

Time-Series Forecasting of Organizational Cash Flow using an LSTM Model

Hemangi Jha
Department of Artificial
Intelligence & Data Science
Thakur College of
Engineering & Technology,
Mumbai, India

Yashashri Mahendrakar
Department of Artificial Intelligence &
Data Science
Thakur College of Engineering &
Technology,
Mumbai, India

Eshwari Pawade
Department of Artificial
Intelligence & Data Science
Thakur College of Engineering
& Technology,
Mumbai, India

Pratibha Sajwan
Department of Artificial
Intelligence & Data Science
Thakur College of Engineering &
Technology,
Mumbai, India

Abstract - Companies require short-term cash flow projections which may be supported by the accounting evidence of past transactions and reconstructed at any point in time subsequent to their creation. This study outlines a method (net cash flows next month) to produce such forecasts and provides a reference framework centered upon this requirement. The forecasting method presented in this study utilizes 6 months of historical company data and a present-day financial picture. The forecasting model consists of three layers. First, an LSTM layer will encode the five monthly variables. Second, a dense layer will encode the ten static variables. Third, the two encoded variables will be fused together in the final layer to provide a single estimated value representing the companies future projected net cash flow. Finally, the estimated value will be combined at runtime using mean net cash flow over the input window. The model was trained on 4851 company sequences and tested against 916 samples that were withheld based on the prediction month. The validation mean absolute errors were: neural model = 33.5069; six month persistence baseline = 40.5298; blended forecast = 32.3725. The blending method used produced an average reduction in error of 20.1% compared to persistence and 3.4% compared to the neural model. A supporting platform has been created to connect ERPNext and file-based accounting data to a canonical transaction model, asynchronous monthly aggregation, versioned inference, persisted forecast lineage, bounded scenario analysis, and finance dashboard. This work fills a practical gap between experimental modeling and actual organizational utilization. It demonstrates how all features of the forecasting model (feature contract, scalars, weights, currency assumptions, forecast inputs, and changes to scenarios), can be maintained as links through a singular implementation. Therefore, the total contribution of this research is an evaluated forecasting model and a replicable pathway for implementing said model as decision support.

Keywords - cash flow forecasting, LSTM, forecast combination, accounting data, decision support, machine learning systems, reproducibility

I. INTRODUCTION

A. 1.1 Organizational motivation

Although an organization may show sufficient income for reporting purposes; nevertheless, this does not exclude the possibility of a short-term cash deficit. In fact, customer payments will probably take longer to receive than anticipated; suppliers' payments will likely be received at once (in one month); and therefore, it follows that expenditure will also occur in conjunction with the obligation to repay debts.

As these timing issues cannot be detected from static accounts or even annual plans; a short horizon view of the financial department is required in this filed to avoid.

A forecast's output is not just a figure. Therefore, when a user considers an alternative assumption, the system should store the previous forecast and indicate the change separately. Therefore, there is a connection between forecasting research and data engineering, software design, and finance-facing representation.

There is a major barrier between most model studies conducted off-line and most finance dashboards that assume that predictions already exist in a trustworthy format. Preparation of features has been done in a training script and must be replicated at run-time. The source systems must be converted into the same Accounting definition that was utilized during training. Weights of models must be transmitted with their respective scalars and ordering of features. Dashboards must distinguish between observed values and hypothetical values. This paper views these concerns as part of the forecasting issue.

B. Research gap

Studies relating to Accounting relate recognised earnings to future realised cash flows [27]. Studies on receivable modelling account for collections and timing of invoices [19, 20, 21]. Contemporary neural studies introduced attention mechanisms and hybrid architectures for financial time series [1, 8, 11, 12] - parallel to which some applied studies have integrated deep learning directly with ERP transactional data [6, 16]. The two streams explain why temporal and financial contexts can be predictive. However, few describe how one artefact trained for organizational cash flow forecasting is deployed within an accounting-data workflow with preserved lineage.

Machine learning system research addresses hidden technical debt and data validation within operational lifecycle [23, 24, 25]. Dashboard research addresses how analytical information is presented for review [26]. Although these studies contribute valuable engineering principles they do not define an end-to-end design for monthly organization cash flow forecasting. The current study unites the model and system questions with one implementation.

C. Research questions

The study is organized around three questions.

- RQ1 Does a neural model that combines ordered monthly activity with static company attributes improve next-month error on cash flow relative to a six-month persistence baseline?
- RQ2 Does a fixed combination of the neural and persistence forecasts reduce validation error relative to either component alone?
- RQ3 Can the promoted model be embedded in a reproducible Accounting-data platform that preserves the feature contract artifact identity forecast inputs and scenario lineage?

RQ1 & RQ2 were answered by chronological validations. RQ3 was answered through the implemented data, inference, persistence & user interface path. The evidence for RQ3 relates to traceability and reproducibility. Performance at enterprise scale would require a separate operational study.

D. Contributions

First contribution is a hybrid forecasting model combining temporal activity and static financial condition separately before fusion. The second contribution is stabilisation rule for transparency comparing the learned estimate with a persistence baseline retaining both in final forecast. Third contribution is artefact contract carrying feature order architecture scaling currency handling blend weights and support rules into runtime inference. Fourth contribution is a reference platform recording Accounting & model evidence behind forecasts & bounded scenarios.

E. Paper organization

Section 2 develops the theoretical basis for the target, temporal model, and forecast combination. Section 3 reviews related work. Section 4 formalizes the organizational prediction problem and derives the design requirements. Section 5 presents data preparation and methodology. Section 6 describes model development and evaluation. Section 7 presents the built platform while Sections 8 and 9 answer the research questions and discuss their implications. Section 10 identifies the next evaluation steps, and Section 11 concludes the paper.

II. THEORETICAL BACKGROUND

A. Timing & Context of Cash Flow Movements

Net cash flow provides an understanding of how much money moved within a certain time frame; whereas net income reflects the cash position at the end of the time frame. For month t , the target is

$$y_t = I_t - O_t \quad (1)$$

where I_t and O_t denote total inflow and outflow. The difference between earnings and cash flow realizations provides two separate indicators of future cash movements [27] since accruals allow for a distinction between the timing of recognition of sales/revenue and the timing of related cash receipt, as well as purchases/expense incurred versus the corresponding date of cash payment.

In addition, the degree of variance in an entity's current asset balances, accounts payable/current liabilities, leverage ratios, credit ratings and past due accounts/payments can influence the

interpretation of changes in the monthly inflows/outflows, which are accounted for in the model. However, the model needs to capture both perspectives because a similar monthly inflow trend may indicate a significantly different level of stability depending on the associated liability obligations

B. Sequence representation using LSTM

In sequence-based models, each observation contributes to the previous observation in terms of input. Traditional RNNs can lose historical data during training either due to vanishing gradients [17], where early weights dominate late weights causing loss of earlier values, or exploding gradients, which causes later weights to become too large relative to their contributions to previous outputs. Long Short-Term Memory (LSTM) networks address these issues via a set of gates to control what is kept and what is discarded as part of updating the hidden state. The input gate determines what information will be allowed into the cell; the forget gate determines what information will be discarded from memory; and the output gate determines what information will be made accessible from memory.

Each sequence used in this paper includes six monthly vectors, and each vector consists of five elements representing invoice amounts; total number of days delayed in receiving payment; outstanding loan repayments; gross inflows; and gross outflows. Although sequences in this paper are short (only six vectors), there is still meaningful ordering involved in distinguishing a consistent pattern from a recent upward/downward trend. Thus, the LSTM representation at the end of each sequence captures and summarizes the ordered patterns of the various vectors for predicting future months' cash flows. There are no dynamic characteristics in this paper that change over the six positions in each sequence.

To prevent repeating identical values for static attributes and blurring their effect, they are sent through a dense layer that learns a fixed-size representation. The temporal representation learned via the recurrent layers and the static representation learned via the dense layers are then fused together prior to generating the final scalar output. The way this fusion is arranged allows the network to learn interaction effects among these representations based on conditional relationships rather than requiring an architectural novelty in doing so.

C. Persistence as a forecasting baseline

The most straightforward type of model for providing a baseline for evaluating performance against is a persistence model. This model replicates performance by mimicking the last known performance metric (in this paper: MAPE) of a system.

$$\hat{y}_{t+1}^p = \frac{1}{6} \sum_{k=0}^5 (I_{t-k} - O_{t-k}) \quad (2)$$

This model lacks the ability to incorporate payment delays, credit status or non-linear relationships but those added inputs are only relevant if they improve upon a reasonable baseline. As such, persistence serves as both an evaluation baseline and runtime stabilization mechanism.

D. Forecast combination

Combining multiple individual forecasts can be beneficial when the components exhibit partially differing types of errors .

A neural model may represent complex relationships between variables but may perform poorly when faced with sparse or irregular inputs. The implemented estimate is

$$\hat{y}_{t+1} = 0.675\hat{y}_{t+1}^N + 0.325\hat{y}_{t+1}^P, \quad (3)$$

where $\hat{y}^N$ is the neural output. The weights belong to the promoted experiment and should be interpreted as artifact settings. On the other hand, persistence exhibits high stability but reacts slowly to structural changes. The weights provided are specific to the experimental design being reported upon herein and should be viewed solely as design choices for artifacts created specifically for this research project. These weights are not intended to be generalized for cash flow forecasting purposes.

Additionally, the code checks for normalized support at runtime. When any input feature is found to have an absolute normalized value greater than 200, the neural contribution is zeroed-out and persistence is used. This protection mechanism ensures that a single extreme transformed variable does not unduly influence the learned path. This is a deterministic support rule and not a confidence interval.

E. Time-ordered evaluation

A model should only be trained on information available up until the start of the evaluation time-frame for that particular model. Randomizing later and earlier labeled months can create a false impression regarding the availability of information for any given model. Cross-validation is valid for time-series predictions when applied appropriately with regard to dependency structures and predictive scenarios [22]. The chronological holdout method employed in this experiment matches exactly with our stated next-month. Mean absolute error is the primary measure :

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4)$$

Mean Absolute Error (MAE) is used primarily. MAE retains units and supports comparisons across all three models evaluated on identical samples.

F. Artifact-controlled model behavior

Model behavior is influenced by many factors beyond merely the weight file. Input feature order, dimensionality, scaling, target transformations, currency normalization, architecture and blending parameters all contribute to model output. Technical debt accrues in machine-learning systems as these dependencies remain unmanaged [25]. Data validation must occur prior to feeding data into any pipeline [24]. One practical solution to managing dependencies is to view all these values as one versioned inference contract and place within a larger machine-learning operations lifecycle [23].

The artifact contract also facilitates performing scenario analyses. A scenario can load the exact model utilized by a baseline; recreate the model's stored input window; apply one permitted modification; and execute the same inference pathway. This approach turns output into a traceable model response rather than an isolated calculator result.

III. LITERATURE SURVEY

A. Organizational Accounting Predictors of Future Flow

Cash flow can be modeled based upon historical behavior of an organization. Current earnings and prior cash flow are two major sources of predictive power regarding future cash movements. A number of researchers have developed nonlinear dynamic models (e.g., [18]) to simulate cash flow at the firm level; other research has focused specifically on the financial reserve and flow variables unique to each organization (e.g., [14]).

Researchers utilizing supervised machine learning have successfully employed their approaches to predict both corporate cash holdings [14], and the growth rate in free cash flow [10]; similarly, they were able to develop forecasts to assist bank management with optimizing their cash flows [7]. Such studies provide rationale for incorporating both the organization's overall financial status and recent transactions history when developing a cash forecast.

B. Receivables and payment behavior

Research related to the topic of receivables examines payments as an unfolding process across time. Specifically, researchers utilized machine learning [19] and other similar algorithmic approaches [20] to develop predictive models to forecast account receivables. Additionally, researchers utilized discrete survival methods to model invoice level payment timing in supply chain settings [21]. Collectively, these studies demonstrate that the timing of collections represents a key source of uncertainty regarding cash availability.

Recently, some researchers have connected these predictive targets with existing Enterprise Resource Planning (ERP) systems. For example, deep learning models currently utilize real-time ERP transaction data and/or economic indicators to generate forecasts of an organization's future cash flow [6]; while others have demonstrated successful application of similar predictive models employing ERP data collected from mid-market manufacturing companies [16]. Collectively, such integration studies support the development of a system that extracts temporal sequences of data from existing accounting records.

C. Statistical & Neural Forecasting Models

Comparative evaluations of neural methods require credible baselines. In this regard, researchers recently conducted comparative evaluations of multilayer perceptron (MLP), long short-term memory (LSTM), ARIMA and Prophet models for forecasting account-receivable cash flow [3]. Standard LSTM networks [17] effectively encode temporal order; however, many researchers conducting financial time series modeling currently rely on hybrid architectures.

Researchers who combined LSTM and transformer blocks improved the robustness of hybrid models in financial forecasting [1, 12]. In addition, researchers found that the inclusion of attention mechanisms allow deep temporal networks to monitor changes in evolving features [8] and

significantly enhanced the quality of standard LSTM forecasts [11]. As such, the shift toward deep learning has resulted in a range of transformer-based architectures being developed for applications including enterprise sales forecasting [15] and generalized reasoning for financial technical analysis [4, 13]. Consistent with the broader deep learning trend, the present study employs a single standard LSTM branch to encode temporal data; this simplifies the required architecture to facilitate subsequent platform engineering.

D. Forecast combination and validation

Each forecasting model should be validated against an appropriate information sequence which reflects its anticipated usage. Bergmeier et al. discussed the validity constraints for applying cross-validation to evaluate autoregressive time series [22]. Financial prediction models are typically subject to the evaluation conditions presented by commercial marketplaces; therefore, additional methodologies may need to be implemented to replicate the complexities of real world markets [9]. Another method for evaluating financial prediction models includes conformal prediction which defines predictive intervals around change points [5]. In contrast to traditional cross-validation, the chronological hold-out split utilized in this study preserves the original ordering of label months allowing for direct comparisons of in sample versus out-of-sample performance.

E. Machine learning systems and decision support

Platform literature suggests that for an offline model to be useful it must possess well-defined operational boundaries. The authors of Sculley et al. identified what they refer to as "hidden technical debt" caused by the entangled data dependency relationships inherent in many data sets and the lack of version control associated with many features in machine learning systems [25]. Similarly, Breck et al. identified the necessary criteria for validating data during entry into machine learning pipelines [24]. Additionally, Kreuzberger et al. provided a holistic framework for addressing the operational and validation issues inherent in the MLOps lifecycle [23]. Consistent with this body of work, the reference platform described in this study enforces one canonical accounting contract and utilizes a versioned artifact representation of all promoted models.

Once a forecast is produced, its presentation will influence how organizations employ it. Yigitbasioglu and Velcu reviewed research related to designing performance dashboards which emphasized information load, flexible presentation options, and drill-down functionality [26]. Therefore, consistent with this emphasis on providing users with flexibility when interacting with performance metrics, this study proposes a workflow that commences with the current base line forecast and then presents users with supporting evidence and bounded scenario options as needed.

F. Position of this study

There are numerous examples of research related to integrating ERP systems, predicting timing of invoice

payments, modeling sequential data and machine learning operations. However, few studies examine how these concerns come together as part of a single organizational cash flow implementation. Therefore, this study represents the intersection point of those various streams of research. It examined one hybrid forecast model and transmitted that artifact through a transparent reference platform. Unlike previous studies that proposed new attention mechanisms or novel temporal architectures, this study's contributions lie in the design path that ties together these concerns and provides evidence in support thereof.

IV. PROBLEM FORMULATION AND DESIGN REQUIREMENTS

A. Operational setting

A company generates invoices, receives payments, makes journal entries, and takes periodic snapshots of their financials. After each month (t), the system must generate a prediction for net cash flow for the subsequent month ($t+1$). Although the company maintains records in their Enterprise Resource Planning System (ERP) or provides a standard format CSV/XLSX file, the finance user reviews the predicted net cash flow generated using these records against a selected buffer.

The finance user may also test the impact of a specific range change in one or more inputs. There are two connected outputs from the problem. The first is a numeric estimate of net cash flow for the subsequent month. The second is an explanation of which period aggregates, static snapshot, model artifacts, and scenario patches were used to determine the estimated value.

B. Prediction formulation

For company j , let $X_{j,t}^T$ be the six-by-five temporal matrix ending at month t . Let $X_{j,t}^S$ be the ten-value static vector available for the forecast date. The neural model estimates

$$\hat{y}_{j,t+1}^N = f_{\theta}(X_{j,t}^T, X_{j,t}^S), \quad (5)$$

and the runtime combines it with the six-month persistence estimate. Each sample must use only months before its label. The static snapshot must be eligible by the forecast date.

C. Design requirements

The research questions imply five system requirements. First, every source must map to one downstream accounting meaning. Second, training and serving must use the same feature names, order, dimensions, and scaling. Third, a forecast must retain links to its six monthly inputs, static snapshot, and artifact version. Fourth, scenarios must preserve the baseline and store their changes separately. Fifth, the user interface must distinguish observed records, predictions, and hypothetical outputs.

These requirements shape the architecture in Figure 1. The model stream ends with a versioned artifact. The platform stream begins with source translation and monthly aggregation, then invokes the artifact through one shared inference service. The handoff is explicit because the platform cannot reproduce the experiment from weights alone.

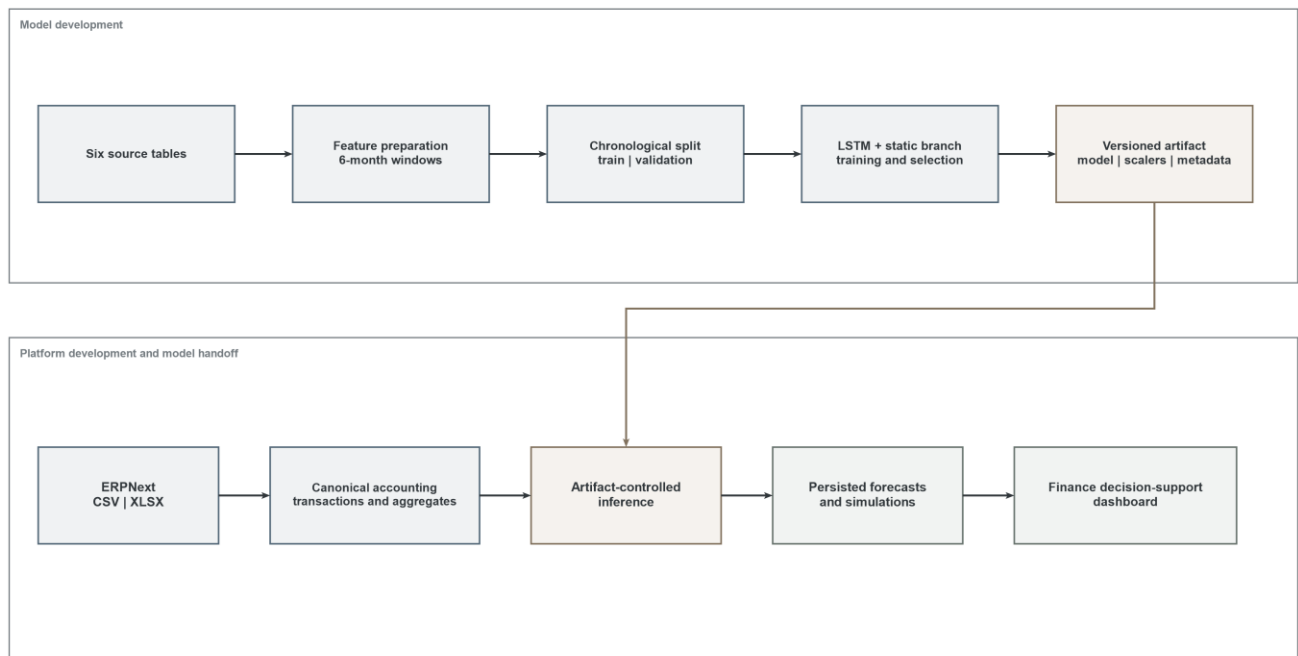


Figure 1. Model and platform development methodology. The artifact carries the training assumptions into runtime inference

D. Decision-support boundary

The platform forecasts net movements and assesses potential input changes. However, the platform does not perform transactions such as payments, collections, borrowings, investments, etc. A scenario presents what the model predicts after applying a limited patch. However, the platform does not assert whether or not the company can create the change or if the proposed change will lead to the predicted outcome. This boundary maintains model-based evidence as useful but prevents it from being automatically translated into a financial command.

V. DATA PREPARATION AND METHODOLOGY

A. Research design

The study utilizes a build-and-evaluate research design methodology. The model development work involves preparing company-month data; constructing ordered samples; training a hybrid network; evaluating multiple versions of the forecast; and creating one artifact. The platform development work involves implementing the necessary source-to-predictive path to utilize that artifact. Accuracy of model estimates addresses RQ1 & RQ2. Evidence regarding repository and runtime utilization address RQ3.

B. Source data

Six SME tables utilized in the experimental evaluation are inherited from previous applied research work. Each table's size and role are summarized in Table 1. The account-receivable table contains the most complete time-series information. Attributes within business, rating, card, credit-account and loan tables contribute to both financial and repayment contexts essential to define inputs.

Table 1. Source tables used for model development

Table	Rows	Role in preparation
Account receivable	79,648	Invoice values and payment timing
Businesses	974	Company financial attributes
Credit account history	1,421	Fallback inflow and outflow relationship
Credit card history	353	Missed-payment indicators
Credit rating	900	Credit, failure, and leverage measures
Loan	90	Repayment history

Company registration number links the tables. Credit score is normalized from a source range of 0 to 1000, and failure score is normalized from 0 to 100. Both enter the model on the runtime 0 to 1 convention.

C. Monthly aggregation

Receivables from companies are aggregated on a per-company and per-month basis in order to derive invoice total and accumulation of payment delay days. Loan records serve as repayment. The initial inflow consists of invoice total while initial outflow consists of repayment. If there is no repayment derived outflow for a particular month, a historical default payout-to-paying ratio for credit accounts serves as documented fallback.

Historical credit-account information lacks monthly timestamps therefore its company totals are not replicated across every month. Static context attributes from business, rating, and card tables are joined together as context attributes. Missing static values are populated with zeros according to a predefined rule. As a result of this preparatory phase, there were 10,593 company-month rows available for analysis.

D. Sample construction

For each eligible company, months $t - 6$ through $t - 1$ form a 6×5 temporal tensor and month t supplies the label. The same

company's ten-value static vector accompanies the window. This process produced 4,851 supervised samples.

The temporal features are invoice amount, payment-delay days, repayment, inflow, and outflow. The static features are capital expenditure, cost of goods sold, current assets, current liabilities, fixed assets, long-term liabilities, credit score, failure score, debt-to-revenue ratio, and missed-payment count. Artifact metadata preserves this ordering.

E. Chronological partition

Training samples consist of those prior to the 80th percentile when labeled by sample date. Validation samples comprise later samples. This resulted in 3,935 training samples and 916 validation samples. Temporal and static scalers are fit only on training data. Every sample is constructed to conclude prior to its corresponding label month and utilizes random seed 32. This design gives a held-out later period and avoids direct leakage from future label months into training. The validation partition also selects the checkpoint, so the reported result is model-development evidence for the promoted baseline.

F. Evaluation measures

The experiment compares the neural output, six-month persistence, and their fixed blend on the same validation

samples. Relative improvement of the blend over comparator c is

$$100 \times \frac{\text{MAE}_c - \text{MAE}_{\text{blend}}}{\text{MAE}_c}. \quad (6)$$

The run also records how many validation samples cross the runtime support guard. Platform evidence is assessed by tracing an input from source adapter through canonical persistence, aggregation, artifact loading, forecast storage, scenario storage, API response, and dashboard presentation.

VI. MODEL DEVELOPMENT AND EXPERIMENTAL SETUP

A. Hybrid architecture

The temporal branch passes six month vectors through an lstm with a hidden width of 96. The last hidden representation goes to a dense layer of width 48. The static branch maps the ten financial attributes to another width-48 representation. All dense blocks utilize ReLU and dropout. The fusion layer which has a width of 48 takes the concatenated output of the two layers and produces a linear scalar output.

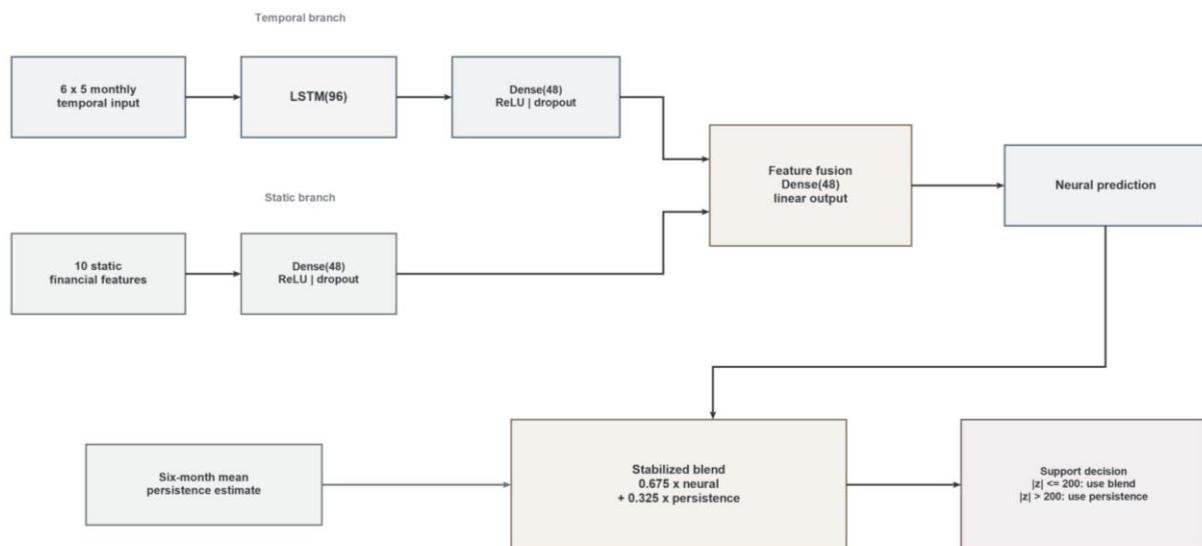


Figure 2. Hybrid model and stabilized inference. The support decision is deterministic, it is not an uncertainty interval.

The temporal and static branches were constructed as per the problem statement. The lstm learned the changes over time for the six months, while the static branch represented the company's financial setting. The fusion occurs after both representations had the same dimensionality (i.e., width). This maintained visibility of the source of each representation within the artifact's meta data as well as within the architecture itself.

B. Training procedure

Training utilized Smooth L1 loss with beta = 1.0 and Adam optimization. The base learning rate was 0.0001; weight decay was 0.001; and there was a 40% drop out probability. Gradients were clipped at 1.0. A plateau scheduler would reduce the learning rate if validation progress stalled. There was a limit of

50 epochs; Training would stop after five epochs without improvement

The checkpoint with the lowest validation MAE was retained. It occurred at epoch 16. Target scaling was disabled. Evaluation of neural output in units of target is done prior to combination with persistence

C. Compared forecasts

The first forecast - neural output. The second forecast - recent six-month mean. The third forecast - weighted neurals of 0.675 and 0.325 persistence weights. Comparison of three answers regarding whether or not learning improves upon recent scale and whether or not stabilization adds value following the availability of the neural estimate.

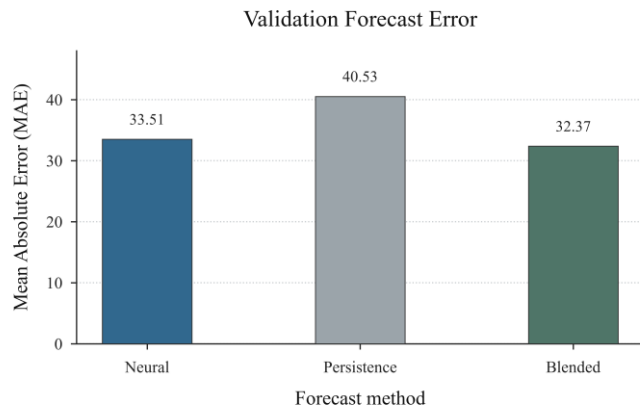


Figure 3. MAE on 916 chronologically held-out samples. Lower is better. Values are in target unit of research dataset.

D. Artifact promotion

Promoted run contains: Model.pth, two fitted scalers, metrics.json, Artifact_metadata.json and a configuration snapshot. Metadata version 3 records feature names, ordering, dimensions, sequence length, architecture, Target-scaling mode, blend settings, support guard, seed, currency used for Training, conversion constants, and artifact filenames.

Both paths (Training and serving) instantiate the same model class from shared machine learning package. Loader

validates Metadata before loading weights; thus, promoted run becomes reproducible inference unit

E. Runtime preparation

Monetary inputs in runtime are converted to reference unit of artifact GBP before scaling. The demonstration mapping for INR divides monetary values by 100 and reverses the conversion after prediction. Ratios, scores, counts, delay values are not converted. Factor is stored with artifact to prevent input being interpreted under unstated currency rule

After scaling service evaluates support guard, neural path, persistence path, blend. Both baseline and scenario requests call this same service; result always carries versions of model and artifact used.

VII. PLATFORM DEVELOPMENT

A. System Architecture derived from the requirements

The platform consists of three different applications (api, worker & dashboard) while being a single application (a modular monolith). Packages common to both the api and worker provide domain rules, schemas, persistence, integration adapters for external systems, event broker behavior & machine learning primitives. The system of record is postgresql. Ingestion work from the api is carried over Redis Streams to the worker.

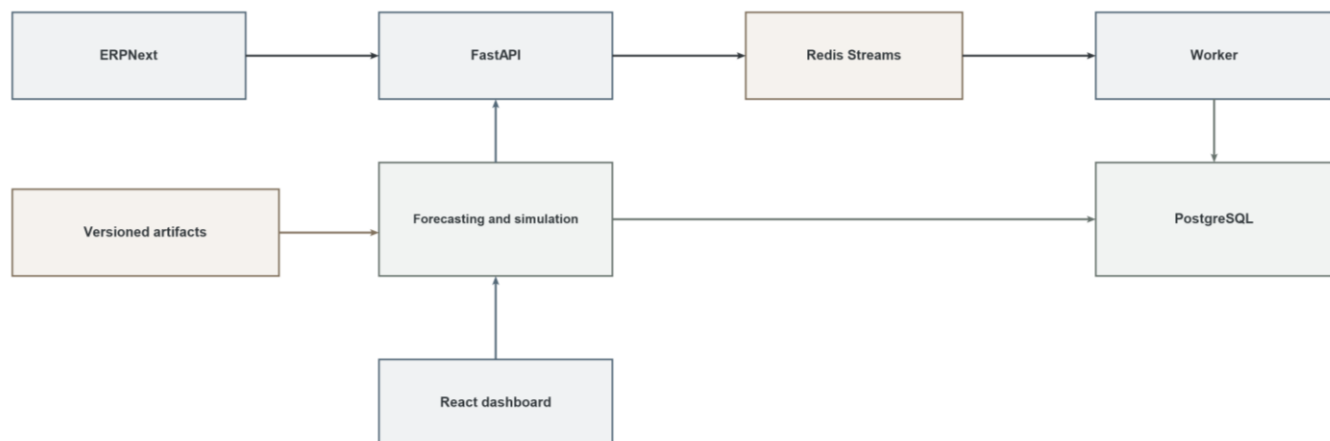


Figure 4. Runtime architecture linking accounting sources, artifact-controlled inference, durable state & dashboard.

The boundaries were formed based on specific requirements from Section 4. Fields representing source-specific information end in the integration package. All asynchronous work ends in canonical transactions & aggregates. Only the canonical model & versioned artifact are consumed by Forecast services. The dashboard reads persisted evidence as opposed to recomputing predictions in the browser.

B. Canonical accounting boundary

All adapters (ext, CSV, XLSX) produce the same transaction shape. It includes enterprise provenance, source provenance, type of transaction, due or settlement dates (if

applicable), positive amount, currency, direction of movement, status, counter party (if applicable), reference text, source record identifier, & payload hash.

The ERPNext adapter consumes sales invoices, purchase invoices, payment entries & journal entries via paginated REST calls. The file adapters validate headers, row shape, dates, amount, currency, enum values & counter-party fields. If there is no source identifier available, it is determined from the file checksum and row number. Forecast code never receives any data related to ERP fields or spreadsheet column choices.

C. Asynchronous ingestion and aggregation

The API writes an id for the ingestion run prior to writing it into Redis Streams. The worker finds the appropriate adapter and pulls down and translates records. The records are then written using the combination of source + source-record identifier as the idempotent boundary. The message is only acknowledged once the worker has successfully processed its routine.

For each month that is impacted by new ingestion activity, the worker will reload all canonical transactions associated with that time frame and rebuild the aggregate. Invoice amount, inflow, outflow, repayment, delay days, number of invoices & number of payments are all calculated during this process. Additionally, this pass will update counter-party balances and observed delays. Since the aggregate can be rebuilt from the stored ledger, an aggregate will remain consistent across multiple or incremental ingestions.

D. Forecast lineage

Any given forecast request will identify the enterprise, target period, run type (Forecast or simulation), and non-negative buffer. The service requires six complete months prior to the target period along with the latest eligible static snapshot. Any history missed or feature contract mismatch will stop the run with an explicit error.

PostgreSQL stores prediction results including buffer gap, run status, timestamps, model version, artifact version, static snapshot url, and six aggregate urls. These records provide answers on what evidence produced a result even after later transactions are ingested. Additionally they provide every scenario a stable baseline.

E. Bounded scenario analysis

Simulation begins from completed Forecast and reloads artifact recorded by that Forecast. The service rebuilds baseline inputs applies approved patch runs shared inference and stores prediction delta. Artifact mismatch is immediately rejected.

Analysis scenarios change selected static values under controlled profiles financial-health analyzes how changes to assumed delay will impact modeled cash flow. Liquidity mitigation evaluates bounded changes to capital expenditure, outflow or repayment when baseline falls below buffer. Each result remains separate from account ledger.

F. Finance workflow

The dashboard made by React presents the process as sequence. The overview establishes the current position. Forecast shows history, predicted movement, buffer status, observed drivers, provenance & run history. Health and Receivables expose sensitivity of models with the underlying evidence. Planning presents feasible options evaluated by model. Data manages sources, synchronization, uploads, and processing status.

TypeScript types used for frontend were generated from FastAPI's OpenAPI schema. Color was added to represent risk

states in addition to text on failure, loss, empty and retry states; values used for scenario display were labeled and separated from historical facts.

G. Reproducible deployment

Docker Compose starts PostgreSQL, Redis, migration scripts, API & worker processes along with deterministic seed processes dashboard. Optional profile adds ERPNext. Python & frontend dependencies locked. Database history provided through Alembic revision names.

Deterministic fixture provides February 2026 through July 2026 as model history and August 2026 as forecast target. Forecast & scenarios created using artifact-controlled path same as production application. It demonstrates completed workflow ensures local reproduction repeatable

VIII. RESULTS AND ANALYSIS

A. RQ1 Neural model against persistence

Table 2 presents the comparison. The neural model achieved MAE of 33.5069 relative to 40.5298 for persistence. The neural path because of this surpassed the naive recent-average prediction by 17.3 percent, which speaks to RQ1 on the observed validation partition. The ordered monthly activity coupled with static corporate context adds value.

Table 2. Held-out chronological validation results

Method	Validation MAE	Improvement obtained by blend
Neural model	33.5069	3.4%
Persistence baseline	40.5298	20.1%
Blended forecast	32.3725	Reference

The result does not imply that every neural architecture would improve on persistence. It establishes that the promoted feature preparation, model, and checkpoint do so on the same later-period samples.

B. RQ2 Effect of forecast combination

The blended forecast had the lowest MAE of 32.3725. It was a 20.1 percent improvement over persistence and 3.4 percent better than the neural model. The larger improvement over persistence indicates the learned path adds a lot of value. The smaller improvement over the neural output indicates that recent mean cash flow contributes much still benefit for stabilization.

No validation sample crossed the support guard. The reported blended metric therefore reflects the configured neural-persistence combination for all 916 samples rather than fallback cases.

C. RQ3 Training-to-platform continuity

RQ3 asks whether the model can retain its contract and evidence path in an implemented platform.

Table 3. Platform implementation evidence

Requirement	Implemented evidence
One downstream accounting meaning	Canonical transaction and monthly aggregate contracts
Training-serving consistency	Shared model class, metadata validation, committed scalars and weights
Forecast traceability	Artifact identifiers, static snapshot, and six aggregate links
Scenario separation	Persisted patches and outputs separate from source transactions
Finance-facing interpretation	Dashboard views backed by generated API contracts
Repeatable execution	Docker Compose, migrations, locked dependencies, and deterministic seed

The platform brings the traceability from source record to the forecast shown. Source identifiers make ingestion idempotent. Monthly aggregate links makes the time range. Artifact identifiers makes the model contract. Scenario records makes, which value were changed. This is the answer to RQ3 at the level of reproducible implementation.

D. Relationship between the two contributions

The model and platform results support one another. The model result would be hard to operationalize if the meaning or order of runtime data changes between the two. The platform would have little value to analysis if its view forecast had no benchmark assessed. The artifact contract links them. It allows the validation result to specify the same preparation and model structure later used by the forecast and simulation services.

IX. DISCUSSION

A. Interpretation of the findings

The comparison shows how each layer contributes. Persistence captures recent scale. The hybrid network adds information from ordered activity and financial condition. Combination retains most of the learned advantage while reducing dependence on the neural estimate alone.

This pattern is consistent with recent hybrid approaches in financial time-series forecasting [1, 12]. It also supports the accounting argument that cash movement and company condition contain related but distinct information [27]. The model architecture mirrors that distinction through separate branches.

B. Comparison with related work

Closest neural analogue exists in Weytjens, Lohmann, and Kleinstuber, but a different measure is used daily accounts-receivable cash flow [3]. Saini et al. and Kureljusic and Metz forecast the timing of invoice payment [20, 21]. Pang et al. use a nonlinear panel specification to estimate firm cash flow [18]. As their datasets, horizons, and measures differ from this one, errors cannot be put into one ranking.

The two comparisons are illustrating a rather useful comparison about design. Once again, specific to each study, temporal organization and baseline choice are central. This work offers transient platform context with the persistence component, which is reported with the neural and blended

outputs. It carries the selected artifact into a platform, which is outside the scope of most forecasting comparisons.

C. Importance of the artifact contract

The artifact is the bridge between empirical evidence and software behavior. A weight file cannot tell the runtime which feature comes first, which scaler applies, how currency is handled, or whether a target transformation must be reversed. Metadata version 3 records those choices. The shared model class removes a second implementation of the architecture.

This design addresses a practical form of technical debt described by Sculley et al. [25]. It also applies the lifecycle reasoning found in machine learning operations research [23] without adopting infrastructure intended for much larger systems. The platform remains compact because the requirement is reproducible application, not continuous model retraining.

D. Accounting provenance as model evidence

Feature values are created by accounting rules. Transaction direction, invoice status, date selection, allocation, and aggregation all affect the model input. Treating ingestion as a separate implementation detail would hide those assumptions. The canonical boundary makes them part of the reported method.

Persisted input links also improve review. A finance user can inspect the periods behind a forecast and distinguish later ledger changes from the state used at prediction time. Scenario patches remain separate from source transactions, preserving the difference between observed and hypothetical evidence.

E. Role of the dashboard

The dashboard follows the evidence chain rather than exposing the model as a standalone tool. Users begin with position and buffer status, then move to drivers, receivables, and bounded scenarios. Model versions and run history remain available as supporting detail. This follows the dashboard literature's emphasis on information load and flexible presentation.

The interface also preserves the decision-support boundary. It presents recommendations for review and does not execute financial actions. This is especially important for scenario outputs, which describe model sensitivity rather than guaranteed business effects.

F. Practical application

The reference implementation is suitable for a controlled organizational pilot. A team can connect ERP Next or upload canonical files, observe ingestion status, create a forecast, compare it with a buffer, review its source periods, and evaluate bounded alternatives. The deterministic environment gives developers and reviewers the same initial state.

Deployment in a specific organization would begin with data mapping and historical backtesting in that context. The artifact can then be retained, replaced, or recalibrated through a controlled promotion process. The current architecture already records the version boundaries needed for that process

X. FUTURE EVALUATION AND EXTENSION

The next model study should repeat evaluation across several forecast origins. Rolling-origin results would show whether the neural and blended advantages remain stable as the cutoff moves through time. Reporting signed error, root mean squared error, and liquidity-shortfall error would add views that MAE does not provide. Industry and company-size breakdowns could show where the static branch contributes most.

Prediction intervals are another useful extension. Quantile loss, conformal calibration, or probabilistic sequence models could attach an empirically tested range to the point forecast. That work should remain separate from the present support guard, whose role is input protection.

The platform can be evaluated with replayed ingestion and forecast workloads. Useful measures include queue delay, end-to-end completion time, recovery after worker interruption, and parity between offline and API inference. A finance-user study could then examine whether provenance and scenario separation improve comprehension and review time. Future platform work can add artifact promotion and rollback records, stronger secret storage, access control, and monitoring. These extensions fit the present boundaries without changing the forecasting method or canonical accounting model.

XI. CONCLUSION

This paper presented a next-month organizational cash flow method and the reference platform needed to apply it with preserved evidence. The model joins a six-month LSTM branch with a ten-feature static branch. On 916 chronologically held-out samples, the neural model improved upon persistence, and the fixed blend produced the lowest MAE at 32.3725.

The platform turns that experiment into a reproducible workflow. Accounting sources converge on one transaction model, monthly aggregates form the input window, artifact metadata controls inference, forecast runs retain their source links, and scenarios record changes separately. The dashboard presents the resulting evidence as bounded finance decision support.

The study answers its three research questions with connected evidence. The hybrid model improves upon recent-mean persistence on the reported split. The forecast blend improves upon both components. The promoted artifact can be carried through an implemented data and application path without losing its feature contract or lineage. This joined result is the paper's main contribution.

REFERENCES

- [1] "LSTM-Transformer-Based Robust Hybrid Deep Learning Model for Financial Time Series Forecasting," IEEE Access / Academic Publication, 2023.
- [2] "ATM Cash Flow Prediction Using Local and Global Model Approaches in Cash Management Optimization," International Journal of Production Economics, 2022.
- [3] H. Weytjens, E. Lohmann, and M. Kleinstaub, "Cash Flow Prediction: MLP and LSTM compared to ARIMA and Prophet," Electronic Commerce Research, vol. 21, no. 2, pp. 371–391, 2021.
- [4] "Reasoning on Time-Series for Financial Technical Analysis," IEEE Transactions on Knowledge and Data Engineering, 2023.
- [5] "Conformal Prediction for Time-series Forecasting with Change Points," Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [6] "AI-Driven Cash Flow Forecasting in ERP Systems: Integrating Economic Indicators and Real-Time Transaction Data Using LSTM-Based Time-Series Models," Journal of Enterprise Information Management, 2023.
- [7] "Forecasting bank cash flows using intelligent systems," Expert Systems with Applications, 2021.
- [8] "Dynamic Forecasting and Temporal Feature Evolution of Stock Repurchases in Listed Companies Using Attention-Based Deep Temporal Networks," IEEE Transactions on Neural Networks and Learning Systems, 2023.
- [9] "FinTSBridge: A New Evaluation Suite for Real-World Financial Prediction with Advanced Time Series Models," ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024.
- [10] "Application Of Machine Learning Algorithms to Free Cash Flows Growth Rate Estimation," Journal of Corporate Finance, 2022.
- [11] "Time series prediction based on LSTM attention-LSTM model," IEEE Access, 2022.
- [12] "A Hybrid LSTM-Transformer Approach for Financial Markets: Forecasting Stock Price Time Series," Applied Soft Computing, 2023.
- [13] "Deep Learning for Financial Time Series Prediction: A State-of-the-Art Review of Standalone and Hybrid Models," Financial Innovation, vol. 9, article 15, 2023.
- [14] A. Özlem and S. Tan, "Predicting Cash Holdings Using Supervised Machine Learning Algorithms," Journal of Financial Research, 2022.
- [15] Y. Sun and X. Li, "A Transformer-Based Framework for Enterprise Sales Forecasting," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024.
- [16] "Predictive Cash Flow Forecasting Using Deep Learning and ERP Transaction Data in Mid-Market Manufacturing Firms," Computers & Industrial Engineering, 2023.
- [17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
- [18] Y. Pang, S. Shi, Y. Shi, and Y. Zhao, "A Nonlinear Dynamic Approach to Cash Flow Forecasting," Review of Quantitative Finance and Accounting, vol. 59, pp. 205–237, 2022.
- [19] A. P. Appel, G. L. Malfatti, R. L. de F. Cunha, B. Lima, and R. de Paula, "Predicting Account Receivables with Machine Learning," arXiv preprint arXiv:2008.07363, 2020.
- [20] M. Kureljusic and J. Metz, "The Applicability of Machine Learning Algorithms in Accounts Receivables Management," Journal of Applied Accounting Research, vol. 24, no. 4, pp. 769–786, 2023.
- [21] S. Saini, G. Manai, W. van den Boom, et al., "Invoice Level Forecasting with Discrete Survival Methods for Effective Forecasting of Account Receivables in Supply Chain," Discover Analytics, vol. 2, article 5, 2024.
- [22] C. Bergmeir, R. J. Hyndman, and B. Koo, "A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction," Computational Statistics & Data Analysis, vol. 120, pp. 70–83, 2018.
- [23] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023.
- [24] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang, and S. Roy, "Data Validation for Machine Learning," in Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA, 2019.
- [25] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," in Advances in Neural Information Processing Systems 28 (NeurIPS), 2015, pp. 2503–2511.
- [26] O. M. Yigitbasioglu and O. Velcu, "A Review of Dashboards in Performance Management: Implications for Design and Research," International Journal of Accounting Information Systems, vol. 13, no. 1, pp. 41–59, 2012.
- [27] R. Ball and V. V. Nikolaev, "On Earnings and Cash Flows as Predictors of Future Cash Flows," Journal of Accounting and Economics, vol. 73, no. 1, article 101430, 2022.