Petroleum Science 23 (2026) 1335–1347

Contents lists available at ScienceDirect

Petroleum Science

journal homepage: www.keaipublishing.com/en/journals/petroleum-science

Original Paper

Unconventional oil production forecasting based on PiAM meta-learning

Yu-Long Zhao [a,*], Qing-Yu Xiao [a], Xing-Jie Zeng [b], Bin Xiong [b], Shuai Wang [a], Song Zhao [c], Yun-Sheng Wei [d]

[a] *National Key Laboratory of Oil and Gas Reservoir Geology and Exploitation, Southwest Petroleum University, Chengdu, 610500, Sichuan, China*
[b] *School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu, 610500, Sichuan, China*
[c] *PetroChina Changqing Oilfield Company, Xi'an, 710018, Shaanxi, China*
[d] *PetroChina Research Institute of Petroleum Exploration \ Development, Beijing, 100083, China*

A R T I C L E I N F O

Article history:
Received 24 June 2025
Received in revised form
1 December 2025
Accepted 8 December 2025
Available online 16 December 2025

Edited by Jia-Jia Fei

Keywords:
Production forecasting
Meta learning
Transformer
Time series

A B S T R A C T

Accurate production forecasting serves as a critical determinant for optimizing extraction strategies, guiding long-term field management in reservoir development. Both conventional methods and deep learning techniques face significant challenges in production forecasting due to the increasing complexities of reservoir extraction. Firstly, traditional production forecasting methods often fail to fully capture the complex reservoir behavior. Finally, these approaches demonstrate suboptimal performance in wells with limited data. These problems can lead to a decrease in prediction accuracy. To address these challenges, this paper introduces the Patching-iTransformer method and applies

^{2} increased by 0.297, RMSE decreased by 11.64% and MAE decreased by 3.49%. PiAM meta-learning method demonstrated superior performance over the Patching-iTransformer model, showing a 0.535-point improvement in the R^2 coefficient along with a reduction of 27.54% in RMSE and a decrease of 28.22% in MAE.

1. Introduction

With the increasing global demand for oil and the increasing complexity of extraction, accurate forecasting of future oil production plays a vital role in ensuring the long-term stability of production. Unconventional reservoir wells are being developed at a faster decline rate, and the number of wells placed in each block is significantly higher than in conventional reservoirs. This results in an increased amount of data for the entire field but less data for individual well extraction, which leads to poor conventional production forecasts. Oil field production generates a significant amount of data each day. When these data are arranged chronologically, they form time series for various parameters, which can be considered as time-series information. These time series often reveal the intrinsic relationships among the different factors involved in the development of the oil field. Production forecasts can be derived using analytical methods, such as the decline curve analysis (DCA) model (Fetkovich, 1973). This method requires manual adjustment of parameters at different stages and is applicable only during the declining phase of field development, resulting in poor predictive performance. Another approach is numerical simulation (Zhang et al., 2016). The method employs a computational model to simulate oil field extraction patterns by

* Corresponding author.
E-mail address: 373104686@qq.com (Y.-L. Zhao).
Peer review under the responsibility of China University of Petroleum (Beijing).

https://doi.org/10.1016/j.petsci.2025.12.015
1995-8226/

integrating historical production data with geological parameters. Numerical techniques are then applied to solve the resulting system of equations and forecast future production. However, increasing the level of detail in oil field simulation substantially slows the computational process.

To overcome these limitations, neural network methods (Vyas et al., 2017) have emerged as a prominent research direction in recent years. Neural networks have been extensively utilized in oil field production forecasting because of their capacity to manage complex nonlinear relationships, deliver rapid predictions, process data efficiently, and effectively capture long-term trends. Numerous neural network models have been developed for yield prediction, including recurrent neural networks (RNN) (Elman, 1990), long short-term memory networks (LSTM) (Graves and Graves, 2012), and Transformer networks (Vaswani et al., 2017). Owing to the outstanding performance of the Transformer architecture in natural language processing (Devlin et al., 2019) and computer vision (Ramachandran et al., 2019), researchers have also introduced its application to time series forecasting and achieved highly promising results. For instance, models such as Autoformer (Wu et al., 2021) and Informer (Zhou et al., 2021) primarily focus on refining the internal structure of the Transformer. Additionally, NSTransformer (Liu et al., 2022) fully leverages the core capabilities of the Transformer, while Crossformer (Zhang and Yan, 2023) incorporates enhancements to both its internal structure and overall design. Recent studies have started to question the efficacy of enhanced Transformer-based neural networks. They even suggest that simple linear layers can outperform these more complex Transformer architectures in both performance and efficiency. Researchers have investigated the reasons why the Transformer architecture exhibits strong predictive performance in other fields but performs poorly in time series prediction. One possible explanation is that the diverse physical meanings of measurements recorded at the same time step imply that a single moment does not fully capture the true nature of an event. In other words, simply integrating different variables into one token overlooks both the inter-variable correlations and the asynchronous aspects of the event. Time series data depends heavily on the order of observations. However, some studies apply order-independent attention mechanisms directly to the temporal dimension. Thus, they disregard the inherent sequential structure of the data.

The challenges of employing Transformer-based neural networks for yield prediction can be summarized as:

- Neglect of interactions among variables at distinct time points, and of asynchronous information, which undermines the model's predictive performance.
- Inadequate intra-variable correlations, especially when long lookback periods are used, compromising prediction validity.
- Scarcity of historical data during the initial stages of exploitation has led to suboptimal forecasts for individual wells.

Production forecasting for an individual well is often based solely on the historical production data of an individual well. Production from individual wells in the same reservoir is highly interdependent. When limited historical data is available for a particular well, it becomes necessary to utilize data from other wells within the same layer to predict production for that small sample. Transfer learning is currently an effective strategy for addressing small-sample well yield prediction. However, most transfer learning approaches rely on pre-training models using public datasets, which hampers the transfer of knowledge from the source domain to the target domain due to distributional differences. Meta-learning facilitates the development of models that generalize across diverse tasks through joint multi-task training, enabling rapid adaptation to a novel task in only a few steps. Because a generic model is trained on data from wells throughout the reservoir, it can perform well even in wells with few samples. The primary contributions of this paper are as follows.

- We employ an inverted Transformer in our approach. In this architecture, each time point in the series is treated as its own token. Relationships between these tokens are then learned through a self-attention mechanism. Subsequently, a feed-forward layer extracts features from each token.
- To address the issue of insufficient attention to the internal correlations among variables, we propose a patching strategy for time series analysis. This approach involves dividing each variable's time series into patches, which not only preserves intra-variable correlations but also enhances prediction accuracy for longer lookback periods.
- Oil field data is complex and contains high levels of noise. Each well in the reservoir also influences recovery to a different degree. To address these challenges, we propose a PiAM meta-learning method. This enables the model to better capture the unique characteristics of the oil field data.

The remainder of the paper is organized as follows: Section 2 discusses preliminary work. Section 3 presents the Paching-iTransformer and PiAM method. Section 4 evaluates the method in an application. Section 5 concludes the paper and introduces future work.

2. Preliminary works

Production forecasting has always been a top priority for oil fields. Early work relied on classical decline curve models, such as the Arps declining model and the logistic model. Subsequently, forecasts used numerical simulation software to improve accuracy. In recent years, neural network approaches have further simplified forecasting and boosted precision.

2.1. Individual well production forecasts

The rise of neural networks for production forecasting is largely due to advances in data measurement and increased computational power. These developments have made it much easier to analyze large volumes of production-related data. Deep learning models that rely on large datasets have specific prerequisites. Production forecasting for unconventional reservoirs has steadily evolved through successive data-driven innovations. Fracture parameters were first incorporated into neural networks to forecast unconventional reservoir output (Li and Han, 2017). A comparative study highlighted clear differences between the two approaches. Classical DCA produces only smooth decline profiles. In contrast, LSTM-based forecasting captures overall production trends more accurately (Sun et al., 2018). A transformer-based multivariate time series framework was introduced for individual well production forecasting. It builds on recurrent architectures and leverages self-attention to model long-term dependencies (Abdrakhmanov et al., 2021). Further improvements came from hybridizing LSTM with CNN and embedding an attention mechanism, which demonstrated superior predictive accuracy compared to conventional designs (Pan et al., 2023). In recent years, the Temporal Fusion Transformer (TFT) has been applied to production forecasting (Al-Ali and Horne, 2023b). It outperforms BlockRNN models in capturing extended time-series dynamics.

2.2. Application of patching

Applications of patching first emerged in computer vision and speech recognition, and were subsequently widely adopted in improved Transformer models. In computer vision, researchers achieved excellent performance on an image classification task by dividing images into 16 $\times$ 16 patches and incorporating a self-attention mechanism (Dosovitskiy et al., 2020). Similarly, in speech recognition, patches are used to extract subsequence-level information from the original speech input (Baevski et al., 2020). In time series forecasting, the PathTST model (Nie et al., 2023) extracts subsequence information by dividing the data into patches, thereby enhancing the accuracy of long-term predictions.

2.3. Meta-learning

The primary reason for the significant limitations in individual well production forecasting is that the limited data from one well hinders the model's ability to capture complex nonlinear relationships in the oil field. Moreover, the weak transferability of individual well models necessitates frequent retraining when predicting new wells. To overcome this problem, it is essential to endow models with "learn to learn" capabilities as an effective solution.

Meta-learning techniques have been employed to boost production forecasting performance. Pre-training the N-BEATS architecture on an open dataset and then fine-tuning it for a specific well resulted in outperformance of a conventional LSTM (Al-Ali and Horne, 2023a). Combining RNN models with the MAML algorithm enabled few-sample well forecasts to leverage prior knowledge, leading to a marked reduction in test error (Wang et al., 2024). Applying MAML to LSTM's training confirmed these benefits and demonstrated additional gains in prediction accuracy (Xu and Yu, 2024).

Meta-learning methods can be classified into three categories, including metric-based (Vinyals et al., 2016), optimization-based (Mishra et al., 2018), and model-based (Finn et al., 2017). Production often involves multiple concurrent tasks. Optimization-based meta-learning methods excel at handling such multi-task scenarios. As a result, these methods are now more widely applied in production forecasting. However, optimization-based meta-learning algorithms require repeated gradient computations in multi-task environments, which not only consume significant computational resources but also tend to converge to local optima. Consequently, a relatively simple base model is often employed in practical production prediction to avoid a gradient explosion. However, such models struggle to effectively capture the complex nonlinear relationships in oil fields. To resolve this contradiction, researchers proposed a MAML++ method (Antoniou et al., 2019) that has significantly reduced computational complexity and mitigated convergence to local optima.

3. Method

In this section, we elaborate on the design details of our method. Firstly, we preprocess the oil reservoirs data and partition the dataset to suit the meta-learning approach. Then, we describe the proposed Patching-iTransformer, which comprises two components: (a) patch-based segmentation of the time-series data to boost intra-variable correlation, and (b) inversion of the time series to enhance inter-variable correlation. Finally, we propose a PiAM meta-learning algorithm for oil reservoirs.

3.1. Data processing

In this paper, data is collected from 49 wells in the oil field with 9 variables per well, including daily oil production, temperature, casing pressure, flowing pressure, and other variables. Conventional noise reduction methods often blur short-term features in the data due to manual well-switching operations inherent in the field data. Kalman filtering, through its state estimation mechanism, efficiently reduces noise in intermittent production data. It preserves the short-term fluctuations in the data. We normalized the data before noise reduction. This step eliminated the effects of magnitude differences.

Kalman filter mathematical model. We assume the system state is represented by vector x_k. The state at time step k is predicted using the previous state and control inputs. This process is mainly governed by

$$x_k = \boldsymbol{F}_{k-1} x_{k-1} + \boldsymbol{B}_{k-1} u_{k-1} + w_{k-1} \tag{1}$$

to predicting the current state, where x_k denotes the state at time k, $\boldsymbol{F}_{k-1}$ denotes the state transition matrix that relates the previous state to the current state, $\boldsymbol{B}_{k-1}$ denotes the control input matrix, u_{k-1} denotes control input at time $k-1$, w_{k-1} denotes the process noise, assumed to be Gaussian. The predictions are then corrected using the observations, according to the observation equation

$$z_k = \boldsymbol{H}_k x_k + v_k, \tag{2}$$

where z_k denotes the measurement at time k, $\boldsymbol{H}_k$ denotes the observation matrix that maps the state to the measurement space, v_k denotes the measurement noise, assumed to be Gaussian.

State prediction. At each time step, the Kalman filter predicts the next state using

$$\widehat{x}_k^- = \boldsymbol{F}_{k-1} \widehat{x}_{k-1} + \boldsymbol{B}_{k-1} u_{k-1}, \tag{3}$$

where $\widehat{x}_k^-$ denotes the predicted state estimate at time k. The uncertainty of the state prediction is also estimated at each step. The predicted covariance matrix is updated by

$$\boldsymbol{P}_k^- = \boldsymbol{F}_{k-1} \boldsymbol{P}_{k-1} \boldsymbol{F}_{k-1}^{\mathrm{T}} + \boldsymbol{Q}_{k-1}, \tag{4}$$

Fig. 1. Meta-learning multivariate time series data partitioning.

where $\boldsymbol{P}_k$ denotes the updated error covariance matrix, $\boldsymbol{I}$ denotes the identity matrix. It reflects the uncertainty in the corrected state estimate.

Next, the data is partitioned using a strategy distinct from conventional deep learning. Fig. 1 illustrates how the multivariate time series data is divided in meta learning. Firstly, wells are assigned as individual tasks and randomly split into meta-learning training and testing tasks in an 8:2 ratio. The data within each well is then divided chronologically into a support and query set in an 8:2 ratio.

The PiAM model is specifically designed to handle few-shot data effectively. It achieves this through the use of meta-learning techniques. In this process, the training and test tasks share similarities, such as time-series data from the same reservoir. However, they also differ, as the test tasks include data from unseen wells. This contrast helps assess the model's generalization ability. The success of transfer learning depends on the similarity between the training and test tasks, especially in terms of reservoir features. This similarity ensures the model can transfer knowledge effectively and make accurate predictions with fewer samples. With this setup, the PiAM model can quickly adapt to test tasks. It makes accurate production predictions by leveraging data from training tasks.

The processed oil field data shows an overall decreasing trend. However, there are clear short-term fluctuations within each interval. This indicates that the data retains significant noise, necessitating that the model better handle the correlation information both between and within variables. In the early stage, the data exhibits minimal fluctuations. In the later stage, the fluctuations become dramatic. The variance evolves over time, and uncertainty increases. This necessitates more adaptive inner-loop optimization strategies.

3.2. Patching-iTransformer

Fig. 2 shows the Patching-iTransformer structure. It employs an encoder-only Transformer architecture and primarily comprises Patching, Flatten, and Transformer blocks.

Previous time series forecasting models based on the Transformer architecture typically input all variables simultaneously as tokens. In contrast, our model first applies patch partitioning to the input time series, treating each patch as an independent token. The attention mechanism then calculates the attention values between patches, and a flatten operation captures correlations both between and within patches. Finally, a linear layer produces the final prediction. The above forecast can be expressed by

$$\begin{aligned} h_n^0 &= \mathrm{Patching}(X_n), \\ \boldsymbol{H}^{l+1} &= \mathrm{TrmBlock}\left(\boldsymbol{H}^l\right) \quad l = 0, \dots, L-1, \\ \widehat{Y}_n &= \mathrm{Flatten}\left(h_n^L\right), \end{aligned} \tag{8}$$

where $\widehat{Y}_n$ denotes prediction data, X_n denotes lookback time series, $\boldsymbol{H} = \{h_1, \dots, h_v\} \in \square^{N\times D}$ denotes that there are N D-dimensional encoded tokens. Both patching transforms $x \in \mathbb{R}^T$ into $x \in \mathbb{R}^D$ and flatten transformers $x \in \mathbb{R}^D$ into $x \in \mathbb{R}^S$ are implemented using multi-layer perceptron (MLP). The Transformer's positional encoding is unnecessary because the inherent order of the sequence provides sufficient positional information.

3.2.1. Patching

Each input time series is divided into either overlapping or non-overlapping patches. We define the patch length as P and the stride as S. Patching is the process of dividing a univariate time series into segments of length P, with a step size of S. After patching, the original time series $x^{(i)} \in \mathbb{R}^{1\times T}$ is transformed into a sequence of patches $x^{(i)} \in \mathbb{R}^{N\times P}$, where $N = \left\lceil \frac{L-P}{S} \right\rceil + 2$. Before patching, we pad the original sequence by repeatedly appending its final value until its length is fully divisible by the patch length. After patching, the number of input tokens is reduced, which not only decreases GPU memory usage but also enhances the model's focus on inter-variable relationships, addressing the inadequate of the inverted model.

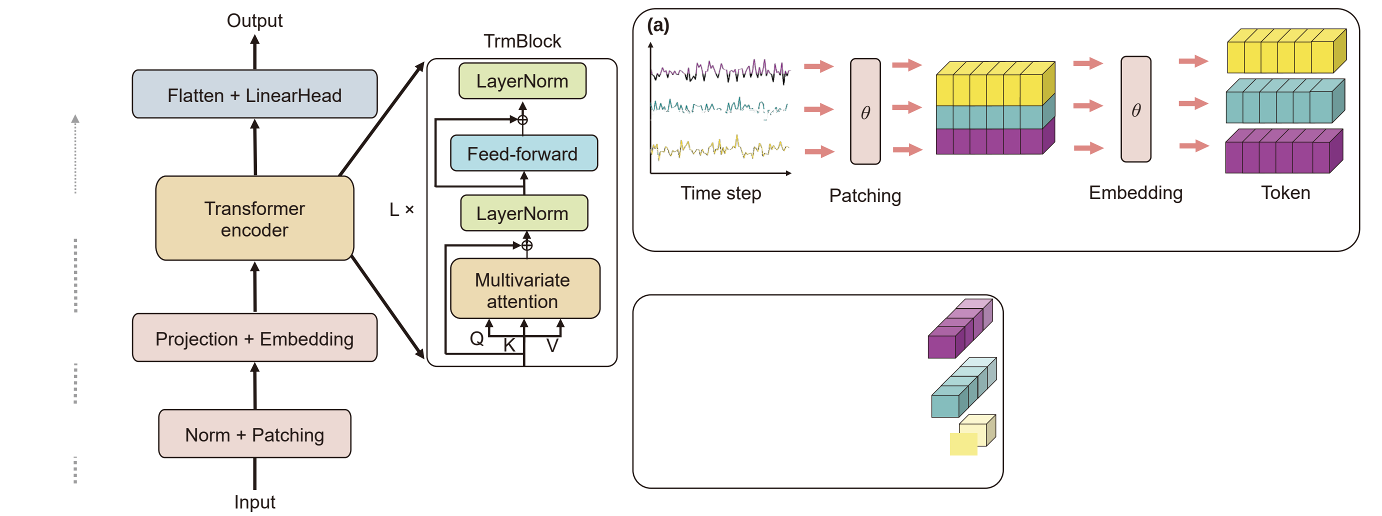

Fig. 2. Overall structure of Patching-iTransformer. **(a)** Patching module: After undergoing patch division processing, each variable should be entered as an individual token. **(b)** Multivariate attention mechanism. **(c)** Feedforward neural network: A model for independently processing multivariate features.

3.2.2. *Layer normalization*

Layer normalization is primarily used to enhance the stability and convergence speed of deep neural network training (Ba et al., 2016). However, because layer normalization normalizes the features of multivariate representations at the same time step, it may cause features from different variables to merge. This can introduce noise when the physical significance of the same time point differs across variables. With inverted layer normalization as

$$\text{LayerNorm}(\boldsymbol{H}) = \left\{ \frac{h_n - \text{Mean}(h_n)}{\sqrt{\text{Var}(h_n)}} \middle| n = 1, \dots, N \right\}, \tag{9}$$

which avoids noise between variables for better handling of complex time series.

3.2.3. *Feed-forward network*

The feed-forward neural network in the traditional Transformer primarily encodes individual tokens. However, in time series data, a token comprises multiple variables simultaneously. These variables may not be consistent, and the brief time span limits the available information, making it difficult to capture long-term trends. In the inverted version, the feed-forward network (FFN) operates directly on the entire time dimension of each variable. According to the approximation principle (Hornik, 1991), an MLP can capture the complex feature transformations of a given variable over time. Consequently, stacking multiple layers enables accurate time series prediction. Different neurons in an MLP can identify various features of a time series, such as amplitude and frequency spectra. Therefore, an MLP can serve as a more accurate and generalized learner for time series. This contrasts with self-attention mechanisms, which are applied only on individual time points.

3.2.4. *Self-attention*

In the traditional Transformer model, the attention mechanism captures temporal relationships. While we have focused on the relationships within individual variables, strong correlations between different variables also exist. In the inverted model, we treat each variable as a distinct channel, with each channel forming a separate sequence. This design captures the interrelationships between variables via the attention mechanism. For each time series $\boldsymbol{H} = \{h_0, \dots, h_N\} \in \square^{N \times D}$, the queries, keys, and values $\boldsymbol{Q}, \boldsymbol{K}, V \in \square^{N \times d_k}$ are obtained by projection, where d_k is the dimension of the projected subspace. $q_i, k_i \in \mathbb{R}^{d_k}$ represent query and key of each variable. The attention score is computed by calculating the $\boldsymbol{Q}$ and $\boldsymbol{K}$ matrices as follows $A_{i,j} = \frac{(\boldsymbol{QK}^\top)_{i,j}}{\sqrt{d_k}} \square q_i^\top k_j$. In this way, the overall attention score reflects the correlation between variables. By multiplying the attention score A and V, highly correlated variables receive higher weights. Consequently, the calculation effectively represents the inter-variable correlations.

3.3. *Adaptive MAML++*

MAML is a meta-learning method proposed by Finn et al. (2017) in 2017. The main idea is to train a general model parameter across a large number of tasks that share similar structures. This trained parameter then lets the model adapt quickly to a new task. Since this algorithm relies on gradient descent for optimization and does not depend on a specific neural network architecture, it's called model-agnostic. MAML consists of two layers of optimization, an inner loop and an outer loop. The inner loop is optimized for a specific task to obtain task-specific parameters θ_i. The outer loop passes the updated loss from the inner loop to the initial model to enhance its generalization. In summary, MAML aims to learn a set of initial model parameters θ that allow rapid adaptation to new tasks.

We define a base neural network model f_θ, where θ indicates model parameters. We define initial parameters $\theta = \theta_0$ sampling a batch of tasks $\{T_i\}$ from the task distribution T, where $i = 1, \dots, N$. For each task T_i, there are support set D_i^{Support} and query set D_i^{Query}. For each specific task T_i, the training process begins with the current model parameters θ_i. Model using D_i^{Support} performs a small number of gradient updates to obtain task-specific parameters θ_i'

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta), \tag{10}$$

where α denotes the learning rate in inner loop, $\mathcal{L}_{\mathcal{T}_i}$ denotes the loss function of specific task T_i. For each particular task T_i, the loss $\mathcal{L}_{\mathcal{T}_i}(\theta_i')$ is computed using D_i^{Query} updated parameters θ_i'. Assuming there are B training tasks, the outer loop loss is computed as

$$\mathcal{L}_{\text{Outer}}(\theta_0) = \sum_{i=1}^{B} \mathcal{L}_{\mathcal{T}_i}(\theta_i'). \tag{11}$$

The outer loop update process is

$$\theta_0 = \theta_0 - \beta \nabla_\theta \sum_{i=1}^{B} \mathcal{L}_{\mathcal{T}_i}(\theta_i'), \tag{12}$$

where β denotes learning rate in outer loop.

3.3.1. Multi-step loss optimization

The outer loop optimization of MAML is implemented by backpropagating through the losses from several inner loops. Specifically, MAML is trained multiple times on the support set, and the losses on the query where T denotes total training epochs, T_{cur} denotes current training epoch, α_{max} denotes maximum learning rate set, α_{min} denotes minimum learning rate set, α_t denotes current learning rate. Fig. 3 illustrates the final PiAM parameter update model. Algorithm 1 illustrates the whole algorithm process.

Algorithm 1 PiAM for production prediction

Input: $p(T)$: distribution over tasks

4. Experiment

In this section, we define our model and conduct several experimental comparisons to evaluate its performance. Specifically, we perform a base model comparison, compare MAML++ with PiAM, and carry out iterative predictions to assess model stability. Fig. 4 presents the loss pairs for all models.

4.1. Base model comparison

We evaluate the models using individual well prediction. The well data was divided into 80% for training and 20% for testing. The lookback window period is set to 120, and the prediction length is 10 days. Each model employs the seq2seq method for prediction, and the loss was computed using the Adam optimizer and mean squared error (MSE). We employ LSTM and iTransformer networks, with the Patching-iTransformer model used for comparative purposes in the experiment. Fig. 5 illustrates the individual well prediction results.

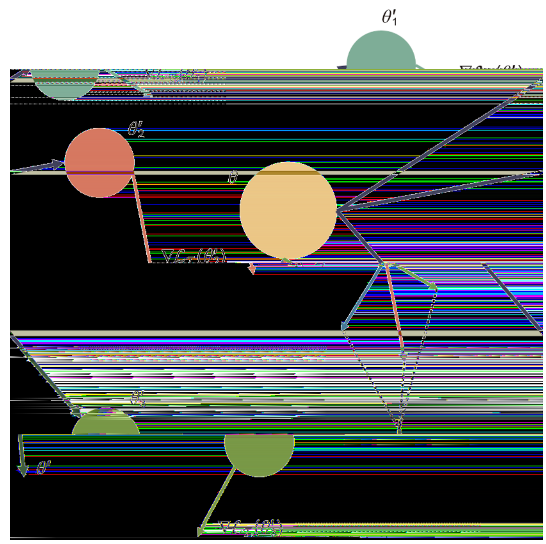

Fig. 3. The update of PiAM parameters.

In the experiments, Patching-iTransformer model's prediction performance was particularly impressive for Well 1 and Well 4. For Well 1, it achieved an R^2 of 0.491, an RMSE of 21.556, and an MAE of 16.602 shown in Table 1. This demonstrates that the model can stably capture long-term dependencies in oilfield production data, with minimal deviation from actual observations. Patching-iTransformer also performs better when faced with anomalous fluctuations in the data. As shown in Fig. 5, Patching-iTransformer predicts data trends smoothly and accurately for different wells, particularly for Well 1 and Well 4. This highlights its strength in handling long-term dependencies and outliers.

However, both iTransformer and LSTM networks show significant performance fluctuations on specific wells, especially when hyperparameters are not correctly selected. This increases the risk of overfitting, resulting in poorer prediction performance. For example, iTransformer model's R^2 for Well 3 is −0.147, with an RMSE of 34.395 and an MAE of 25.084. In the same well, LSTM model's R^2 is −15.792, with an RMSE of 150.773 and an MAE of

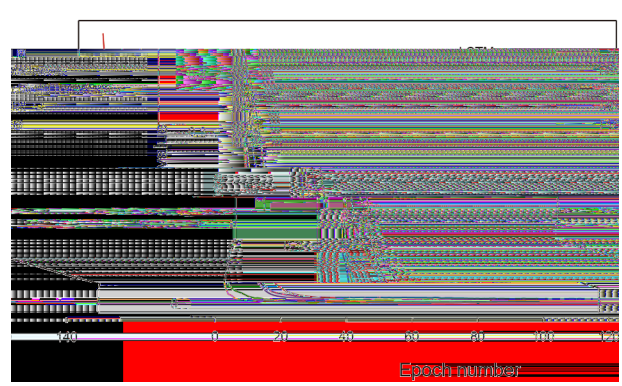

Fig. 4. Model loss on the training set.

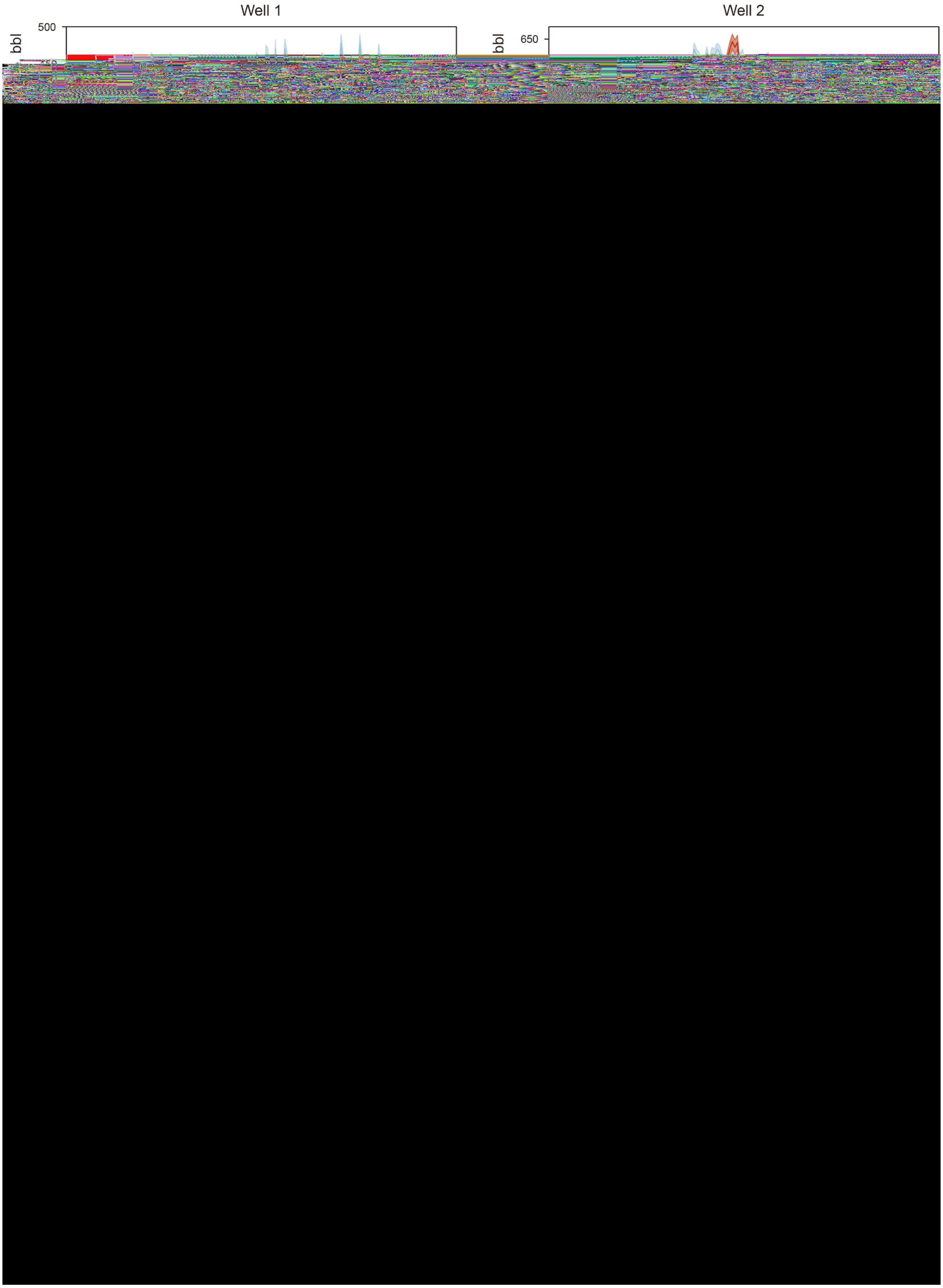

Fig. 5. Comparative results of individual well model predictions.

128.648 as shown in Table 1. These results show that when hyperparameters are not set correctly, the model fails to adapt effectively to the characteristics of different wells. This leads to a significant increase in prediction error. The prediction curves in Fig. 5 further highlight significant fluctuations in the predictions of iTransformer and LSTM, especially at Well 3. In this case, the model's performance deviates greatly from the actual production trends.

Table 1
Comparative results of individual well model predictions. The best results are in **bold**.

Models	Patching-iTransformer			iTransformer			LSTM		
	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE
Well 1	**0.491**	**21.556**	**16.602**	0.264	23.996	19.625	−0.922	41.717	32.845
Well 2	**−0.347**	**48.817**	**36.784**	−0.570	52.705	35.225	−0.319	75.737	60.919
Well 3	**−0.103**	**33.723**	**24.200**	−0.147	34.395	25.084	−15.792	150.773	128.648
Well 4	**0.471**	**32.157**	**23.902**	0.403	34.160	24.902	0.221	43.436	35.166
Well 5	**0.368**	**54.127**	**37.348**	0.107	64.331	46.182	−0.12	52.547	40.949
Well 6	**−0.139**	**19.137**	**16.327**	−1.230	26.777	21.103	−0.522	23.720	18.685
Well 7	**−0.001**	**74.632**	**55.303**	−0.298	84.999	68.842	−0.073	77.283	63.356
Well 8	**−0.960**	**76.913**	**66.639**	−1.220	81.848	66.379	−2.347	100.495	89.819
Well 9	**0.615**	**55.122**	**42.863**	0.417	67.800	44.918	−1.899	151.212	135.534
Average	**0.044**	**46.243**	**35.552**	−0.253	52.335	39.14	−2.419	79.657	67.324

Overall, the Patching-iTransformer model outperforms iTransformer and LSTM in capturing long-term dependencies and handling outlier fluctuations. It also shows greater flexibility and stability in hyperparameter tuning. While iTransformer excels in short-term prediction, it falls short when dealing with complex time series data, particularly in handling long-term dependencies and outliers.

4.2. Comparison of meta-learning methods

We compare the prediction results of the Patching-iTransformer, MAML++, and the PiAM model. For the comparison between PiAM and MAML++, we conduct 150 epochs of outer loops, 2 inner loops, and 10 testing loops. Fig. 6 presents the results.

In our experiments, we evaluated the MAE, RMSE, and R^2 coefficients for the PiAM, MAML++, and Patching-iTransformer models. When the data is sufficient, the Patching-iTransformer and meta-learning models provide comparable predictions. The Patching-iTransformer model excels at capturing long-term dependencies in time series, especially when using patching and the inverted strategy. Compared to MAML++, Patching-iTransformer is better at capturing complex relationships between variables, leading to more accurate predictions in most cases. As shown in Table 2, PiAM achieves R^2 of 0.294, an RMSE of 73.505, and an MAE of 52.887 for Well 1. This indicates that PiAM maintains high prediction accuracy despite data scarcity.

The PiAM model performs exceptionally well when data is scarce. It is able to converge quickly with very few gradient updates. In Well 3, PiAM reached R^2 0.649, RMSE 115.679, MAE 64.486 as shown in Table 2. These results highlight the advantages of PiAM in few-shot learning. A comparison with MAML++ shows that PiAM significantly outperforms MAML++ in prediction performance. This improvement is attributed to PiAM's better handling of noisy data and complex oil field data structures.For example, in Well 2, PiAM achieves an R^2 of −0.152, an RMSE of 34.990, and an MAE of 27.903. In comparison, MAML++ has an R^2 of −0.115, an RMSE of 46.882, and an MAE of 37.882 shown in Table 2.

For single-well prediction, traditional models struggle to fit the data due to insufficient samples. In contrast, the meta-learning approach is more robust. Meta-learning requires only one or two gradient updates for a new task to achieve a better fit, significantly reducing computation time. This advantage makes PiAM ideal for real-time predictions and resource-constrained environments.

Although PiAM and MAML++ show similar MAE values for Well 3 and Well 5, PiAM achieves a better R^2 value, indicating a more accurate fit to the actual data. Some of the MAE deviations may result from large discrepancies in individual parameter points. However, overall, PiAM outperforms MAML++. Thus, despite the MAE differences, PiAM better captures the data trend and demonstrates superior predictive ability.

Fig. 6 compares PiAM's prediction errors across different wells, highlighting its stability under data-scarce conditions. In particular, PiAM significantly outperforms MAML++ and Patching-iTransformer in Well 2 and Well 3. The figure clearly shows PiAM's advantages in sample-efficient learning, especially when handling complex data structures. Its smaller prediction error compared to other models verifies its strong robustness in practical applications.

4.3. Stability experiment

To assess the model's performance in long-term forecasting, we increase the prediction time lookback length $S \in \{10, 15, 30, 60, 120, 240\}$ and prediction length $T \in \{10, 20\}$ to evaluate its performance on long time-series data. As shown in Fig. 7, the PiAM model outperforms others in terms of MAE as the lookback length S increases. Notably, the error of the PiAM model consistently decreases for longer prediction length $T = 20$. This demonstrates PiAM's ability to capture long-term dependencies and trends effectively. In contrast, models like MAML++ and iTransformer exhibit an increase in error rate and volatility as the backtracking window extends. This suggests that their performance becomes more unstable when handling longer time series.

Taking Well 1 as an example, the PiAM model maintains a low MAE as the lookback window increases from 10 to 240. This demonstrates its ability to capture long-term dependencies effectively. In contrast, models like iTransformer and LSTM show a significant rise in error with the increasing lookback window. This suggests they struggle to handle longer time series effectively.

We assessed the model's stability, robustness, and generalization ability. We run iterative forecasts in 10-day increments over periods of 30, 50, 70, and 100 days. We use mean absolute error (MAE), root mean squared error (RMSE), and R^2 coefficients as evaluation criteria.

Table 3 shows that the model's prediction error generally increases with the prediction length, especially when the data is scarce or noisy. For example, in Well 1, PiAM achieves an RMSE of 41.820 and an MAE of 39.464 for a 30-day prediction, demonstrating strong accuracy and stability. However, as the lookback window was extended to 100 days, the RMSE increased to 94.024 and the MAE to 90.305. Similarly, for Well 2, PiAM performed better in short-term forecasting, with an RMSE of 87.826 and an MAE of 72.885 for a 30-day forecast. We find that the prediction error increases as the prediction length grows. However, in short-term predictions, PiAM better captures the data patterns, especially when data is scarce, achieving high accuracy with few gradient updates. The model excels in short-term and small-sample learning tasks, but its performance declines in long-term predictions.

We tested the impact of patch length P and stride S on the performance, and the results are shown in Table 4. In this study, we adjusted P and S to assess their impact on the receptive field of each layer in the

Table 2
Meta-learning model prediction comparison results.The best results are in **bold**.

Models	PiAM			MAML++			Patching-iTransformer		
	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE
Well 1	**0.294**	**73.505**	**52.887**	0.268	82.185	61.640	−0.194	114.578	74.754
Well 2	**−0.152**	**34.990**	**27.903**	−0.115	46.882	37.882	−0.187	41.245	31.434
Well 3	**0.649**	**115.679**	64.486	0.618	117.261	**62.237**	0.594	126.917	73.623
Well 4	**0.470**	**33.858**	**26.407**	0.409	39.372	31.745	0.219	39.039	27.398
Well 5	**0.514**	**51.630**	35.049	0.469	51.920	**34.315**	0.354	54.737	34.599
Well 6	**0.096**	**13.314**	**11.408**	−0.545	22.284	19.405	−2.536	17.843	15.114
Well 7	**−0.376**	**69.344**	**53.047**	−1.245	70.521	45.609	−0.792	99.864	76.522
Well 8	**0.440**	**35.355**	**27.624**	0.389	40.105	31.389	−1.805	91.995	80.468
Well 9	**0.398**	**50.170**	**38.656**	0.070	53.104	43.424	−0.130	68.815	51.922
Average	**0.259**	**53.094**	**37.496**	−0.180	57.160	39.954	−0.276	73.275	52.236

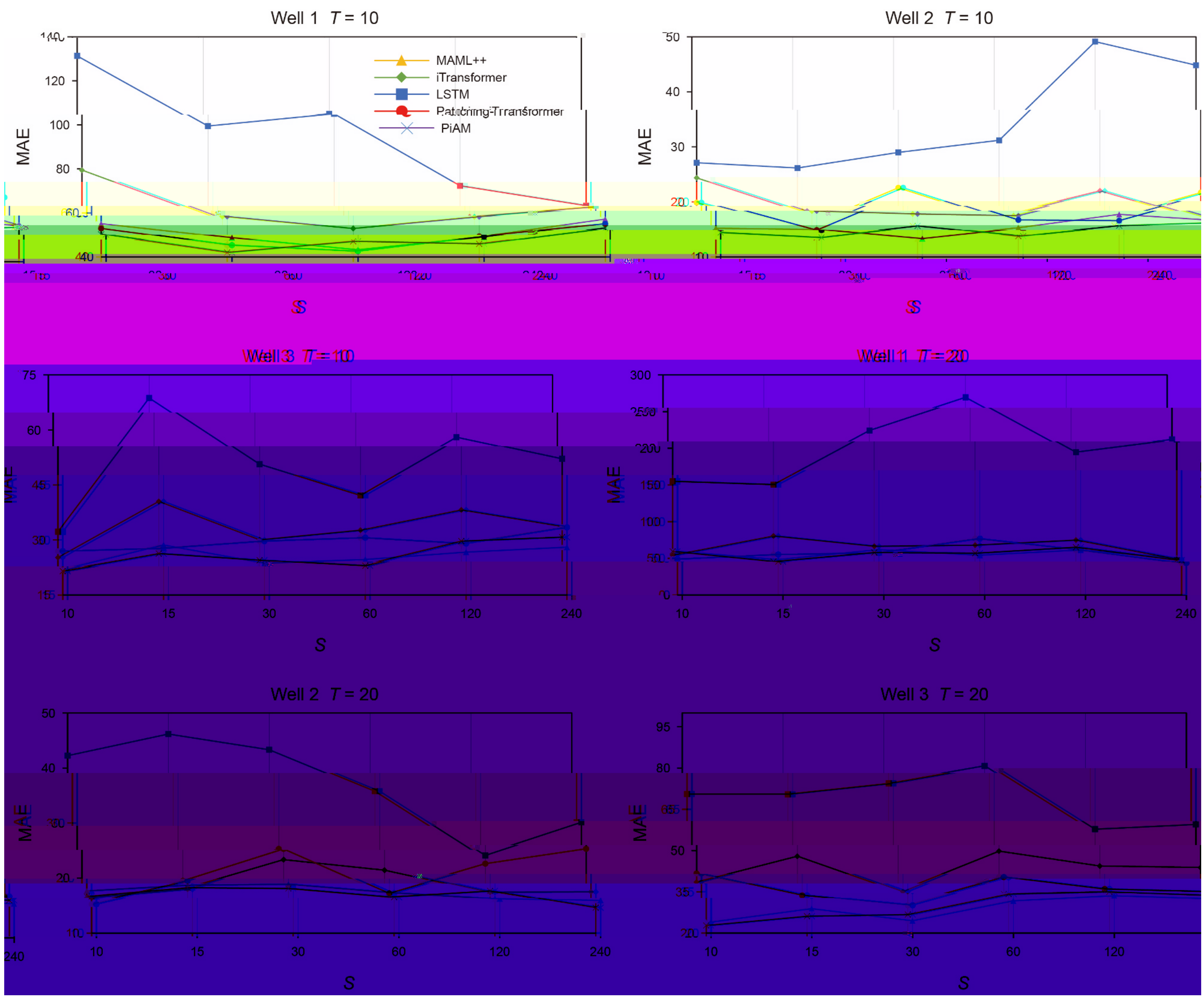

Fig. 7. Different lookback windows model prediction performance (MAE).

prediction accuracy. The experimental results indicate that for Well 1, the model performs best when $P = 12$ and $S = 20$. Increasing P further leads to higher RMSE due to model overfitting. In contrast, smaller P and S in Well 2 and Well 3 yield the best predictions. This is because Well 1 had less data and required a more complex model to learn its production regime, while increasing well data for Wells 2 and 3 only needed a simpler model. Additionally, as P and S increase, the learning complexity of the model rises, significantly increasing computational resource consumption. This experiment highlights that selecting the optimal combination of P and S is crucial to balancing model accuracy and computational resource efficiency.

This experiment was conducted using 4060 GPU. The PiAM framework takes approximately 50 min for computation, showing a significant improvement in efficiency compared to traditional numerical modeling. It also consumes fewer computational resources.

5. Conclusions and future works

The Patching-iTransformer method integrates a patching mechanism into the inverted

Al-Ali, Z.A.A.H., Horne, R., 2023a. Meta learning using deep N-BEATS model for production forecasting with limited history. In: SPE Gas & Oil Technology Showcase and Conference. https://doi.org/10.2118/214214-MS.
Al-Ali, Z.A.A.H., Horne, R., 2023b. Probabilistic well production forecasting in volve field using temporal fusion transformer deep learning models. In: SPE Gas & Oil Technology Showcase and Conference. https://doi.org/10.2118/214133-MS.
Antoniou, A., Edwards, H., Storkey, A., 2019. How to train your maml. arXiv preprint. arXiv:1810.09502. https://arxiv.org/abs/1810.09502.
Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. arXiv preprint. arXiv: 1607.06450. https://arxiv.org/abs/1607.06450.
Baevski, A., Zhou, Y., Mohamed, A., et al., 2020. wav2vec 2.0: a framework for self-supervised learning of speech representations. Adv. Neural Inf. Process. Syst. 33, 12449–12460. https://doi.org/10.48550/arXiv.2006.11477.
Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the association for Computational Linguistics: Human language Technologies, Volume 1 (Long and Short Papers). https://aclanthology.org/N19-1423/.
Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al., 2020. An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint. arXiv: 2010.11929. https://arxiv.org/abs/2010.11929.
Elman, J.L., 1990. Finding structure in time. Cogn. Sci. 14 (2), 179–211. https://doi.org/10.1207/s15516709cog1402_1.
Fetkovich, M.J., 1973. Decline curve analysis using type curves. In: SPE Annual Technical Conference and Exhibition. https://doi.org/10.2118/4629-MS.
Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. https://arxiv.org/abs/1703.03400.
Graves, A., Graves, A., 2012. Long short-term memory. Supervised sequence labelling with recurrent neural networks, pp. 37–45. https://doi.org/10.1007/978-3-642-24797-2_4.
He, K., Zhang, X., Ren, S., et al., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.48550/arXiv.1512.03385.
Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Netw. 4 (2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T.
Li, Y., Han, Y., 2017. Decline curve analysis for production forecasting based on machine learning. In: SPE Symposium: Production Enhancement and Cost Optimisation. https://doi.org/10.2118/189205-MS.
Li, Z., Zhou, F., Chen, F., et al., 2017. Meta-sgd: learning to learn quickly for few-shot learning. arXiv preprint. arXiv:1707.09835. https://doi.org/10.48550/arXiv.1707.09835.
Liu, Y., Wu, H., Wang, J., et al., 2022. Non-stationary transformers: exploring the stationarity in time series forecasting. Adv. Neural Inf. Process. Syst. 35, 9881–9893. https://doi.org/10.48550/arXiv.2205.14415.
Loshchilov, I., Hutter, F., 2017. SGDR: stochastic gradient descent with warm restarts. arXiv preprint. arXiv:1608.03983. https://arxiv.org/abs/1608.03983.
Mishra, N., Rohaninejad, M., Chen, X., et al., 2018. A simple neural attentive meta-learner. arXiv preprint. arXiv:1707.03141. https://arxiv.org/abs/1707.03141.
Nichol, A., Achiam, J., Schulman, J., 2018. On first-order meta-learning algorithms. arXiv preprint. arXiv:1803.02999. https://arxiv.org/abs/1803.02999.
Nichol, A., Schulman, J., 2018. Reptile: a scalable metalearning algorithm, 2 (3), 4.
Nie, Y., Nguyen, N.H., Sinthong, P., et al., 2023. A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint. arXiv:2211.14730. https://doi.org/10.48550/arXiv.2211.14730.
Pan, S., Yang, B., Wang, S., et al., 2023. Oil well production prediction based on cnn-lstm model with self-attention mechanism. Energy 284, 128701. https://doi.org/10.1016/j.energy.2023.128701.
Ramachandran, P., Parmar, N., Vaswani, A., et al., 2019. Stand-alone self-attention in vision models. Adv. Neural Inf. Process. Syst. 32. https://doi.org/10.48550/arXiv.1906.05909.
Sun, J., Ma, X., Kazi, M., 2018. Comparison of decline curve analysis DCA with recursive neural networks RNN for production forecast of multiple wells. In: SPE Western Regional Meeting. https://doi.org/10.2118/189205-MS.
Vaswani, A., Shazeer, N., Parmar, N., et al., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30. https://doi.org/10.48550/arXiv.1706.03762.
Vinyals, O., Blundell, C., Lillicrap, T., et al., 2016. Matching networks for one shot learning. Adv. Neural Inf. Process. Syst. 29. https://doi.org/10.48550/arXiv.1606.04080.
Vyas, A., Datta-Gupta, A., Mishra, S., 2017. Modeling early time rate decline in unconventional reservoirs using machine learning techniques. In: Abu Dhabi International Petroleum Exhibition and Conference. https://doi.org/10.2118/188231-MS.
Wang, H.C., Zhang, K., Chen, N., et al., 2024. Better use of experience from other reservoirs for accurate production forecasting by learn-to-learn method. Pet. Sci. 21 (1), 716–728. https://doi.org/10.1016/j.petsci.2023.04.015.
Wu, H., Xu, J., Wang, J., et al., 2021. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. Adv. Neural Inf. Process. Syst. 34, 22419–22430. https://openreview.net/forum?id=J4gRj6d5Qm.
Xu, Z., Yu, G., 2024. A time series forecasting approach based on meta-learning for petroleum production under few-shot samples. Energies 17 (8), 1947. https://doi.org/10.3390/en17081947.
Zhang, R.H., Zhang, L.H., Luo, J.X., 2016. Numerical simulation of water flooding in natural fractured reservoirs based on control volume finite element method. J. Petrol. Sci. Eng. 146, 1211–1225. https://doi.org/10.1016/j.petrol.2016.08.024.
Zhang, Y., Yan, J., 2023. Crossformer: transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: The Eleventh International Conference on Learning Representations. https://openreview.net/forum?id=vSVLM2j9eie.
Zhou, H., Zhang, S., Peng, J., et al., 2021. Informer: beyond efficient transformer for long sequence time-series forecasting. Proc. AAAI Conf. Artif. Intell. 35 (12), 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325.