water

MDPI

Article

Comparative Analysis of Snowmelt-Driven Streamflow Forecasting Using Machine Learning Techniques

Ukesh Thapa [1,*], Bipun Man Pati [1], Samit Thapa [2], Dhiraj Pyakurel [3] and Anup Shrestha [4]

1 AI Research Center, Advanced College of Engineering and Management, Kathmandu 44600, Nepal; bipunmanpati@acem.edu.np
2 Department of Civil Engineering, Advanced College of Engineering and Management, Kathmandu 44600, Nepal; samit.thapa@acem.edu.np
3 Department of Electronics and Computer Engineering, Advanced College of Engineering and Management, Kathmandu 44600, Nepal; dhiraj@acem.edu.np
4 Department of Electronics and Computer Engineering, National College of Engineering, Lalitpur 44700, Nepal; anup@nce.edu.np
* Correspondence: ukesh.thapa@acem.edu.np

Abstract: The rapid advancement of machine learning techniques has led to their widespread application in various domains, including water resources. However, snowmelt modeling remains an area that has not been extensively explored. In this study, we propose a state-of-the-art (SOTA) deep learning sequential model, leveraging a Temporal Convolutional Network (TCN), for snowmelt forecasting of the Hindu Kush Himalayan (HKH) region. To evaluate the performance of our proposed model, we conducted a comparative analysis with other popular models, including Support Vector Regression (SVR), Long Short-Term Memory (LSTM), and Transformer models. Furthermore, nested cross-validation (CV) was used with five outer folds and three inner folds, and hyperparameter tuning was performed on the inner folds. To evaluate the performance of the model, the Mean Absolute Error (MAE), Root-Mean-Square Error (RMSE), R square (R^2), Kling–Gupta Efficiency (KGE), and Nash–Sutcliffe Efficiency (NSE) were computed for each outer fold. The average metrics revealed that the TCN outperformed the other models, with an average MAE of 0.011, RMSE of 0.023, R^2 of 0.991, KGE of 0.992, and NSE of 0.991 for one-day forecasts of streamflow. The findings of this study demonstrate the effectiveness of the proposed deep learning model as compared to traditional machine learning approaches for snowmelt-driven streamflow forecasting. Moreover, the superior performance of this TCN highlights its potential as a promising deep learning model for similar hydrological applications.

Keywords: support vector regression (SVR); long short-term memory (LSTM); transformer; temporal convolutional network (TCN); nested cross-validation; snowmelt

check for updates

Citation: Thapa, U.; Pati, B.M.; Thapa, S.; Pyakurel, D.; Shrestha, A. Comparative Analysis of Snowmelt-Driven Streamflow Forecasting Using Machine Learning Techniques. *Water* **2024**, *16*, 2095. https://doi.org/10.3390/w16152095

Academic Editor: Jueyi Sui

Received: 28 May 2024
Revised: 20 June 2024
Accepted: 25 June 2024
Published: 25 July 2024

1. Introduction

Snowmelt is the process by which snow or ice on the ground or various surfaces changes into water due to the temperature rising. Snowmelt plays a pivotal role in the environment and development of human society. Snowmelt is also a major source of fresh water in the world. Figure 1 shows that Central Asia contains a large amount of ice in the Hindu Kush Himalayan (HKH) region and is highly sensitive to global climate change, with it having faced significant warming ($0.21 \pm 0.08°$/decade) [1] over the past few decades. In a study by Pandey et al. [1], a dynamic threshold-based method was applied to enhance QuickSCAT ku-band backscatter observations from 2000 to 2008, wherein the average melt duration was calculated and a longer melt season (∼5 weeks) was found to occur in the Eastern Himalayan region relative to the Central and Western Himalayas and the Karakoram region. The river basin sourced from the HKH region provides the habitat's vital supplies [2]. The seasonal snowmelt process significantly affects the ecosystem, water availability, agriculture, and water resources. Additionally, the snowmelt directly influences the river flow, aquatic habitats, and the functioning of the hydroelectric power system. The geographical conditions and extreme weather in this region make it difficult to gather adequate information to develop an optimal strategy for the sustainable utilization of its water resources. Therefore, precise forecasting of the snowmelt runoff is essential in this area for efficient planning and management.

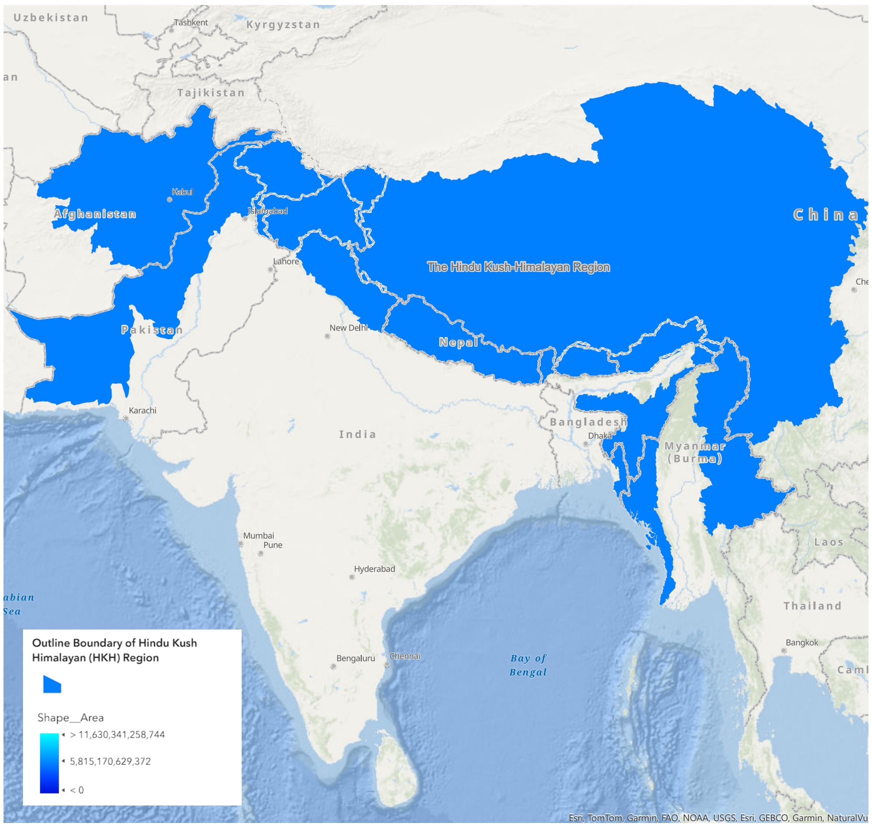

Figure 1. Map of the Himalayan Hindu Kush region.

The prediction of snowmelt runoff can be carried out through various approaches, which are classified into three different models, namely Energy-Based (EB) models, Temperature Index (TI) models, and Data-Driven (DD) models. EB models are computationally heavy models that require a huge amount of data for forecasting as well as a deep understanding of thermodynamics. This type of model cannot provide optimal predictions in data-scarce regions [3]. TI models are often used for forecasting, as they use air temperature data to estimate the energy that melts the snow [4]. Moreover, TI models can not outperform EB models, and calibration should be conducted to find the suitable parameters, which is tedious work [5]. DD models include machine learning (ML) models, which have the capability to learn complex data distributions with high precision for forecasting without prior knowledge of the process. However, the application of ML models in snowmelt runoff prediction is still limited.

ML has diverse applications across interdisciplinary fields, showcasing its versatility and impact. The application of ML in the domain of water resources has been well investigated in previous studies [6,7]. However, traditional ML models such as Support Vector Regression (SVR), Decision Tree (DT), and other regression models are used in practice for snowmelt runoff prediction. In one study [8], snowmelt forecasting of the Italian Alps area was carried out using SVR where the Snow Cover Area (SCA), antecedent discharge (Q), and meteorological data such as the temperature (T), precipitation (P), humidity of the atmosphere, and weather conditions were taken as the features, which outperformed a simple linear auto-regressive model with an average relative Root-Mean-Square Error

(RMSE) of 33% on the tested samples. Another study, by De Gregorio et al. [9], proposed an SVR model for real-time river flow prediction, where the SVR model performed better than the average of the previous 10 years on a single gauging station, with a mean improvement of about 48% in the RMSE. Artificial Neural Networks (ANNs) are also used for various applications as they can handle non-linear data better than traditional ML models. In the study in [10], the authors used an ANN for snowmelt runoff prediction, where SCA, Q, T, and P were used as inputs to the model, and research verified that this ANN model outperformed any TI model as its Nash–Sutcliffe Efficiency (NSE) increased from 0.51 to 0.71 in forecasting. Similarly, in [11], the authors conducted a comparative analysis of Random Forest (RF), SVR, and Multi-Layered Perceptron (MLP) models for the forecasting of soil moisture levels on a volumetric basis, where the MLP model showed the best performance, with a coefficient of determination (R^2) value of 0.957.

In past research studies, traditional ANNs were used for the time-series problem, but this was not sufficient for capturing all of the necessary or complex information from the data. So, the Recurrent Neural Network (RNN) was introduced to capture temporal information with the help of memory or a feedback loop, although it has its own problems like the problems of vanishing and exploding gradients [12]. The advancement of technology has led to the development of more complicated architectures. Accordingly, Long Short-Term Memory (LSTM) was introduced to overcome the problems of RNNs and capture temporal information with better precision. In the study in [13], two-layered LSTM was used for rainfall-runoff modeling with an NSE of 0.63, and the authors argued that the associativity between winter precipitation and spring runoff can be replicated by learning the process of snowmelt, but they did not investigate the influence of hyperparameter tuning on model performance. Similarly, an LSTM model proposed in another study was able to obtain an average RMSE of 150 m^3/s with an NSE above 99% while forecasting Q for one day [14]. Additionally, in [14,15], it was claimed that the window size in forecasting problems is an important hyperparameter to be tuned for achieving the best model performance, but in both studies, other hyperparameters, such as the number of LSTM layers and optimizers, were not observed. In a recent study, Thapa et al. [16] compared snowmelt prediction across different models by hypertuning parameters with the hit-and-trial method and setting the window size for prediction. Also, gamma testing was performed, where SCA, T, and Q were chosen as the inputs for the different models, out of which the LSTM model had the best performance with 0.997, 0.112, 0.173, 0.99, and 0.995 as its R^2, Mean Absolute Error (MAE), RMSE, Kling–Gupta Efficiency (KGE), and NSE, respectively. This study did not evaluate the performance of the model using nested cross-validation (CV), and the created state-of-the-art (SOTA) model only performed better for the specified dataset, not for any unseen dataset. In most studies, we found that RNN and LSTM models are commonly used for time-series forecasting. However, Temporal Convolutional Neural Networks (TCNs), which have outperformed other sequential models on publicly available datasets [17], have not yet been used in snowmelt forecasting. So, the main contributions of this article are summarized as follows:

1. We compare SOTA deep learning (DL) architectures such as the Transformer and TCN models for snowmelt prediction with traditional ML methods like SVR and previous DL techniques such as LSTM. Notably, our study incorporates a TCN architecture that has not yet been widely utilized for comprehensive comparison in snowmelt forecasting.
2. We use nested CV to evaluate the model's generalization capability in the context of snowmelt prediction, which has not been carried out in previous studies.

The remainder of the paper is organized as follows: Section 2 provides the materials and methods of the study. Section 3 then describes the methodology. The results of the study are then discussed in Section 4. The discussion part is presented in Section 5. Finally, Section 6 provides the conclusion and directions for future research.

2. Materials and Methods

2.1. Study Area

Within the Langtang basin, which is situated in Nepal's Central Himalayas, this study's main focus was on predicting the runoff from snowfall. The Langtang River basin covers an area of 353.59 km^2, with an elevation ranging from 3647 to 7213 m above sea level and with a mean slope of 26.7°. Glaciers occupy almost 39% of the total basin area and the remaining 61% is covered by rock and vegetation. The Randolph Glacier Inventory (RGI), part of the Global Land Ice Measurements from Space (GLIMS) initiative, version 6, calculated the extent of these glaciers [18]. The average annual discharge is 6.57 m^3/s according to the data collected from 2002 to 2012. An Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Digital Elevation Model (DEM) of a 30 m resolution from 2002 to 2012 was used. The entire amount of water discharge in this watershed is largely determined by the melting of glaciers and snow [19]. Figure 2 shows that a significant amount of water discharge occurs during the summer months (June to September). Figure 3 shows that a significant amount of snow is present during the winter (December to February) and the minimum SCA is present during the summer months (June to September). Figure 4 shows the HKH area that was used for the dataset collection.

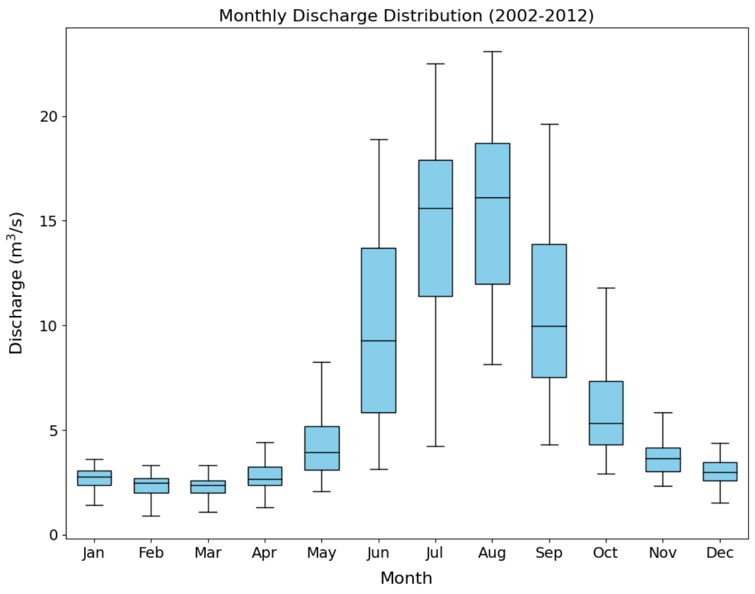

Figure 2. Monthly observed discharge for the Langtang basin from 2002 to 2012.

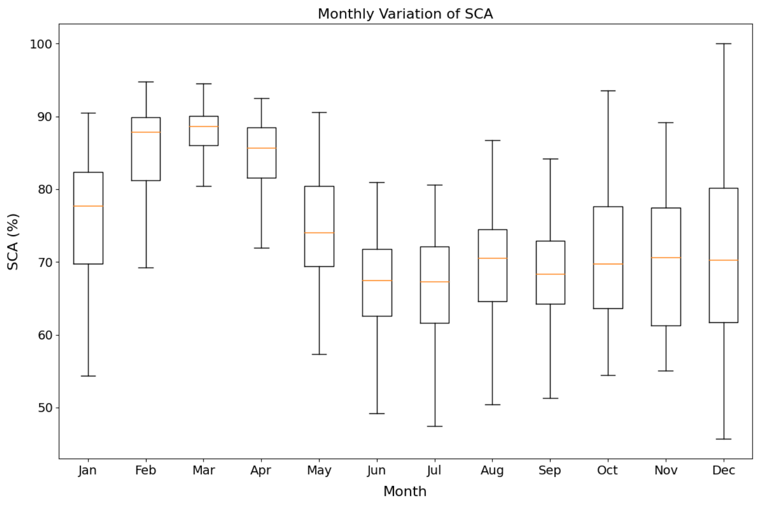

Figure 3. Monthly observed SCA for the Langtang basin during the research period (2002–2012).

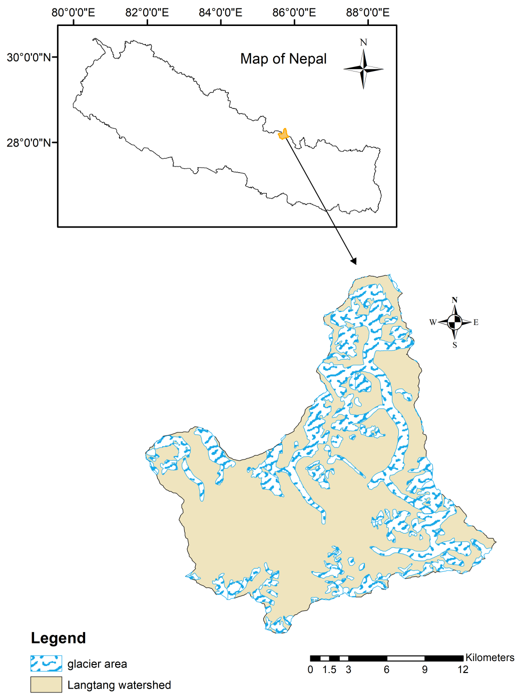

Figure 4. Map of the Langtang basin area.

2.2. Hydrometeorological Data

In Nepal, the Department of Hydrology and Meteorology provided the hydrological data from the Kyangjing station (28.216° latitude, 85.55° longitude) located in the Rasuwa District of Bagmati Province. The APHRODITE product (APHRO_TAVE_MA_V1808) provided diurnal temperature data with a $0.25° \times 0.25°$ grid for 2002 to 2012 [20]. The high associativity between APHRODITE products and the surface observations of the Langtang basin has been shown in a previous study [21]. The precipitation-related data were obtained from multiple detectors, including satellite infrared data to compute the precipitation rate, acquired from the Tropical Rainfall Measuring Mission (TRMM). The 3B42RT TRMM dataset was created by combining data from rain needles. TRMM products are available at https://pmm.nasa.gov and are constantly used in the Himalayan region [22].

2.3. Snow Cover

In the MODIS MOD10A2 interpretation, six products were employed to analyze snow cover. Stigter et al. [23] observed an accuracy of 83.1% with on-point snow compliances in the Langtang basin with this snow-mapping algorithm, which utilizes MODIS bands 4 and 6 to calculate the regularized-difference snow indicator [24]. In MOD10A2, a pixel is labeled as snow if snow is observed at least once over eight days. No snow label is marked if snow is absent throughout these eight days and cloud is marked if cloud cover is observed on all eight days. In this research work, we used datasets from the Langtang basin of the HKH, where we collected 496 images from 2002 to 2012. The area boundary was traced from the DEM and all of the images of MOD10A2 were converted to the coordinated system of World Geodetic System 1984 (WGS84) and the Universal Transverse Mercator (UTM) zone 45. The daily SCA for 365 days a year was interpolated or extrapolated from the 8-day maximum SCA using the cubical spline method to obtain 4015 total samples.

2.4. Dataset Preparation

SCA, Q, T, and P were used as inputs for forecasting the snowmelt runoff. We used MinMaxScaler to scale the data without losing their distribution for faster convergence during the training process. MinMaxScaler uses a data value and subtracts it from the minimum value of the overall features, which is then divided by the difference of the maximum value and the minimum value of features:

$$MinMaxScaler(m) = \frac{x - xmin}{xmax - xmin}, \quad (1)$$

where

x is the input value;
$xmin$ is the minimum value of the column;
$xmax$ is the maximum value of the column.

We used data from two days to predict the third day's SCA, i.e., a window size of 2. Furthermore, nested CV of the model was carried out to check the generalizability of any unseen data, with five folds for the outer loop and three folds for the inner loop. The tensor shape for our models was (window size, number of features), i.e., (2, 3) for three inputs and (2, 4) for four inputs. Keras' K-Fold was used for splitting the dataset with a random state of 42, which ensured that the data split could be replicated in the future. The number 42 was the seed value for the consistent reproduction of random data. The dataset was split into 80% for training (3212 samples) and 20% for testing (803 samples).

2.5. Experimental Setup

The simulations were performed for four different architectures:

1. SVR;
2. LSTM;
3. Transformer;
4. TCN.

Each of the four different architectures was evaluated for two different inputs: M1, with inputs T, Q, SCA, and P; and M2, with T, Q, and SCA. A list of the hyperparameters used for our experiments is given in Table 1.

The performance of each of the four architectures was evaluated using the metrics in the following table:

Table 1. Hyperparameters used to tune four different models for optimal performance.

Model	Hyperparameters	Values
SVR	C	[0.1, 1, 10]
	Epsilon	[0.01, 0.1, 0.2]
	Kernel	[Linear, RBF]
LSTM	LSTM Layers	[1, 2, 3]
	LSTM Units	[32, 64, 128]
	Dropout Rate	[0.2, 0.3, ..., 0.5]
	Optimizer	[Adam, Adamax, RMSProp, SGD]
	Learning Rate	[0.0001, 0.001, ..., 0.1]
Transformer	Transformer Blocks	[2, 4, 6, 8]
	Head Size	[8, 16, ..., 256]
	Number of Heads	[2, 4, ..., 16]
	FF Dim	[4, 8, ..., 64]
	Dropout Rate	[0.1, 0.2, ..., 0.6]
	Number of MLP Layers	[1, 2, 3]
	MLP Units	[32, 64, ..., 256]
	MLP Dropout	[0.1, 0.2, ..., 0.6]
	Optimizer	Adam, Adamax, RMSProp, SGD
	Learning Rate	[0.0001, 0.001, ..., 0.1]
TCN	TCN Layers	[1, 2, 3]
	Number of Filters	[32, 64, ..., 128]
	Kernel Size	[2, 3, 4]
	Optimizer	[Adam, Adamax, RMSProp, SGD]
	Learning Rate	[0.0001, 0.001, ..., 0.1]

2.5.1. Kling–Gupta Efficiency

The KGE is an evaluation metric that checks the performance of a model as it considers the data, correlation, bias, and variability. A value of KGE equal to 1 indicates a perfect agreement between simulations and observations [25].

$$KGE = 1 - \sqrt{(r-1)^2 + (\alpha - 1)^2 + (\beta - 1)^2}, \tag{2}$$

where
r is the Pearson's correlation coefficient;
$\beta = \frac{\bar{Q}'}{\bar{Q}}$ is the bias ratio;
$\bar{Q}$ is the average observed discharge;
$\bar{Q}'$ is the average simulated discharge;
$\alpha = \frac{CV_s}{CV_o}$ is the variability;
CV_o is the observed coefficient of variation;
CV_s is the simulated coefficient of variation

2.5.2. Nash–Sutcliffe Efficiency

The magnitude of residual variance concerning the observed data variance is provided by the NSE.

$$NSE = 1 - \frac{\sum_{t=1}^{N}(Q_t - Q'_t)^2}{\sum_{t=1}^{N}(Q_t - \bar{Q})^2}, \tag{3}$$

where
Q_t is the observed discharge at time t;
Q'_t is the simulated discharge at time t;
$\bar{Q}$ denotes the average observed discharge.

2.5.3. R Square (R^2)

The coefficient of determination R^2 evaluates the variation of dependent features with the independent features in a regression model. It ranges from 0 to 1, where 1 indicates a high correlation and 0 indicates a low correlation.

$$R^2 = \left[\frac{\sum_{t=1}^{n}(Q'_t - \bar{Q}')(Q_t - \bar{Q})}{\sqrt{\sum_{t=1}^{n}(Q'_t - \bar{Q}')^2}\sqrt{\sum_{t=1}^{n}(Q_t - \bar{Q})^2}}\right]^2, \tag{4}$$

where
Q_t is the observed discharge at time t;
Q'_t is the simulated discharge at time t;
$\bar{Q}$ is the average observed discharge;
$\bar{Q}'$ is the average simulated discharge.

2.5.4. Root-Mean-Square Error

The RMSE evaluation metric is used to measure the average differences between predicted and measured values. It represents the standard deviation of the error, with a lower RMSE value indicating a better fit. The RMSE is sensitive to large errors and is calculated using the following equation:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(Q'_t - Q_t)^2}{n}}, \tag{5}$$

where

Q_t is the observed discharge at time t;
Q'_t is the simulated discharge at time t.

2.5.5. Mean Absolute Error

The MAE measures the average magnitude of errors between actual and predicted values. A low MAE value represents a poor performance of the model. A lower MAE represents lower error. The equation for calculating the MAE is given below:

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |Q_t - Q'_t|, \tag{6}$$

where
Q_t is the observed discharge at time t;
Q'_t is the simulated discharge at time t.

3. Methodology

In this section, the proposed methodology for snowmelt runoff prediction is shown in Figure 5. First, we preprocessed the dataset using MinMaxScaler to scale the data and set the value of the window size to 2 for the third day's SCA prediction. The prepared data were then split into outer and inner folds in a random state of 42, with a splitting ratio of 80%, i.e., 3212 samples, and 20%, i.e., 803 samples, for training and testing, respectively. These split data were used for the traditional ML and DL architectures, as shown in Figure 5. The trained models were evaluated using metrics such as the RSME, MAE, R^2, KGE, and NSE, which were then used for comparing the models. These performance metrics helped to create a baseline for model performance comparison and highlight the best-performing traditional ML or DL model.

In Figure 5, the model section contains traditional ML and deep learning architectures. Detailed explanations of the four different models used in our simulations are described below.

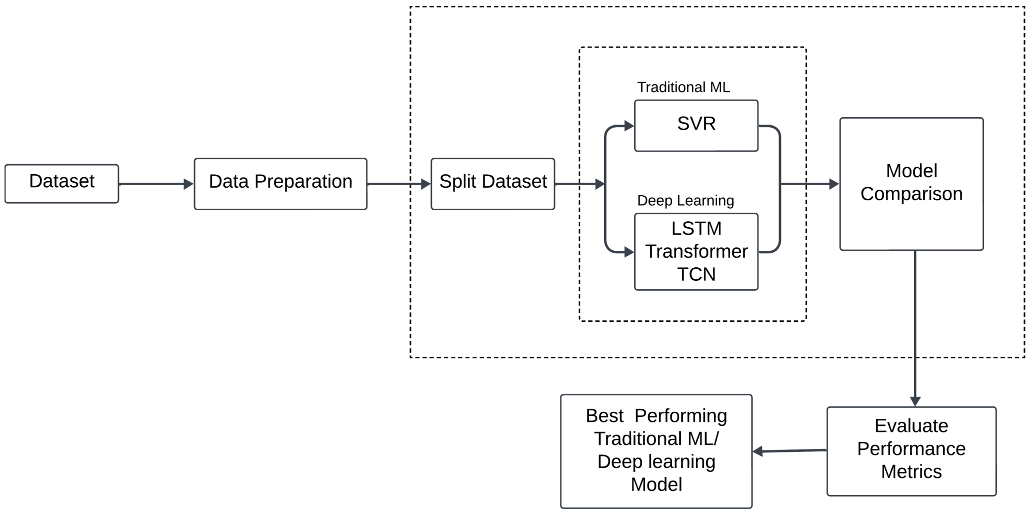

Figure 5. Methodological framework for assessing the performance of ML techniques in snowmelt forecasting.

3.1. SVR

The SVR model, proposed by Vapnik in 1999 [26], is a traditional ML algorithm designed for classification and regression. SVR can use both linear and non-linear kernels. SVR determines the hyperplane in high-dimensional space to separate the classes or output values. The main objective of SVR is to minimize the error value between the predicted and actual values to obtain the correct prediction. In this study, we used a Radial Basic

Function (RBF) kernel with SVR; an RBF kernel is capable of capturing non-linear patterns. The RBF kernel mathematical expression is given below.

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{7}$$

where

x_i and x_j are the input feature vectors;
σ is a parameter controlling the width of the kernel.

In this study, we used 0.1, 1, and 10 as regularization parameters (C) and 0.01, 0.1, and 0.2 as epsilon (ϵ) values for the hyperparameter tuning. We also used both linear and RBF techniques for the kernel. All of these hyperparameters were determined using the grid-search approach. The best hyperparameters after tuning were 0.1 for C, 0.01 for the epsilon value, and the linear kernel.

3.2. LSTM

LSTM, proposed by Hochreiter and Schmidhube [27], is a variation of an RNN used widely in DL architecture. Unlike the vanilla RNN, LSTM is capable of capturing long-term dependencies and solving the problems of exploding and vanishing gradients. LSTM consists of feedback connections, which allow it to process sequences of data. It consists of three different gates: an input gate, a forget gate, and an output gate. The forget gate removes information that is no longer useful in the cell state. Additional new information is then added into the cell state by the input gate. The output gate extracts the useful information from the cell state to be presented as the output. A classic LSTM cell is shown in Figure 6, and its related equations are below.

Forget gate: $f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f)$
Input gate: $i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i)$
Update vector: $\hat{c} = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c)$
Cell state: $f_t \odot C_{t-1} + i_t \odot \hat{c}_t$
Output gate: $\sigma(w_o \cdot [h_{t-1}, x_t] + b_o)$

where

w_f, w_i, w_c, and w_o are weights for the forget gate, input gate, cell state, and output gate, respectively;
b_f, b_i, b_c, and b_o are biases for the forget gate, input gate, cell state, and output gate, respectively;
σ represents the sigmoid activation function;
$[h_{t-1}, x_t]$ represents the concatenation of the current input and the previous hidden state.

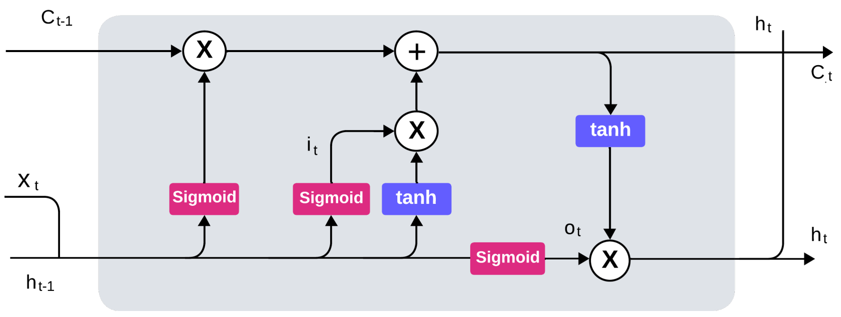

Figure 6. The detailed LSTM architecture, where *C* denotes the cell state, *h* denotes the hidden state, *o* denotes the output gate, *i* denotes input vectors at time step *t*, and *tanh* represents a hyperbolic tangent function [27].

The hyperparameters we used to tune our LSTM model were three different numbers of LSTM layers; three different LSTM hidden units, with values of 32, 64, and 128; dropout values ranging from 0.2 to 0.5; learning rate values ranging from 0.0001 to 0.1; and five different optimizers, including Adam, Adamax, RMSProp, and SGD. The hyperparameter tuning was carried out using a random search in the Keras tuner. For the optimal hyperparameters in our model, we utilized the Adam optimizer with a learning rate of 0.004 and a dropout rate of 0.2, and incorporated two LSTM layers, each consisting of 96 hidden units.

3.3. Transformer

A Transformer, proposed by Vaswani et al. [28], is a DL architecture that includes an encoder and a decoder. The encoder and decoder are connected through an attention mechanism. This architecture requires less training time than LSTM, as its latest version is highly used for training Large Language Models (LLMs). This architecture is applicable for Natural Language Processing (NLP) and computer vision, but also for audio and multi-modal processing. Figure 7 shows the Transformer model architecture, and its components are described below.

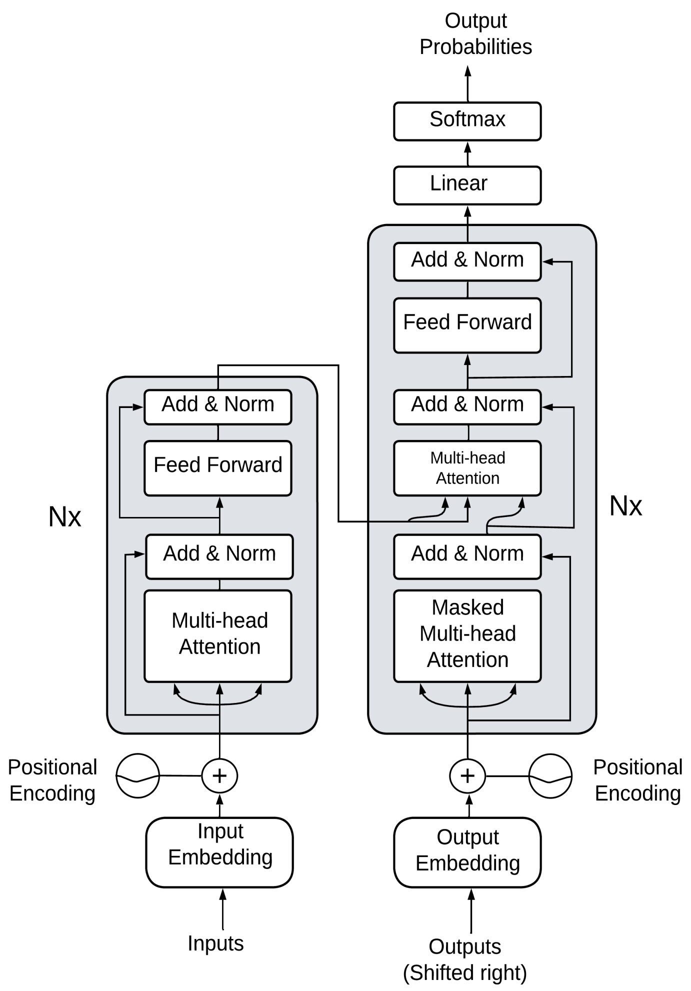

Figure 7. Detailed architecture of the Transformer model [28].

3.3.1. Self-Attention Mechanism

A self-attention mechanism is responsible for capturing any crucial information in a long sequence of data. The calculation for attention can be expressed as a large matrix calculation using the softmax function:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{8}$$

where

$\mathbf{Q}$ represents the query;
$\mathbf{K}$ represents the key;
$\mathbf{V}$ represents the value;
d_k represents the dimensionality of the key in the items.

3.3.2. Multi-Head Attention

Multi-head attention is a feature of the Transformer that helps the model process different aspects of an input sequence simultaneously. In this process, multiple self-attention heads work in parallel, and outputs are concatenated and linearly transformed.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{h}_1, \ldots, \mathbf{h}_n)\mathbf{W}_o, \tag{9}$$

where
$\mathbf{h}$ represents the heads;
$\mathbf{W}_o$ represents the matrix of the entire multi-head attention mechanism.

3.3.3. Position Encoding

Position encoding in the model architecture is responsible for the order of the words in a sequence, and this is then added to the input embedding.

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{1000^{(2i/d_{\text{model}})}}\right), \tag{10}$$

$$\text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{1000^{(2i/d_{\text{model}})}}\right), \tag{11}$$

where
pos is the position of the sequence;
d_{model} is the dimensionality of the model.

In our Transformer, we used four different Transformer blocks, with values of 2, 4, 6, and 8; head sizes ranging from 8 to 256; different numbers of heads, ranging from 2 to 16; dropout rate values ranging from 0.1 to 0.6; three different numbers of MLP layers; MLP unit values ranging from 32 to 256; MLP dropout values ranging from 0.1 to 0.6; four different optimizers, namely Adam, SGD, RMSProp, and Adamax; and learning rate values from 0.0001 to 0.1 as hyperparameters to tune the model. We used a Keras tuner for the hyperparameter tuning. The optimal hyperparameters were two Transformer blocks, each with a head size of 136, and two attention heads. The two MLP layers were employed with units of 192 and 160, respectively. Dropout regularization is a technique that involves ignoring some layers' outputs during training to prevent overfitting, and in our experimental setup, 0.41, which means 41%, of the neurons were randomly dropped. Similarly, the dropout values were 0.10 and 0.11 for the second and third layers, respectively. The Adam optimizer with a learning rate of 0.0044 was used for training.

3.4. TCN

A TCN, proposed by Bai et al. [17], is a DL architecture that is capable of capturing long-term dependencies and temporal patterns. The TCN architecture is capable of preventing data leakage, using causal convolution by ensuring that the model's prediction is solely based on past and present information. The TCN architecture can handle sequences of varying lengths, allowing for the efficient capture of long-range dependencies. Figure 8 shows the TCN model architecture, and its key components are described below.

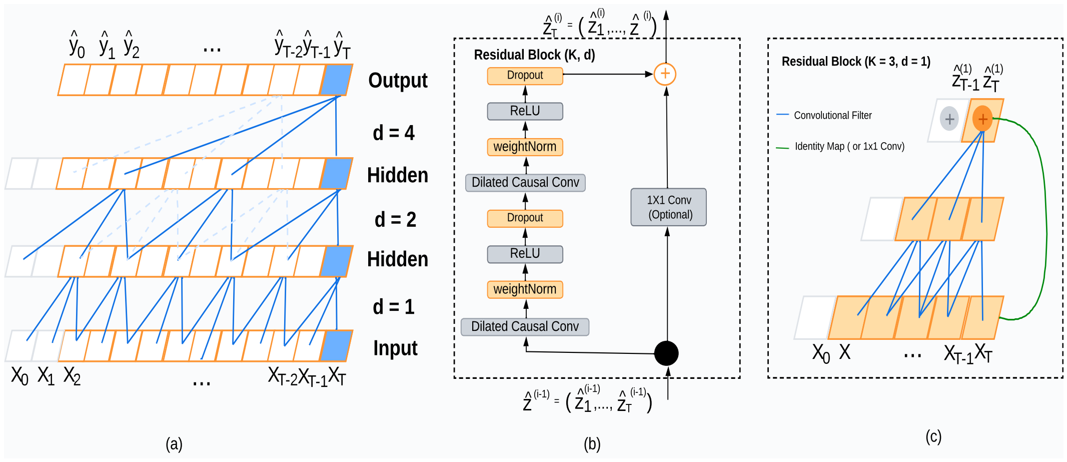

Figure 8. TCN architectural elements. (**a**) A dilated causal convolution, with dilated factors d = 1, 2, 4 and filter size k = 3. (**b**) A TCN residual block. A 1 × 1 convolution matrix is added when the residual input and output have different dimensions. (**c**) An example of a residual connection in a TCN, where the blue lines represent the residual function and the green line represents identity mapping [17].

3.4.1. Dilated Causal Convolutions

In a TCN, the dilated causal convolutions can process any length of sequence at varying receptive field sizes. The dilated convolutions are responsible for capturing long-range dependencies and are useful for memory management.

$$y_t = \sum_{i=1}^{k} w_i \cdot x_{t-d_t}, \tag{12}$$

where

y_t is the output at time t;
w_i is the weight of the convolutional filter;
x_t is the input at time t;
d is the dilation rate;
k is the filter size.

3.4.2. Residual Blocks

A TCN uses residual connections within its blocks that help to solve the vanishing-gradient problem and allow for the effective optimization of its deeper architecture. A TCN residual block consists of two layers of dilated causal convolutions and non-linearity, for which a rectified linear unit is used.

3.4.3. Temporal Skip Connection

TCNs have skip connections that connect every other layer, which makes the network capable of capturing and reusing the information. The skip connection feature is also useful for promoting gradient flow and can also enhance model's capability to learn hierarchical representations.

In this study, the TCN model used three different TCN layers. We used 32 to 256 filters; three kernel sizes, such as 2, 3, and 4; five different optimizers, including Adam, Adamax, RMSProp, and SGD; and learning rate values ranging from 0.0001 to 0.1 as hyperparameters for tuning the model. We used a Keras tuner for the hyperparameter tuning. The optimal hyperparameters were a single TCN layer with 96 filters and a kernel size of 3, with the Adam optimizer, alongside a learning rate of 0.0057.

4. Results

This section presents a comprehensive analysis of each model's performance across different scenarios. We evaluate the effectiveness of both three-input and four-input models in each fold, comparing their performance against other models. Additionally, we provide insights into the testing time required for each model.

4.1. Performance Analysis of ML Models with Four Inputs

This subsection describes the performance of ML models with four inputs in each fold of nested CV, as given in Table 2.

Table 2. Performance comparison of different models with four inputs, assessing each fold using the MAE, RMSE, R^2, KGE, and NSE.

Model	Folds	MAE	RMSE	R^2	KGE	NSE
SVR	1	0.0341	0.0451	0.9657	0.8967	0.9657
	2	0.0312	0.0401	0.9723	0.9291	0.9723
	3	0.0306	0.0406	0.9710	0.9227	0.9710
	4	0.0338	0.0450	0.9672	0.9175	0.9672
	5	0.0313	0.0395	0.9734	0.9220	0.9734
LSTM	1	0.0161	0.0301	0.9847	0.9727	0.9847
	2	0.0130	0.0235	0.9905	0.9889	0.9905
	3	0.0126	0.0239	0.9901	0.9693	0.9903
	4	0.0130	0.0242	0.9905	0.9713	0.9905
	5	0.0121	0.0220	0.9918	0.9735	0.9918
Transformer	1	0.0195	0.0308	0.9840	0.9855	0.9840
	2	0.0177	0.0274	0.9870	0.9880	0.9870
	3	0.0186	0.0283	0.9860	0.9819	0.9860
	4	0.0185	0.0281	0.9872	0.9859	0.9872
	5	0.0177	0.0258	0.9887	0.9883	0.9887
TCN	1	0.0126	0.0269	0.9878	0.9914	0.9878
	2	0.0110	0.0227	0.9911	0.9927	0.9911
	3	0.0107	0.0214	0.9919	0.9913	0.9919
	4	0.0109	0.0230	0.9914	0.9926	0.9914
	5	0.0098	0.0192	0.9937	0.9931	0.9937

Table 2 shows the performance of the DL architectures and the traditional ML algorithms. The SVR model shows the worst performance for each fold of nested CV among all other models, with average MAE, RMSE, R^2, KGE, and NSE values of 0.032, 0.042, 0.97, 0.918, and 0.97, respectively. The LSTM model shows a significant performance improvement over SVR in terms of its average MAE, RMSE, R^2, KGE, and NSE, at 0.013, 0.025, 0.989, 0.975, and 0.989, respectively. The Transformer achieved a better performance compared to the LSTM model, with an average KGE value of 0.986. However, the TCN shows superior performance compared to the other models due to its capability of capturing long-range dependencies, where the average performance metric values for the MAE, RMSE, R^2, KGE, and NSE are 0.011, 0.023, 0.991, 0.992, and 0.991, respectively. It is important to note that we used data points from the testing set of the best-predicted model fold to avoid data leakage resulting from systematic data separation using nested CV. In the scatter-plot diagram, the discharged river is estimated correctly near the line, whereas points above and below the diagonal overestimate and underestimate the actual flow, respectively. Figure 9 shows that the SVR and Transformer models predicted medium flows well, but the higher and lower flows were overestimated and underestimated, respectively. The low and medium flows were estimated accurately by the LSTM model but the high flows were only accurately estimated by the TCN model. Therefore, the TCN model predicted all types of flows accurately when compared to the other models.

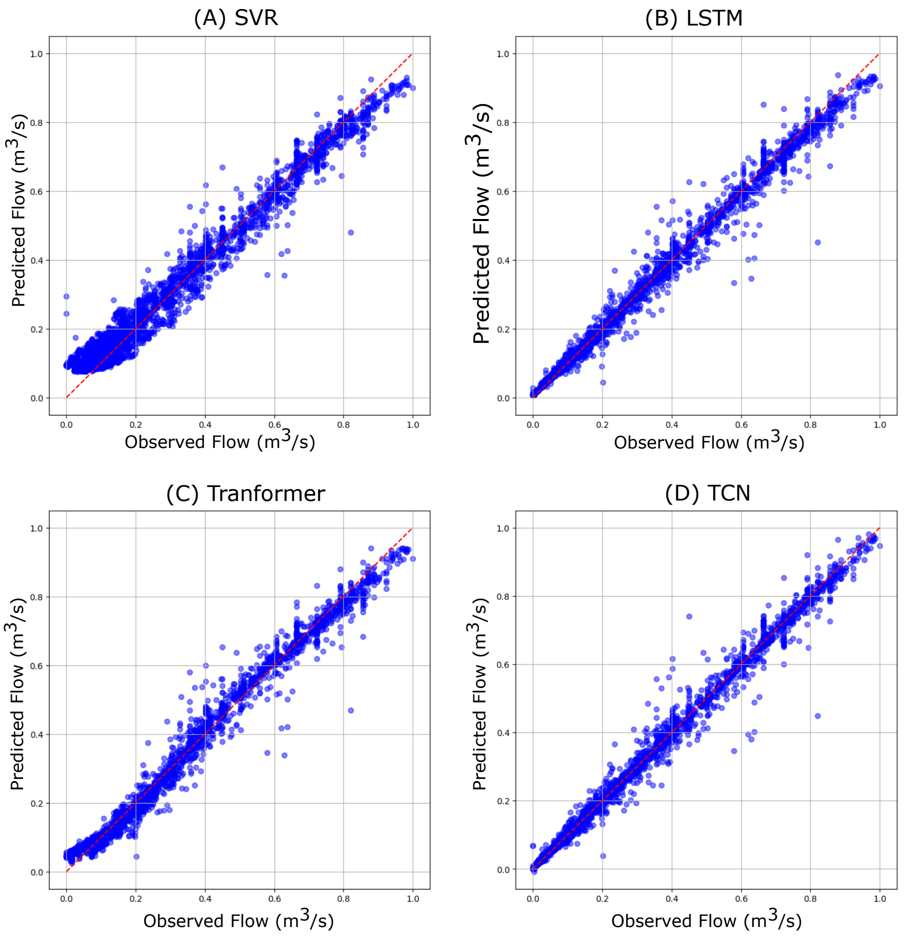

Figure 9. Scatter plot of model predictions, with the red dotted 1:1 solid line having four inputs.

4.2. Performance Analysis of ML Models with Three Inputs

In this subsection, the performance of ML models with three inputs in each fold of nested CV is given in Table 3.

Table 3. Performance comparison of different models with three inputs, assessing each fold using the MAE, RMSE, R^2, KGE, and NSE.

Model	Folds	MAE	RMSE	R^2	KGE	NSE
SVR	1	0.0317	0.0426	0.9693	0.9316	0.9693
	2	0.0300	0.0386	0.9743	0.9403	0.9743
	3	0.0288	0.0374	0.9755	0.9405	0.9755
	4	0.0300	0.0395	0.9747	0.9335	0.9746
	5	0.0284	0.0368	0.9769	0.9369	0.9769
LSTM	1	0.0143	0.0272	0.9875	0.9768	0.9875
	2	0.0136	0.0236	0.9903	0.9757	0.9903
	3	0.0112	0.0231	0.9905	0.9812	0.9905
	4	0.0116	0.0232	0.9912	0.9840	0.9912
	5	0.0106	0.0206	0.9927	0.9875	0.9927
Transformer	1	0.0178	0.0305	0.9842	0.9663	0.9842
	2	0.0161	0.0262	0.9881	0.9781	0.9881
	3	0.0162	0.0266	0.9875	0.9751	0.9875
	4	0.0126	0.0243	0.9904	0.9826	0.9904
	5	0.0115	0.0213	0.9922	0.9831	0.9922

Table 3. *Cont.*

Model	Folds	MAE	RMSE	R^2	KGE	NSE
TCN	1	0.0122	0.0269	0.9877	0.9806	0.9877
	2	0.0108	0.0217	0.9918	0.9891	0.9918
	3	0.0113	0.0223	0.9913	0.9865	0.9913
	4	0.0115	0.0228	0.9915	0.9888	0.9915
	5	0.0104	0.0201	0.9931	0.9904	0.9931

The SOTA DL architectures achieved higher levels of performance when compared to the traditional ML algorithms, as shown in Table 3. The SVR model shows the minimum performance over each fold in comparison to the other models, with average MAE, RMSE, R^2, KGE, and NSE values of 0.030, 0.039, 0.974, 0.937, and 0.974, respectively. The LSTM model shows a significant performance improvement over SVR, with average MAE, RMSE, R^2, KGE, and NSE values of 0.012, 0.024, 0.99, 0.981, and 0.99, respectively. The LSTM model shows a better performance, with an average KGE value of 0.981, than the Transformer model, which shows an average KGE value of 0.977. However, the TCN also shows superior performance for the three inputs due to its capability of capturing long-range dependencies, where the average performance metric values for the MAE, RMSE, R^2, KGE, and NSE are 0.011, 0.023, 0.991, 0.987, and 0.991, respectively. Figure 10 shows that the low and medium flows were estimated well by the other models. However, the high flows were only accurately predicted with the TCN model.

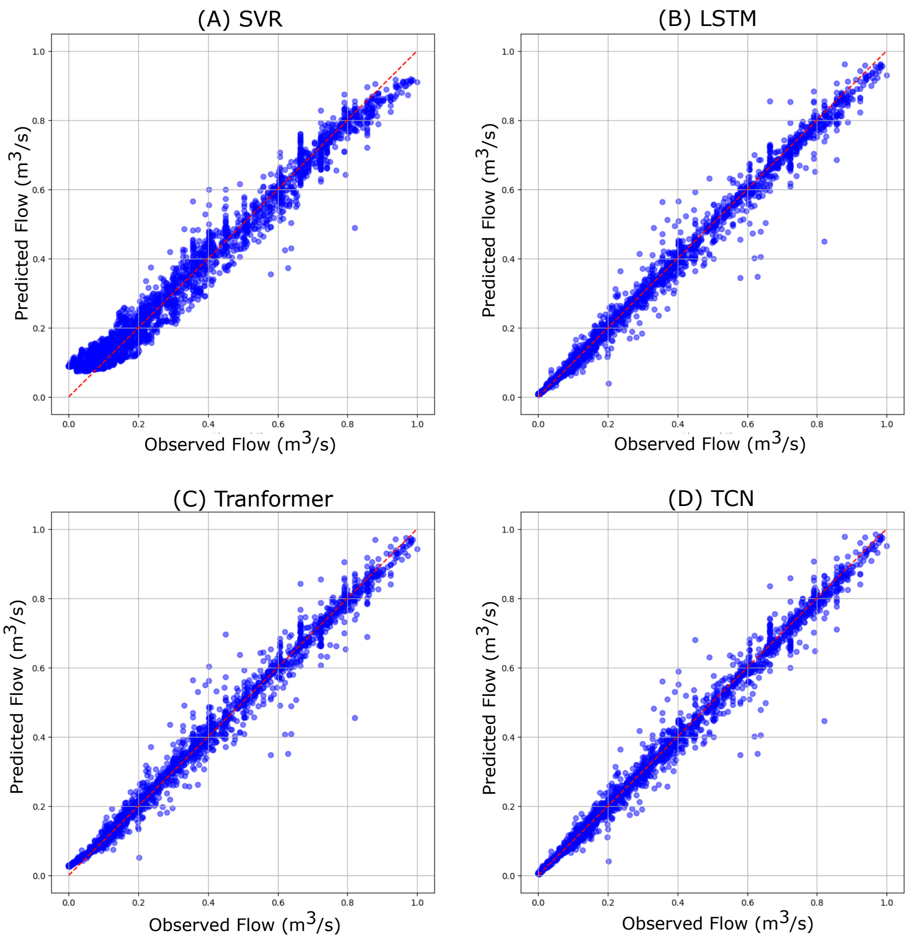

Figure 10. Scatter plot of model predictions, with the red dotted 1:1 solid line having three inputs.

4.3. Overall Comparison of ML models

This section describes the overall performance of the models by averaging all of the performance metrics for the M1 and M2 input types given in Tables 2 and 3.

Table 4 shows the values of different performance metrics obtained for all of the models over the five outer folds. In this table, four different models' sensitivity analyses are compared for two input conditions, i.e., all four inputs (M1) and three inputs (M2) without precipitation. The performance of the SVR and LSTM models was better for the M2 input, while the Transformer model achieved better results with M2 for most of metrics except the KGE. However, the TCN model performed systematically better than all other models, as shown in Table 4, and the inclusion of the precipitation (P) input slightly improved the verification statistics for the TCN model. However, in this overall comparison, the results we obtained for inputs M1 and M2 are almost identical. The results obtained in the study by Thapa et al. [16] show a very similar RMSE value for the M1 and M2 inputs. The final value of RMSE reached in the aforementioned paper was achieved using the hit-and-trial method, which is not the best practice for tuning hyperparameters to increase a model's performance. Hence, our research demonstrates the fact that the method of systematic hyperparameter tuning with the Keras tuner can significantly reduce the RMSE value for both M1 and M2 when compared to the hit-and-trial method used by Thapa et al. [16]. Furthermore, we used a persistence model as a baseline model in order to evaluate its performance with the other models. The TCN model outperformed the benchmark set achieved with the persistent model after the addition of the P parameter.

Table 4. Overall performance comparison with four inputs (M1) and three inputs (M2) for various models, where the performance metrics are the MAE, RMSE, R^2, KGE, and NSE.

Model	**Inputs**	**MAE**	**RMSE**	R^2	**KGE**	**NSE**
SVR	M1	0.032	0.042	0.970	0.918	0.970
	M2	0.030	0.039	0.974	0.937	0.975
LSTM	M1	0.013	0.025	0.989	0.975	0.989
	M2	0.012	0.024	0.990	0.981	0.990
Transformer	M1	0.018	0.028	0.987	0.986	0.987
	M2	0.015	0.026	0.989	0.977	0.989
TCN	M1	0.011	0.023	0.991	0.992	0.991
	M2	0.011	0.023	0.991	0.987	0.991
Persistence	**N/A**	0.0143	0.0310	0.9837	0.991	0.983

4.4. Testing Time

For this subsection, we computed the testing time in each outer fold for different models and averaged these testing times, as shown in Table 5. We do not report the testing time of the SVR model based on several considerations, including the computational efficiency of SVR relative to the other models evaluated in our study, i.e., in comparison to more complex models like the LSTM, TCN, and Transformer models. The training and testing times for SVR are generally lower due to its simpler architecture and optimization algorithms, which are specifically designed for efficiency in high-dimensional feature spaces. By focusing our testing-time analysis on the LSTM, TCN, and Transformer models, we aim to highlight the computational characteristics of these newer and more sophisticated models, which are of greater interest to the research community given their potential for handling complex sequential data. The TCN model had a faster computational testing time than any other DL architecture, as shown in Table 5.

Table 5. Model testing time (seconds) comparison.

Models	Testing Time (s)
LSTM	0.694
Transformer	0.652
TCN	0.372

5. Discussion

Snowmelt modeling is crucial for water resource management, optimizing hydroelectricity power generation, irrigation, and disaster preparation for events like landslides and avalanches, so it is necessary to explore new techniques for appropriate snowmelt forecasting. Taking the Langtang basin as an example, we applied SOTA DL architectures such as Transformer and TCN models for snowmelt prediction, which have not been used before for snowmelt forecasting. The values of NSE for the SVR, LSTM, Transformer, and TCN models were 97.5%, 99%, 98.9%, and 99.1%, respectively. However, an ANN model proposed in a past study showed a 93% model efficiency for the Upper Euphrates basin of Turkey, which is comparatively less than that of traditional ML models [10]. The NSE value of the LSTM model in [14] was 99.2%, which is comparable to our LSTM model result as our study and their study both focus on predicting discharge.

Previous studies show that single-layered LSTM is being used for prediction [14,15], but the model proposed by Kratzert et al. [13] used two-layered LSTM; however, these studies do not explore the performance of models using different layers of LSTM. In Thapa et al.'s study [16], the performance of LSTM models with different layers of LSTM, optimizers, and window sizes was compared and evaluated; however, the manual tuning of hyperparameters without the use of any systematic approach is questionable for achieving the best model performance. In our study, we used a Keras tuner for the hyperparameter tuning to increase the model performance. After hyperparameter tuning, we found that two-layered stacked LSTM performs better than single-layered LSTM. In this research work, we evaluated the performance of five optimizers across different layers and found that the Adam optimizer with a learning rate of 0.004 performed better. The performance of all of the models for each fold is shown in Tables 2 and 3.

The majority of research work using ML models does not exploit the most recent SOTA deep learning technologies [10,13,15,16]. In past endeavors, traditional approaches such as SVR, ANNs, and LSTM were used for snowmelt forecasting. In our study, we used recent DL architectures such as Transformer and TCN models; these models have been shown to be capable of capturing long-range dependencies, and they outperformed all other traditional ML models. We found that the TCN model exhibited superior performance compared to the other models, with a model efficiency of 99.1% in terms of the NSE. In the model development phase, input selection is an important task, but it is often neglected. In the study by Thapa et al. [16], it was found that gamma testing can help to determine the appropriate input combination; so, in this study, we compared each model based on different input types and evaluated that the models with the M2 input type had a better performance, which is shown in Table 4. However, the TCN model had a better model efficiency with the M1 input, while all other models performed better with the M2 input, which might be due to variations in the spatial and temporal resolution or the relevance of the input variable. In this study, we used nested CV to make the models generalizable. However, the model generalizability depends on the distribution of the input data. Therefore, we used 10 years (2002–2012) of data with features such as Q, P, SCA, and T. This dataset encompasses a diverse range of data distributions, enabling the model to learn different distributions. Nevertheless, the model performance will be lower when using data with unknown distributions, such as those influenced by climatic changes, like SCA, T, and P. Given that the model encounters completely different distributions, when compared to a model trained with a specific distribution, the ML model will have a poor performance [29]. Furthermore, the authors of [30] evaluated the performance of a DL model under different distributions and found that the proposed SOTA model was still far from solving this problem. Ground-truth observation in the Himalayan basins is difficult due to the high elevation difference between the stations. Hence, the remotely sensed SCA and meteorological products are important assets. This research proves the effectiveness of ML models in regions with limited data, and it also signifies that SOTA models are more applicable for snowmelt forecasting as they capture long-range dependencies and are more generalizable.

6. Conclusions

In this work, we investigated snowmelt-driven streamflow forecasting using ML techniques. The experiments were conducted over the HKH area. We compared the performance of SVR, LSTM, Transformer, and TCN architectures with M1 and M2 inputs for forecasting snowmelt. We found no significant differences in performance between the M1 and M2 inputs, which indicates that precipitation is not important for forecasting streamflow during the snowmelt period. In this study, we have also highlighted the importance of hyperparameter tuning to obtain a better model. The TCN architecture showcased superior performance compared to the other ML techniques. Furthermore, we disclosed the benefits of nested CV for evaluating the performance of different ML techniques in snowmelt-driven streamflow forecasting. This contribution provides valuable insights for decision-makers in environmental monitoring and resource management, emphasizing the importance of leveraging advanced deep learning architecture for more reliable and precise forecasting.

Author Contributions: Conceptualization, U.T., B.M.P. and S.T.; investigation, U.T. and B.M.P.; methodology, U.T.; resources, S.T.; supervision, B.M.P. and S.T.; writing—original draft preparation, U.T.; writing—review and editing, U.T., B.M.P., S.T., D.P. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Advanced College of Engineering and Management, Tribhuvan University.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors extend their heartfelt gratitude to all Electronics and Computer Department members at the Advanced College of Engineering and Management. Special appreciation is also extended to the AI Research Center and Advanced College of Engineering and Management for their generous financial support and management of the experimental setup.

Conflicts of Interest: No conflicts of interest are declared by the authors regarding the experimental setup; data analyses; interpretation of the results; or documentation of the manuscript.

References

1. Panday, P.K.; Frey, K.E.; Ghimire, B. Detection of the timing and duration of snowmelt in the Hindu Kush-Himalaya using QuikSCAT, 2000–2008. *Environ. Res. Lett.* **2011**, *6*, 024007. [CrossRef]
2. Wester, P.; Mishra, A.; Mukherji, A.; Shrestha, A.B. *The Hindu Kush Himalaya Assessment: Mountains, Climate Change, Sustainability and People*; Springer Nature: New York, NY, USA, 2019.
3. Griessinger, N.; Schirmer, M.; Helbig, N.; Winstral, A.; Michel, A.; Jonas, T. Implications of observation-enhanced energy-balance snowmelt simulations for runoff modeling of Alpine catchments. *Adv. Water Resour.* **2019**, *133*, 103410. [CrossRef]
4. Ohmura, A. Physical basis for the temperature-based melt-index method. *J. Appl. Meteorol. Climatol.* **2001**, *40*, 753–761. [CrossRef]
5. Massmann, C. Modelling snowmelt in ungauged catchments. *Water* **2019**, *11*, 301. [CrossRef]
6. ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. Artificial neural networks in hydrology. I: Preliminary concepts. *J. Hydrol. Eng.* **2000**, *5*, 115–123. [CrossRef]
7. ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. Artificial neural networks in hydrology. II: Hydrologic applications. *J. Hydrol. Eng.* **2000**, *5*, 124–137. [CrossRef]
8. Callegari, M.; Mazzoli, P.; De Gregorio, L.; Notarnicola, C.; Pasolli, L.; Petitta, M.; Pistocchi, A. Seasonal river discharge forecasting using support vector regression: A case study in the Italian Alps. *Water* **2015**, *7*, 2494–2515. [CrossRef]
9. De Gregorio, L.; Callegari, M.; Mazzoli, P.; Bagli, S.; Broccoli, D.; Pistocchi, A.; Notarnicola, C. Operational river discharge forecasting with support vector regression technique applied to alpine catchments: Results, advantages, limits and lesson learned. *Water Resour. Manag.* **2018**, *32*, 229–242. [CrossRef]

10. Uysal, G.; Şensoy, A.; Şorman, A.A. Improving daily streamflow forecasts in mountainous Upper Euphrates basin by multi-layer perceptron model with satellite snow products. *J. Hydrol.* **2016**, *543*, 630–650. [CrossRef]
11. Granata, F.; Di Nunno, F.; Najafzadeh, M.; Demir, I. A stacked machine learning algorithm for multi-step ahead prediction of soil moisture. *Hydrology* **2022**, *10*, 1. [CrossRef]
12. Nagesh Kumar, D.; Srinivasa Raju, K.; Sathish, T. River flow forecasting using recurrent neural networks. *Water Resour. Manag.* **2004**, *18*, 143–161. [CrossRef]
13. Kratzert, F.; Klotz, D.; Brenner, C.; Schulz, K.; Herrnegger, M. Rainfall–runoff modelling using long short-term memory (LSTM) networks. *Hydrol. Earth Syst. Sci.* **2018**, *22*, 6005–6022. [CrossRef]
14. Le, X.H.; Ho, H.V.; Lee, G.; Jung, S. Application of long short-term memory (LSTM) neural network for flood forecasting. *Water* **2019**, *11*, 1387. [CrossRef]
15. Fan, H.; Jiang, M.; Xu, L.; Zhu, H.; Cheng, J.; Jiang, J. Comparison of long short term memory networks and the hydrological model in runoff simulation. *Water* **2020**, *12*, 175. [CrossRef]
16. Thapa, S.; Zhao, Z.; Li, B.; Lu, L.; Fu, D.; Shi, X.; Tang, B.; Qi, H. Snowmelt-driven streamflow prediction using machine learning techniques (LSTM, NARX, GPR, and SVR). *Water* **2020**, *12*, 1734. [CrossRef]
17. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
18. RGI Consortium; Randolph Glacier Inventory. *A Dataset of Global Glacier Outlines: Version 6.0*; Global Land Ice Measurements from Space: Boulder, CO, USA, 2017.
19. Ragettli, S.; Pellicciotti, F.; Immerzeel, W.W.; Miles, E.S.; Petersen, L.; Heynen, M.; Shea, J.M.; Stumm, D.; Joshi, S.; Shrestha, A. Unraveling the hydrology of a Himalayan catchment through integration of high resolution in situ data and remote sensing with an advanced simulation model. *Adv. Water Resour.* **2015**, *78*, 94–111. [CrossRef]
20. Yasutomi, N.; Hamada, A.; Yatagai, A. Development of a long-term daily gridded temperature dataset and its application to rain/snow discrimination of daily precipitation. *Glob. Environ. Res.* **2011**, *15*, 165–172.
21. Thapa, S.; Li, B.; Fu, D.; Shi, X.; Tang, B.; Qi, H.; Wang, K. Trend analysis of climatic variables and their relation to snow cover and water availability in the Central Himalayas: A case study of Langtang Basin, Nepal. *Theor. Appl. Climatol.* **2020**, *140*, 891–903. [CrossRef]
22. Immerzeel, W.W.; Droogers, P.; De Jong, S.; Bierkens, M. Large-scale monitoring of snow cover and runoff simulation in Himalayan river basins using remote sensing. *Remote Sens. Environ.* **2009**, *113*, 40–49. [CrossRef]
23. Stigter, E.E.; Wanders, N.; Saloranta, T.M.; Shea, J.M.; Bierkens, M.F.; Immerzeel, W.W. Assimilation of snow cover and snow depth into a snow model to estimate snow water equivalent and snowmelt runoff in a Himalayan catchment. *Cryosphere* **2017**, *11*, 1647–1664. [CrossRef]
24. Hall, D.K.; Riggs, G.A.; Salomonson, V.V.; DiGirolamo, N.E.; Bayr, K.J. MODIS snow-cover products. *Remote Sens. Environ.* **2002**, *83*, 181–194. [CrossRef]
25. Kling, H.; Fuchs, M.; Paulin, M. Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. *J. Hydrol.* **2012**, *424*, 264–277. [CrossRef]
26. Vapnik, V. *The Nature of Statistical Learning Theory*; Springer Science & Business Media: New York, NY, USA, 1999.
27. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
29. Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv. (CSUR)* **2014**, *46*, 1–37. [CrossRef]
30. Guillory, D.; Shankar, V.; Ebrahimi, S.; Darrell, T.; Schmidt, L. Predicting with confidence on unseen distributions. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 10–17 October 2021; pp. 1134–1144.