

DAFM-Supplementary Material - Section 5.8

Computational Efficiency and Parameter Optimization

1 Computational Efficiency and Parameter Optimization

This section provides detailed analysis of DAFM’s computational performance, hyperparameter optimization methodology, and sensitivity analysis supporting the model’s real-time deployment capability.

1.1 Runtime Performance Analysis

Table 7 presents comprehensive computational efficiency metrics comparing DAFM against baseline forecasting models across training and inference phases.

Table 7: Computational Efficiency Comparison of Forecasting Models

Model	Training Time (s)	Inference Time (ms)	GPU Util. (%)	CPU Util. (%)
Decision Tree	15	12	2	15
Random Forest	42	27	3	28
XGBoost	58	34	4	31
BiLSTM	145	72	68	35
DAFM	189	49	54	32
TCN-KAN [2]	285	95	72	41
LSTM-COA [4]	312	108	75	38
Stacking Ensemble [1]	198	156	45	52
Stacked Generalization [3]	224	178	51	58

Hardware Configuration All experiments were conducted across two complementary computing environments. Model implementation and initial testing were performed on a local workstation, while final training and inference benchmarking were carried out on a high-performance computing (HPC) server:

- **HPC Server:**
 - **GPU:** NVIDIA A100 (32 GB HBM2 memory)

- **CPU:** Intel Xeon Gold 6248R (3.0 GHz, 24 cores)
- **RAM:** 128 GB DDR4
- **Local Workstation:**
 - **GPU:** NVIDIA GeForce RTX 3050 (8 GB VRAM)
 - **CPU:** 12th Gen Intel Core i7-12700KF (12 cores / 20 threads, 3.0 GHz)
 - **RAM:** 128 GB DDR4/DDR5, **Storage:** NVMe SSD
- **Software Framework:** PyTorch 2.0.1 with CUDA 11.8
- **Batch Size:** 32 samples for inference benchmarking

Key Performance Insights 1. Real-Time Inference Capability

DAFM achieves an inference time of **49 ms per sample (0.049 seconds)**, enabling:

- **Throughput:** Approximately 20 samples per second
- **Latency:** Sub-second response for operational decision support
- **Real-time viability:** Suitable for continuous monitoring with 6-minute time steps (0.1-hour intervals)

This represents a **32% improvement** over BiLSTM (72 ms) and **48-72% faster** than recent state-of-the-art adaptive ensemble models [2, 4, 1, 3].

2. Training Efficiency Trade-off

While DAFM requires longer training time (189 s) compared to tree-based methods, this represents a one-time offline cost. The critical advantage lies in:

- **23% faster inference** than BiLSTM despite similar architecture complexity
- **Neural gating mechanism** enables efficient online weight adaptation without full retraining
- **Edge computing compatibility:** Lower GPU utilization (54%) compared to deep learning alternatives (68-75%)

3. Resource Utilization

DAFM demonstrates balanced resource consumption:

- **GPU utilization:** 54% (vs. 68% BiLSTM, 72-75% for TCN-KAN/LSTM-COA)
- **CPU utilization:** 32% (vs. 52-58% for stacking ensembles)
- **Memory footprint:** Approximately 8.2 GB peak GPU memory (vs. 12.5 GB for TCN-KAN)

1.2 Sliding-Window Sensitivity Analysis

To determine the optimal temporal window for adaptive forecasting, we conducted systematic sensitivity analysis across window lengths ranging from 12 to 48 months.

Table 7a: Sliding-Window Configuration Sensitivity Analysis

Window Length	MAE (bbl/day)	RMSE (bbl/day)	R ²	Adaptation Lag (min)
12 months	9.8 ± 1.2	18.9 ± 2.1	0.94 ± 0.02	42
18 months	8.9 ± 0.9	17.2 ± 1.8	0.96 ± 0.01	45
24 months	8.4 ± 0.8	16.7 ± 1.5	0.97 ± 0.01	48
30 months	8.5 ± 0.9	16.9 ± 1.6	0.97 ± 0.01	52
36 months	8.7 ± 1.0	17.4 ± 1.9	0.96 ± 0.02	58
48 months	9.2 ± 1.3	18.1 ± 2.3	0.95 ± 0.02	67

Optimal Window Selection: 24 Months The **24-month sliding window** was selected based on:

1. **Prediction Accuracy:** Achieves lowest MAE (8.4 bbl/day) and RMSE (16.7 bbl/day)
 - 14% improvement over 12-month window
 - Statistically equivalent to 30-month window ($p = 0.42$, paired t-test)
 - Captures seasonal cycles and inter-annual reservoir dynamics
2. **Adaptation Responsiveness:** Maintains sub-hour adaptation lag (48 minutes)
 - 28% faster than 36-month window (58 min)
 - 40% faster than 48-month window (67 min)
 - Balances historical context with recent trend sensitivity
3. **Computational Efficiency:**
 - Memory footprint: 8.2 GB (vs. 11.4 GB for 36-month, 15.1 GB for 48-month)
 - Inference time remains at 49 ms (increases to 63 ms for 36-month window)
 - Training convergence: 95% stable within 150 epochs
4. **Robustness to Perturbations:**
 - ±10% window length perturbations (21.6-26.4 months): MAE variance < 2%
 - Cross-validation stability: R² standard deviation = 0.01
 - Consistent performance across heterogeneous reservoir types (5-fold CV)

1.3 Hyperparameter Optimization Methodology

Table 2 (main manuscript) summarizes the grid search optimization process used to derive the final DAFM configuration. For completeness, the same table is reproduced in the Supplementary Materials.

Table 2: Grid Search Hyperparameter Optimization Details

Component	Parameter	Search Range	Optimal Value
Neural Gating	Depth (Layers)	1, 2, 3, 4	3
	Hidden Units	16, 32, 64, 128	32
	Activation	ReLU, Tanh, ELU	ReLU
	Momentum (γ)	0.5, 0.6, 0.7, 0.8, 0.9	0.8
Decision Tree	Max Depth	5, 10, 15, 20	10
	Min Split	2, 5, 10	5
	Criterion	Gini, Entropy	Gini
Random Forest	N Trees	50, 100, 150, 200	100
	Max Depth	None, 10, 20, 30	None
	Min Leaf	1, 2, 5	1
XGBoost	N Rounds	50, 100, 150, 200	100
	Learning Rate	0.01, 0.05, 0.1, 0.3	0.1
	Max Depth	3, 6, 9, 10	6
BiLSTM	Hidden Units	32, 64, 96, 128	64
	Dropout	0.1, 0.2, 0.3, 0.5	0.2
	Seq. Length	15, 30, 45, 60 days	30 days

Optimization Procedure

1. **Search Strategy:** Exhaustive grid search with 5-fold cross-validation
2. **Evaluation Metric:** Primary - MAE; Secondary - RMSE, R^2 , adaptation lag
3. **Training Data:** 70% of dataset (post-sliding window initialization)
4. **Validation Data:** 15% held-out for hyperparameter selection
5. **Test Data:** 15% final evaluation
6. **Convergence Criterion:** Early stopping with patience=20 epochs
7. **Total Configurations Tested:** 3,456 combinations
8. **Computational Cost:** Approximately 18 hours on 4 NVIDIA V100 GPUs

1.4 Comparison with State-of-the-Art Models

Table 4 (main manuscript) expands the computational comparison against recent adaptive ensemble models referenced in the main manuscript.

Table 4: Extended Computational Efficiency Comparison

Model	Inference (ms)	Memory (GB)	Edge Compatible
DAFM (Ours)	49	8.2	Yes
TCN-KAN [2]	95	12.5	Limited
LSTM-COA [4]	108	10.8	Limited
Stacking Ensemble [1]	156	6.4	Yes
Stacked Generalization [3]	178	9.7	No

Key Advantages:

- **48-72% faster inference** than deep learning alternatives (TCN-KAN, LSTM-COA)
- **69-72% faster** than multi-layer stacking approaches
- **Moderate memory footprint:** Enables edge deployment unlike TCN-KAN (12.5 GB)
- **Real-time weight adaptation:** Unlike static ensemble methods requiring full re-training

1.5 Summary

The computational efficiency and optimization analysis establishes:

1. **Real-time inference capability:** 49 ms per sample enables operational deployment
2. **Optimal 24-month window:** Balances historical context, adaptation speed, and accuracy
3. **Systematic hyperparameter tuning:** 3,456 configurations tested via grid search
4. **Edge computing viability:** Moderate resource requirements (8.2 GB, 54% GPU utilization)
5. **Superior to state-of-the-art:** 48-72% faster than recent adaptive ensemble models

References

- [1] M. Ghodsi, P. Vaziri, M. Kanaani, and B. Sedaee, “Optimizing oil production forecasts in Iranian oil fields: a comprehensive analysis using ensemble learning techniques,” *J Pet Explor Prod Technol*, vol. 15, no. 4, Apr. 2025, doi: 10.1007/s13202-025-01976-y.

- [2] Y. Hu, X. Xin, G. Yu, and W. Deng, “Deep insight: an efficient hybrid model for oil well production forecasting using spatio-temporal convolutional networks and Kolmogorov–Arnold networks,” *Sci Rep*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-91412-2.
- [3] M. Zhang et al., “Novel enhanced oil recovery screening methodologies by implementing improved stacking ensemble learning algorithms,” *Geoenergy Science and Engineering*, vol. 255, Dec. 2025, doi: 10.1016/j.geoen.2025.214104.
- [4] N. Makarov, M. Al-Shargabi, D. A. Wood, E. Burnaev, and S. Davoodi, “Prediction of oil production rate in multiple wells of a producing field applying combined deep-learning and optimization techniques,” *Fuel*, vol. 406, Feb. 2026, doi: 10.1016/j.fuel.2025.136847.