could adopt a semi-decentralized approach with a rotating federator to balance coordination and fairness, the architecture can be extended further into a fully decentralized, peer-to-peer (P2P) design as shown in Figure 15. In such a setup, there is no federator at all, not even temporarily. Instead, every task independently participates in aggregation, and coordination is achieved through cryptographic protocols similar to semi-decentralized but without any role assignment. In a fully peer-to-peer setting:

- No single task is ever elected as a leader.
- Each participant locally trains its model, masks its update, and engages in a collaborative aggregation protocol with every other peer.

Our existing use of Bonawitz-style SecAgg remains compatible with this topology, as long as participants cooperatively execute the aggregation without a federator. Though communication overhead increases with full decentralization, this mode maximizes autonomy and fits environments where equal trust and governance symmetry are essential.

In summary, while our semi-decentralized setup balances trust and coordination of decentralized federator, and efficiency of centralized federator and this design allows for evolution toward a peer-to-peer topology where aggregation becomes a collective computation rather than a delegated tfask.

8.5 Threat Model.

To evaluate the confidentiality guarantees of our federated framework, we analyze its behavior under some standard adversarial models specifically honest-but-curious and protocol-conforming-but-active participants across all critical entities: the HF party, the Customer-side LF party, and the Federator. Since aggregation in federated learning can take different forms, we structure our analysis around three federation schemas:

- Centralized Federator – a trusted third part that aggregates all updates.
- Semi-Decentralized Federator – our proposed rotating-leader design using verifiable election to assign aggregation temporarily to a task.
- Fully Decentralized (Peer-to-Peer) – no federator; aggregation is jointly computed using symmetric protocol.

We begin with the centralized setting, which forms the strongest adversarial baseline: since a permanent federator observes all aggregate updates across time, any confidentiality guarantees proven in this model serve as upper bounds on adversarial access in more decentralized alternatives. All subsequent designs strictly reduce the federator's temporal visibility and concentration of trust. To make the guarantees concrete, we analyze the framework's behavior under two widely accepted adversarial models.

- Honest-but-curious (semi-honest): Each party(HF, LF and Federator) follows the protocol but passively inspects every value it receives.
- Active-but-protocol-conforming: Parties may drop out or collude with other independent data-holding entities (including the Federator) to increase their knowledge, yet they do not deviate from the prescribed message flow (no arbitrary message injection or Byzantine behavior). Concrete colluding threat scenarios are presented in Table 6.

9 Additional experiments results analysis.

9.1 Modeling Strategies

To evaluate best performing model explained in Appendix 8.1, we use three different dataset SOLAR, Crypto, and Sales to test behavior of models across different strategies.

The following tables 5-11 report mean absolute error (MAE) with SEM. In almost every setting, MTL, i.e. explicit collaboration, either matches or outperforms the non-collaborative and implicit-collaborative baselines, highlighting the value of shared representations while maintaining task specific representation when task distributions differ. Notably, Simple LSTM and Deep Transformer performs the best with MTL on the new SOLAR and Crypto datasets,

Table 6: Collusion threat scenarios in a centralized federated learning setting and corresponding privacy defenses.

Colluding Entities	Practical Goal	Why the Design Still Protects
High-Frequency (HF) Party + Server	Reconstruct low-frequency (LF) targets or gradients	HF party does not observe LF data; the server receives only masked and DP-noised aggregated updates, ensuring individual contributions remain private.
Low-Frequency (LF) Party + Server	Infer proprietary HF representations	HF encoder outputs are clipped and Gaussian-noised before sharing; Secure Aggregation hides per-party updates.
Multiple HF Parties (Cross-task)	Triangulate a single party's update by subtracting their own	Requires breaking Secure Aggregation threshold; even then, DP noise limits what can be inferred if masks are exposed.
Multiple LF Parties (Cross-task)	Infer internal HF signals	Same protection as above; DP noise added on the HF side ensures LF parties never observe unclipped latent vectors.
HF Party and Paired LF Party within the Same Task	Reveal hidden activations via gradient inspection	Gradients crossing between HF and LF parties are clipped and perturbed with DP noise before leaving HF, ensuring (ϵ, δ)-DP and making single record contributions indistinguishable.

whereas ARIMAX remains competitive only for very short horizons ($H = 1–2$).

9.2 Non Collaborative and Collaborative Strategies with Longer horizons.

Table 15 shows experiment results for longer horizons i.e. horizon $H = 1, 2, 4, 8, 16$ which is the expanded version of Table 2, 3 and 4. For *Collaborative* setting we clearly see that our method consistently outperforms existing baselines for longer horizons as well which proves the stability of our model for such longer horizons due to regularization effect of our method that does not let any task dominate. Further longer horizons have higher uncertainty, making gradient conflicts more severe and our GradBal method alleviates it. This hypothesis is supported by *Non-Collaborative* setting in Table 15 where we see our model performance degrades considerably on longer horizons. We also see that UniTS performs competitively with could mean a transformer based model could possibly outperform our LSTM based model given we have more data.

Further in Table 16 we show what happens when we add random noise to the gradients during training. The comparison between our primary method ("Ours^{+}") and the random noise baseline ("Ours-random noise") demonstrates that simply adding random noise to gradients is not sufficient. Across all datasets and forecasting horizons, the random noise approach performs significantly worse than our sophisticated gradient balancing method. For example, on the Spain dataset at H=16, our method achieves 0.21 while random noise yields 0.46 - more than doubling the error. This substantial performance gap proves that our improvements come from intelligent gradient conflict resolution rather than simple noise regularization, validating the importance of our principled approach to multi-client learning in mixed-frequency time series forecasting. We further see that without any conflict resolution techniques(Ours^{-}), we get inconclusive results when compared to Ours^{+}, which clearly shows that a strategic conflict mitigation method helps better model heterogeneous tasks in a multi-task learning environment.

9.3 Confidentiality Analysis.

Here we perform confidentiality analysis for our federated setting. In Figure 16-19 we observe how for all datasets having secured aggregation improves the model utility due to reduced noise needed for similar privacy guarantee. We also see in dual DP scenario which noise effects the model performance the most and we find that DP-SGD(σ_{grad}) hurts the model utility the most which shows using equation by truex makes most sense to reduce noise needed for the DP-SGD. Figure 20-27 shows how changing encoder level noise(σ_{enc}) doesn't have much effect on model utility. This is further supported by the results shown from Figure 28-31 where we clearly see changing encoder noise(σ_{enc}) does not cause much change in model utility except for the Wind dataset.

For Wind dataset we see encoder noise effects model performance by a lot and Figure 29 even shows that a certain encoder noise helps the model converge better. This could mean that model struggles to fit on this particular dataset and a certain noise acts as a regularizer and ends up improving model utility. For these plots we also use privacy accountant to see when we run out of privacy budget for each dataset given the parameters needed by the privacy accountant opacus and we stop training as soon as we run out of privacy budget. For public dataset we run out of privacy budget for lower noise levels but not for Industry dataset. This can be attributed to relatively larger amount of data that we have specifically for Industry dataset.

9.4 Qualitative Results

This section presents a qualitative evaluation of model performance on the Spain Load Forecasting dataset, comparing our method against the best-performing baselines. Figure 32 provides a visual comparison of the three top-performing approaches: our CrossFreqNet-GradBal model, the CrossFreqNet baseline, and CrossFreqNet-PCGRAD.

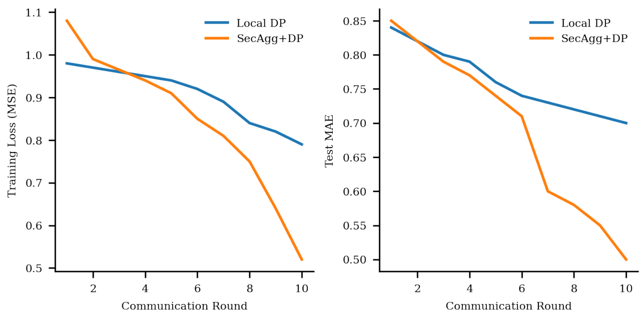


Figure 16: FedAvg Training with $\epsilon_{enc} = 1$ & $\epsilon_{grad} = 1$ for Air Quality Dataset for 10 epochs per round

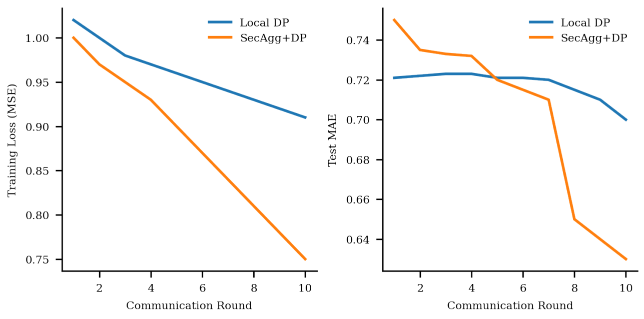


Figure 17: FedAvg Training with $\epsilon_{enc} = 1$ & $\epsilon_{grad} = 1$ for Load Dataset for 10 epochs per round

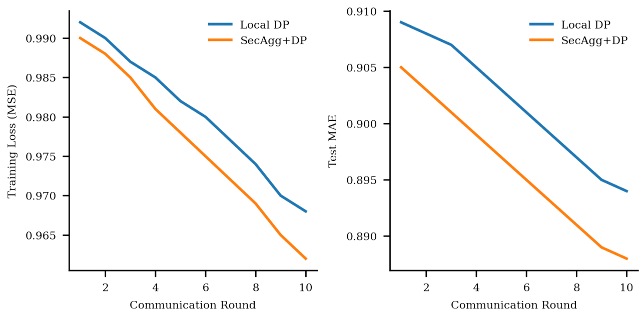


Figure 18: FedAvg Training with $\epsilon_{enc} = 1$ & $\epsilon_{grad} = 1$ for Wind Dataset for 10 epochs per round

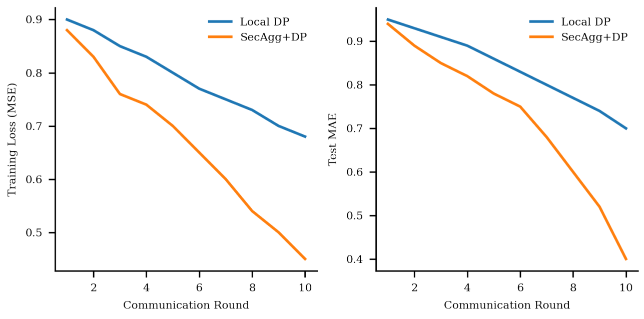


Figure 19: FedAvg Training with $\epsilon_{enc} = 1$ & $\epsilon_{grad} = 1$ for Industry Dataset for 10 epochs per round

Table 7: MAE scores of Simple LSTM Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.716	0.77	0.77	0.87	0.92
		Solar	0.227	0.30	0.30	0.40	0.47
	No Collaboration	Crypto	0.51	0.55	0.66	0.73	0.71
		Sales	0.32	0.39	0.45	0.49	0.51
		Industry	0.70	0.71	0.68	0.68	0.68
		Air Quality	0.35	0.44	0.53	0.73	0.90
		Solar	0.15	0.14	0.16	0.21	0.24
Simple LSTM	Implicit Collaboration	Crypto	0.19	0.20	0.22	0.26	0.29
		Sales	0.32	0.35	0.36	0.38	0.44
		Industry	0.60	0.77	0.67	0.66	0.65
		Air Quality	**0.14**	**0.15**	**0.21**	**0.32**	**0.44**
		Solar	**0.11**	**0.13**	**0.15**	**0.21**	**0.30**
	Explicit Collaboration	Crypto	**0.15**	**0.15**	**0.17**	**0.20**	**0.25**
		Sales	**0.17**	**0.18**	**0.20**	**0.21**	**0.26**
		Industry	**0.62**	**0.54**	**0.55**	**0.50**	**0.53**

Table 8: MAE scores of Simple Transformer Encoder Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.51	0.55	0.61	0.70	0.81
		Solar	0.22	0.23	0.24	0.25	0.28
	No Collaboration	Crypto	0.19	0.19	0.21	0.24	0.29
		Sales	0.23	0.24	0.25	0.27	0.34
		Industry	0.71	0.71	0.93	0.77	0.66
		Air Quality	0.43	0.51	0.56	0.65	0.79
		Solar	**0.10**	**0.11**	**0.15**	**0.19**	**0.25**
Simple Transformer Encoder	Implicit Collaboration	Crypto	0.17	0.19	0.23	0.26	0.34
		Sales	0.21	0.22	0.27	0.30	0.37
		Industry	**0.55**	**0.60**	**0.51**	**0.46**	**0.48**
		Air Quality	**0.12**	**0.14**	**0.20**	**0.29**	**0.39**
		Solar	0.18	0.19	0.21	0.22	0.26
	Explicit Collaboration	Crypto	**0.14**	**0.15**	**0.18**	**0.22**	**0.27**
		Sales	**0.16**	**0.18**	**0.21**	**0.25**	**0.30**
		Industry	0.71	0.67	0.62	0.58	0.65

Table 9: MAE scores of Deep LSTM Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.20	0.24	0.28	0.43	0.47
		Solar	0.15	0.18	0.16	0.21	0.27
	No Collaboration	Crypto	0.11	0.13	0.16	0.19	0.29
		Sales	0.15	0.17	0.21	0.24	0.32
		Industry	0.71	0.87	1.06	0.95	0.79
		Air Quality	0.35	0.44	0.54	0.73	0.90
		Solar	0.14	0.13	0.16	0.20	0.23
FATHOM[14]	Implicit Collaboration	Crypto	0.19	0.20	0.22	0.26	0.29
		Sales	0.23	0.24	0.27	0.30	0.33
		Industry	**0.60**	**0.77**	**0.67**	**0.66**	**0.65**
		Air Quality	**0.13**	**0.17**	**0.21**	**0.31**	**0.40**
		Solar	**0.10**	**0.12**	**0.15**	**0.19**	**0.26**
	Explicit Collaboration	Crypto	**0.11**	**0.10**	**0.15**	**0.16**	**0.25**
		Sales	**0.14**	**0.13**	**0.18**	**0.20**	**0.27**
		Industry	0.80	1.01	1.00	0.80	0.61

The results demonstrate that our centralized model with gradient balancing achieves superior prediction accuracy, particularly in the highlighted regions where significant deviations occur between ground truth and predictions. Our method consistently tracks the actual values more closely than competing approaches, especially during periods of high volatility. Additional qualitative comparisons with extended baselines are presented in Figures 33 through 40, further confirming the robustness of our approach across different forecasting scenarios.

Table 10: MAE scores of Deep Transformer Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.13	0.16	0.21	0.29	0.42
		Solar	0.10	0.11	0.15	0.20	0.26
	No Collaboration	Crypto	0.14	0.15	0.18	0.20	0.28
		Sales	0.17	0.18	0.21	0.23	0.31
		Industry	0.71	1.01	1.18	0.97	0.89
		Air Quality	**0.43**	**0.51**	**0.56**	**0.65**	**0.79**
		Solar	0.18	0.19	0.21	0.22	0.26
Shared-Private Attention[17]	Implicit Collaboration	Crypto	0.17	0.19	0.23	0.26	0.34
		Sales	0.21	0.22	0.25	0.28	0.37
		Industry	**0.55**	**0.60**	**0.51**	**0.46**	**0.43**
		Air Quality	0.10	0.14	0.20	0.28	0.40
		Solar	**0.09**	**0.12**	**0.13**	**0.17**	**0.22**
	Explicit Collaboration	Crypto	**0.12**	**0.14**	**0.18**	**0.20**	**0.24**
		Sales	**0.15**	**0.17**	**0.21**	**0.23**	**0.27**
		Industry	1.07	0.90	1.08	1.18	1.03

Table 11: MAE scores of ARIMAX Across in Independent Setting and across Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.10	0.27	0.31	0.45	0.58
		Solar	0.01	0.01	0.02	0.03	0.13
ARIMAX	No Collaboration	Crypto	0.05	0.10	0.03	0.33	0.36
		Sales	0.83	0.44	0.75	0.99	0.51
		Industry	0.24	0.34	0.37	0.48	0.54

Table 12: MAE scores of Shared-HF-LF Direct model Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.20	0.29	0.37	0.46	0.45
		Solar	0.18	0.21	0.25	0.28	0.41
	No Collaboration	Crypto	0.18	0.24	0.25	0.28	0.33
		Sales	0.10	0.13	0.18	0.21	0.28
		Industry	**0.17**	**0.26**	**0.33**	**0.39**	**0.47**
		Air Quality	0.24	0.38	0.42	0.51	0.61
		Solar	0.14	0.17	0.21	0.29	0.38
Shared HF-LF	Implicit Collaboration	Crypto	0.17	0.21	0.29	0.34	0.41
		Sales	0.14	0.20	0.23	0.31	0.36
		Industry	0.25	0.38	0.50	0.59	0.69
		Air Quality	0.20	0.33	0.57	0.77	0.97
		Solar	0.12	0.15	0.19	0.23	0.29
	Explicit Collaboration	Crypto	0.10	0.13	0.18	0.25	0.28
		Sales	0.12	0.15	0.24	0.30	0.32
		Industry	0.20	0.41	0.57	0.81	0.90

Table 13: MAE scores of Shared-HF-LF Direct model Across Learning Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
		Air Quality	0.28	0.39	0.49	0.60	0.61
		Solar	0.26	0.31	0.37	0.42	0.57
	No Collaboration	Crypto	0.26	0.34	0.37	0.42	0.49
		Sales	0.18	0.23	0.30	0.35	0.44
		Industry	0.30	0.43	0.58	0.80	1.45
		Air Quality	0.34	0.50	0.56	0.67	0.79
		Solar	0.24	0.29	0.35	0.45	0.56
Secured HF-LF	Implicit Collaboration	Crypto	0.27	0.33	0.43	0.50	0.59
		Sales	0.24	0.32	0.37	0.47	0.54
		Industry	0.24	0.41	0.59	0.76	0.91
		Air Quality	0.28	0.32	0.45	0.69	0.79
		Solar	0.24	0.29	0.35	0.41	0.49
	Explicit Collaboration	Crypto	0.22	0.27	0.34	0.43	0.48
		Sales	0.24	0.29	0.40	0.48	0.52
		Industry	0.22	0.38	0.55	0.75	1.13

Table 14: MAE scores Federated Strategies and Horizons

Model Type	Learning Strategy	Dataset	H=1	H=2	H=4	H=8	H=16
Federated Learning	FedAvg	Air Quality	0.19	0.30	0.46	0.66	0.80
		Solar	0.14	0.20	0.32	0.43	0.57
		Crypto	0.24	0.28	0.42	0.43	0.49
		Sales	0.20	0.31	0.37	0.38	0.50
		Industry	0.12	0.26	0.50	0.65	0.85
	FedKD	Air Quality	0.38	0.50	0.57	0.66	0.82
		Solar	0.26	0.33	0.39	0.49	0.62
		Crypto	0.27	0.33	0.43	0.50	0.59
		Sales	0.28	0.36	0.41	0.51	0.58
		Industry	0.22	0.39	0.59	0.76	0.90
	FedDKP	Air Quality	0.34	0.43	0.52	0.66	0.81
		Solar	0.22	0.28	0.34	0.44	0.58
		Crypto	0.27	0.31	0.43	0.48	0.57
		Sales	0.24	0.32	0.39	0.38	0.56
		Industry	0.18	0.38	0.56	0.73	0.86

Learning Strategy	*Dataset*	*H=1*	*H=2*	*H=4*	*H=8*	*H=16*
Non-Collaborative(ARIMAX)	Air Quality	0.33±0.002	0.35±0.002	0.38±0.001	0.42±0.002	0.48±0.002
	Load	0.47±0.001	0.56±0.001	0.69±0.003	0.87±0.003	0.95±0.005
	Wind	0.82±0.003	0.83±0.002	0.83±0.006	0.84±0.004	0.85±0.004
	Spain	1.01±0.001	1.05±0.001	1.06±0.001	1.13±0.002	1.20±0.002
	Industry	0.24±0.001	0.34±0.002	0.38±0.002	0.50±0.002	0.60±0.003
Non-Collaborative(UniTS)	Air Quality	0.30±0.030	0.35±0.020	0.38±0.020	0.46±0.030	0.56±0.040
	Load	0.47±0.030	0.55±0.025	0.63±0.027	0.76±0.030	0.88±0.033
	Wind	0.70±0.015	0.75±0.018	0.84±0.022	0.88±0.024	0.91±0.026
	Spain	0.23±0.018	0.28±0.020	0.32±0.022	0.36±0.025	0.44±0.028
	Industry	0.51±0.020	0.52±0.040	0.54±0.040	0.54±0.060	0.56±0.080
Non-Collaborative(TSDiff)	Air Quality	1.89±0.040	1.78±0.050	1.50±0.050	1.28±0.060	1.10±0.080
	Load	1.26±0.020	0.96±0.020	0.88±0.040	0.82±0.050	0.79±0.070
	Wind	1.07±0.020	1.03±0.040	1.03±0.040	1.02±0.060	0.99±0.080
	Spain	1.37±0.040	1.17±0.030	1.08±0.040	1.03±0.040	1.00±0.030
	Industry	2.14±0.040	1.98±0.040	1.67±0.030	1.69±0.040	1.32±0.050
Non-Collaborative(Ours$^-$)	Air Quality	0.20$^{-33\%}$±0.004	0.29$^{-17\%}$±0.004	0.37$^{-3\%}$±0.004	0.46$^{+10\%}$±0.005	0.45$^{-6\%}$±0.006
	Load	**0.11**$^{-77\%}$±0.003	0.19$^{-65\%}$±0.003	0.19$^{-70\%}$±0.003	0.20$^{-74\%}$±0.004	0.30$^{-62\%}$±0.005
	Wind	**0.53**$^{-24\%}$±0.004	**0.54**$^{-28\%}$±0.004	0.58$^{-30\%}$±0.005	0.65$^{-23\%}$±0.006	0.70$^{-18\%}$±0.007
	Spain	0.27$^{+17\%}$±0.005	0.43$^{+54\%}$±0.006	0.45$^{+41\%}$±0.006	0.60$^{+67\%}$±0.007	0.52$^{+18\%}$±0.008
	Industry	0.19$^{-29\%}$±0.002	0.30$^{-23\%}$±0.004	0.74$^{-29\%}$±0.003	0.81$^{-34\%}$±0.006	1.11$^{-30\%}$±0.005
Collaborative(ARIMAX)	Air Quality	0.35±0.003	0.31±0.003	0.38±0.003	0.41±0.003	0.46±0.004
	Load	0.43±0.002	0.57±0.003	0.68±0.003	0.83±0.004	0.91±0.004
	Wind	0.74±0.003	0.76±0.003	0.78±0.004	0.83±0.004	0.88±0.005
	Spain	0.83±0.004	0.94±0.005	1.10±0.005	1.39±0.006	1.56±0.006
	Industry	0.28±0.003	0.35±0.003	0.38±0.003	0.51±0.004	0.60±0.004
Collaborative(TSDiff) [30]	Air Quality	1.05±0.040	0.95±0.050	0.89±0.050	0.87±0.060	0.86±0.080
	Load	1.26±0.020	0.96±0.020	0.88±0.040	0.82±0.050	0.79±0.070
	Wind	0.96±0.020	0.94±0.040	0.92±0.040	0.91±0.060	0.90±0.080
	Spain	1.18±0.040	1.05±0.030	1.02±0.040	1.99±0.040	0.97±0.030
	Industry	1.75±0.040	1.31±0.040	1.13±0.030	1.06±0.040	1.04±0.050
Collaborative(UniTS) [23]	Air Quality	0.29±0.030	0.39±0.040	**0.32**±0.040	0.39±0.060	0.50±0.070
	Load	0.40±0.040	0.50±0.040	0.60±0.050	0.72±0.060	0.85±0.080
	Wind	0.77±0.020	0.79±0.050	0.80±0.080	0.83±0.080	0.86±0.090
	Spain	0.21±0.040	0.25±0.050	0.29±0.050	0.33±0.070	0.40±0.080
	Industry	0.30±0.020	0.37±0.040	0.42±0.040	0.50±0.060	0.51±0.080
Collaborative(Ours$^+$)	Air Quality	**0.18** $^{-38\%}$±0.002	**0.26** $^{-33\%}$±0.002	0.37 $^{+16\%}$±0.003	**0.38** $^{-3\%}$±0.003	**0.40** $^{-20\%}$±0.005
	Load	**0.11** $^{-73\%}$±0.001	**0.16** $^{-68\%}$±0.002	**0.19** $^{-68\%}$±0.003	**0.23** $^{-68\%}$±0.004	**0.28** $^{-67\%}$±0.003
	Wind	0.63 $^{-18\%}$±0.002	**0.72** $^{-9\%}$±0.003	**0.72** $^{-10\%}$±0.004	**0.73** $^{-12\%}$±0.004	**0.72** $^{-16\%}$±0.007
	Spain	**0.11** $^{-48\%}$±0.003	**0.14** $^{-44\%}$±0.004	**0.16** $^{-45\%}$±0.004	**0.18** $^{-46\%}$±0.005	**0.21** $^{-48\%}$±0.006
	Industry	**0.09** $^{-67\%}$±0.002	**0.17** $^{-51\%}$±0.004	**0.26** $^{-36\%}$±0.003	**0.30** $^{-42\%}$±0.006	**0.34** $^{-22\%}$±0.005

Table 15: Mean Absolute Error (↓) with standard errors (SEM). The best result for each dataset–horizon is shown in bold; the runner-up is underlined. % improvements are shown against best performing Independent setting and best performing centralized setting respectively.

Learning Strategy	*Dataset*	*H=1*	*H=2*	*H=4*	*H=8*	*H=16*
TSDiff-PCGrad	Air	1.18±0.04	0.95±0.05	0.92±0.05	0.89±0.06	0.87±0.08
	Load	1.25±0.02	0.80±0.02	0.86±0.04	0.92±0.05	0.92±0.08
	Wind	0.96±0.02	0.92±0.04	0.91±0.04	0.91±0.04	0.90±0.03
	Spain	1.15±0.04	1.03±0.03	0.99±0.04	0.96±0.04	0.97±0.03
	Industry	2.11±0.04	1.98±0.04	1.67±0.03	1.56±0.04	1.58±0.05
TSDiff-CAGrad	Air	0.94±0.04	0.91±0.05	0.90±0.05	0.90±0.06	0.90±0.05
	Load	0.71±0.02	0.71±0.02	0.70±0.05	0.69±0.05	0.68±0.08
	Wind	1.01±0.03	1.01±0.03	1.01±0.03	1.00±0.05	0.98±0.08
	Spain	1.09±0.04	1.09±0.03	1.11±0.04	1.13±0.04	1.14±0.03
	Industry	0.98±0.04	0.99±0.04	1.02±0.04	1.03±0.04	1.04±0.05
UniTS-PCGrad	Air	0.22±0.03	0.26±0.04	0.33±0.04	0.42±0.06	0.50±0.07
	Load	0.37±0.04	0.46±0.04	0.55±0.05	0.66±0.06	0.82±0.08
	Wind	0.73±0.04	0.72±0.06	0.74±0.08	0.80±0.08	0.84±0.09
	Spain	$\underline{0.19\pm0.04}$	$\underline{0.24\pm0.05}$	$\underline{0.26\pm0.05}$	$\underline{0.30\pm0.07}$	$\underline{0.38\pm0.08}$
	Industry	0.28±0.02	0.36±0.04	0.38±0.04	0.48±0.06	0.50±0.08
UniTS-CAGrad	Air	0.26±0.03	0.30±0.04	0.37±0.05	0.44±0.06	0.54±0.07
	Load	0.41±0.04	0.50±0.05	0.61±0.06	0.71±0.07	0.83±0.09
	Wind	0.81±0.05	0.77±0.06	0.78±0.07	0.86±0.08	0.90±0.09
	Spain	0.55±0.04	0.55±0.04	0.56±0.05	0.57±0.05	0.58±0.06
	Industry	0.17±0.03	0.25±0.03	0.36±0.04	0.44±0.05	0.51±0.06
UniTS-GradBal	Air	0.23±0.03	0.25±0.04	0.35±0.04	0.40±0.06	0.51±0.07
	Load	0.40±0.04	0.48±0.04	0.58±0.05	0.62±0.06	0.81±0.08
	Wind	0.71±0.05	0.74±0.06	0.80±0.07	0.84±0.08	0.82±0.08
	Spain	0.22±0.04	0.26±0.05	0.30±0.05	0.34±0.07	0.37±0.08
	Industry	0.25±0.03	0.32±0.04	0.40±0.04	0.50±0.06	0.52±0.07
TSDiff-GradBal	Air	1.03±0.03	0.94±0.04	0.89±0.04	0.86±0.06	0.84±0.07
	Load	1.26±0.03	0.92±0.03	0.89±0.04	0.82±0.05	0.79±0.07
	Wind	0.96±0.04	0.92±0.04	0.91±0.04	0.90±0.06	0.90±0.08
	Spain	0.22±0.04	0.26±0.05	0.30±0.05	0.34±0.07	0.37±0.08
	Industry	1.95±0.03	1.98±0.03	1.50±0.03	1.40±0.04	1.80±0.05
Ours^{-}	Air Quality	0.20±0.003	$\underline{0.33\pm0.003}$	0.57±0.002	0.77±0.007	0.97±0.008
	Load	$\underline{0.12\pm0.001}$	**0.14±0.002**	$\underline{0.19\pm0.003}$	0.25±0.004	0.33±0.005
	Wind	0.82±0.002	0.94±0.004	0.99±0.004	0.98±0.004	0.97±0.007
	Spain	0.42±0.003	0.6±0.003	0.91±0.005	1.06±0.006	1.16±0.009
	Industry	0.17±0.002	0.36±0.003	$\underline{0.67\pm0.003}$	0.84±0.006	$\underline{1.15\pm0.005}$
Ours-PCGrad	Air Quality	**0.18±0.002**	$\underline{0.28\pm0.002}$	$\underline{0.42\pm0.003}$	$\underline{0.51\pm0.005}$	$\underline{0.60\pm0.006}$
	Load	$\underline{0.12\pm0.002}$	**0.14±0.003**	**0.17±0.003**	$\underline{0.24\pm0.003}$	$\underline{0.32\pm0.004}$
	Wind	$\underline{0.73\pm0.003}$	$\underline{0.82\pm0.004}$	$\underline{0.86\pm0.004}$	$\underline{0.85\pm0.005}$	$\underline{0.88\pm0.007}$
	Spain	$\underline{0.40}\pm0.003$	$\underline{0.69}\pm0.003$	$\underline{0.86}\pm0.004$	$\underline{1.04}\pm0.005$	$\underline{1.13}\pm0.007$
	Industry	$\underline{0.21\pm0.002}$	$\underline{0.38\pm0.002}$	0.65±0.004	$\underline{0.83\pm0.006}$	1.11±0.008
Ours-CAGrad	Air Quality	0.78±0.002	0.78±0.002	0.79±0.003	0.79±0.005	0.80±0.006
	Load	0.80±0.002	0.80±0.003	0.80±0.003	0.80±0.003	1.04±0.004
	Wind	0.80±0.003	0.80±0.004	0.81±0.004	0.81±0.005	0.96±0.007
	Spain	0.77±0.003	0.78±0.003	0.80±0.004	0.80±0.005	0.83±0.007
	Industry	0.55±0.002	0.55±0.002	0.55±0.004	0.54±0.006	0.54±0.008
Ours^{+}	Air Quality	$\mathbf{0.18}^{(0\%)}\pm0.002$	$\mathbf{0.26}^{(-7.1\%)}\pm0.002$	$\mathbf{0.37}^{(-11.9\%)}\pm0.003$	$\mathbf{0.38}^{(-25.5\%)}\pm0.004$	$\mathbf{0.40}^{(-33.3\%)}\pm0.005$
	Load	$\mathbf{0.11}^{(-8.3\%)}\pm0.001$	$0.16^{(+14.3\%)}\pm0.002$	$\underline{0.19}^{(+11.8\%)}\pm0.003$	$\mathbf{0.23}^{(-4.2\%)}\pm0.004$	$\mathbf{0.28}^{(-12.5\%)}\pm0.003$
	Wind	$\mathbf{0.63}^{(-13.7\%)}\pm0.002$	$\mathbf{0.72}^{(-12.2\%)}\pm0.003$	$\mathbf{0.72}^{(-16.3\%)}\pm0.004$	$\mathbf{0.73}^{(-14.1\%)}\pm0.004$	$\mathbf{0.72}^{(-18.2\%)}\pm0.007$
	Spain	$\mathbf{0.11}^{(-48\%)}\pm0.003$	$\mathbf{0.14}^{(-56\%)}\pm0.004$	$\mathbf{0.16}^{(-59\%)}\pm0.004$	$\mathbf{0.18}^{(-42\%)}\pm0.005$	$\mathbf{0.21}^{(-58\%)}\pm0.006$
	Industry	$\mathbf{0.09}^{(-41.2\%)}\pm0.002$	$\mathbf{0.17}^{(-40.0\%)}\pm0.004$	$\mathbf{0.26}^{(-55.0\%)}\pm0.003$	$\mathbf{0.30}^{(-59.2\%)}\pm0.006$	$\mathbf{0.34}^{(-62.6\%)}\pm0.005$
Ours-random noise	Air Quality	0.25±0.002	0.32±0.002	0.48±0.003	0.51±0.004	0.58±0.005
	Load	0.38±0.001	0.16±0.002	0.41±0.003	0.48±0.004	0.53±0.003
	Wind	0.83±0.002	0.94±0.003	0.92±0.004	0.97±0.004	0.97±0.007
	Spain	0.23±0.003	0.25±0.004	0.30±0.004	0.34±0.005	0.46±0.006
	Industry	0.29±0.002	0.32±0.004	0.55±0.003	0.59±0.006	0.61±0.005

Table 16: Mean Absolute Error (↓), standard error of mean (SEM), and % improvement over PCGrad for different conflict-resolution strategies across multiple horizons (H). Bold indicates best, underline indicates second-best.

Learning Strategy	*Dataset*	*H=1*	*H=2*	*H=4*	*H=8*	*H=16*
FedAvg – No confidentiality	Air Quality	**0.19**±0.001	$\underline{0.30}$±0.003	0.46±0.003	$\underline{0.66}$±0.004	$\underline{0.80}$±0.007
	Load	$\underline{0.12}$±0.002	**0.14**±0.002	**0.25**±0.003	**0.30**±0.005	**0.32**±0.008
	Wind	$\underline{0.71}$±0.003	**0.72**±0.004	**0.72**±0.003	**0.86**±0.007	**0.87**±0.008
	Spain	$\underline{0.41}$±0.003	0.75±0.004	1.05±0.004	1.01±0.006	1.04±0.008
	Industry	**0.28**±0.001	**0.50**±0.003	**0.64**±0.004	**0.69**±0.005	**1.01**±0.008
FedAvg – Private	Air Quality	$\underline{0.20}$±0.001	0.32±0.003	$\underline{0.49}$±0.003	0.69±0.004	0.83±0.006
	Load	0.17±0.002	0.28±0.002	0.43±0.004	0.41±0.006	0.36±0.007
	Wind	0.74±0.003	$\underline{0.75}$±0.003	$\underline{0.75}$±0.004	$\underline{0.89}$±0.004	$\underline{0.90}$±0.006
	Spain	0.45±0.002	**0.62**±0.002	$\underline{0.74}$±0.003	$\underline{0.92}$±0.005	$\underline{0.95}$±0.008
	Industry	$\underline{0.16}$±0.001	$\underline{0.33}$±0.003	$\underline{0.50}$±0.003	$\underline{0.74}$±0.003	$\underline{0.97}$±0.007
FedProx – No confidentiality	Air Quality	**0.19**±0.002	**0.28**±0.002	**0.44**±0.003	**0.65**±0.004	0.82±0.006
	Load	**0.11**±0.001	0.20±0.002	$\underline{0.30}$±0.003	$\underline{0.32}$±0.003	$\underline{0.35}$±0.004
	Wind	**0.70**±0.003	0.84±0.004	0.85±0.004	0.90±0.005	0.92±0.006
	Spain	**0.32**±0.002	**0.55**±0.003	**0.62**±0.004	**0.66**±0.004	**0.73**±0.006
	Industry	0.14±0.001	0.32±0.003	0.64±0.003	0.82±0.004	1.14±0.003
FedProx – Private	Air Quality	**0.19**±0.002	$\underline{0.31}$±0.003	0.47±0.003	0.68±0.004	0.82±0.006
	Load	$\underline{0.16}$±0.002	$\underline{0.21}$±0.002	0.42±0.004	0.38±0.005	**0.33**±0.006
	Wind	$\underline{0.72}$±0.003	0.88±0.004	0.88±0.004	0.90±0.005	0.91±0.006
	Spain	$\underline{0.44}$±0.002	0.61±0.003	0.73±0.004	0.91±0.005	0.96±0.007
	Industry	0.18±0.002	0.34±0.003	0.53±0.004	0.72±0.004	0.89±0.006
SCAFFOLD – No confidentiality	Air Quality	0.45±0.004	0.50±0.005	0.63±0.006	0.70±0.007	$\underline{0.81}$±0.008
	Load	0.32±0.003	0.45±0.004	0.54±0.005	0.61±0.006	0.67±0.007
	Wind	0.83±0.004	0.88±0.005	0.89±0.005	0.92±0.006	0.96±0.007
	Spain	0.76±0.004	0.78±0.005	0.85±0.006	0.93±0.007	0.98±0.008
	Industry	1.59±0.003	1.87±0.004	1.92±0.005	1.98±0.004	2.10±0.008
SCAFFOLD – Private	Air Quality	0.50±0.004	0.54±0.005	0.64±0.006	0.74±0.007	0.85±0.008
	Load	0.35±0.003	0.49±0.004	0.59±0.005	0.72±0.007	0.78±0.008
	Wind	0.88±0.004	0.91±0.005	0.92±0.005	0.98±0.006	1.07±0.008
	Spain	0.78±0.004	0.80±0.005	0.90±0.006	1.07±0.008	1.14±0.009
	Industry	2.27±0.003	4.21±0.005	2.52±0.006	1.97±0.008	2.96±0.008

Table 17: Federated learning on heterogeneous mixed-frequency tasks. We compare FedAvg, FedProx and SCAFFOLD with and without differential-confidentiality noise ($\epsilon_{enc} = 3$ and $\epsilon_{grad} = 3$). Values are MAE ± SEM (lower is better); bold = best, underline = second best.

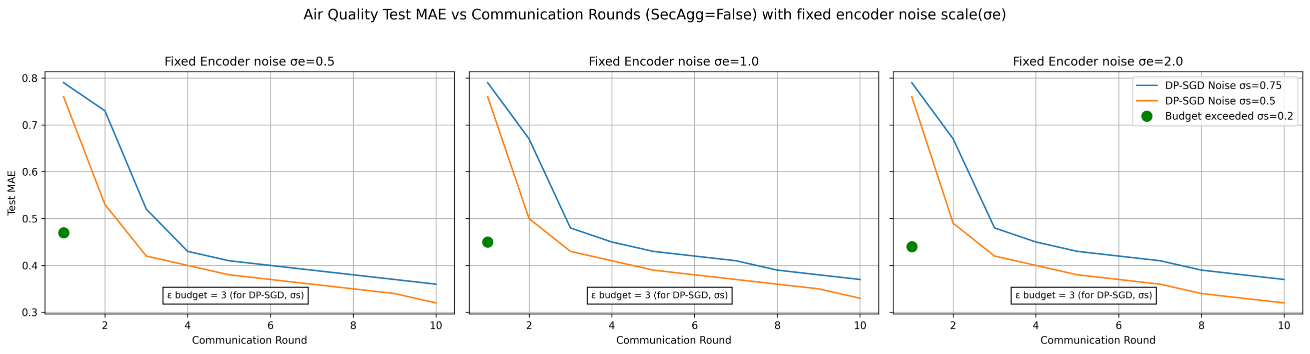


Figure 20: MAE(↓) for Air Quality dataset with fixed encoder output noise(σ_{enc}) and varying gradient noise(σ_{grad}) with $\epsilon_{grad} = 3$ and without secured aggregation

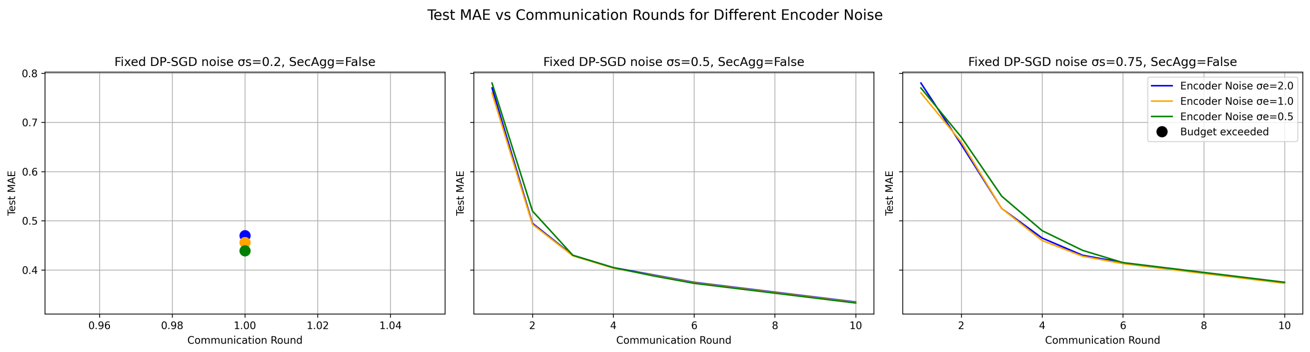


Figure 21: MAE(↓) for Air Quality dataset with fixed DP-SGD noise(σ_{grad}) and varying encoder output noise(σ_{enc}) with $\epsilon_{grad} = 3$ and without secured aggregation

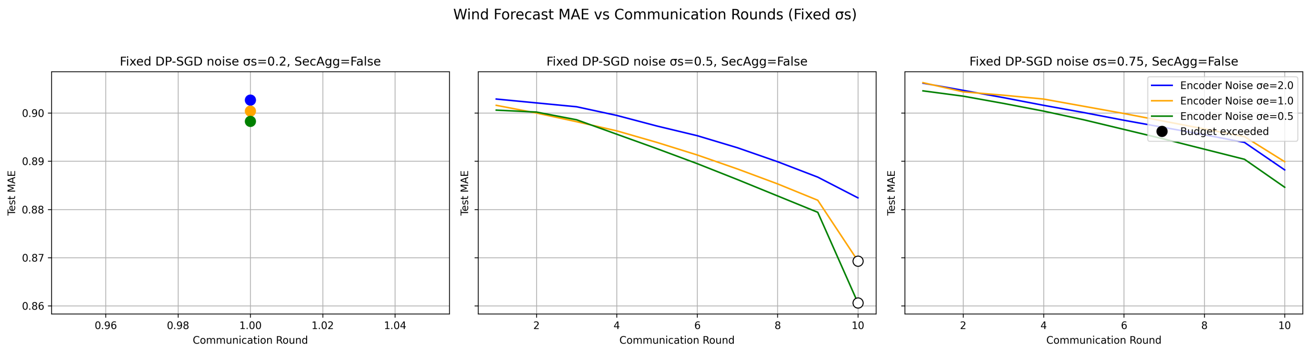


Figure 22: MAE(↓) for Wind dataset with fixed encoder output noise(σ_{enc}) and varying gradient noise(σ_{grad}) with $\epsilon_{grad} = 3$ and without secured aggregation

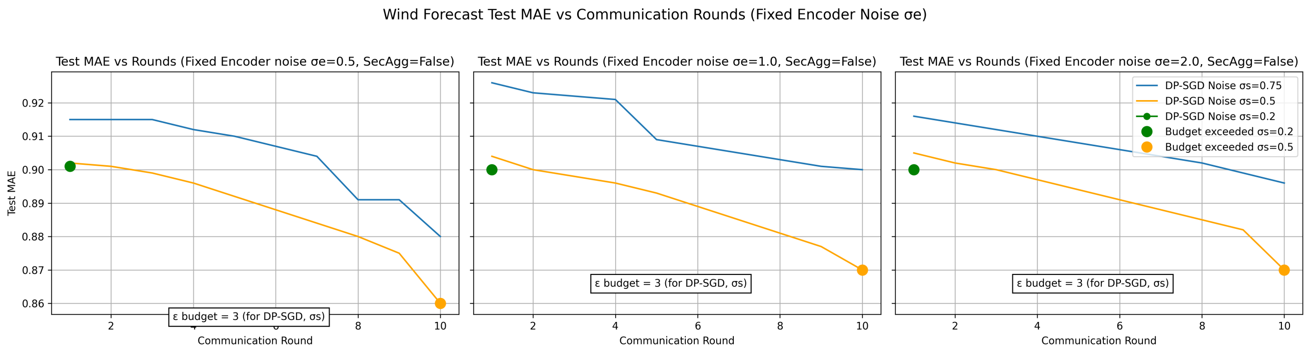


Figure 23: MAE(↓) for Wind dataset with fixed DP-SGD noise(σ_{grad}) and varying encoder output noise(σ_{enc}) with $\epsilon_{grad} = 3$ and without secured aggregation

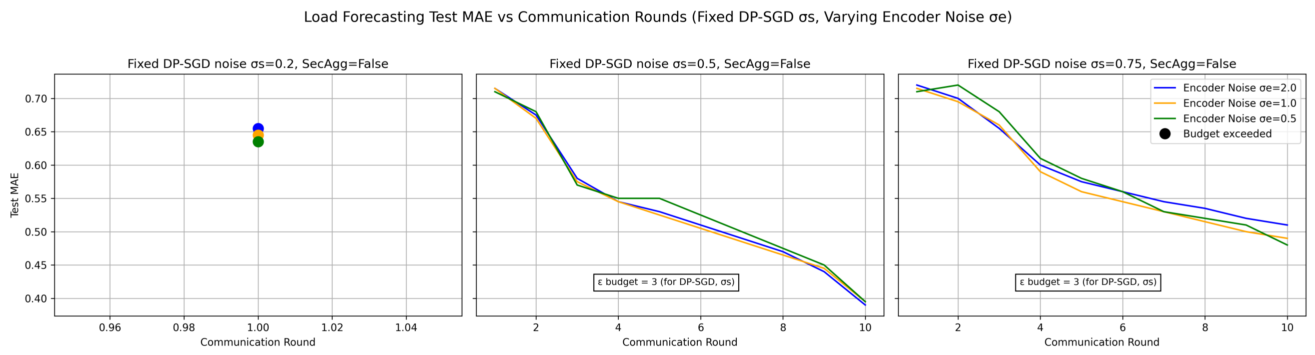


Figure 24: MAE(↓) for Load dataset with fixed encoder output noise(σ_{enc}) and varying gradient noise(σ_{grad}) with ϵ_{grad} = 3 and without secured aggregation

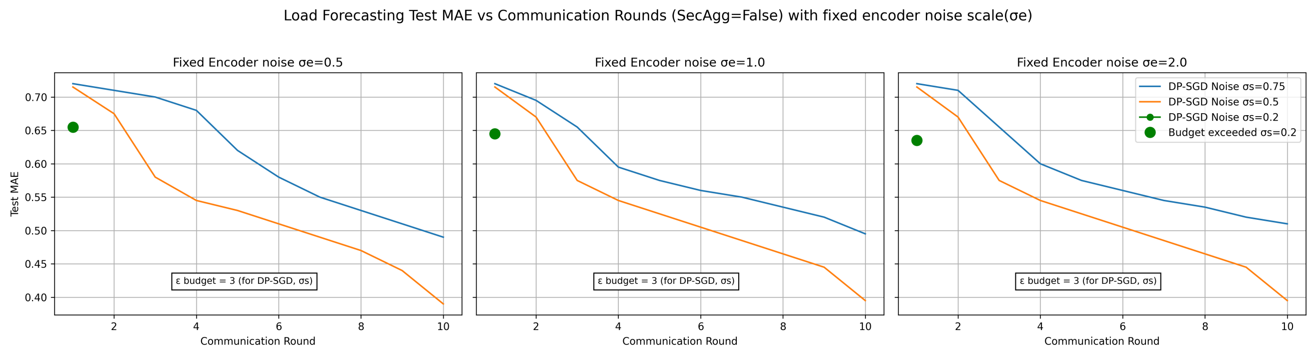


Figure 25: MAE(↓) for Load dataset with fixed DP-SGD noise(σ_{grad}) and varying encoder output noise(σ_{enc}) with ϵ_{grad} = 3 and without secured aggregation

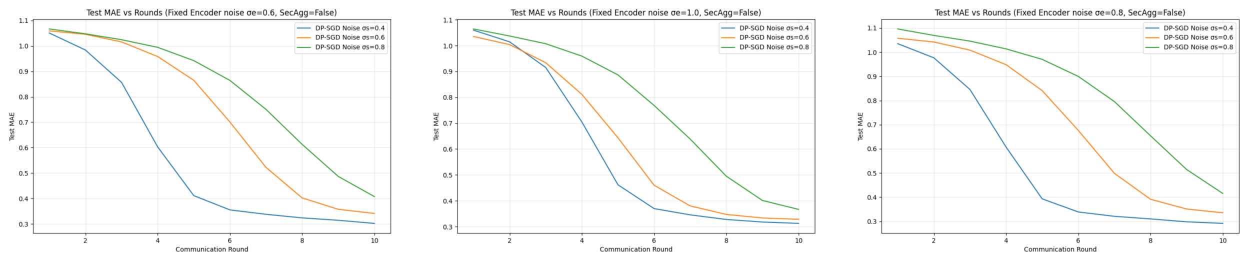


Figure 26: MAE(↓) for Industry dataset with fixed encoder output noise(σ_{enc}) and varying gradient noise(σ_{grad}) with ϵ_{grad} = 3 and without secured aggregation

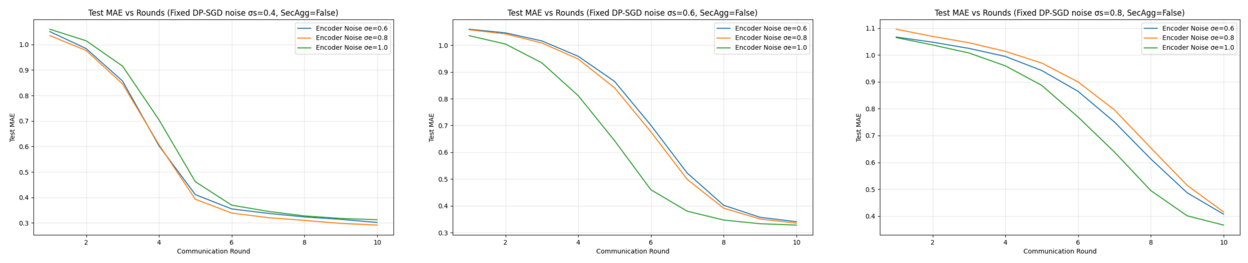


Figure 27: MAE(↓) for Industry dataset with fixed DP-SGD noise(σ_{grad}) and varying encoder output noise(σ_{enc}) with $\epsilon_{grad} = 3$ and without secured aggregation

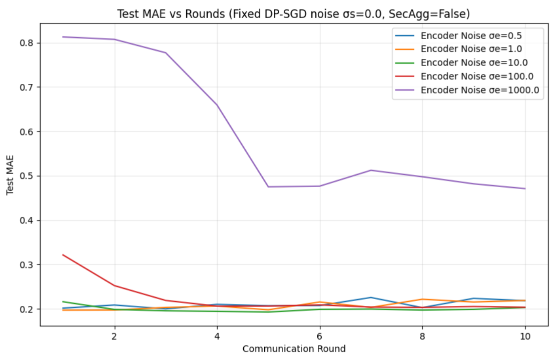


Figure 28: MAE(↓) for Air Quality dataset with $\sigma_{grad} = 0$

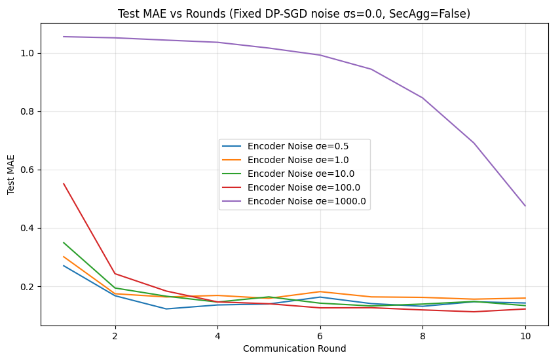
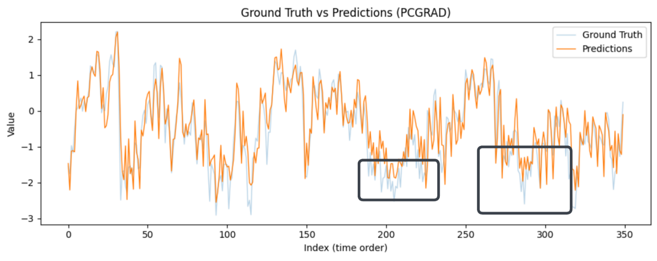
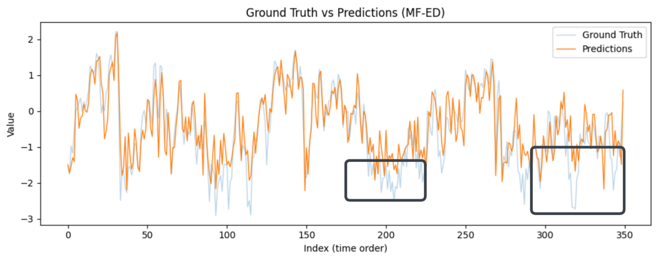


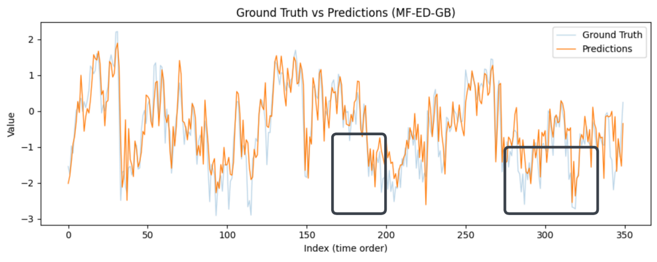
Figure 31: MAE(↓) for Industry dataset with $\sigma_{grad} = 0$



(a) CrossFreqNet-GradBal



(b) CrossFreqNet



(c) CrossFreqNet-PCGrad

Figure 32: Ground Truth vs Predictions comparison across different methods

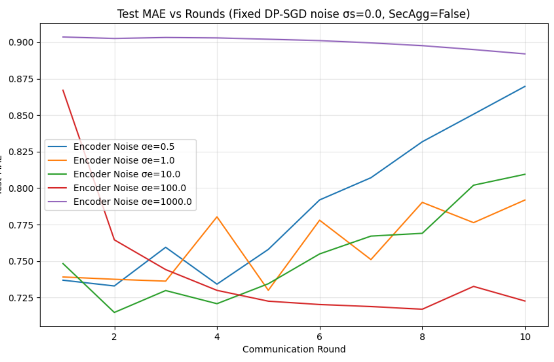


Figure 29: MAE(↓) for Wind dataset with $\sigma_{grad} = 0$

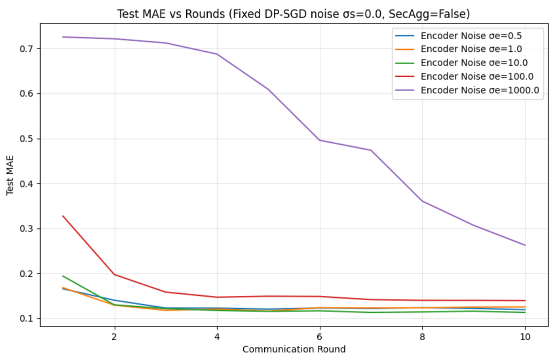


Figure 30: MAE(↓) for Load dataset with $\sigma_{grad} = 0$

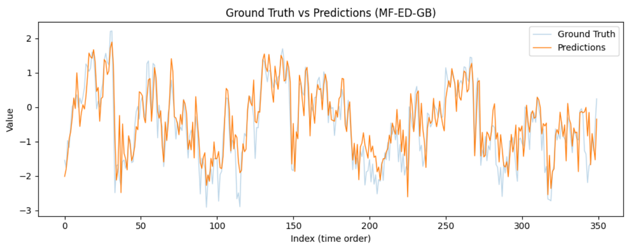


Figure 33: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with our centralized model with gradient balancing

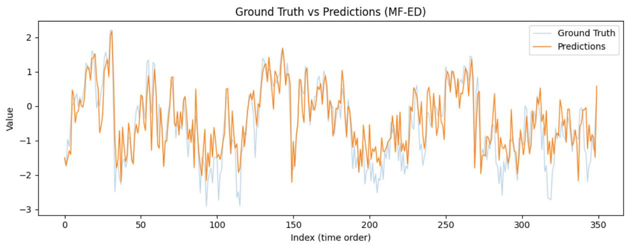


Figure 34: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run without and with multi-task optimization

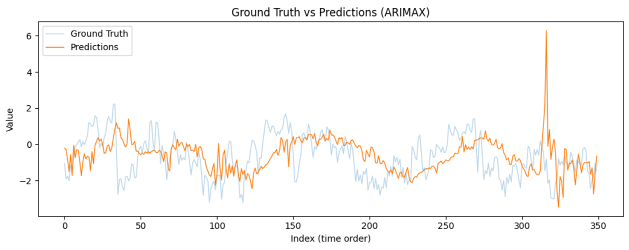


Figure 35: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with supervised ARIMAX

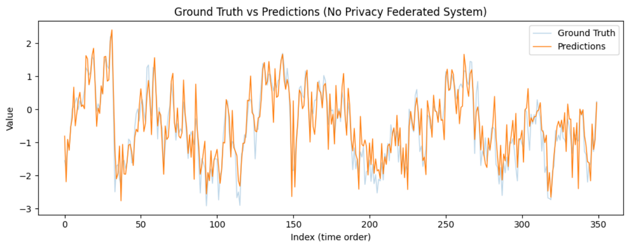


Figure 36: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with no confidentiality-preserving federated learning

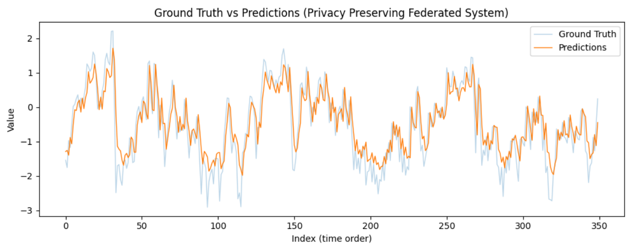


Figure 37: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with confidentiality-preserving federated learning

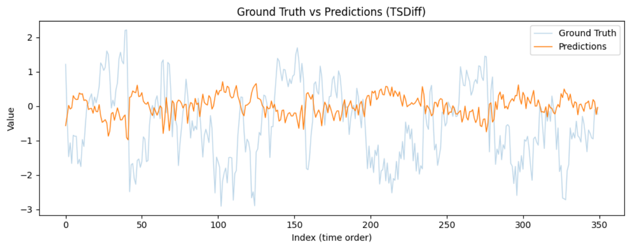


Figure 38: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with diffusion TSDiff[30]

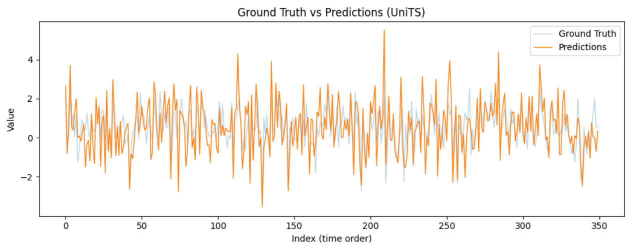


Figure 39: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with UniTS[23] foundational model

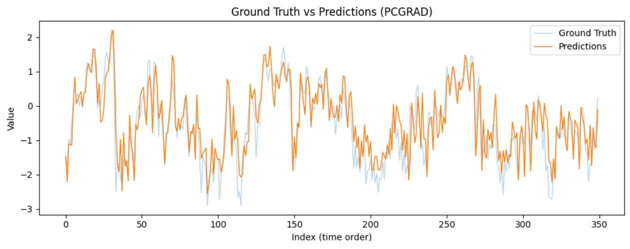


Figure 40: One step ahead (Horizon=1) forecasting of time series from Spain Load Forecasting dataset run with our centralized model with PCGrad[66]

3 Background

3.1. Time Series Forecasting

3.1.1. Definition

Time series forecasting refers to methods that predict future values based on historical data sequences collected over regular or irregular time intervals [17]. Effective forecasting provides a critical advantage across various fields including finance, supply chain management, healthcare, energy systems, and industrial maintenance, as it enables informed decision-making and strategic planning by accurately anticipating future trends, events, or demands.

Formally, a time series is defined as a chronological sequence of observations:

$$\mathbf{y} = \{y_1, y_2, \dots, y_T\}, \tag{3.1}$$

where $y_t \in \mathbb{R}$ represents the observed value at discrete or continuous time step t. Forecasting seeks to estimate future values beyond the known sequence:

$$\hat{y}_{T+1}, \hat{y}_{T+2}, \dots, \hat{y}_{T+H}, \tag{3.2}$$

where H is the forecasting horizon, reflecting how far into the future the model aims to predict.

Time series can exhibit various characteristics such as trends, seasonality, cyclicality, and irregular fluctuations (noise). Identifying and modeling these characteristics are critical for achieving accurate forecasts.

3.1.2. Types of Forecasting Problems

Time series forecasting problems are generally classified according to the dimensionality of inputs and outputs:

Univariate Forecasting: In univariate forecasting, predictions depend exclusively on the historical observations of a single target variable. It is represented mathematically as:

$$y_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-p}), \tag{3.3}$$

where p denotes the lookback window or the number of past observations influencing the prediction. Common applications include stock price prediction, electricity consumption forecasts, and weather forecasting using historical temperature data. Popular models for univariate forecasting include traditional statistical approaches such as AutoRegressive Integrated Moving Average (ARIMA), and modern deep learning models like Long Short-Term Memory (LSTM) neural networks.

Multivariate Forecasting: Multivariate forecasting leverages not only the past observations of the primary variable but also incorporates other related variables known as exogenous predictors or features. It is formally defined as:

$$y_{t+1} = f(y_t, x_t^1, x_t^2, \ldots, x_t^d, y_{t-1}, x_{t-1}^1, \ldots, x_{t-1}^d), \tag{3.4}$$

where $x_t^i \in \mathbb{R}$ denotes the value of the i^{th} exogenous variable at time t, and d is the number of external variables considered. Incorporating multiple variables often leads to increased accuracy as it captures interdependencies among different time series. Practical examples of multivariate forecasting include predicting sales based on historical sales and marketing expenditures, forecasting electricity demand based on past load and weather conditions, and predicting industrial system failures using multiple sensor measurements.

3.1.3. Challenges in Time Series Forecasting

Time series forecasting faces several inherent challenges due to the characteristics of the data. Common challenges include:

1. **Non-stationarity**: Real-world time series data frequently exhibit changing mean, variance, or other statistical properties over time. This non-stationarity complicates modeling as most methods assume a stationary data distribution[9, 19].
2. **Missing or irregular data**: Many practical scenarios feature gaps, irregular intervals, or asynchronous multi-frequency sampling rates, creating challenges for standard forecasting methods[6].
3. **Long-term dependencies**: Capturing complex, long-term temporal relationships is challenging, particularly for traditional models, motivating the use of deep learning architectures capable of modeling these dependencies effectively[12, 21].

Classical statistical forecasting methods, like AutoRegressive Integrated Moving Average (ARIMA), typically assume the time series is stationary. A stationary series has a constant mean and variance over time. Non-stationary series often require transformations, such as differencing or log transformations, to achieve stationarity before modeling.

In contrast, modern deep learning methods, including Long Short-Term Memory (LSTM) networks and Transformer architectures, can model complex temporal dependencies and patterns directly from raw data. These methods often reduce the need for extensive manual feature selection and engineering.

Despite differences in methodology, all forecasting approaches share a common goal: leveraging past information and any relevant external factors to generate accurate and reliable predictions about future values.

3.1.4. Challenges in Time Series Forecasting

Time series data often exhibit challenges such as:

1. **Trend**: A long-term increase or decrease in values.
2. **Seasonality**: Regular patterns repeating at fixed intervals.
3. **Noise**: Random fluctuations that obscure underlying patterns.

Classical statistical methods, such as ARIMA, often require the series to be stationary (constant mean and variance), which may necessitate transformations like differencing. Modern deep learning approaches, such as LSTMs and Transformers, can capture complex temporal dependencies without requiring explicit feature engineering.

Regardless of the approach, the fundamental objective remains the same: using historical patterns and related factors to generate accurate future predictions.

3.1.5. Linear Models

Linear statistical models are widely adopted in time series forecasting due to their simplicity, interpretability, and computational efficiency. Among these approaches, AutoRegressive Integrated Moving Average (ARIMA) models are especially popular and have been extensively used for decades [4].

ARIMA models predict future observations by combining historical values and past errors, explicitly addressing key challenges in time series data, such as trends, seasonal effects, and random fluctuations. Furthermore, ARIMA models assume stationarity, meaning that data properties like mean and variance are stable over time. For cases where external influences impact the series, ARIMA with eXogenous inputs (ARIMAX) enhances forecasting capability by incorporating additional related variables into the prediction, thereby capturing external effects that ARIMA alone may not adequately describe.

ARIMA and ARIMAX.

The AutoRegressive Integrated Moving Average (ARIMA) model is a widely used statistical approach for time series forecasting. It combines three components: an autoregressive (AR) term, an integrated (I) term for differencing, and a moving average (MA) term. The ARIMA(p, d, q) model is defined by:

- p: Number of autoregressive lags (past values used as predictors).
- d: Number of times the series is differenced to remove trends and achieve stationarity.
- q: Number of moving average terms, which model past forecast errors.

Autoregressive (AR) Component: The AR part models the relationship between the current observation and its past values:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t, \tag{3.5}$$

where ϕ_i are the autoregressive coefficients, and ϵ_t is a white noise error term.

Integrated (I) Component: To handle non-stationarity, the series is differenced d times:

$$y'_t = y_t - y_{t-1}, \quad \text{(for first-order differencing, } d = 1) \tag{3.6}$$

Higher-order differencing applies the operation multiple times.

Moving Average (MA) Component: The MA component captures the effect of past forecast errors:

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}, \tag{3.7}$$

where θ_i are the moving average coefficients.

General ARIMA(p, d, q) Model: The complete ARIMA model is expressed as:

$$\Phi_p(B)(1 - B)^d y_t = \Theta_q(B)\epsilon_t, \tag{3.8}$$

where:

- B is the backshift operator ($By_t = y_{t-1}$),
- $\Phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is the AR polynomial,
- $\Theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is the MA polynomial.

ARIMAX: Incorporating Exogenous Variables.

While ARIMA relies solely on past values of the target variable, ARIMAX(AutoRegressive Integrated Moving Average with Exogenous Variables) extends ARIMA by incorporating external (exogenous) variables that can help improve forecasting accuracy. These exogenous variables are denoted as $\mathbf{x}_t = \{x_t^1, x_t^2, \ldots, x_t^k\}$, where k is the number of external predictors.

The ARIMAX model introduces an additional regression component:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{m=1}^{k} \beta_m x_t^m + \epsilon_t. \tag{3.9}$$

Here:

- β_m represents the influence of the exogenous variable x_t^m on the target variable y_t.
- The terms $\phi_i y_{t-i}$ and $\theta_j \epsilon_{t-j}$ represent the autoregressive and moving average components, respectively.
- ϵ_t is the white noise error term.

ARIMAX is useful when external factors have a significant impact on the target variable. Examples include Economic indicators (e.g., interest rates influencing stock prices) Weather conditions (e.g., temperature affecting energy consumption) Marketing campaigns (e.g., advertisements affecting sales). Hence ARIMA is purely time series-based and models dependencies only on past values of the target variable and ARIMAX incorporates external variables, which can improve forecast accuracy when relevant external information is available.

In practice, ARIMAX can provide better predictive performance than ARIMA when meaningful exogenous variables are included. However, selecting relevant external factors and ensuring their stationarity is crucial for effective modeling.

3.1.6. Deep Models

In recent years, deep learning has emerged as a powerful alternative to classical linear techniques for time–series forecasting. Neural networks excel at discovering non-linear, high-order relationships that are difficult to specify in traditional statistical models. By stacking multiple layers of learnable transformations, deep models can automatically extract relevant temporal features, handle large sets of correlated variables, and adapt to complex patterns such as abrupt regime changes or multi-scale seasonality. This capacity makes them well-suited for real-world data that often violate the stationarity and linearity assumptions of approaches like ARIMA. In the following subsections, we review two prominent families of deep architectures applied to sequential data: recurrent networks (including LSTM) and the Transformer.

Recurrent Neural Networks (RNNs).

Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed to handle sequential data. Unlike traditional feedforward networks, RNNs maintain a hidden state that carries information from previous time steps, allowing them to model temporal dependencies.

At each time step t, an RNN takes the current input x_t and the previous hidden state h_{t-1} to compute a new hidden state h_t:

$$h_t = F(Wx_t + Uh_{t-1} + b), \tag{3.10}$$

where:

- W and U are learnable weight matrices,
- b is a bias term,
- $F(\cdot)$ is a nonlinear activation function (typically a hyperbolic tangent or ReLU).

The key advantage of RNNs is their ability to process sequences of arbitrary length while maintaining context from past observations. However, in practice, standard RNNs struggle with learning long-term dependencies due to the vanishing and exploding gradient problem. As new inputs arrive, old information is gradually overwritten, making it difficult for the network to retain meaningful patterns from earlier time steps. This limits the ability of RNNs to capture long-range dependencies.

Long Short-Term Memory (LSTM).

Long Short-Term Memory (LSTM) networks are a specialized form of RNNs designed to mitigate the issue of vanishing gradients by introducing an internal cell state (C_t) and a set of gating mechanisms that regulate information flow.

The LSTM architecture consists of three key gates:

1. **Forget Gate:** Decides how much past information to retain in the cell state.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.11}$$

2. **Input Gate:** Determines how much of the new input to write to the cell state.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.12}$$

3. **Cell State Update:** Updates the memory cell with new candidate information.

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.13}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{3.14}$$

4. **Output Gate:** Controls how much of the cell state should be exposed as the new hidden state.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.15}$$

$$h_t = o_t \tanh(C_t) \tag{3.16}$$

Here, $\sigma(\cdot)$ represents the sigmoid activation function, ensuring that the gates produce values between 0 and 1, allowing selective memory retention.

LSTMs improve upon traditional RNNs by maintaining memory over long sequences, making them highly suitable for problems where past information is crucial for predicting future outcomes.

Transformers.

The Transformer is a neural network architecture that has revolutionized sequence modeling by relying entirely on a mechanism called **self-attention** while omitting recurrence altogether [**<empty citation>**]. In traditional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, processing sequential data is inherently step-by-step, which can be computationally inefficient for long sequences. Transformers, introduced by Vaswani et al. (2017) [**<empty citation>**], overcome this limitation by using self-attention, which allows each position in the sequence to directly attend to all other positions.

Key Concept: Self-Attention Unlike RNNs, where past information is propagated sequentially through hidden states, the Transformer computes relationships between all elements in a sequence simultaneously. This is achieved through the self-attention mechanism, which assigns attention scores that determine how much each time step should influence another.

For a given input sequence, self-attention operates on three learned representations:

- **Query (Q):** Represents the current time step's feature vector.
- **Key (K):** Represents other time steps that may influence the query.
- **Value (V):** Contains the actual information to be aggregated.

The self-attention output is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V, \tag{3.17}$$

where:

- $Q, K, V \in \mathbb{R}^{T \times d}$ are matrices of query, key, and value vectors (where T is the sequence length and d is the embedding dimension).
- $\frac{QK^\top}{\sqrt{d}}$ computes scaled dot-product attention, where division by $\sqrt{d}$ stabilizes gradients.
- softmax$(\cdot)$ ensures that the attention weights sum to 1 across the sequence.

This mechanism allows the model to dynamically focus on relevant parts of the sequence, making it highly effective for learning long-range dependencies.

Parallelization and Efficiency One major advantage of Transformers over RNNs is their ability to process all time steps in parallel. Since self-attention does not rely on sequential computations like RNNs, Transformers significantly reduce training time and improve scalability, especially for long sequences.

Multi-Head Attention and Stacking Layers To enhance learning, Transformers employ **multi-head attention**, where multiple self-attention layers are run in parallel, each with different weight matrices:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O. \tag{3.18}$$

Each attention head learns different aspects of the relationships in the sequence, improving model capacity.

The full Transformer model consists of multiple stacked attention layers, interleaved with feedforward sub-layers and residual connections, forming a deep network capable of learning complex representations.

Application to Time Series Forecasting Although Transformers were originally developed for natural language processing, their ability to capture long-range dependencies has made them increasingly popular for time series forecasting. Since time series data is inherently ordered, **positional encodings** are added to the input embeddings to preserve temporal order:

$$PE(t, 2i) = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad PE(t, 2i+1) = \cos\left(\frac{t}{10000^{2i/d}}\right). \tag{3.19}$$

These encodings help the Transformer recognize the sequence structure, compensating for the absence of recurrence.

Transformers offer several clear benefits for time-series forecasting. Because self-attention considers every position in a sequence at once, the model can capture long-range dependencies that often cause recurrent networks to struggle. In addition, all time steps are processed in parallel, so training is faster than the sequential updates used in RNNs and LSTMs. The attention weights are also learned adaptively, allowing the model to focus on the most relevant parts of the input without hand-crafted rules.

These strengths come with important limitations. The computational cost of self-attention grows quadratically with sequence length, which can make very long time series expensive to handle. Successful training usually requires large data sets; with limited data, Transformers may overfit or fail to learn stable patterns. Finally, unlike recurrent models, Transformers do not have an inherent notion of temporal order. They rely on added positional encodings to represent time, so they may perform poorly if those encodings do not match the true structure of the data.

3.2. Multi-Task Learning

Multi-Task Learning (MTL) trains a single model to tackle several related tasks simultaneously, rather than fitting a separate model for each task [5]. By sharing parameters across tasks, the model can exploit statistical regularities that recur in different data sources. As a result, the learned representations are often more robust and general, which in turn improves predictive accuracy and reduces over-fitting [26].

The usual architectural pattern combines two types of layers:

- **Shared layers**. These layers encode features that are expected to be useful for every task. Their parameters are updated with gradient signals aggregated from all tasks, which encourages cross-task information transfer.
- **Task-specific layers**. Placed on top of the shared backbone, these layers adapt the common representation to the unique requirements of each individual task, allowing the model to capture task-dependent nuances.

The interplay between shared and task-specific components provides an inductive bias that guides learning toward solutions that explain multiple objectives at once, thereby making more efficient use of limited data and accelerating convergence during training [22].

Multi-task learning (MTL) trains a single model to handle multiple tasks by sharing parameters, improving efficiency and generalization [5]. The loss functions used in MTL frameworks are designed to balance task-specific needs while leveraging shared information. Below, we outline key loss functions from three MTL frameworks: FATHOM, MTL-Trans, and UniTS.

FATHOM, a federated MTL framework, supports classification and regression across devices, treating each as a separate task [7]. Its objective minimizes the average loss over K tasks:

$$\widetilde{\theta} = \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{K} \sum_{k=1}^{K} L\left(\hat{Y}^{(k)}, Y^{(k)}\right). \tag{3.20}$$

For classification, FATHOM uses a regularized cross-entropy loss to prevent overconfidence on sparse labels:

$$L(\hat{Y}, Y) = -(1-\alpha) \sum_{m=1}^{M} \sum_{t=1}^{N} y_{mt} \log(\hat{y}_{mt}) - \alpha \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{N} \hat{y}_{mt}, \tag{3.21}$$

where $\alpha = 0.3$ balances fitting true labels and output diversity. For regression, it uses Mean Absolute Error (MAE):

$$L(\hat{Y}, Y) = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{t=1}^{N} \left|\hat{y}_{mt} - y_{mt}\right|. \tag{3.22}$$

MTL-Trans, a Transformer-based framework for time series forecasting, uses a Mean Squared Error (MSE) loss across tasks:

$$L = \frac{1}{N} \sum_{n=1}^{N} (x_n - y_n)^2, \tag{3.23}$$

where x_n and y_n are predicted and true values in a batch. The total objective averages losses over K tasks:

$$\widetilde{\theta} = \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{K} \sum_{m=1}^{K} L^{(m)}(\theta). \tag{3.24}$$

This MSE suits continuous outputs like traffic volume, with shared and private attention layers capturing cross-task patterns.

UniTS supports both predictive and generative tasks like forecasting and classification [12]. Its objective is a weighted sum of task-specific losses:

$$L_{\text{total}} = \sum_{i=1}^{I} \lambda_i \cdot L_i(D_i). \tag{3.25}$$

For generative tasks, UniTS uses MSE:

$$L_{\text{MSE}} = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{t=1}^{N} (\hat{x}_{mt} - x_{mt})^2, \tag{3.26}$$

and for classification, it applies cross-entropy:

$$L_{\text{CE}} = -\sum_{m=1}^{M} y_m \log(\hat{y}_m). \tag{3.27}$$

UniTS also uses a pretraining loss for masked reconstruction:

$$L_{\text{pretrain}} = L_{\text{MSE}}(H_{\text{GEN}}(z_p, z_x), x) + L_{\text{MSE}}(H_{\text{GEN}}(H_{\text{CLS}}(z_{\text{Pred}}), z_x), x). \tag{3.28}$$

3.2.1. General technique for optimizing Multi-Task Learning

Above thoery can be generalized as follows where multi-task learning (MTL) trains a single model to solve several related tasks at once, sharing parameters [5]. Let $\theta \in \mathbb{R}^m$ be the shared parameter vector and let $L_i(\theta)$ denote the loss associated with task i for $i = 1, \ldots, N$. A common strategy is to form a weighted sum of the task losses,

$$L_{\text{total}}(\theta) \;=\; \sum_{i=1}^{N} \alpha_i \, L_i(\theta), \tag{3.29}$$

where each non-negative weight α_i reflects the relative importance of task i. The weights can be fixed a priori or adapted during training, for example on the basis of task uncertainty or learning speed. The optimisation problem is then

$$\widetilde{\theta} \;=\; \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{N} \sum_{i=1}^{N} L_i(\theta), \tag{3.30}$$

which reduces to (3.29) when the weights are uniform ($\alpha_i = 1/N$).

Benefits of Multi-Task Learning Compared with single-task learning (training an independent model for each task), multi-task learning provides three principal advantages. First, shared parameters act as an implicit regularizer, improving generalization and reducing over-fitting to any single dataset. Second, parameter sharing makes more efficient use of data: tasks with limited observations can exploit representations learned from data-rich tasks. Third, common structures are learned only once, so optimisation often converges faster than training N separate models [5].

3.3. Multi-task optimization: Conflicting Gradients.

Multi-task learning (MTL) is when a single model learns to perform multiple tasks at the same time. A common challenge in multi-task learning is that tasks can interfere with each other. This happens because the gradient updates from different tasks might point in opposite directions. Such conflicting gradients slow down training and worsen performance. This happens because in multi-task learning the parameter update is typically driven by the *mean* gradient $g_0 = K^{-1} \sum_{i=1}^{K} g_i$, where $g_i = \nabla L_i(\theta)$. When two tasks disagree, i.e. $\langle g_i, g_0 \rangle < 0$, the step can *harm* task i and cause *negative transfer*. A toy example of such a scenario is shown in Figure 3.1.

Recent approaches like GradNorm [8], PCGrad [25], and CAGrad [18] address this problem in different ways.

GradNorm tries to balance how quickly each task learns. Some tasks learn faster because their gradients or losses are larger, causing the model to prioritize those tasks. GradNorm fixes this by adjusting the loss weights dynamically, making sure tasks learn at similar speeds.

Mathematically, if we have a total loss from all tasks:

$$L = \sum_{i} w_i(t) \cdot L_i(t)$$

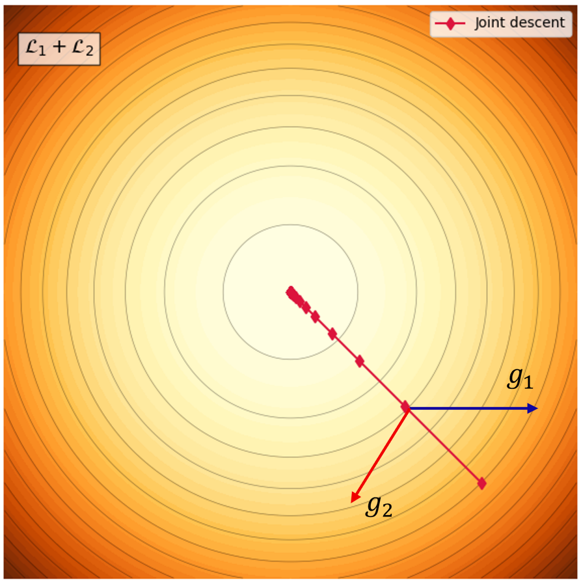


Figure 3.1: Illustration of conflicting gradients in multi-task learning. The mean gradient g_0 points in a direction that harms task i due to $\langle g_i, g_0 \rangle < 0$.

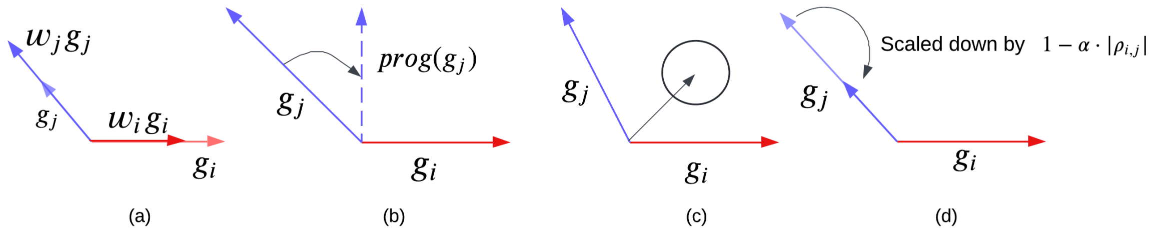


Figure 3.2: Gradient–conflict strategies. **(a) GradNorm:** rescales each task's gradient to equalise their lengths $(w_i g_i,\ w_j g_j)$ while leaving directions unchanged. **(b) PCGrad:** removes head-on conflict by projecting g_j onto the sub-space orthogonal to g_i, deleting the opposing component. **(c) CAGrad:** selects a single update (black arrow) via a convex optimisation that blends g_i and g_j just enough to make non-negative progress for both tasks. **(d) GradBal:** preserves directions but softly down-scales a conflicting gradient by the factor $1 - \alpha\,|\rho_{i,j}|$; aligned gradients are left untouched.

GradNorm updates each task's weight $w_i(t)$ by minimizing:

$$\mathcal{L}_{grad}(t) = \sum_i \left| \|\nabla_W \left[w_i(t)L_i(t)\right]\|_2 - G_W(t) \cdot r_i(t)^\alpha \right|$$

Here, $G_W(t)$ is the average gradient norm, and $r_i(t)$ measures how slowly or quickly task i is training compared to the average rate.

PCGrad directly modifies conflicting gradients. If two tasks have gradients pointing in opposite directions, PCGrad adjusts one gradient by removing the conflicting component:

$$g_i \leftarrow g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} g_j$$

CAGrad addresses the gradient conflict by finding an optimal weighted combination of gradients. It solves an optimization problem to select the shortest gradient step that still improves all tasks:

$$g_0^* = \arg \min_{g_0 \in \text{conv}(g_1, \ldots, g_n)} \|g_0\|^2 - c \sum_{i=1}^{n} g_i^\top g_0$$

All the approaches are summarized in Figure 3.2.

3.4. Federated Learning (FL)

Federated Learning (FL) is a machine learning paradigm that enables multiple decentralized clients to collaboratively train a shared global model while keeping their local data private [24]. Rather than transmitting raw data, clients send model parameters or gradients to a central server, which aggregates them to update the global model. This approach is particularly suited for privacy-sensitive applications.

At each communication round t, client k receives the global model parameters θ_t from the server, performs local updates on its private data, and sends back the model update $\Delta\theta_t^{(k)}$ for aggregation.

Federated Averaging (FedAvg)

Federated Averaging (FedAvg) [20] is the foundational aggregation method in FL. It updates the global model as a weighted average of local client models:

$$\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \cdot \theta_t^{(k)}, \tag{3.31}$$

where:

- K is the number of participating clients.
- n_k is the number of local data points at client k.
- $n = \sum_{k=1}^{K} n_k$ is the total number of data points across all clients.
- $\theta_t^{(k)}$ is the locally updated model from client k.

FedAvg is effective when data distributions across clients are similar. However, it can face convergence issues with non-identically distributed (non-IID) client data.

Multi-Task Optimization in FL

To address optimisation challenges with heterogeneous clients, advanced methods extend FedAvg:

- **FedProx** [15] introduces a proximal term in the local objective, defined as:

$$L_k(\theta) = f_k(\theta) + \frac{\mu}{2}\|\theta - \theta_t\|^2, \tag{3.32}$$

where $f_k(\theta)$ is the loss on client k's data, and μ controls the strength of the proximal term that tethers local models to the global model.

- **SCAFFOLD** [14] reduces client drift using control variates. Each client maintains a control variable c_k and updates its model as:

$$\theta_k^{(t+1)} = \theta_k^{(t)} - \eta\left(\nabla f_k(\theta_k^{(t)}) + c - c_k\right), \tag{3.33}$$

where c is the server control variate and η is the learning rate. This correction aligns local gradients with the global objective, improving convergence on heterogeneous data.

3.4.1. Knowledge Distillation (KD)

Knowledge Distillation (KD) [13] is a model compression and transfer learning technique where a smaller model, referred to as the student, is trained to replicate the behavior of a larger, high-capacity teacher model. Rather than using hard labels, KD relies on the soft probability distributions (soft targets) generated by the teacher. These soft targets contain richer information about inter-class relationships, improving the student's ability to generalize.

Given a temperature parameter T, the softened probability distribution produced by the teacher for class i is calculated as:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{3.34}$$

where z_i is the teacher's logit for class i. The student model minimizes the Kullback–Leibler (KL) divergence between its output distribution p_s and the teacher's softened distribution p_t:

$$L_{\text{KD}} = \text{KL}(p_t, p_s). \tag{3.35}$$

Beyond response-based KD using logits, feature-based KD extends this concept by matching intermediate layer activations between teacher and student. The corresponding loss is defined as:

$$L_{\text{Fea}} = \|\Phi_t(f_t) - \Phi_s(f_s)\|_2^2, \tag{3.36}$$

where f_t and f_s represent feature maps from the teacher and student, respectively, and Φ denotes any necessary transformation to align dimensions.

Additionally, relation-based KD focuses on preserving the structural relationships between data samples, using correlation functions such as:

$$L_{\text{Rel}} = \|\Psi_t(f_t) - \Psi_s(f_s)\|_2^2, \tag{3.37}$$

where Ψ encodes pairwise similarities or other relational information from the feature maps.

Together, these approaches enable KD to serve as a flexible framework for model compression and collaborative learning, including in multi-task and federated learning settings[16].

3.5. Confidentiality

In machine learning, ensuring the confidentiality of sensitive data is important, particularly when dealing with personal or proprietary information. Traditional centralized machine learning approaches require aggregating data from various sources, which poses significant privacy risks. As we saw in previous section Federated learning (FL) addresses these concerns by enabling model training across decentralized devices or servers without exchanging raw data [20] but when updates themselves may leak information, FL is commonly combined with *differential privacy* (DP).

Differential privacy[10] provides statistical guarantee that no single task's or client's data can be reversed engineered from the model by incorporating calibrated noise to the entity being protected. Formally, a learning algorithm is (ϵ, δ)-DP protected if its outputs are nearly indistinguishable when run on two two neighboring datasets D and D'(differing by record) i.e. for any subset of outputs S, the following holds:

$$\Pr[M(D) \in S] \leq e^{\epsilon} \Pr[M(D') \in S] + \delta \tag{3.38}$$

, where ϵ controls the privacy loss, and δ represents the probability of failure.

In deep learning, differential privacy can also be interpreted as comparing the shapes of two probability distributions [2]. Formally, this is expressed as:

$$\log \frac{P(\mathcal{A}(\mathcal{D}) = \mathcal{O})}{P(\mathcal{A}(\mathcal{D}_{-r}) = \mathcal{O})} < \epsilon \quad \text{with probability } 1 - \delta, \tag{3.39}$$

In literature smaller ϵ is preferred as it provides better guarantee and δ < $1\|N\|$ where N is the size of the dataset being protected[1].

In deep learning, differential privacy is most often enforced through DP-SGD[1]. It clips per-sample gradients and injects Gaussian noise, yielding an (ε, δ) guarantee that any single record has little influence on the released update but degrades model utility.

Hence, for each sample i with gradient g_i, we apply per-example gradient clipping:

$$\tilde{g}_i = \frac{g_i}{\max\left(1, \frac{\|g_i\|_2}{B}\right)} \tag{3.40}$$

Here:

- g_i is the raw gradient for sample i.
- $\|g_i\|_2$ is the ℓ_2-norm (Euclidean norm) of g_i.
- B is the clipping threshold. If $\|g_i\|_2 > B$, the gradient is scaled down to have norm B; otherwise, it remains unchanged.
- $\tilde{g}_i$ is the clipped gradient.

After clipping, Gaussian noise is added to the mean of the clipped gradients:

$$\tilde{g} = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{g}_i + \mathcal{N}(0, \sigma_{\text{grad}}^2 B^2 I)\right) \tag{3.41}$$

Where:

- N is the number of samples in the batch.
- $\mathcal{N}(0, \sigma_{\text{grad}}^2 B^2 I)$ denotes Gaussian noise with zero mean and covariance matrix $\sigma_{\text{grad}}^2 B^2 I$.
- σ_{grad} is the noise multiplier that controls the trade-off between privacy and utility.
- I is the identity matrix of appropriate dimension.

In deep learning settings, however, we typically apply differentially private mechanisms repeatedly (e.g., across many training iterations. This raises the question of how privacy losses accumulate. Fortunately, there are several composition theorems that bound the overall privacy leakage when composing multiple DP operations called composition theorms[10], as well as related work like amplification and post-processing[10] that help manage the required noise for a desired privacy guarantee. Lets look at them one by one:

Composition Theorems.

When multiple differentially private mechanisms are applied, composition theorems determine the overall privacy guarantee [10]. Key theorems include:

- **Basic Composition**: For k mechanisms, each providing (ϵ_i, δ_i)-DP, the composition provides $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.
- **Advanced Composition**: Offers tighter bounds, such as (ϵ', δ')-DP where $\epsilon' = \sqrt{2k \ln(1/\delta'')}\epsilon + k\epsilon(e^{\epsilon} - 1)$, for some δ'', and $\delta' = k\delta + \delta''$.
- **Moment Accountant**: Tracks privacy loss using the moments of the privacy loss random variable, providing tighter bounds for iterative algorithms like DP-SGD [1]. It is particularly useful in federated learning for analyzing privacy over multiple training rounds.

These theorems are relevant where multiple DP mechanisms, such as dual DP, are applied to ensure privacy across local training and aggregation phases.

Privacy Amplification by Subsampling.

Another important result is privacy amplification, which shows that running a DP mechanism on a random subset of data(batches during training) can yield stronger privacy guarantees than running it on the full dataset. Intuitively, if each individual's data has a chance not to be included in a given operation, that uncertainty provides additional privacy protection. The most common scenario is Poisson subsampling (or sampling without replacement) of records or clients. In other words wehn we apply a differentially private (DP) mechanism not on the full dataset, but on a randomly selected subsample of the dataset, the privacy guarantee (ε, δ) improves compared to applying the same mechanism on the full dataset[11].

$$\varepsilon' = \log\left(1 + p \cdot (e^{\varepsilon} - 1)\right), \quad \delta' = p \cdot \delta \tag{3.42}$$

Where:

- ε' and δ' are the amplified (improved) privacy parameters,
- p is the probability that any single data point is included in the subsample,
- ε and δ are the original privacy parameters of the mechanism $\mathcal{M}$.

We can also look at it as when $p \ll 1$, we observe that $\varepsilon' \approx p \cdot \varepsilon$ to first order. This means subsampling effectively scales down the privacy cost proportional to the sampling rate, providing stronger privacy guarantees without modifying the underlying mechanism.

This property is widely used in private machine learning, such as in DP-SGD, where random minibatch sampling naturally results in privacy amplification.

Utility Implications of Noise in Federated Learning

In federated learning (FL), ensuring strong privacy (small ε) requires adding noise to each client's updates, as done in DP-SGD [11]. However, too much noise can harm model accuracy, slow convergence, or even cause training to fail. This creates a known privacy-utility trade-off: lowering ε improves privacy but reduces model performance.

The problem is more noticeable in FL because each client may have limited data, so noisy updates carry less useful information. To balance privacy and utility, practitioners tune learning rates, increase the number of training rounds, or involve more clients.

Privacy amplification by subsampling helps reduce the noise needed since only a batches of data is processed during training and we get stronger privacy.

Secured Aggregation for federated setting

Secure Aggregation (SecAgg) is a cryptographic protocol that protects the confidentiality of individual model updates in federated learning (FL) by ensuring that only the aggregate of all updates is revealed to the server. This mechanism addresses a critical privacy concern not covered by Differential Privacy (DP): exposure of raw client updates during the training process.

In a standard FL round, each client $c \in C$ computes a local update Δ_c based on its private data. Without protection, transmitting these updates would risk leaking sensitive information. Secure Aggregation mitigates this risk using a combination of pairwise masking and threshold secret sharing [3, 23].

Pairwise Masking Mechanism. At the beginning of each training round, clients establish pairwise shared secrets through cryptographic key exchange protocols such as Diffie-Hellman (DH). Specifically, each client c generates a DH key pair and exchanges public keys with every other client c'. From the shared secrets $s_{c,c'}$ derived from DH key agreement, clients generate perturbation masks using cryptographically secure pseudorandom generators (PRGs).

The pairwise masking protocol operates as follows:

- For each client pair (c, c'), if $c < c'$, client c adds the mask PRG$(s_{c,c'})$ to its update, while client c' subtracts the same mask.
- Additionally, each client samples an independent self-mask m_c and distributes secret shares of it using a (t, n) threshold secret-sharing scheme.

The masked update transmitted by client c is given by:

$$\tilde{\Delta}_c = \Delta_c + m_c + \sum_{c>c'} \text{PRG}(s_{c,c'}) - \sum_{c<c'} \text{PRG}(s_{c,c'}). \tag{3.43}$$

When the server aggregates all masked updates, all pairwise masks cancel out due to their structured addition and subtraction, and the secret-shared self-masks m_c are removed via reconstruction from shares. The final aggregate is:

$$S = \sum_{c \in C} \tilde{\Delta}_c = \sum_{c \in C} \Delta_c. \tag{3.44}$$

Trust Threshold and Hybrid Privacy Guarantee. Truex et al. [23] extend the Secure Aggregation framework by formally analyzing the trust threshold concept. If at least t out of N total clients are honest and non-colluding, Secure Aggregation ensures confidentiality of individual updates. This threshold assumption enables combining Secure Aggregation with Differential Privacy (DP) while reducing the required noise level.

Specifically, if each client adds Gaussian noise with variance σ^2 to its update to achieve local (ϵ, δ)-DP, the total noise variance across all clients would naively scale as $N\sigma^2$. However, under a trust threshold t, the effective noise variance reduces to:

$$\sigma^2_{\text{eff}} = \frac{\sigma^2}{t-1}, \tag{3.45}$$

which results in a noise magnitude reduction by a factor of $\sqrt{t-1}$.

This hybrid design ensures that even if up to $N - t$ clients collude with the server, the privacy guarantee holds as long as t honest clients remain. The combination of Secure Aggregation and DP thus enables federated learning systems to scale while maintaining strong privacy guarantees and minimizing the negative impact on model utility.

Cumulative Privacy Considerations. When federated learning operates over multiple rounds, privacy loss accumulates according to composition theorems [10]. Running a (ϵ, δ)-DP mechanism for T rounds leads to an overall $(T\epsilon, T\delta)$ privacy budget. By reducing the per-round noise through Secure Aggregation, the framework helps maintain an acceptable trade-off between privacy and model performance across many training iterations.

System Implications. Protocols based on Bonawitz et al. [3] and Truex et al. [23] form the basis for real-world FL deployments that require both communication security and statistical privacy guarantees. While Secure Aggregation protects against exposure of raw updates, it relies on trust threshold assumptions and is most effective when combined with formal DP mechanisms. This layered approach is critical for building scalable, privacy-preserving FL systems used in sensitive application domains such as healthcare and finance.

4 Conclusion and Future Works

4.1. Conclusion

In this work, we present *CrossFreqNet*, a novel multi-frequency multi-task framework designed to address the challenges of forecasting in distributed industrial systems where data streams exhibit varying sampling rates and task heterogeneity. By extending encoder-decoder architectures to incorporate shared knowledge across tasks while preserving task-specific patterns through dedicated output layers, our model effectively integrates high- and low-frequency signals without resorting to up- or down-sampling, which can introduce artifacts or information loss. This approach builds upon prior efforts in mixed-frequency forecasting, such as the multi-task encoder-dual-decoder framework, but advances the field by enabling collaborative learning in multi-task environments, where traditional single-task models like mTAN or Scaleformer fall short.

A central innovation of our framework is *GradBal*, a gradient-balancing mechanism that mitigates inter-task conflicts by scaling gradient contributions based on pairwise cosine similarities and loss magnitudes, without altering their directions. This strategy recognizes that mild gradient conflicts in temporal data can serve as informative signals rather than mere noise, providing implicit regularization that enhances model stability and generalization. Unlike aggressive conflict elimination methods such as PCGrad, which project gradients to remove oppositions entirely and may discard valuable temporal cues, or CAGrad, which enforces simultaneous descent across tasks via constrained optimization, GradBal's softer attenuation preserves lead-lag relationships inherent in time series. Empirical results demonstrate that this leads to superior performance, with CrossFreqNet achieving up to 72% reduction in Mean Absolute Error (MAE) compared to the strongest multi-task baseline, UniTS, and up to 48% over PCGrad across four public benchmarks (Air Quality, Wind Forecast, Load Forecast, and Spain Electricity Shortfall) and one proprietary industrial dataset. These gains align with findings in multi-task learning literature, where balanced gradient handling has been shown to improve convergence in heterogeneous settings.

Furthermore, we extend our evaluation to privacy-sensitive scenarios through federated learning (FL), incorporating differential privacy (DP) and secure aggregation to safeguard data confidentiality. In FL setups, raw data remains decentralized, and only model updates are aggregated, addressing real-world constraints in domains like healthcare and energy systems. Despite the noise injection and masking required for (ϵ, δ)-DP guarantees, our federated variant of CrossFreqNet retains 50%–90% of centralized accuracy, underscoring the robustness of our architecture. Among tested optimizers—FedAvg, FedProx, and SCAFFOLD—FedProx emerges as particularly effective in handling non-i.i.d. task distributions, as it introduces a proximal term to counteract client drift, consistent with prior studies on heterogeneous FL for time series. This balance between privacy and utility highlights the potential of FL in scaling multi-frequency forecasting to distributed environments, where centralized data pooling is infeasible due to regulatory or proprietary concerns.

Our contributions not only advance the state-of-the-art in multi-frequency multi-task forecasting but also provide practical insights for deploying such models in industrial contexts, where data heterogeneity and privacy are paramount. By outperforming baselines in both centralized and federated regimes, CrossFreqNet demonstrates the value of integrating frequency-aware encoding with conflict-aware optimization, paving the way for more accurate and efficient time series models.

4.2. Future Works.

Even though this work takes an important step towards a better Time Series Modeling, there are still a lot of avenues through which it can be improved and extended. Some of them are as follows: (1) Developing advanced conflict-aware gradient methods tailored for temporal data exhibiting phase shifts and lead-lag relationships, potentially incorporating adaptive similarity thresholds or dynamic weighting schemes inspired by recent works on proactive gradient mitigation and momentum-calibrated approaches; (2) Exploring federated learning paradigms specifically optimized for time series forecasting, such as hybrid models combining FedAvg, FedProx, and SCAFFOLD with domain-specific adaptations for mixed-frequency data, building on emerging frameworks like Time-FFM and prompt-based FL to better handle seasonality and non-stationarity; and (3) Designing federated frameworks that enable secure gradient-level sharing while preserving privacy, thereby allowing sophisticated methods like GradBal to function in fully decentralized settings, possibly through homomorphic encryption or advanced secure multi-party computation techniques. These avenues promise to further bridge the gap between theoretical advancements and real-world applicability in distributed time series analysis.

References

[1] Martin Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[2] Adrien Banse, Jan Kreischer, et al. "Federated learning with differential privacy". In: *arXiv preprint arXiv:2402.02230* (2024).

[3] Keith Bonawitz et al. "Practical secure aggregation for privacy-preserving machine learning". In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[4] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[5] R. Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997), pp. 41–75. DOI: `10.1023/a:1007379606734`.

[6] Zhengping Che et al. "Recurrent neural networks for multivariate time series with missing values". In: *Scientific reports* 8.1 (2018), p. 6085.

[7] Yujing Chen et al. "Federated Multi-task Learning with Hierarchical Attention for Sensor Data Analytics". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: `10.1109/IJCNN48605.2020.9207508`.

[8] Zhao Chen et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 794–803.

[9] Fatoumata Dama and Christine Sinoquet. "Time series analysis and modeling to forecast: A survey". In: *arXiv preprint arXiv:2104.00164* (2021).

[10] Cynthia Dwork. "Differential privacy". In: *International colloquium on automata, languages, and programming*. Springer. 2006, pp. 1–12.

[11] Juanru Fang and Ke Yi. "Privacy Amplification by Sampling under User-level Differential Privacy". In: *Proceedings of the ACM on Management of Data* 2.1 (2024), pp. 1–26.

[12] S. Gao et al. "UniTS: A Unified Multi-Task Time Series Model". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 37. 2024, pp. 140589–140631.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[14] Sai Praneeth Karimireddy et al. "Scaffold: Stochastic controlled averaging for federated learning". In: *International conference on machine learning*. PMLR. 2020, pp. 5132–5143.

[15] T. Li et al. "Federated Optimization in Heterogeneous Networks". In: *Proceedings of Machine Learning and Systems (MLSys)*. Vol. 2. 2020, pp. 429–450.

[16] W. H. Li and Hakan Bilen. "Knowledge Distillation for Multi-task Learning". In: *European Conference on Computer Vision (ECCV)*. Cham: Springer International Publishing, 2020, pp. 163–176.

[17] B. Lim and S. Zohren. "Time-Series Forecasting with Deep Learning: A Survey". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2194 (2021), p. 20200209. DOI: `10.1098/rsta.2020.0209`.

[18] Bo Liu et al. "Conflict-averse gradient descent for multi-task learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18878–18890.

[19] Yong Liu et al. "Non-stationary transformers: Exploring the stationarity in time series forecasting". In: *Advances in neural information processing systems* 35 (2022), pp. 9881–9893.

[20] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics.* PMLR. 2017, pp. 1273–1282.

[21] Yuqi Nie et al. “A time series is worth 64 words: Long-term forecasting with transformers”. In: *arXiv preprint arXiv:2211.14730* (2022).

[22] S. Ruder. “An Overview of Multi-Task Learning in Deep Neural Networks”. In: *arXiv preprint* (2017). arXiv: 1706.05098 [cs.LG].

[23] Stacey Truex et al. “A hybrid approach to privacy-preserving federated learning”. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, pp. 1–11.

[24] Kang Wei et al. “Federated learning with differential privacy: Algorithms and performance analysis”. In: *IEEE transactions on information forensics and security* 15 (2020), pp. 3454–3469.

[25] Tianhe Yu et al. “Gradient surgery for multi-task learning”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5824–5836.

[26] Y. Zhang and Q. Yang. “A Survey on Multi-Task Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609. DOI: 10.1109/TKDE.2021.3070203.